Urban Wireless Mesh Network Planning: The Case of Directional Antennas

Chun-cheng Chen and Chandra Chekuri
Department of Computer Science, University of Illinois at Urbana-Champaign

Abstract—As more and more cities plan to deploy municipal wireless mesh networks to provide publicly-accessible infrastructure, it becomes critical to come up with a deployment plan that provides sufficient accessibility with financial and technological constraints. In this paper, we consider deploying the mesh routers that are equipped with directional antennas to form the mesh backbone. We propose GPSR, an efficient and near-optimal algorithm that maximizes the profit of the deployment, ensures the deployment cost is under budget, and guarantees connectivity and system robustness. Furthermore, GPSR is general enough to address many practically-encountered variants of the mesh network planning problem. Our performance evaluation shows that GPSR almost always generates the deployment plan within 95% of the optimal for all of our experiments.

I. INTRODUCTION

Municipal wireless mesh network holds great potential to provide cost-efficient publicly-accessible broadband services. For example, Singapore’s Wireless@SG [2] aims to deploy a nationwide publicly-accessible mesh network with speed up to 512 Kbps. The City of Cambridge borrows the Roofnet [1] idea developed by MIT. The advent of publicly-accessible broadband services can be predicted. When an entire city is to be covered by wireless mesh network, however, a critical problem follows: how to place the mesh routers to guarantee certain accessibility with financial and technological constraints? Arbitrary deployment of too many routers would be impracticable under the limited budget while too few routers would be unsatisfactory in terms of accessibility and system robustness. Given this, it is crucial to seek for a profitable-driven, cost-conscious, and robustness-guaranteed solution for a wireless mesh network deployment plan.

In a small-scale experiment, a ready approach of mesh network planning is to manually examine and place the mesh routers. This entails experienced personnel and probably interactive visualization tools to come up with an efficient deployment plan. However, this approach becomes a formidable task when it applies to a real city where hundreds or thousands of mesh routers will be deployed. Moreover, reasonable locations for router placement would be on the city-owned properties such as public buildings, lamppoles, traffic signals, and road signs. These candidate locations for mesh router placement could add up to tens of thousands or even more. Thus, it is vital to provide an efficient and optimal/near-optimal solution for wireless mesh network planning, serving as either a tentative reference for the planning personnel to start from or the final solution to the deployment plan.

To provide a city-wide mesh coverage, we envision that the backhaul links will be connected using point-to-point (p2p) directional antennas, i.e. wireless routers may use omni-directional antenna for client access but connect to other mesh routers using directional antennas. Such prediction is because (1) unlike its omni-directional counterpart, directional antenna reduces interference and multi-path fading, provides higher gain, and increases spatial reuse; (2) existing measurement study [9] indicates the behavior of error rate with respect to receive signal strength conforms closely to theory and link abstraction does hold. This makes the on-site measurement and network planning for mesh deployment more reliable. (3) even with directional antenna’s near-field effect, it has been shown [18], [19] that a mesh node equipped with multiple directional antennas can transmit/receive packets at all the interfaces simultaneously in a synchronous manner, further enhancing the backbone throughput and confirming the feasibility of using p2p directional antennas. In fact, several projects [5], [14] have started to evaluate the performance of wireless mesh networks using directional antennas.

For urban wireless mesh network planning, we assume that gateway locations are pre-determined. This is because (1) before the wireless mesh network deployment, some locations may already have Internet connectivity and could serve as gateways; (2) for performance consideration, it is seemingly reasonable to place the gateways in a city roughly evenly so as to avoid forming long hops in the backbone. Furthermore, depending on the density of population and usage pattern different candidate locations could provide different amount of services and utility. Hence, we assume that each candidate location is associated with some ‘profit’ so that we want to maximize the profit of the wireless mesh network deployment. In addition, locations such as hospitals, libraries, or slum areas might be crucial in terms of social welfare, we thus assume some of the locations are required for router placement. Furthermore, these required locations need to have robust connectivity to the gateways, i.e. multiple paths to the gateways, to address unpredictable failures. We also assume that when a location is deployed with a router, a fixed cost consisting of hardware/software equipments and labor installation will
be involved\(^1\). Given a limited budget, we therefore allow a fixed number of locations to be installed with routers. Our goal is to provide an large scale mesh deployment solution so that (1) the profit of wireless mesh network deployment is maximized, (2) the cost of deployment is under the budget, (3) required locations (if any) and any other deployed locations are connected to the gateways, and (4) the mesh network is robust and fault-tolerant.

In this paper, we propose GPSR, an efficient algorithm that delivers a near optimal solution to urban wireless mesh network planning so that the total profit is maximized, the cost is maintained within a certain level, and the formed mesh network is robust and fault-tolerant. GPSR is based on a provably good linear program (LP) that guarantees certain connectivity requirements. It solves the problem by first pre-selecting high profit locations, connecting the pre-selected and required locations through iteratively solving the LP, and then post-selecting the high profit locations if possible. GPSR not only addresses a typical wireless mesh network planning problem but is general enough to be easily extended to other variants of the wireless mesh network planning problems encountered in practice such as asymmetric links, various robustness constraints, heterogeneous costs, and different profit functions. To the best of our knowledge, we are the first to study urban wireless mesh network planning for the case of directional antennas to provide a profit-driven, cost-conscious, and robustness-guaranteed solution.

Our contributions are three folds. First, we define wireless mesh network planning problem (MPP) and show its hardness. Second, we develop an efficient algorithm, GPSR, for solving MPP and provide the performance lower bound of MPP. We further extend GPSR to incorporate various situations that could happen in practice, e.g. asymmetric links, various robustness constraints and cost/profit functions. In addition, we formulate an integer program (IP) for MPP and discuss the techniques of enhancing the IP solver. Third, through extensive evaluation, we show that our proposed algorithm is able to help deploy a robust city-sized wireless mesh network with high profit and under the budget, while satisfying the robustness requirements.

The rest of the paper is organized as follows. We describe our network model and preliminaries in Section II. The hardness for MPP, proposed efficient algorithm GPSR, its performance bound, and IP formulation are then discussed in Section III. We further relax several assumptions of MPP and discuss how GPSR and the IP formulation can be easily modified to handle these scenarios. We evaluate our algorithm in Section V and discuss the related work in Section VI and finally conclude the work in Section VII.

II. Problem Formulation and Preliminaries

In this section, we describe the network model for MPP and formulate the problem with a graph theoretic approach. We further provide necessary preliminaries and backgrounds.

