SPARSE DEPLOYMENT OF LARGE SCALE WIRELESS NETWORKS FOR MOBILE TARGETS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By
Zizhan Zheng, B.E., M.S.
Graduate Program in Computer Science and Engineering

The Ohio State University

2010

Dissertation Committee:
Prof. Prasun Sinha, Adviser
Prof. Ness Shroff
Prof. Yusu Wang
Deploying wireless networks at large scale is challenging. Despite various effort made in the design of coverage schemes and deployment algorithms with static targets in mind, how to deploy a wireless network to achieve a desired quality of service for mobile targets moving in a large region without incurring prohibitive cost largely remains open. To address this issue, this dissertation proposes Sparse Coverage, a deployment scheme that provides guaranteed service to mobile targets while trading off service quality with cost in a deterministic way.

The first part of this dissertation discusses two sparse coverage models for deploying WiFi access points (APs) along a city-wide road network to provide data service to mobile vehicles. The first model, called Alpha Coverage, ensures that a vehicle moving through a path of length $\alpha$ is guaranteed to have a contact with some AP. This is the first partial coverage model (in contrast to the more expensive full coverage model) that provides a performance guarantee to disconnection-tolerant mobile users. We show that under this general definition, even to verify whether a given deployment provides Alpha Coverage is co-NPC. Thus, we propose two practical metrics as approximations, and design efficient approximation algorithms for each of them. The concept of Alpha Coverage is then extended by taking connectivity into account. To characterize the performance of a roadside WiFi network more accurately, we propose the second sparse coverage model, called Contact Opportunity, which measures the
fraction of distance or time that a mobile user is in contact with some AP. We present an efficient deployment method that maximizes the worst-case contact opportunity under a budget constraint by exploiting submodular optimization techniques. We further extend this notion to the more intuitive metric – average throughput – by taking various uncertainties involved in the system into account.

The second part of this dissertation studies sparse deployment techniques for placing sensor nodes in a large 2-d region for tracking movements. We propose a sparse coverage model called Trap Coverage, which provides a bound on the largest gap that a mobile target, e.g., an intruder or a dynamic event, is missed by any sensor node. In contrast to the current probabilistic partial coverage models, this is the first 2-d coverage model that can trade off the quality of tracking with network lifetime in a deterministic way. For an arbitrarily deployed sensor network, we propose efficient algorithms for determining the level of Trap Coverage even if the sensing regions have non-convex or uncertain boundaries. We then discuss a roadmap assisted geographic routing protocol to support efficient pairwise routing in large sensor networks with holes, which embodies a novel hole approximation technique and makes desired tradeoff between route-stretch and control overhead.
Dedicated to my parents Yongti Zheng and Fengzhen Chen
I would not have been able to finish this dissertation without the guidance, support, and encouragement I received from many great people. I am indebted to them for helping me get this far.

I am especially grateful to my advisor, Prof. Prasun Sinha, from whom I have learned a lot. I appreciate those enjoyable and inspiring weekly discussions in his office in the last nearly five years. His wisdom, acute sense, and passion in conducting frontend research always impressed me. As a great advisor, he encouraged me to think independently while always being supportive. He was there with me in the last moments before several early morning submission deadlines I have caught.

I have also greatly benefited by close interactions with Prof. Santosh Kumar at University of Memphis. We have worked together on several papers. It has always been rewarding to discuss research with Prof. Kumar. His insight and persistence in pursuing big research issues inspired me a lot.

I also owe a lot to my great teachers in both the CSE department and Math department at OSU. Their patient instructions are invaluable in helping me to set up a solid background in both engineering and scientific fields.

I want to thank my colleagues in Prof. Sinha’s research group: Kai-Wei Fan, Sha Liu, Dongwook Lee, Ren-shiou Liu, Gayathri Chandrasekaran, Zhixue Lu, Dong Li,
Daeyoung Choi, Tarun Bansal, and Shengbo Chen. We have shared enjoyable time in discussing research ideas, conducting experiments, and preparing for submissions.

I am also thankful to my friends Jingsong Chai, Dong Du, Boxuan Gu, Ronggang Shi, Wei Sun, Enhua Tan, Pihui Wei, Zhizhang Xie, Zhimin Yang, Xun Yu, and several others at OSU. I appreciate their help in my graduate study and enjoy those wonderful time spent with them.

Finally, I would like to thank my family. A few words cannot adequately express the thanks I own to my parents, Yongti Zheng and Fengzhen Chen. They were always there offering understanding and encouragement every time I felt frustrated and depressed. I would also like to thank my sister, Qihua Zheng and her husband, Xiaojun Tan, and my uncles and their families, for their continuous support in the past years.
VITA

November 8, 1979 .......................... Born - Jiangsu, China

2002 ............................................ B.E. Polymer Engineering,
                                   Sichuan University, China

2005 ............................................ M.S. Computer Science,
                                   Peking University, China

2005-2006 ................................. Graduate Fellow,
                                   The Ohio State University

2006-present .............................. Graduate Research and Teaching Asso-
                                   ciate, The Ohio State University

PUBLICATIONS

Research Publications


**FIELDS OF STUDY**

Major Field: Computer Science and Engineering

Studies in:

- Wireless and Sensor Networks
  - Prof. Prasun Sinha
  - Prof. Anish Arora
  - Prof. Ten H. Lai