A. Problem Formulation

MPP can be formulated using a graph theoretic approach. Given a city to be deployed with wireless mesh network, the set \( V \) of candidate locations for mesh router deployment represents the vertices in a graph \( G \). A set \( GW \subseteq V \) of locations is predetermined for gateway placement. A set \( R \subseteq V \) of locations is required to be deployed with mesh routers. Moreover, each node \( v \in R \) is required to have \( \alpha_v \) vertex disjoint paths to the gateways to ensure fault-tolerance\(^2\). Each candidate location \( v \in V \) is associated with a profit \( w(v) \), representing the amount of services a location can provide if it is deployed with a router. There is a link \( (u,v) \) if and only if Line-of-Sight (LOS) exists between location \( u \) and \( v \) and their distance is within the communication range of each other. As mentioned in Section II, wireless link abstraction exists and its behavior approaches theory, we expected that communication range can be derived from max transmission power or using the measurement-based approach similar to \( 7 \). The graph \( G \) is thus an undirected graph. Note that we do not consider the inter-dependency between the links due to interference. The reason is because (1) we expect the candidate locations in a city would mostly be along the roads or at intersections with predictable patterns, the resulting graph \( G \) would probably be a grid-like graph. Even if roads are irregular, the angular degree between two crossing roads is expected to be well larger than \( 8^\circ \) which is the beamwidth of a regular parabolic directional antenna used in \( 21 \). (2) Even with the near-field effect, a mesh node equipped with multiple antennas can still transmit/receive at all the interfaces as shown in \( 18, 19 \).

Besides the above assumptions, we are allowed to installed at most \( \kappa \) routers due to the budget constraints. We are asking how to construct a subgraph \( H \) such that (1) the total profit of \( H \) is maximized (2) the number of installed mesh routers is at most \( \kappa \), (3) all deployed locations are connected to the gateways and there are at least \( \alpha_v \) vertex disjoint paths connecting each required location \( v \) to the gateways. We formally define MPP as follows:

**Definition 1 (MPP):** The input consists of an undirected graph \( G = (V,E) \), a set \( GW \subseteq V \) of gateways, a set \( R \subseteq V \) of required nodes, a budget \( \kappa \in \mathbb{N} \), and for each vertex \( v \in V \) an associated profit \( w(v) \geq 0 \), and for each vertex \( u \in R \) an associated integer robustness requirement \( \alpha_v > 0 \). The objective is to find the max-profit subgraph \( H \subseteq G \) so that \( GW \subseteq V(H), R \subseteq V(H), |V(H)| \leq \kappa, \) each \( v \in H \) is connected to some gateway, and each \( u \in R \) has \( \alpha_u \) vertex disjoint paths to gateways.

**Remark 1:** Without loss of generality we can assume that there is a single gateway node by merging all the gateway nodes into one node. The problem remains the same in terms of feasibility and cost.

B. Preliminaries

Clearly, MPP is a fairly complicated problem. Its difficulty lies in the fact that it tries to simultaneously maximize the

\(^1\)We anticipate the dominating cost will probably be hiring the people climbing up the poles for installation.

\(^2\)We discuss the robustness requirements for other nodes in Section IV-B.
vertex profit, minimize vertex cost in a sense, and maintain the connectivity requirements. These dual objectives make it hard to define a the standard notion of approximation ratio. We note that even relaxing the MPP by considering only the the cost minimization aspect results in the following problem which currently is an interesting open problem with no known provably good approximation algorithms (we discuss hardness of related problems in Section III-A).

**Definition 2 (MPP-min):** 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 \( v \).

It turns out that the problem becomes easier if one seeks to minimize edge costs. We therefore consider the following min-edge-cost vertex-connectivity problem (MVP).

**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 \( v \).

Recently, Fleischer et al. [11] proposed a linear programming formulation for the min-edge-cost vertex-connectivity problem (MVP). We include their formulation here for the continuity of the reading. Readers are referred to [11] for further details.

Let \( \delta(S, S') \) be the edges connecting \( S \) and \( S' \). Formally speaking \( \delta(S, S') = \{(u, v) \in E : u \in S, v \in S', S \subseteq V, S' \subseteq V \} \). 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 \( r_{u,v} \in \{0, 1, 2, ..., k\} \). Consider the following linear program in Figure 1:

\[
\begin{align*}
\text{min} & \quad \sum_{e \in E(G)} c(e)x(e) \\
\text{s.t.} & \quad \sum_{e \in \delta(S, S')} x(e) \geq f_k(S, S') - |V - S - S'| \\
& \quad \forall S, S' \subseteq V, S \cap S' = \emptyset \quad (1) \\
& \quad 0 \leq x(e) \leq 1 \quad \forall e \in E(G) \quad (2)
\end{align*}
\]

Fig. 1. (LP-MVP) Linear program for the min-cost vertex-connectivity problem

Since there can be at most one vertex disjoint path going through each of the \(|V - S - S'| \) vertices, there must be at least \( f_k(S, S') - |V - S - S'| \) number of edges to be chosen from \( \delta(S, S') \) to guarantee the connectivity requirement. The condition in Eq. (1) is thus necessary. Furthermore, it turns out that the condition is also sufficient.

**Lemma 2:** The set of integral solutions to the linear program LP-MVP in Figure 1 equals the set of solutions to the min-edge-cost vertex-connectivity problem.

Despite the fact that LP-MVP consists of exponential number of constraints, the optimal solution can be computed in polynomial time as long as there exists a polynomial-time separation oracle [12]. Given a solution to the LP, a separation oracle either verifies the solution is feasible or it generates a constraint of the LP violated by the solution. Furthermore, LP-MVP possesses some nice property as follows.

**Theorem 3:** Any basic solution to LP-MVP in Figure 1 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 Theorem 3 and the fact that there exists a polynomial-time separation oracle of LP-MVP, Fleischer et al. [11] proposed a 2-approximation algorithm. Initially, the solution set of edges \( F \) is empty. It first solves the LP. From Theorem 3, the basic feasible solution must have at least one variable \( x(e) \) with value at least 1/2. The variables with values at least 1/2 are rounded up to 1, and the corresponding edges are added to the solution set \( F \). The process of solving the LP and rounding up the variables is continued until all the requirements \( r_{u,v} \), \( u, v \in V \) are satisfied. Note that once some edges are rounded up to 1, the residual problem is no longer a simple pair-wise connectivity problem; nevertheless the existence of an edge variable with fractional value 1/2 is shown for the residual problem as well which enables the above iterative process.

The cost induced by the solution set \( F \) returned by the algorithm is at most 2 times of the optimal value. The argument is that since an edge \( e \) is added to the solution \( F \) only when its corresponding variables \( x(e) \) is at least 1/2. An edge, after the rounding, contributes its cost to the final solution by at most 2 times of the cost of the LP relaxation. Furthermore, since the optimal solution of the LP relaxation must be the lower bound for the corresponding optimal integral solution, the algorithm is a 2-approximation algorithm.

Although LP-MVP has an exponential number of constraints, it can be solved in polynomial time using the Ellipsoid method since a separation oracle for the linear program exists. However the Ellipsoid method is very inefficient in practice and no solver using this method is readily available. For the connectivity problem one can instead use a compact formulation based on additional flow variables as we will show in Section III-B3.

### III. Mesh Network Deployment

In this section, we discuss the hardness of of MPP and propose our approximation algorithm Greedy Pre-Selection Rounding (GPSR). We further discuss the properties of GPSR and formulate MPP with an integer program.

#### A. Hardness

As we remarked earlier, the MPP problem has dual objectives of (1) maximizing the profit of nodes that are connected and (2) keeping the number of used nodes below \( \kappa \), while ensuring that all nodes in \( R \) have the desired connectivity. Consider the simpler problem where the profit of every node in \( V \setminus R \) is zero - in other words the only question is whether we can connect \( R \) to the gateways using at most \( \kappa \) nodes. This is essentially the MPP-min problem that we defined earlier. Even when \( \alpha_v = 1 \) for each \( v \in R \), this problem is equivalent...
to the node-weighted Steiner tree problem \cite{15} and hence NP-hard. Note that the problem remains NP-hard even when \( c(v) = 1 \) for each \( v \). It is natural to ask whether there exist good approximation algorithms for MPP-min. The following known indicates the hardness of the problem.

**Theorem 4:** Unless \( P = NP \) there is no polynomial time algorithm for MPP-min that achieves an approximation ratio better than \( c \log |V| \) for some absolute constant \( c \) even when \( \alpha_v = 1 \) for each \( v \in R \). If all vertex costs are identical, there is no \( c' \) approximation for some absolute constant \( c' > 1 \).

The first part of the above theorem follows directly from the hardness of approximation for node-weighted Steiner trees shown in \cite{15}. The second part of the above theorem can be deduced from the known hardness result for the vertex cover problem, again based on a reduction similar to the one in \cite{15}. When \( \alpha_v = 1 \) for each \( v \in R \) an \( O(\log |V|) \) approximation is given in \cite{15}.

When the vertex connectivity requirements are large, the problem is considerably harder.

**Theorem 5:** Unless \( NP \subseteq DTIME(n^{polylog(n)}) \) there is no polynomial time algorithm for MPP-min that achieves an \( O(\log^{1−\epsilon} |V|) \) approximation for MPP-min when \( \alpha_v \) are arbitrary.

The above theorem follows from \cite{16} which showed the hardness even in the context of edge costs. For MPP we are interested in small connectivity requirements; that is \( \alpha_v \in \{0, 1, 2, \ldots, k\} \) for some small \( k \). For \( k \geq 3 \) there is no non-trivial approximation known even for edge-costs. As we discussed already for \( k = 2 \) there is a 2-approximation for edge costs \cite{11} but no algorithm is known for vertex costs. For uniform vertex costs one can utilize the edge-cost algorithm as we show later in the paper.

### B. Greedy Pre-Selection Rounding

For the ease of presentation we assume \( \alpha_v \) to be at most 2 for this moment. This will be relaxed in Section \cite{LVBB}. Our proposed GPSR algorithm is based on a provably good LP formulation that solves the min-edge-cost vertex-connectivity problem (MVP). In particular we formulate an LP, called LP-GPSR, that is equivalent to LP-MVP with only polynomial number of constraints and variables. Therefore, we can have the nice property LP-MVP has, i.e. the 2-approximation algorithm as described in Section \cite{II-B} without relying on the heavy Ellipsoid solver and polynomial-time separation oracles.

GPSR consists of four main components. It first distributes the vertex costs to edges. It then pre-selects some high profit nodes that are not required nor gateways, assigns them robustness requirements 1, and adds them to the required set \( R \). Note that the problem can now be transformed to an min-edge-cost vertex-connectivity problem (MVP) by simply minimizing the cost. With the formulation of LP-GPSR, a 2-approximation is readily available, and we solve the transformed MVP so that the edge costs are minimized and vertex connectivity requirements are satisfied. If the returned solution contains more than \( \kappa \) nodes, we go back and pre-select fewer number of high profit nodes and re-solve the transformed MVP again.

GPSR(
\begin{enumerate}
  \item distribute vertex costs to edges
  \item \( P_u \leftarrow \text{pre-select } \kappa - |GW| - |R| \) possible nodes
  \item \( Q_u \leftarrow P_u \cup R \)
  \item \( W_u \leftarrow \text{solveMVP}(Q_u) \)
  \item \( S_u \leftarrow GW \cup W_u \)
  \item \text{if } |S_u| < \kappa \text{ then}
    \item \( T_u \leftarrow \text{post-select } \kappa - |S_u| \) high profit nodes
    \item \( W_u \leftarrow \text{solveMVP}(R) \)
    \item \( S_u \leftarrow GW \cup W_u \)
  \item \text{else}
    \item binarySearch(\( S, l, u \))
    \item return \( S \)
    \item numToTry \leftarrow (l + u)/2
    \item \( P \leftarrow \text{preselect numToTry} \)
    \item \( Q \leftarrow P \cup R \)
    \item \( W \leftarrow \text{solveMVP}(Q) \)
    \item \( S' \leftarrow GW \cup W \)
    \item if \( |S'| > \kappa \) then
      \item binarySearch(\( S, l, numToTry \))
      \item return \( S \)
    \item else
      \item binarySearch(\( S', numToTry, u \))
      \item return \( S \)
\end{enumerate}
\)

**Fig. 2.** Pseudo-code for GPSR

Otherwise, we greedily post-select high profit nodes if the returned solution still has fewer than \( \kappa \) nodes. The final solution is then returned.

Since at most \( \kappa \) routers can be installed, it would not make much sense to pre-select the rest of all \( \kappa - |R| - |GW| \) vertices. Remember we still need some nodes serving as intermediate points connecting the required nodes to the gateway. The question is how many high profit nodes shall we pre-select? We believe using a binary search is a reasonable approach. The reasons follow. First, it is faster than linearly going through all possibilities. Second, due to the way we distribute the costs (as discussed next) and the fact MVP minimizes edge costs, an optimal MVP solution would not go through other vertices and accumulate costs unnecessarily. The pseudo-code of GPSR is shown in Figure 2. In particular, line 2-8 is to test the upper bound of pre-selected nodes. line 9-12 is to test the lower bound of pre-selected nodes. And line 13 recursively search for the best number of pre-selected high profit nodes.

In what follows, we go through the details of each of the four main components of GPSR.

1. \textit{Distribute vertex costs to edges}: Recall that the number of installed routers is at most \( \kappa \). In other words, each \( v \in V \) is associated with a cost 1 and the total cost of the returned graph is at most \( \kappa \). We distribute the cost from the vertices to
the edges as follows: for vertex $v \in V$, distribute half of its cost to each of its incident edges. That is, each edge $e = (u, v)$ takes half from $u$ and half from $v$ as its cost.

2) Pre-select high profit vertices: We greedily pre-select high profit nodes, ties are broken arbitrarily. Furthermore, we need to choose the vertices that are reachable from the gateways because the given graph $G$ may not be connected. This can be done simply through a depth-first-search (DFS) to identify the connected components where the gateways reside. After pre-selecting the set $P$ of high profit nodes, we assign their connectivity requirements so that each $v \in Q$ has at least 1 path to a gateway (this will be relaxed in Section IV-B). We aggregate the nodes in $P$ and $R$ to form a new set $Q$ of required locations, $Q = P \cup R$.