- Computational Geometry and Topology
  - Prof. Yusu Wang
  - Prof. Ian Leary

- Statistical Inference and Learning
  - Prof. Douglas E. Critchlow
  - Prof. Mikhail Belkin
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Vita</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td>Chapters:</td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Challenges in Deploying Large Scale Wireless Networks</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Contributions of Dissertation</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Organization of Dissertation</td>
<td>9</td>
</tr>
<tr>
<td>2. Sparse Deployment of Roadside WiFi Networks</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Model, Contributions, Applications, and Related Work</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1 Model</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2 Summary of Contributions</td>
<td>14</td>
</tr>
<tr>
<td>2.1.3 Applications of Our Techniques</td>
<td>16</td>
</tr>
<tr>
<td>2.1.4 Related Work</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Alpha Coverage</td>
<td>19</td>
</tr>
<tr>
<td>2.2.1 Problem Formulation</td>
<td>20</td>
</tr>
<tr>
<td>2.2.2 Coverage Verification</td>
<td>24</td>
</tr>
<tr>
<td>2.2.3 Proof of Hardness</td>
<td>29</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Approximation Algorithms</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Budgeted Version</td>
</tr>
<tr>
<td>2.2.6</td>
<td>Discussion</td>
</tr>
<tr>
<td>2.3</td>
<td>Connected Alpha Coverage</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Problem Formulation</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Two-Stage Approximation</td>
</tr>
<tr>
<td>2.4</td>
<td>Contact Opportunity</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Problem Formulation</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Approximation via Submodular Set Cover</td>
</tr>
<tr>
<td>2.4.3</td>
<td>From Contact Opportunity to Average Throughput</td>
</tr>
<tr>
<td>2.5</td>
<td>Evaluation</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Simulations</td>
</tr>
<tr>
<td>2.5.2</td>
<td>A Small Scale Controlled Experiment</td>
</tr>
<tr>
<td>3.1</td>
<td>Model, Contributions, and Related Work</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Model</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Summary of Contributions</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Related Work</td>
</tr>
<tr>
<td>3.2</td>
<td>Trap Coverage Verification</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Challenges</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Discovering Hole Boundary</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Diameter Computation</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Coping with Sensing Region Uncertainty</td>
</tr>
<tr>
<td>3.3</td>
<td>Hole Bypassing Routing</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Motivation</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Roadmap Aided Geographic Routing</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Hole Approximation and Advertisement</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Worst-Case Stretch in Continuous Domain</td>
</tr>
<tr>
<td>3.3.5</td>
<td>An Optimization of HBR</td>
</tr>
<tr>
<td>3.3.6</td>
<td>Evaluation</td>
</tr>
<tr>
<td>4.1</td>
<td>The Thesis</td>
</tr>
<tr>
<td>4.2</td>
<td>Future Work</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Better Approximation for Alpha Coverage</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Network Design with Uncertainty</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Open Problems for Trap Coverage</td>
</tr>
<tr>
<td>Bibliography</td>
<td>122</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Symbols used in the system model</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>2.1</td>
<td>(a) A graph $G_R$ representing a small road network. Every edge has unit length except $(v_1, v_6)$, which is of length 2. (b) A path $f_{st}$ of length 4 is highlighted, where $s$ is the midpoint of $(v_2, v_3)$ and $t$ is the midpoint of $(v_6, v_7)$. $\text{dist}(s, t) = 3$. ..................................................</td>
</tr>
<tr>
<td>2.2</td>
<td>A road network with four roads (lines) and three candidate locations with coverage regions shown as disks that partition the roads into subsegments such as $va, vb, vc, vd, ce$, etc. ..................................................</td>
</tr>
<tr>
<td>2.3</td>
<td>Two grid graphs with (a) a Hamiltonian circuit highlighted and (b) a Hamiltonian path highlighted. ..................................................</td>
</tr>
<tr>
<td>2.4</td>
<td>A small road network with unit length edges where a single AP located at the midpoint of the middle edge provides $\alpha_P$-coverage but not $\alpha_N$-coverage for $\alpha = 5$. ..................................................</td>
</tr>
<tr>
<td>2.5</td>
<td>A graph with a single path $(v_1, v_2, ..., v_n)$ where the squares represent APs. Each edge has unit length, and $\alpha \in (1, 2)$. (a) A minimum cover that provides $\alpha_N$-coverage. (b) A suboptimal solution that uses only vertices to provide $\alpha_N$-coverage. ..................................................</td>
</tr>
<tr>
<td>2.6</td>
<td>Four road networks used in simulations. ..................................................</td>
</tr>
<tr>
<td>2.7</td>
<td>The percentage of the paths containing $\alpha$-pairs among all the selected paths of length at least $\gamma$. ..................................................</td>
</tr>
<tr>
<td>2.8</td>
<td>The minimum $\alpha$ that is achievable for each budget to provide $\alpha_N$-coverage. ..................................................</td>
</tr>
<tr>
<td>2.9</td>
<td>The percentage of N-equivalent $\alpha_{VC}$-pairs that are not covered by random deployment where $\alpha_{VC}$ corresponds to the values achieved by VC in Figure 2.8. ..................................................</td>
</tr>
</tbody>
</table>
2.10 The minimum $\alpha$ that is achievable for each budget to provide $\alpha_P$-coverage.

2.11 The percentage of P-equivalent $\alpha_{SC}$-pairs that are not covered by random deployment where $\alpha_{SC}$ corresponds to the values achieved by SC in Figure 2.10.

2.12 The cumulative distribution of the interconnection gap under a random waypoint mobility model. The maximum gap for our algorithm is bounded by twice of $\alpha_{SC}$ (see the text for an explanation.) The maximum gap for a random deployment (not shown in the figure) can be as large as 23 km for Rand, 37 km for Grid, and 67 km for DS in the dense network, and 33 km for Rand, 51 km for Grid, and 32 km for DS in the sparse network.

2.13 The standard deviation of the interconnection gap under a random waypoint mobility model. In the dense network, when budget is 50, the standard deviation of the max-min distance sampling is more than 48 (not shown).

2.14 Connected $\alpha_N$-Coverage with various diameters and number of gateways. In (a) and (b), * denotes the number of anchor nodes required to meet the coverage requirement only, and the number of gateways are 1, 5, 10, 20, and 40, respectively. In (c), $\alpha$ is fixed to 2 km.

2.15 Comparing the Two-Stage Connected $\alpha_N$-Coverage (TSCC) algorithm with Connected Distance Sampling. The average results over all the gateway deployments are shown.

2.16 An instance of AP’s coverage region with its boundary highlighted.

2.17 The minimum and average contact opportunity (CO) across all the movements. For (c) and (d), the contact opportunity in time corresponding to the worst-case traffic condition is plotted. For the two baseline algorithms, the results are averaged over all the deployments.

2.18 The complementary cumulative distribution of contact opportunity (for all the shortest paths $\geq 2$km) in distance and time. For (c) and (d), the contact opportunity in time corresponds to the worst-case traffic condition is plotted.
The complementary cumulative distribution of average throughput (over every 400s interval). (a) and (b): a single mobile user in the large network; (c): 5 mobile users in the small network. The results are cumulative across all the users. 75

The average throughput of the 6 paths under evaluation, where Rand represents the average of 5 random deployments. 77

A coverage hole with its hole loop along the outermost boundary highlighted. 80

In this deployment, $d$ is the diameter of the largest hole. Notice that although the diameter line intersects with a covered section, it still represents the largest displacement that a moving object can travel without being detected. 81

$s_2$ and $s_3$ are the sensing neighbors of $s_1$. The four sensing segments $ab$, $bc$, $cd$, and $da$ of $s_1$ are highlighted, where $da$ is a 1-covered sensing segment. 81

An instance of deployment with eight coverage holes, $H_1$ to $H_8$. The rectangle shows the boundary of the target field. Notice that, only the holes within the target field are counted. The small disks are sensing regions. $D_1$ to $D_6$ are coverage components, and $C_1$ to $C_{12}$ are coverage cycles. 88

The partial order among all coverage cycles in Figure 3.4. The cycles at the same level are either all negative(-) or all positive(+). Each box represents a coverage cycle group. $\{C_1, C_7\}$ is type 2, and rest are type 1. 90

The approximation of covered region by a planar graph. The dashed lines show part of $\partial A$. The dashed curves show the real sensing regions (unknown) of node $s_1$ and $s_2$. $e_1$ and $e_2$ are two events detected by both of them. $a$, $c$, and $d$ are points on the edges of $\partial A$ intersecting the two sensing regions, and $b$ is a vertex of $\partial A$. Three faces, $\overline{as_1cb}$, $\overline{s_1e_1s_2e_2s_1}$, and $\overline{s_1e_2s_2dcs_1}$ are shown. 93

Geographic routing can result in sub-optimal routes and high route-stretch. The dashed curve is the routing path of GPSR, and the solid curve is the routing path if the hole can be bypassed before its boundary is touched. The boundary nodes of the hole are highlighted. 97
3.8 A shortest-path roadmap and a path from s to t computed by HBR in the continuous domain. The three polygons are routing holes and the dashed polygons are their cores. The dotted segments are the bitangent edges of the map. The path computed using cores is (s, v₀, v₂, v₄, t). The real path is highlighted. x and y are the intersections of line v₂v₄ with the boundary of the hole in the middle. .......................... 100

3.9 The approximation of convex holes. .......................... 104

3.10 The approximation of a routing hole with one convex chain and one concave chain. α = π/2 for the convex chain. The hole is simplified to the polygon with vertices v₁, v₂, v₅, v₈, v₁₀, w₁₀.1. .......................... 106

3.11 A line segment in f vs. the corresponding segment in f. The convex chain (v₀, v₁, v₂, v₃, v₄) is approximated by the line segment v₀v₄. (a) v₀v₄ vs. (v₀, v₁, v₂, v₃, v₄). (b) xv₄ vs. (x, v₂, v₃, v₄). (c) xy vs. (x, v₁, v₂, v₃, y). .......................... 109

3.12 Convex chain (v₀, v₁,..., v₄) is approximated by v₀v₄. A packet is forwarded by v towards v₄. (a) The optimization of HBR by early bypassing, (v, x, v₁, v₂, v₃, v₄) and (v, v₂, v₃, v₄) are pathes found by HBR without and with optimization, respectively. (b) The optimization cannot be applied if the extreme node is not visible. .......................... 110

3.13 EVRs of a convex chain of hole H. For instance, regions I, II, and III are EVRs of (v₂, v₃), (v₂, v₄), and (v₃, v₄), respectively. .......................... 111

3.14 Two typical simulation scenarios .......................... 115

3.15 Average route-stretches .......................... 117

3.16 Normalized number of packet transmissions in routing procedure .......................... 117
CHAPTER 1

INTRODUCTION

1.1 Challenges in Deploying Large Scale Wireless Networks

Large scale wireless networks are emerging to support various applications involving mobile targets, such as tracking the movements of people or animals, or of phenomena such as fire, via wireless sensor networks deployed in large 2-d regions, or providing multimedia content and ubiquitous services to mobile users through WiFi hotspots distributed in city-wide areas. A key challenge to the success of such large scale deployments is to support desired network functionalities without incurring prohibitive cost for both deployment and management. In this section, we discuss the challenges involved in large scale deployment of wireless networks for providing data service to mobile vehicles and for tracking 2-d movements.

City-Wide Vehicular Internet Access: With increasing popularity of media enabled hand-helds, the need for high data-rate services for mobile users is evident. WiFi hotspots are rapidly mushrooming in every city to meet this demand. But, their primary target is static users. These networks fail to provide any assured level of service to a mobile user. Although large deployments of WLANs can be used to
provide high data-rate services over large areas, the cost becomes prohibitive due to the sheer number of access-points (APs) required. For instance, as of 2009, to cover a 12 square miles area in Mountain View, Google needed to deploy more than 500 access points [35] to barely provide coverage at the base data rate. In addition to the deployment cost, the maintenance and management complexity has led to abandonment or scaling back of several WLAN projects from San Francisco to Philadelphia [3].

New Wireless Wide-Area Networking (WWAN) technologies such as 3GPP LTE (Long Term Evolution) and mobile WiMAX are expected to provide either long range coverage or high data rates, but practical numbers are far from the promised levels. For example WiMAX is intended to support data rates as high as 75 Mbps per 20 MHz channel, or a range of 30 miles [85]. However, one of the first deployments of WiMAX in US is reported to provide a downlink bandwidth of 3 Mbps [3], which is only within a factor of 2 better than the current 3G networks. Note that these resources will potentially be shared by a large number of active users within the respective sector of the antenna. Given the resistance from majority of users to pay high monthly fees for mobile data access, which is essential for supporting expensive new deployments, ubiquitous service from such new deployments could take several years, and possibly decades.