3) Transform MPP to MVP and solve the MVP: After distributing the vertex costs to edges and pre-selecting high profit nodes, an MPP is readily transformed to MVP. Although the given graph $G$ is assumed to be undirected, for the ease of presentation, we first make the graph $G$ a directed one by transforming each undirected edge $e = (u, v)$ to two directed edges $\vec{e} = (u, v)$ and $\vec{e} = (v, u)$ with opposite directions. The corresponding set of directed edges is thus $\vec{E}$, $|\vec{E}| = 2|E|$. In what follows, we distinguish the directionality using a ‘$\rightarrow$’ mark.

Our LP is a compact formulation based on the idea of multi-commodity flows and is standard in the network design area. In particular, each $v \in Q$ originates flows for one commodity. We want to construct a min-cost subgraph of $G$ consisting of the flows and guarantee that the flows originating from the same source are vertex disjoint. We define the variables as follows:

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

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

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

Note that variable $y_{\vec{e}}$ represents whether there is a flow through the undirected edge $e$ in either of the directions. Since each edge $e = (u, v)$ is assigned cost 1 (half from $u$, half from $v$), our objective is thus $\min \sum_{\vec{e} \in \vec{E}} y_{\vec{e}}$. A complete LP formulation is shown in Figure 3.

Since there must be at least $\alpha_v$ vertex disjoint paths from each node in $Q$ to some gateway, the sum of the flows originating from $v \in Q$ must be at least $\alpha_v$, as is shown in Eq. (3). The trick we use to ensure vertex disjoint paths is that for a vertex $u$, there can be at most one incoming flow originating from $v \in Q$, as shown in Eq. (4). It is not hard to see that this condition is sufficient and necessary to guarantee vertex disjoint paths. Furthermore, Eq. (5) is the flow conservation constraint to ensure the amount of flow-in is the same as the amount of flow-out at each vertex, except

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

s.t.

$$\sum_{\vec{e} \in \vec{E}} f_{\vec{e}, v} \geq \alpha_v \quad \forall v \in Q \quad (3)$$

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

$$f_{\vec{e}, v} \leq z_{\vec{e}} \quad \forall \vec{e} \in \vec{E}, \forall v \in Q \quad (5)$$

$$f_{\vec{e}, v} = 0 \quad \forall v \in Q, \forall \vec{e} = (w, v) \in \vec{E} \quad (6)$$

$$z_{\vec{e} = (u, v)} = z_{\vec{e} = (v, u)} = y_{\vec{e}} \quad \forall \vec{e} = (u, v) \in E \quad (7)$$

$$0 \leq f_{\vec{e}, v}, y_{\vec{e}}, z_{\vec{e}} \leq 1 \quad (8)$$

Fig. 3. (LP-GPSR) Linear program for MVP with only polynomial number of constraints and variables.

As the source and destination, Eq. (6) is the capacity constraint so that the amount of each commodity flow will not exceed the capacity. Note that different commodity flows may share the same edge, but the two flows do not add up together. 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. Note that Eq. (8) automatically prevents the flows from forming cycles at the intermediate nodes on the way to the gateway(s). Eq. (9) relates the undirected edge variable $y_{\vec{e}}$ with the capacity $z_{\vec{e}}$ of the directed edges. Although it may not be clear at this moment why we have this constraint, it will become more sensible when we show that LP-GPSR in Figure 3 is equivalent to the LP-MVP in Figure 4.

The following theorems give some useful properties of the formulation. The proofs are standard in the literature and follow from the Ford-Fulkerson maxflow-mincut theorem for single-commodity flows and the properties of the flow polyhedron. We refer the reader to [4], [20], [13] for more details.

**Theorem 6:** When all the $y_{\vec{e}}$ variables in Figure 3 are assigned with integral values, in any basic solution, the flow variables $f_{\vec{e}, v}$ are integral.

**Theorem 7:** LP-GPSR in Figure 3 is equivalent to LP-MVP in Figure 4 in the following sense:

- for every feasible solution $\bar{x}$ to LP-MVP in Figure 4, there is a feasible solution $(\bar{y}, \bar{f}, \bar{z})$ to LP-GPSR in Figure 3 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 4.

From Theorem 7, LP-GPSR possesses the nice properties of LP-MVP and a 2-approximation algorithm is readily available and we call this algorithm **GPSR-HELPER**. The algorithm works as follows. Initially the solution set $S$ of vertices is empty. We solve the LP iteratively. After solving the LP at
each run, the algorithm rounds up to 1 the $y_e=(u,v)$ variables whose values are at least half and adds the end-points $u, v$ to $S$ if they are not added yet. The process is continued until all the $y_e$’s are either 1 or 0. Since LP can be solved in polynomial time and there are polynomial number of variables, the whole process runs in polynomial time.

There is one caveat for rounding up the variables. Although LP-GPSR is equivalent to LP-MVP, a basic feasible solution in LP-GPSR may not always correspond to a basic feasible solution in LP-MVP. In other words, it might be possible that after solving LP-GPSR all the un-rounded variables $y_e$’s are less than half. We fix this as follows. Since the two LP’s are equivalent, there must exist an assignment of the un-rounded variables $y_e$ so that at least one has value at least 1/2 while maintaining the same optimal objective value. We can test each un-rounded $y_e$ by temporarily adding the constraint $y_e \geq 1/2$ and see if the objective value remains unchanged. The variable that maintains the same objective value is rounded up. This idea appears in [3].

4) Post-select high profit vertices: Recall that the number of installed routers is at most $\kappa$. In the case that the returned solution $|S_b|$ shown in Figure 2 is strictly less than $\kappa$ (this happens rarely in our experimental study), we can still increase the profit by grabbing more nodes to the solution. Post-selection is done in a greedy fashion too. We choose the highest profit node $v$ that is adjacent to some node $u \in S_b$, and added it to $S_b$. We continue adding the highest profit node until $S_b = \kappa$ or there is no more nodes adjacent to $S_b$.

C. Properties of GPSR

If there is no feasible solution to MPP, GPSR always fails to find a solution. However, GPSR does not guarantee it will always find a feasible solution if one exists. Nevertheless, it possesses the fail-safe property in the case of failure. Before we dig out the details, we introduce the necessary definition and theorem. Readers are referred to [23] for further details.

Definition 4 (Ear Decomposition): An ear on a graph $G$ is a path whose endpoints are in $G$ and whose internal vertices (if any) are not in $G$. An ear decomposition of $G$ is a decomposition $Q_0 \cup \ldots \cup Q_k$ such that $Q_0$ is a cycle and $Q_i$ for $i \geq 1$ is an ear on $Q_0 \cup \ldots \cup Q_{i-1}$.

(a) A sample ear decomposition (b) A 2-connected graph where $|E| \sim 2|V|

Fig. 4. An example of ear decomposition and 2-connected graph with $|E| \sim 2|V|$

Theorem 8: A graph is 2-connected if and only if it has an ear decomposition. Furthermore, when such a decomposition exists it can start with any cycle.

Figure 4(a) shows an example of ear decomposition. With the above background, we will show the following lemma.

Lemma 9: Let $H$ be the graph returned by GPSR-HELPER when connectivity requirements are at most 2, we have $V(H) - c \leq E(H) < 2V(H)$, where $c$ is the number of trees in $H$.