On the other hand, evaluation of wireless data access by mobile users using “in situ” (or “open”) WiFi networks [7, 15, 24, 65], and in various controlled environments [7, 32, 65, 69] have confirmed the feasibility of WiFi based vehicular Internet access for non-interactive applications. The possibility and challenges to support certain interactive applications, such as Web browsing, have also been studied [7,8]. Most
existing work, however, consider an unplanned deployment of APs based on open-APs \([7,15,24,32,65,69]\). Consequently, these solutions fail to provide any throughput assurance to a mobile user; they can only provide opportunistic services to mobile users.

**Tracking Movements in a Large 2-d Region:** It is now widely understood that to maximize the utility of a large scale resource constrained wireless sensor network in surveillance or monitoring applications, requiring every point in the region to be covered (i.e., full coverage or blanket coverage [54]) at all times is cost prohibitive, as has been the case in prototype deployments [41, 52, 75]. Instead, a partial coverage model that can make a tradeoff between tracking quality and network lifetime is preferred. It is first observed in [37] that for surveillance applications, a sparse subset of sensors may be sufficient to detect moving objects within a delay bound. Only when an object is detected, all or most of the sensors need to become active to track the object. Similarly, for monitoring large scale events, a small subset of sensors may be enough to catch large events [29]. Once an event is detected, additional sensors may be activated to obtain an accurate estimation of the boundary of the event.

Based on this observation, Gui et al. envisioned trading the quality of surveillance with network lifetime [37]. In particular, they defined a metric called Quality of Surveillance (QoSv) as the reciprocal of the average distance that a moving target, starting at a random location and moving in a random direction can travel in a straight line before it is detected by a sensor. Such a metric is then used to trade off tracking quality with the network lifetime by controlling the density and distribution of active nodes. Various probabilistic and partial area coverage models have been
studied since [12,19,29,60,72,77,83,87,91], most of which use a probabilistic metric like QoSv or detection probability. The main weakness of these approaches is that they rely on expected performance and do not provide any deterministic guarantee on the worst case tracking quality.

In summary, the following two challenges are fundamental in enabling mobile targets oriented large scale deployment of wireless networks:

• First, how to provide a deterministic performance guarantee without incurring prohibitive cost?

• Second, how to tradeoff the performance and cost in a fine-grained and deterministic way?

1.2 Contributions of Dissertation

To address the above challenges, this dissertation proposes the notion of Sparse Coverage and develops three sparse coverage models that enable large scale roadside as well as 2-d deployment. The core idea of Sparse Coverage is to trade off service quality and cost by exploiting the mobility of mobile targets. The key observation is that many applications can tolerate certain amount of delay or disconnection, and due to the fact that targets are moving, even if the target region is only partially covered, we may still be able to provide a guarantee on the long term or aggregated service that a mobile target can obtain. Based on this observation, this dissertation develops the foundation of Sparse Coverage by carrying out the following tasks:

Sparse Coverage for Roadside WiFi: In the scenario of vehicular Internet access, many applications are non-interactive and disconnection tolerant, as in the case
of road surface monitoring [25] and cargo tracking. This fact has been used to motivate the opportunistic service through “in-situ” or open WiFi. However, as we stated above, such a service cannot provide any guarantee on performance. In particular, the gap between two consecutive contacts with APs could be arbitrarily large in an unplanned deployment. Being able to predict the next contact with some AP could be important for the scheduling of resources in mobile applications. Furthermore, an upper bound on the worst-case interconnection gap leads to a lower bound on the amount of data that can be accessed within certain amount of time assuming the quality of each contact and travel speed can be estimated. The first sparse coverage model we proposed provides such a guarantee on interconnection gap.

- **Alpha Coverage**: We model a road network as a graph and a deployed access point as a point on the graph (a short user-AP communication range is assumed here). Informally, a deployment of APs provides $\alpha$-coverage [95, 96] to a road network, if any path of length at least $\alpha$ along the road network meets with at least one AP. For a given deployment (which could be empty) and a given budget on the number of additional APs that can be used, we are looking for an optimal deployment that provides $\alpha$-coverage for a minimum possible $\alpha$ under the budget constraint. Unfortunately, for this general definition, even the verification of $\alpha$-coverage is co-NPC (there is a reduction from the Hamiltonian Path problem to the complement of this decision problem). The essential difficulty is due to the fact that the two ends of a long path could actually be very close to each other. We therefore propose two practical metrics as approximations and propose efficient approximation algorithms for optimizing each of these metrics.
– *Alpha Network Coverage*, where instead of requiring all the paths of length at least $\alpha$ to be covered, that is, to touch at least one AP, only those paths that connect two locations that are at least $\alpha$ distance away (in terms of graph distance) are required to be covered. This model provides a realistic characterization of coverage gaps in a partially covered road network.

– *Alpha Path Coverage*, where only a given set of paths, e.g., a set of shortest paths or most frequently traveled path of length at least $\alpha$, are required to be covered. This definition is less powerful than the previous one, but allows more efficient approximation.

- **Connected Alpha Coverage**: The notion of Alpha Coverage has been extended by taking connectivity into account and applied in an asset tracking system [64], where besides coverage requirement, each AP is required to be connected to at least one of the given gateway locations with Internet backhaul through long range AP-AP links. We propose a two-stage solution to the problem of finding a deployment using minimum number of APs to provide the desired joint coverage and connectivity, and derive a joint approximation factor.