Proof: The connectivity requirements are either 1 or 2. When the connectivity requirements are all 1’s, the graph $H$ returned by GPSR-HELPER is a forest. Since for each tree $T$, $|E(T)| = |V(T)| - 1$. Enumerating each tree in $H$ gives us $|E(H)| = |V(H)| - c$, where $c$ is the number of trees in $H$. When all the connectivity requirements are 2’s, $H$ is a 2-connected graph. From Theorem 3, there must exist an ear decomposition. We can start from a cycle and gradually count the number of edges and vertices of the ears. The worst case is when the initial cycle has 2 edges and 2 vertices, and each of the ears consists of 2 edges and 1 vertex as shown in Figure 4(b). Thus $|E(H)| \sim 2|V(H)|$. When some connectivity requirements are 1 and some are 2, $H$ is a graph where each connected component is either a tree, an ear decomposition, or a combination of both. Following similar reasoning, we reach the conclusion that $|V(H)| - c \leq |E(H)| < 2|V(H)|$.

Theorem 10: When zero nodes are pre-selected and connectivity requirements are at most 2, GPSR-HELPER returns at most 5 $\kappa$ nodes.

Proof: Let $H^*_MPP$ be the graph that is the optimal solution to MPP. Clearly $|V(H^*_MPP)|$ can be at most $\kappa$. Following similar reasoning in Lemma 9 the number of edges $|E(H^*_MPP)|$ is at most $2|V(H^*_MPP)|$. Now let $H^*_MVP$ be the graph that is the optimal solution to the MVP transformed from the MPP, $|E(H^*_MVP)| \leq |E(H^*_MPP)|$ due to optimality. Furthermore, let $H^*_MVP$ be the graph returned by GPSR-HELPER. Since GPSR-HELPER is a 2-approximation algorithm, $|E(H^*_MVP)| \leq 2|E(H^*_MVP)|$. From Lemma 9, we have $|V(H^*_MVP)| - c \leq |E(H^*_MVP)| \leq 2|E(H^*_MVP)| \leq 2|V(H^*_MPP)| \leq 4\kappa$. Since $c$ is at most $\kappa$, $|V(H^*_MVP)|$ is at most 5$\kappa$.

Theorem 10 provides a desirable property for practical purposes. Many requirements in reality may not be stringent and it can be tolerated by a certain amount. In the case that GPSR fails to find a feasible solution as shown in line 12 of Figure 2, it is assured that the number of nodes found by GPSR will be at most 5$\kappa$. In our performance study, we observe all the returned number of nodes, in the case that GPSR failed to find a feasible solution given there exists one, are within 1.2 times of $\kappa$.

D. Integer Programming Formulation

In this section, we formulate the MPP in an integer program (IP) based on the extension of LP-GPSR. We first define the following variable as follows:

$$x_v = \begin{cases} 1 & \text{if } v \in V(G) \text{ is selected in the solution subgraph} \\ 0 & \text{otherwise} \end{cases}$$

Figure 5 shows our proposed IP that only requires polynomial number of variables and constraints. Eq. (10), Eq. (11), Eq. (12), and Eq. (13) are essentially the same as the constraints of LP-GPSR in Figure 3. The only difference is we
do not pre-select high profit nodes anymore. Since any non-required node may start a flow now, our flow conservation constraints are therefore for all potential sources, as reflected in Eq. \((12)\). Eq. \((13)\) means that if \(\vec{e} = (p, q) \in \bar{E}\) carries the flow originating from \(v\), then \(p\) and \(q\) must be placed with routers. Eq. \((15)\) ensures that if a location \(v\) is placed with a router, then there must be at least one flow originating from \(v\) to some gateway. Eq. \((16)\) ensures that all required locations and gateway locations are placed with routers. Eq. \((17)\) ensures that there can be at most \(\kappa\) locations for router placement. 

\[
\max \sum_{v \in V} w_v x_v \\
\text{s.t.} \sum_{\vec{e}: \vec{e} = (v, u) \in \bar{E}} f_{\vec{e}, v} \geq \alpha_v \quad \forall v \in R \quad (10) \\
\sum_{\vec{e}: \vec{e} = (u, v) \in \bar{E}} f_{\vec{e}, u} \leq 1 \quad \forall u \in V - GW \\
\sum_{\vec{e}: \vec{e} = (v, u) \in \bar{E}} f_{\vec{e}, v} = \sum_{\vec{e}: \vec{e} = (u, v) \in \bar{E}} f_{\vec{e}, u} \quad \forall u \in V - GW, \forall v \in V - GW, \ u \neq v \quad (11) \\
f_{\vec{e}, v} = 0 \quad \forall u \in R, \forall \vec{e} = (u, v) \in \bar{E} \quad (13) \\
f_{\vec{e}, v} \leq x_p, \ f_{\vec{e}, v} \leq x_q \quad \forall \vec{e} = (p, q) \in \bar{E} \quad \forall v \in V - GW \quad (14) \\
\sum_{\vec{e}: \vec{e} = (v, u)} f_{\vec{e}, v} \geq x_v \quad \forall u \in V - GW - R \quad (15) \\
x_v = 1 \quad \forall v \in GW \cup R \quad (16) \\
\sum_{v \in V} x_v \leq \kappa \quad (17) \\
f_{\vec{e}, v}, x_v \text{ binary} \quad (18)
\]

Fig. 5. (IP-MPP) Integer programming formulation for MPP

IV. DISCUSSIONS

Our proposed GPSR algorithm and the IP formulation in Figure 5 is general enough to address many scenarios that could arise in practice. In this section, we discuss how GPSR and IP-MPP can be extended if several assumptions in MPP are relaxed.

A. Asymmetric Links

Although we believe that wireless links will likely be symmetric with the use of point-to-point directional antennas, our IP formulation still holds without any modification even when the links are asymmetric. It is not surprising since IP-MPP in Figure 5 already assumes the graph is a directed one. Our GPSR algorithm also remains the same essentially. The only caveat is that GPSR-HELPER will no longer be a 2-approximation algorithm. In other words, we might need to round up the variables with values less than half.

B. Robustness Requirements

If the nodes in \(R\) have higher connectivity requirements, IP-MPP nicely extends by simply setting the \(\alpha_v\) to be a larger value. Similarly, GPSR-HELPER works by changing the \(\alpha_v\) value in LP-GPSR with the caveat that it is no longer a 2-approximation algorithm. Furthermore, GPSR easily extends to the case when the pre-selected/post-selected nodes are required to have higher connectivity, say \(k\), to the gateways. For pre-selecting nodes, we need to know which ones have \(k\)-vertex disjoint paths from \(w\). It can be done by contracting all the gateway nodes to a single node \(w\). Then we assign cost 1 and capacity 1 to each vertex. By running the standard single source min-cost flow algorithm, it is easy to see which nodes have \(k\)-vertex disjoint paths from \(w\). Those nodes are the candidates for pre-selection. For post-selecting nodes, we assign cost 0 to the nodes in the solution \(S_0\) (line 13 of Figure 2) and 1 to the rest. We also assign capacity 1 to all the nodes in \(G\). Similar to pre-selection, we find the nodes that have \(k\)-vertex disjoint paths to \(w\). We greedily add node \(v\) (and the nodes connecting \(v\) to the solution so that \(w(v)/\text{cost}(v)\) is the largest, where \(\text{cost}(v)\) is the number of nodes not in \(S_0\) needed to connect \(v\) to the gateway.