- **Contact Opportunity**: While Alpha Coverage provides a bound on worst-case interconnect gap, it only considers the number of contacts but ignores the quality of each contact. We hence propose a more expressive sparse coverage model, called Contact Opportunity [94]. Informally, the contact opportunity for a given deployment measures the fraction of distance or time that a mobile user is in contact with some AP when moving through a certain path. Such a metric is closely related to the quality of data service that a mobile user might
experience while driving through the system. However, it is also significantly more challenging to optimize such a metric because the performance of each AP including its coverage region and capacity, as well as the various uncertainties involved in the system, such as unpredictable traffic conditions, unknown moving patterns of mobile users, and the dynamics involved in the performance of APs. To address these issues, we first extend the AP coverage model from a single point to an arbitrary geometric shape. We then use an incremental approach as well as a worst-case perspective to deal with the uncertainties. We start with Contact Opportunity in distance by taking the coverage region of each AP into account, and present an efficient deployment method that maximizes the worst-case Contact Opportunity in distance under a budget constraint, which involves least amount of uncertainties. We then show how to extend this notion and the deployment techniques to Contact Opportunity in time by using an interval based road traffic modeling, and further to the more intuitive metric – average throughput along a path – by taking various dynamic elements into account.

**Sparse Coverage for 2-d Sensor Deployment:** As stated above, for tracking movements in a large 2-d region, certain amount of detection delay is acceptable. The main weakness of the current probabilistic partial coverage models is that such a delay could be arbitrarily large, which is unsatisfactory. We propose Trap Coverage, a sparse coverage model that provides a bound on the worst-case detection gap.
• **Trap Coverage**: A sensor network providing Trap Coverage [9] guarantees that any moving object or phenomena can move only a (known) bounded displacement before it is guaranteed to be detected by the network for any trajectory and speed. For a deployment that provides trap coverage with a diameter of $d$, the sensor network guarantees that every moving object or phenomena of interest will surely be detected for every displacement $d$ that it travels in the target region. At any instant, we can either pinpoint the location of a moving object precisely, or can point to a coverage hole of diameter at most $d$ in which it is trapped. If the value of $d$ is set to 0, then trap coverage is equivalent to full coverage. By relaxing the requirement of having every point covered, trap coverage generalizes the model of full coverage. Once sensors have been deployed on the ground (perhaps randomly), it may be necessary to determine the level of trap coverage that they provide since some may fail at or after the deployment for unforeseen reasons. We therefore propose polynomial time algorithms to determine the level of trap coverage that an arbitrary deployed sensor network provides. Our algorithm not only works for non-convex models of sensing regions, but it also works when sensing regions are uncertain. Further, it takes into consideration the complications that may arise due to the boundary of the deployment region.

• **Hole Approximation**: Large holes in terms of either communication or sensing gaps may exist in a network due to the presence of large obstacles such as buildings or lakes, or a deployment strategy that allows holes as in the case of Trap Coverage. Knowing the positions and the shapes of these holes could be important for both networking and application layer protocols. However,
advertising the accurate shape of each hole to the entire network incurs high overhead. We hence propose a core-set based approach for summarizing the critical information of a 2-d hole of roughly convex shape, where the size of the core-set is controlled by a single parameter and is independent of the size of the original hole [93]. We further extend this approach to general holes.

- **Hole Bypassing Routing**: One application of our hole approximation technique is a hole bypassing geographic routing protocol. In the presence of large communication holes, many existing geographic routing protocols either fail at the “local minima” or place a heavy requirement on the control plane, as in the case of network-wide flooding, and usually do not provide a bound on route-stretch. Both the high control overhead and the unbounded route-stretch lead to energy inefficiency. We propose a distributed shortest-path roadmap based routing paradigm [93] to address this issue. In the preprocessing stage, the core-sets of communication holes are first advertised within limited neighborhoods and a shortest path roadmap is setup at every node. A hole avoiding shortest path between any two nodes can then be efficiently approximated within a bound by using the core-sets instead of the real hole boundaries. We show that our approach makes a desired tradeoff between the worst-case route-stretch and control overhead through both analysis and simulations.

### 1.3 Organization of Dissertation

The rest of this dissertation is organized as follows. Chapter 2 discusses the two sparse coverage models and their extensions, and corresponding approximation algorithms for roadside WiFi deployment. Chapter 3 discusses algorithms for verifying
Trap Coverage, the hole approximation algorithms, and the hole bypassing routing protocol. Chapter 4 summarizes the results of the thesis, and discusses possible future work.
2.1 Model, Contributions, Applications, and Related Work

2.1.1 Model

We model a road network $\mathcal{R}$ as a connected undirected geometric graph $G_{\mathcal{R}}$, where vertices represent the points where the road centerline segments and the road intersections meet, and edges represent the road centerline segments connecting the road intersections. For a curved road segment, we introduce artificial road intersections, so that each edge represents a straight line segment. Let $V(G_{\mathcal{R}})$ and $E(G_{\mathcal{R}})$ denote the vertex set and the edge set, respectively. Each edge $e$ has a length, denoted as $|e|$, which is the length of the corresponding road centerline segment. This model has been used by some publicly available road network databases, such as [1]. Although we are assuming an undirected graph model, most of our results can be extended to directed graphs as well.

For a point $p$ on an edge $(u, v) \in E(G_{\mathcal{R}})$, if $p$ is not a vertex, we can make it a vertex by adding $p$ to $V(G_{\mathcal{R}})$ and by subdividing edge $(u, v)$ into two edges $(p, u)$ and $(p, v)$ with $|(p, u)|$ and $|(p, v)|$ equal to the length of the corresponding road centerline segments starting at $p$. The resulting graph is denoted as $G_{\mathcal{R}} \lor \{p\}$. For instance, by
Figure 2.1: (a) A graph $G_R$ representing a small road network. Every edge has unit length except $(v_1, v_6)$, which is of length 2. (b) A path $f_{st}$ of length 4 is highlighted, where $s$ is the midpoint of $(v_2, v_3)$ and $t$ is the midpoint of $(v_6, v_7)$. $\text{dist}(s, t) = 3$.

inserting the midpoint of edge $(v_1, v_6)$ to the graph in Figure 2.1(a), we get a graph where all the edges have the same length as shown in Figure 2.1(b). $G_R \lor \{p\} = G_R$ if $p$ is a vertex in $G_R$.

We assume that $A$ is a set of known candidate locations in the 2D region covering the road network where access points (APs) can be deployed. Associated with each candidate location $a \in A$, there is a fixed cost $w_a \in \mathbb{R}^+$ for installing an AP at $a$, and a coverage region $C_a$, which is a connected region in the 2D space consisting of the set of points where the received SNR from an AP deployed at $a$ is higher than a fixed threshold. The coverage regions $C_a, \forall a \in A$ partition the road network graph into smaller segments called subsegments. Figure 2.2 shows a road network with four roads (lines) and three candidate locations with coverage regions shown as disks that partition the roads into subsegments such as $va, vb, vc, vd, ce$, etc. Although the coverage regions are plotted as disks in Figure 2.2, our problem definitions and solutions are independent of the shape of the coverage region. Also note that such
a partition is with respect to a particular deployment and each subsegment is either fully covered or not covered at all.

**Remark.** For Alpha Coverage model and its extensions, we assume \( C_a \) is a point for reasons explained in Section 2.2, which can be viewed as a special case of the general case discussed here. The general AP coverage model will be fully utilized in our Contact Opportunity model.

![Figure 2.2: A road network with four roads (lines) and three candidate locations with coverage regions shown as disks that partition the roads into subsegments such as \( va, vb, vc, vd, ce \), etc.](image)

Let \( \mathcal{L} \) denote the set of all the subsegments in a road network graph. For each \( l \in \mathcal{L} \), let \( d_l \in \mathbb{R}^+ \) denote the length of the corresponding road centerline segment. For any deployment \( S \subseteq A \), let \( \mathcal{L}_S \subseteq \mathcal{L} \) denote the set of subsegments covered by \( S \), that is, \( \mathcal{L}_S = \{l \in \mathcal{L} : l \subseteq \bigcup_{a \in S} C_a \} \).

The trajectory of a moving vehicle is modeled as a set of consecutive *general* paths in the road network graph defined as follows.
Definition 2.1.1. A general path in a graph: A general path $f_{ab}$ in a graph $G_{\mathcal{R}}$ between $a$ and $b$, both of which are points on some edges of $G_{\mathcal{R}}$ ($a = b$ is allowed), is a simple path (or a simple circuit) in $G_{\mathcal{R}} \cup \{a, b\}$. The length of $f_{ab}$, denoted as $|f_{ab}|$, equals to the sum of the lengths of the edges composing the path in $G_{\mathcal{R}} \cup \{a, b\}$.

A general path will be simply called a path when there is no ambiguity. For instance, Figure 2.1(b) highlights a path of length 4.

We assume that there is a set of movements, denoted as $\mathbb{P}$, given as part of the input to the deployment decision maker. The concrete definition of $\mathbb{P}$ is independent of our problem definitions and solutions, while the size of the set $\mathbb{P}$ impacts the computational complexity and performance guarantee of our solutions as discussed below. For instance, $\mathbb{P}$ could be a set of shortest (or fastest) paths or a set of most frequently traveled paths connecting any two road intersections. Such information can be learned from a road network database [1] and historical traffic data [34]. For each $p \in \mathbb{P}$, let $\mathcal{L}_p \subseteq \mathcal{L}$ denote the set of subsegments that constitute $p$. The symbols used in the system model are summarized in Table 2.1.

2.1.2 Summary of Contributions

In this chapter, we make the following contributions on the issue of deploying roadside WiFi networks:

- We present the first sparse coverage model for mobile users with intermittent connectivity called Alpha Coverage [95, 96], which places an upper bound on the interconnection gap in a roadside WiFi network. We define two practical coverage metrics that can be used in various scenarios.
Table 2.1: Symbols used in the system model

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_\mathcal{R}$</td>
<td>The graph model of a road network $\mathcal{R}$</td>
</tr>
<tr>
<td>$A$</td>
<td>The set of all candidate locations for deploying APs</td>
</tr>
<tr>
<td>$w_a$</td>
<td>The cost of installing an AP at location $a \in A$</td>
</tr>
<tr>
<td>$C_a$</td>
<td>The coverage region of an AP deployed at $a \in A$</td>
</tr>
<tr>
<td>$\mathcal{L}$</td>
<td>The set of subsegments</td>
</tr>
<tr>
<td>$d_l$</td>
<td>The length of a subsegment $l \in \mathcal{L}$</td>
</tr>
<tr>
<td>$\mathcal{L}_S$</td>
<td>The set of subsegments covered by $S \subseteq A$</td>
</tr>
<tr>
<td>$A_l$</td>
<td>The set of candidate locations that cover $l \in \mathcal{L}$</td>
</tr>
<tr>
<td>$\mathcal{P}$</td>
<td>The set of movements</td>
</tr>
<tr>
<td>$\mathcal{L}_p$</td>
<td>The set of subsegments constituting $p \in \mathcal{P}$</td>
</tr>
<tr>
<td>$A_p$</td>
<td>The set of candidate locations that cover $p \in \mathcal{P}$</td>
</tr>
</tbody>
</table>

• We present efficient algorithms for verifying Alpha Coverage, and factor $O(\log n)$ approximation algorithms for finding a close to optimal deployment that uses minimum number of APs to achieve Alpha Coverage. We also consider a budgeted version of the problem and apply a binary search technique to achieve a bicriteria approximation.

• We extend the notion of Alpha Coverage to Connected Alpha Coverage [64], where besides coverage, the set of APs are required to be connected to a set of gateway locations with Internet backhaul. We present a two-stage approximation algorithm to jointly optimize coverage and connectivity.

• We then present the second sparse coverage metric, called Contact Opportunity [94], as a more expressive characterization of roadside WiFi deployment,
which is closely related to the quality of data service that a mobile user might experience when driving through the network.

- We present an efficient deployment method that maximizes the worst-case Contact Opportunity given a budget constraint by utilizing submodular optimization techniques.

- We extend the notion of Contact Opportunity and the corresponding deployment techniques to average throughput by modeling various dynamic parameters.

- We evaluate the performance of our solutions by doing simulations over real road network data [1] and show that our proposed algorithms perform significantly better than several commonly used baseline algorithms. A small scale controlled experiment further demonstrates the advantage of our algorithm in improving average throughput.