C. Cost Functions

Although MPP assumes each vertex is assigned with cost 1 with total budget \(\kappa\), GPSR is applicable to the case when costs are assigned to both vertices and edges. In addition to the existing edge costs, we distribute the vertex costs to the edges as before. Since LP-GPSR is minimizing the edge costs anyway, the worst case bound for GPSR follows similarly. We omit the proof for the interest of space.

**Theorem 11:** When costs are homogeneous on vertices and heterogeneous on edges with total budget \(\kappa\) and the connectivity requirements are at most 2, GPSR-HELPER with pre-selected 0 high profit vertices returns a subgraph with total costs at most 5\(\kappa\).

When the costs are assigned to vertices and edges heterogeneously, we still distribute each vertex cost to the edges so that each edge \(e = (u, v)\) has a modified cost \(c'(e) = c(e) + \frac{1}{2}(c(u) + c(v))\) where \(c(e)\) is the original cost of \(e\) and \(c(u)\) and \(c(v)\) are the vertex costs of \(u\) and \(v\) respectively. GPSR returns a subgraph of \(G\) whose cost is at most 2\(\Delta \cdot \kappa\), where \(\Delta\) is the maximum vertex degree in \(G\).

**Theorem 12:** When costs could be heterogeneous on vertices and edges with total budget \(\kappa\) and \(\alpha_v \leq 2\) for all \(v \in R\), GPSR-HELPER with pre-selected 0 high profit vertices returns a subgraph with total costs at most 2\(\Delta \cdot \kappa\). Moreover, if \(\alpha_v = 2\) for all \(v \in R\), the solution returned by GPSR-HELPER has total cost at most \(\Delta \cdot \kappa\).

**Proof:** We will assume that \(\Delta \geq 2\) for otherwise the problem is trivial. Let \(H_{MPP}^*\) be an optimum solution for the given instance of MPP and without loss of generality assume its total cost \(\kappa\). Let the vertex cost portion of \(\kappa\) be \(\rho\) and therefore the edge cost portion is \(\kappa - \rho\). Note that \(H_{MPP}^*\) is also a feasible solution to MVP - the cost of \(H_{MPP}^*\) with the modified edge costs is at most \(\Delta \rho/2 + (\kappa - \rho)\). Since
GPSR-HELPER is a 2-approximation algorithm, the subgraph \( H \) it returns has total edge cost at most \( \Delta \rho + 2(\kappa - \rho) \leq \Delta \kappa \) (since \( \Delta \geq 2 \)). We observe that the modified costs are sufficient to pay for half the cost of \( H \) with the original edge and vertex costs; for any edge \( e = (u, v) \) in \( H \), since \( c'(e) = c(e) + (c(u) + c(v))/2 \), \( c'(e) \) has paid for half the vertex costs of \( u \) and \( v \) and the edge cost \( c(e) \). Therefore the real cost of \( H \) is at most \( 2\Delta \kappa \).

Finally we observe that if \( \alpha_v = 2 \) then every vertex in \( V \) has degree at least 2. In this case the modified costs are sufficient to pay for the original vertex costs and edge costs and hence the cost of \( H \) is at most \( \Delta \kappa \).

Remark 13: If there is a \( \beta \) approximation for MVP (for arbitrary \( \alpha_v \)) then there is a \( \Delta \beta \) approximation for MPP-min (for arbitrary \( \alpha_v \)). Moreover, if \( \alpha_v \geq 2 \) for all \( v \) then the approximation for MPP-min improves to \( (\Delta \beta)/2 \).

Our IP formulation can also be easily modified to incorporate homogeneous/heterogeneous costs on vertices and edges. We only need to replace Eq. (17) and (18) of IP-MPP in Figure 5 with the the following ones:

\[
\sum_{v \in V} c(v)x_v + \sum_{e \in E} c(e)y_e \leq \kappa \\
f_{\vec{e},v}, x_v, z_{\vec{e}} \text{ binary}
\]

and add the following constraints to IP-MPP. Eq. (21) and (22) ensure that an edge \( \vec{e} \) is active iff. there is a flow passing through \( \vec{e} \). Eq. (23) and (24) ensure that edge \( e = (u, v) \) is in the solution graph iff. either \( \vec{e} = (u, v) \) or \( \vec{e} = (v, u) \) is active.

\[
f_{\vec{e},v} \leq z_{\vec{e}} \quad \forall \vec{e} \in \vec{E}, \forall v \in V - GW \quad (21)
\]

\[
\sum_{v : v \in V - GW} f_{\vec{e},v} \geq z_{\vec{e}} \quad \forall \vec{e} \in \vec{E} \quad (22)
\]

\[
z_{\vec{e} = (u, v)} + z_{\vec{e} = (v, u)} \leq 2y_e \quad \forall e \in E \quad (23)
\]

\[
z_{\vec{e} = (u, v)} + z_{\vec{e} = (v, u)} \geq y_e \quad \forall e \in E \quad (24)
\]

D. Profit Functions

Our MPP formulation takes into account that the profit \( w_v \) at each node \( v \) could be different. Even though this reflects many of the practical situations, the profit at location \( v \) is assumed to be only dependent on \( v \) itself. In reality, the profit in a region could be shared among candidate locations. Consequently, when a location is selected for router placement its surrounding locations’ profits will drop. The objective function of IP-MPP is thus no longer a simple linear combination of the \( x_v \) variables. However, this type of objective functions possess the property called submodularity.

Definition 5 (Submodularity): A function \( f : 2^V \to \mathbb{Z}^+ \) 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 \( f(A) - f(B) \leq f(A - B) \) where \( B \subseteq A \subseteq V \); 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 MPP 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 GPSR 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 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 GPSR-HELPER remain the same once \( P \) is chosen. We can also modify the post-selection process 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 [17]. Thus we expect the pre and post selection steps to perform well even for submodular profit functions.

E. Possible Improvements on Solving the IP

Besides relaxing various constraints for practical consideration, IP-MPP in Figure 5 can also be improved for computation efficiency and scalability. For example, we can first run GPSR and use the output as the initial solution for the branch-and-cut algorithm in solving the IP. This two-phase approach spends extra time in finding a good initial solution, trading for the time spent in the IP solver to search for the optimal integral solution. Furthermore, it is not hard to see that the IP contains a lot more number of constraints than variables. When solving the IP formulation with current state-of-the-art branch-and-cut solver, the number of constraints will eventually become intractable. Fortunately, this can be improved by generating the constraints ‘on demand’—a well-known row generation technique in operations research. We leave this as the future work.