2.1.3 Applications of Our Techniques

A number of applications can benefit from a roadside WiFi network deployed using our techniques. Remote monitoring and tracking of shipments is one such application. For example, Walmart currently depends on a satellite based system [2] for tracking its trailers, which is an expensive solution. Similarly, businesses with mobile workforce can benefit from media-rich communication over such a system. Recently, the feasibility and usefulness of a system that provides road condition updates has been studied [46], which is another use case of the system.
The framework and solutions presented in this chapter are immediately usable by various service provider companies for enabling WLAN based services for mobile users. Using our solution for planning incremental deployment, service providers can install few APs for large values of interconnection gap or small values of contact opportunity to begin with, and over time add new APs to gradually improve the service quality. In addition to providing a low cost solution for supporting various current applications mentioned earlier, making worst-case service guarantee may enable new applications based on intermittent connectivity.

2.1.4 Related Work

Roadside WiFi: The idea of Drive-thru Internet by connecting to existing roadside Access Points is introduced in [69], which shows that a single moving vehicle connected via 802.11b with an AP located at roadside of an empty street can access several megabytes of TCP or UDP traffic, even when the velocity is as high as 180 km/h. Subsequently, evaluations in various controlled environments [7,32,65,69] and in situ WiFi networks [7,15,24,65] have been conducted, further confirming the feasibility of WiFi-based Vehicular Internet Access for non-interactive applications. The possibility and challenges to support certain interactive applications, such as Web browsing, have also been studied [7,8].

Vehicular communication through WiFi infrastructure is characterized by short-lived and intermittent connections, which challenges both the 802.11 MAC and the transport protocols. It was shown in [69] that the channel quality when a vehicle is moving through an AP can be roughly divided into three phases: the entry phase when connectivity is weak and loss and delay are both high, the production phase
when effective communication can take place, and the exit phase when loss and delay are high again. The initial setting of parameters and the default behavior of 802.11 and TCP can lead to performance loss in all the three phases as shown in [40]. In [15], the time spent in each stage of connection setup was measured, and a simple IP caching approach was proposed to reduce the delay induced by DHCP. It was reported in [24] that the mean connection setup time can be reduced from over 10s to 400ms by using a streamlined client-side connection setup process, which also increases the number of usable short connections significantly. Transport protocols that hide the wireless losses from the wired side, and the temporary unavailability of connections from the client were proposed in [24] and [70]. Interesting ideas have been proposed regarding the scenarios where a moving vehicle is in the transmission ranges of multiple APs [7,65], or multiple vehicles are associated with a single AP [39]. In particular, [65] shows that the use of directional antennas at vehicle side and beam steering techniques can improve the performance of the 802.11 link via carefully designed handoff algorithms. In [7], a lightweight coordination protocol was designed that allows a vehicle to communicate with multiple APs simultaneously to reduce disruption in connectivity. It was noted in [39] that multiple vehicles associated with the same AP may choose different transmission rates and therefore suffer from the 802.11 performance anomaly, that is, the data rates of all these vehicles will eventually be slowed down to the lowest one. A medium access protocol that grants the channel to the vehicle with best SNR was suggested in [39].

It should be noted that the deployment issues with respect to WiFi-based Vehicular Internet Access have not been carefully studied so far. Instead, an unplanned deployment of APs is commonly assumed in most previous works. A simple non-uniform
strategy that places more stationary nodes in the network core was considered in a recent work [10]. However, it was completely based on intuition without providing any performance guarantees.

**Submodular Optimization:** We have applied submodular optimization in both Alpha Coverage model (implicitly) and Contact Opportunity model (explicitly). A set function is called submodular if it satisfies a decreasing marginal property (a formal definition is given in Section 2.4.2), which can be viewed as the discrete counterpart of continuous concave functions. Submodular functions play a critical role in combinatorial optimization. The theory was first developed half a century ago. Since then, various submodular optimization problems have been intensively studied [28,66]. Although the submodular minimization problem is polynomial time solvable, the submodular maximization problem is NP-hard. However, a simple greedy algorithm gives an \((1 - 1/e)\) approximation ratio when the function is also nondecreasing and normalized. The submodular set covering problem was first studied in [86]. Recently, submodular optimization has been applied to several network deployment problems, including placing sensors to efficiently detect outbreak [59], or to provide robust observations [50]. The former is formulated as a submodular maximization problem while the latter is formulated as a budgeted submodular set covering problem.

### 2.2 Alpha Coverage

Our first sparse coverage model concerns interconnection gap, an important parameter for characterizing the performance of intermittent data service. Informally, a deployment of APs provides \(\alpha\)-coverage to a road network, if any simple path of length \(\alpha\) on the road network meets with at least one AP. If the expected service from
an AP is known, then the cumulative service received by a mobile user over a path of length \( \alpha \) can be computed. For a given road network and a parameter \( \alpha \), we ask the following two questions: 1) does a given deployment provide \( \alpha \)-coverage? (Section 2.2.2); 2) if not, how to deploy a minimum number of additional APs to ensure \( \alpha \)-coverage? (Section 2.2.4). A budgeted version of the problem is also considered (Section 2.2.5).

In this section, we model APs as points on \( G_{\mathcal{R}} \) by the following consideration. First, we are seeking a solution that provides the worst-case guarantee to the interconnection gap, and therefore make the most conservative assumption about the contribution of each AP. Second, we are considering a sparse deployment, and therefore, we ignore the case where the coverage regions of multiple APs overlap with each other. Hence each AP is represented by a point in \( G_{\mathcal{R}} \) closest to it.

### 2.2.1 Problem Formulation

We are now ready to formally define \( \alpha \)-coverage. We say that a path is covered by a point in \( G_{\mathcal{R}} \) if the point is on that path.

**Definition 2.2.1.** \( \alpha \)-coverage: A deployment of APs provides \( \alpha \)-coverage to \( \mathcal{R} \), if every path \( f_{ab} \) in \( G_{\mathcal{R}} \) with \( |f_{ab}| \geq \alpha \) is covered by at least one AP.