V. PERFORMANCE EVALUATION

We evaluate GPSR by comparing the profit it generates with the optimal profit returned by the IP solver. In particular, we have GPSR-B where GPSR-HELPER rounds up the variables in a batch manner, and GPSR-NB where GPSR-HELPER rounds up the variables with values 1. If there does not exist such a variable, it rounds up only one legitimate variable (at least 1/2) at random and re-solve the LP. Although GPSR-HELPER is a 2-approximation algorithm, rounding up the variables in batch may inevitably choose some variable of little contribution. Therefore, a non-batch rounding provides
an approach for avoiding variables with little contribution, on the cost of spending more time in solving the LP. Note that rather than rounding the variables randomly in non-batch mode, there could also be other better choices. Our purpose is to study how effective this random non-batch rounding would be.

We study GPSR’s performance over grid graphs to emulate the typical deployment scenarios where candidate locations are road intersections. We also study GPSR’s performance over unit disk graphs where nodes are randomly generated and there is a link between node $u$ and $v$ iff. their distance is within some threshold. Each unit disk graph is generated so that each node has around 3.5 neighbors on average to ensure that the robustness requirements are easily satisfied. The study over unit disk graphs is to emulate the deployment situation when the roads and intersections are irregular so that the potential deployment locations are less predictable. We expect the real world performance would be between the grid graphs and unit disk graphs. When generating the graphs, we randomly assign node profits between 1 and 10. Each node is also assigned with cost 1. The gateways and required locations are randomly chosen and required locations’ robustness requirements are randomly chosen between 1 and 2. The GPSR-HELPER is implemented in C++ with CPLEX 10.0 LP solver using dual simplex method. All of our evaluations are run on multiple Intel(R) Xeon(TM) CPU 3.20GHz machines each with 6GB memory.

A. The Effect for Varied Number of Potential Locations

We first study how GPSR would perform when the number of potential locations increases. For this study, we gradually increase $|V|$ while maintaining the ratio $\kappa/|V| = 0.6$ and $|GW|/|V| = |R|/|V| = 0.2$. We choose these ratios so that a feasible solution is relatively easy to exist while making the problem challenging to solve. Figure 6(a) and 6(b) shows the median objective value of GPSR normalized by the optimal value for grid graphs and unit disk graphs respectively. Each of the data points represents the median of the successfully returned solutions of 10 randomly generated scenarios. As we can see, when the number of potential locations increases, GPSR consistently finds the subgraphs that are within 90% of the optimum for grid graphs. We also observe that the GPSR-NB performs around 2% better than GPSR-B for the returned objective values. For unit disk graphs, GPSR performs even better (all of the medians are within 98% of the optimum). This is because each node has variant number of neighbors for unit disk graphs, reducing the space of feasible solutions and leading GPSR towards the solution closer to the optimum.

B. The Effect for Varied $\kappa$

We then study GPSR’s performance when the budget $\kappa$ is varied. The total number of nodes $|V|$ is set to 100 and the number of gateways $|GW|$ and required locations $|R|$ are both set to 20. Since most of the randomly chosen scenarios failed to generate feasible solutions when $\kappa$ is small (45 for grid graphs and 40 for unit disk graphs), we start $\kappa$ from 50 for grid graphs and $\kappa = 45$ for unit disk graphs. Figure 6(c) and 6(d) show the performance of GPSR for grid and unit disk graphs respectively. When budget $\kappa$ is barely above the sum of gateways and required locations GPSR performs around 90% of the optimum. As $\kappa$ increases, it approaches optimum coherently. The result is not surprising since when $\kappa$ is small, there is already fewer nodes left for just connecting the required nodes to the gateway. Finding a feasible solution becomes challenging, let alone the optimum solution. Also, the results of GPSR-B and GPSR-NB are more distinguishable when $\kappa$ is small.

C. The Effect for Varied Number of Gateways

We also study how GPSR would behave when the number of gateways varies. We set $|V| = 100$, $\kappa = 80$, $|R| = 20$ and increase $|GW|$ from 20 to 50. In the case where $|GW| = 50$, there will be only 10 nodes left serving as intermediate nodes to connect the required nodes. Figure 7(a) and 7(b) show the success rate of GPSR over the two types of graphs. The gain from GPSR-NB over GPSR-B is also apparent. GPSR over unit disk graphs, on the other hand, performs better for unit disk graphs (almost optimal) than for grid graphs. Moreover, the performance does not seem to significantly vary with the number of gateways for both grid and unit disk graphs. This is because when the number of gateways increases, although it is easier to find a gateway to be connected to some required node, the available intermediate nodes becomes less ($\kappa$ is fixed). As a result, the gain from having more gateways is offset by the loss of intermediate nodes. Also, GPSR-NB consistently performs better than GPSR-B for grid graphs.

D. The Effect for Varied Number of Required Nodes

We further study the performance when the number of required nodes changes. We again set $|V| = 100$, $\kappa = 80$, $|GW| = 20$ and increase $|R|$ from 20 to 50. Figure 7(c) and 7(d) show the GPSR’s normalized performance for both grid and unit disk graphs respectively. As we expected, when the number of required nodes increases GPSR’s performance degrades for grid graphs. This is because both $\kappa$ and $|GW|$ are fixed, the more required nodes the less available intermediate nodes for connection. Nevertheless, both GPSR-B and GPSR-NB maintain their median normalized performance above 95% of the optimum. The gain from GPSR-NB over GPSR-B is also apparent. GPSR over unit disk graphs, on the other hand, performs almost perfect for both batch and non-batch rounding. This surprising result is probably due to inhomogeneous node degrees of the unit disk graphs.

E. Success Rate and Fail-safe Behavior

We finally study how often GPSR succeeds in generating a solution when there exists one. We randomly generate 20 scenarios, set $|V| = 100$, $|GW| = |R| = 20$, and increase $\kappa$ from 50 for grid graphs and 45 for unit disk graphs. Figure 8(a) and 8(b) show the success rate of GPSR over the two types of graphs. We can see that GPSR achieves higher success rates for unit disk graphs than grid graphs. Moreover, GPSR
Fig. 6. Objective value of GPSR normalized by optimal value for (a) grid graphs (b) unit disk graphs, both with varied $|V|/|V| = 0.6$, $|R|/|V| = |GW|/|V| = 0.2$ for (c) grid graphs (d) unit disk graphs, both with varied $\kappa$ ($|V| = 100, |R| = 20, |GW| = 20$).

Fig. 7. Objective value of GPSR normalized by optimal value for (a) grid graphs (b) unit disk graphs, both with varied $|GW|/|V| = 100, \kappa = 80, |R| = 20$ for (c) grid graphs (d) unit disk graphs, both with varied $|R|$ ($|V| = 100, \kappa = 80, |GW| = 20$).

Fig. 8. The rate where GPSR successfully finds a solution for (a) grid graphs (b) unit disk graphs, both with varied $\kappa$ ($|V| = 100, |R| = 20, |GW| = 20$), and the number of nodes returned given that GPSR failed to find a feasible solution for (c) grid graphs (d) unit disk graphs, both at different $\kappa$ values ($|V| = 100, |R| = 20, |GW| = 20$).

Fig. 9. Evaluation of GPSR for grid graphs with $|V| = 100, |R| = 20, |GW| = 20$ for (a) returned objective value normalized by optimal value with varied $\kappa$ (b) success rate of finding a solution with varied $\kappa$ (c) cost in terms of the number of nodes returned given that GPSR failed to find a feasible solution at $\kappa = 60$ and $\kappa = 65$.

quickly reaches 100% success rate although it could start from low. With non-batch rounding GPSR-NB’s lowest success rate
improves to around 57% for grid graphs and 75% for unit disk graphs. This further confirms the effectiveness of random non-batch rounding.

Although GPSR may fail to generate a feasible solution when there is one, we study how ‘bad’ the returned solution is in terms of cost. We randomly generate scenarios and pick the first 10 for which GPSR-B and GPSR-NB would have failed to find a feasible solution. We set $|V| = 100$, $|GW| = |R| = 20$. Note that when $\kappa \leq 45$ for grid graphs and $\kappa \leq 41$ for unit disk graphs, a feasible solution can hardly exist. When $\kappa \geq 60$ for grid and $\kappa \geq 50$ for unit disk graph, GPSR always finds a feasible solution. We therefore evaluate the performance for $\kappa = 50$ and 55 for grid graphs and $\kappa = 42$ and 45 for unit disk graphs. Figure 8(c) and 8(d) show the number of nodes returned by GPSR. When $\kappa = 55$ for grid graphs, GPSR-NB succeeds in producing a feasible solution for 98% of all the generated graphs, we therefore report the result only for GPSR-B. For both of the graphs, We observe that the returned solutions are well below the worst case bound $5\kappa$ although the they exceed budget $\kappa$. This further confirms the desirable fail-safe property of GPSR.

**F. Higher Robustness**

We also evaluate GPSR with higher robustness requirements. For grid graphs, we randomly assign the requirement between 1 and 4 at each internal node, between 1 and 3 at each edge node, and between 1 and 2 at each corner node. For unit disk graphs, since the location of each node is unpredictable, we simply randomly assign the requirement between 1 and 4 to each node no matter how many neighbors it has. Note that we generate the graph so that each node has around 5 neighbors on average.

We first study the grid graphs. Figure 9(a) shows the effect of varied $\kappa$ on the normalized objective values returned by GPSR. Each data point represents the median of the 10 randomly generated scenarios. When the budget $\kappa$ is below 60, there can hardly exist a feasible solution. We therefore only report the results for $\kappa$ between 60 and 100. Even though GPSR-HELPER is not a 2-approximation algorithm when the robustness requirement exceeds 2, it is clearly seen that the normalized objective value does not degrade with the increased robustness requirements. Furthermore, we randomly generate 20 scenarios and study how often GPSR succeeds in generating a solution given that there exists one. Figure 9(b) shows the success rate of GPSR over varied $\kappa$. Comparing Figure 9(b) with Figure 8(a), we see that the success rate of GPSR starts only at 30% when robustness requirement increases up to 4. This is not too surprising because the higher the robustness requirement, the more difficult for GPSR to find a feasible solution. Nevertheless, GPSR quickly approaches 100% success rate as $\kappa$ increases to 70. To further study how ‘bad’ is the returned solution may exceed the cost requirement, we randomly generate 10 scenarios for $\kappa = 60$ and $\kappa = 65$ given that GPSR fails to find a feasible solution. Figure 9(c) shows the returned cost in terms of the number of nodes for 10 randomly generated scenarios. Apparently, even though the returned cost exceeds the corresponding requirement, it is close to the requirement and far way from the theoretical bound $\Delta \cdot \kappa$ as shown in Theorem 12.

Similarly, we evaluate GPSR over unit disk graphs. Figure 10(a) shows the normalized objective values over varied $\kappa$. Following the same trend as before, the objective value approaches the optimal as $\kappa$ increases. Figure 10(b) shows the success rate of finding a feasible solution over 20 randomly generated scenarios. We finally show in Figure 10(c) the cost in terms of the number of nodes returned by GPSR given that it fails to find a feasible solution when there exists one. Over the 10 randomly generated scenarios, we can still see a nicely bounded cost that is far away from its theoretical bound. It is apparent that even with higher robustness requirements, GPSR’s performance shown in the results is comparable to the one with low robustness requirements shown in the earlier sections.

**VI. RELATED WORK**

Several projects utilize omni-directional antennas for mesh backbon connection, for example Roofnet [3]. [6] deploys the wireless mesh network in an unplanned manner while
reliability and shadowing factors, to compare performance of the predefined reliability and throughput over grid topology and random topology to help mesh deployment. Our approach differs in that we use directional antennas for the underline mesh backbone and we do not assume the topology is fixed at front. In stead, we search for the best placement of the wireless routers with the goal of maximum profit, constrained budget, and robust mesh network.

DGP [3], [18], [19], [9] deployed the mesh network to cover a large rural area in India, Magnets [4] measures the link-level performance over the mesh network deployed in Berlin, and Wireless Leiden [22] deployed a low-cost mesh network in Netherlands. All of the above use directional antennas to provide viable networking services, further supporting our assumptions.

Sen et al. [21] proposed a planning solution for the Ashwini project in India. Their goal is to provide rural area network services while ours aims at urban wireless mesh network infrastructure. Since the two purposes differ, the assumptions and objectives are also different. Their goal is to minimize the cost since antennas would be installed on designated costly towers, while ours is to maximize the deployment profit and maintain the cost within the budget. Furthermore, their solution aims to generate a low-cost tree topology with assigned transmission powers while ours incorporate robustness into consideration and leave power/channel assignment to the next planning step. Chandra et al. [8] proposed to minimize the number of gateways while satisfying the client demands using a network flow model. They formulate the problem in the context of community mesh networks where the client houses are fixed, leaving only the placement of gateways to be decided. Ours differ in that we plan the wireless mesh network from scratch, aiming to provide a citi-scaled wireless mesh network solution.

VII. Conclusion

In this paper, we propose GPSR, an efficient and near optimal algorithm for addressing urban wireless mesh network planning problem. GPSR is based on the greedy pre-selection/post-selection of high profit nodes and a 2-approximation algorithm GPSR-HELPER that solves a provably good linear program. It not only maintains high profits and constrained costs for the mesh deployment but also provides robust and fault-tolerant networks. In the case of failure in finding a feasible solution, GPSR is always able to find an alternative with bounded costs. Furthermore, GPSR can easily be extended for different practical situations such as asymmetric links, high robustness requirements, heterogeneous vertex/edge costs, and various profit functions. Through extensive evaluation, we have shown that GPSR consistently approaches optimality under various parameter settings. Following the strand of this planning framework, we will next investigate problems such as channel assignments, power and rate adaptation for the wireless mesh network with directional antennas in the future.

ACKNOWLEDGMENT

We would like to thank Diego Klabjan, Nitish John Korula, Jin Suk Kim, and Fang-Kai Jao for their helps and comments in the fruitful brainstorming discussions.

REFERENCES