For instance, Figure 2.1(b) shows a deployment that provides \( \alpha \)-coverage for \( \alpha = 2 \) by placing APs at vertices \( v_2, v_5, v_7 \), and point \( v_9 \), the midpoint of \( (v_1, v_6) \). Although \( \alpha \)-coverage closely models our intuition, it is hard to even determine in polynomial time whether a deployment provides \( \alpha \)-coverage since such a problem is co-NP-complete (see Section 2.2.2 for a proof), the class of problems most likely not to
be in P. Since even verifying whether a graph is $\alpha$-covered is hard, we propose two new metrics to approximate $\alpha$-coverage. First, we define the following terms:

**Definition 2.2.2. Distance on a graph:** For any two points $a$ and $b$ in graph $G_{R}$, the distance between them, denoted as $\text{dist}(a, b)$, is the length of a shortest path between $a$ and $b$ in $G_{R} \lor \{a, b\}$.

Let $F_{ab}$ denote the set of all the possible paths between points $a$ and $b$.

**Definition 2.2.3. $\alpha_N$-coverage:** A deployment of APs provides **Network Coverage** of distance $\alpha$ ($\alpha_N$-coverage for short) to $R$, if for every pair of points $a$ and $b$ in $G_{R}$ with $\text{dist}(a, b) \geq \alpha$, every path $f \in F_{ab}$ is covered by at least one AP. A pair of points $(a, b)$ is said to be $\alpha_N$-covered if every path $f \in F_{ab}$ is covered by at least one AP.

*Remark.* (1) To provide $\alpha_N$-coverage, it suffices to only consider those pairs of points that are *exactly* $\alpha$ distance apart, since for any path connecting two points that are at least $\alpha$ distance apart, there must be two points on the path that are exactly $\alpha$ distance apart by the continuity of the distance function. (2) If a deployment provides $\alpha$-coverage, it also provides $\alpha_N$-coverage. The converse is not true. For instance, the deployment in Figure 2.1(b) also provides $\alpha_N$-coverage for $\alpha = 2$. Now suppose $\alpha = 5$, then since the diameter of the graph is 4, the distance between any pair of points in the graph is at most 4, $\alpha_N$-coverage is satisfied without deploying any APs. However, an empty deployment does not provide $\alpha$-coverage since the longest path in the graph has length 8.

**The approximation of $\alpha_N$-coverage to $\alpha$-coverage in grid graphs:** Consider a $m \times n$ grid graph with $m$ vertices in each row and $n$ vertices in each column and
Figure 2.3: Two grid graphs with (a) a Hamiltonian circuit highlighted and (b) a Hamiltonian path highlighted.

suppose the length of each edge is $u$. Suppose $\alpha$ equals to the diameter of the graph, that is $\alpha = (m+n-2)u$. When either $m$ or $n$ is even, the grid graph has a Hamiltonian circuit (see Figure 2.3(a)). Otherwise, it has a Hamiltonian path (see Figure 2.3(b)).

It follows that the length of a longest general path in the graph, denoted as $\lambda$, equals to $mnu$ (in the first case) or is arbitrarily close to $mnu$ (in the second case). We observe that when $m = n$, the larger the ratio $\alpha/u$ is (or equivalently, the larger the $m$ is), the larger $\lambda/\alpha$ is and less efficient the approximation is. The simulation result (see Section 2.5.1) over a real road network confirms this observation. If either $m$ or $n$ is close to 1, $\lambda/\alpha$ is also close to 1. Intuitively, for a sparse road network that is far from a regular grid, the approximation is much better than in a dense road network, which is also confirmed by the simulation result.

Given $G_\mathcal{R}$ – the graph model of a road network, $A_0$ – a set of points in $G_\mathcal{R}$ that models the APs previously deployed, we ask the following two questions – 1) given a value $\alpha > 0$, determine if $A_0$ provides $\alpha_N$-coverage (coverage verification), and if not 2) find a minimum set of points $A$ in $G_\mathcal{R}$ so that when new APs are deployed
at these points, $A_0 \cup A$ provides $\alpha_N$-coverage for the given $\alpha$ (optimal deployment). Notice that the second problem addresses both the new deployment and incremental deployment. We show in Section 2.2.2 that it can be verified in polynomial time whether a deployment provides $\alpha_N$-coverage. However, finding an optimal deployment that provides $\alpha_N$-coverage is NP-hard as proved in Section 2.2.3. We provide an $O(\log n)$ factor approximation algorithm in Section 2.2.4, where $n = |V(G_R) \cup A_0|$. A budgeted version of the problem is studied in Section 2.2.5. We note that as a real road network could be far from a regular grid, especially for a sparse road network, a simple greedy solution based on grid assumption is not desired.

Notice that, $\alpha_N$-coverage requires that for any pair of points that are $\alpha$ distance apart, all paths connecting them are covered, and the number of such paths could be exponential. In reality, however, there are a small subset of paths most frequently traveled between any two places, which can be learned from historical traffic data [34]. For instance, people usually follow a close to shortest path from their source to the destination. Note that although there may have multiple (or even exponential number of) shortest paths between two points in a grid graph, this rarely happens in a real road network, where there is typically a unique shortest path between any two locations. The next metric, $\alpha_P$-coverage, captures this observation. Although finding an optimal deployment that achieves $\alpha_P$-coverage is still NP-hard, we are able to find a more efficient approximation algorithm for it (see Section 2.2.4). Let $F_{ab}^*$ denote the set of paths between $a$ and $b$ most frequently traveled and we assume $|F_{ab}^*|$ is bounded by a small constant.

**Definition 2.2.4. $\alpha_P$-coverage** A deployment of APs provides Path Coverage of distance $\alpha$ ( $\alpha_P$-coverage for short) to $\mathcal{R}$, if for every pair of points $a$ and $b$ in $G_R$
with \( \text{dist}(a, b) \geq \alpha \), every path \( f \in F^*_ab \) is covered by at least one AP. A pair of points \((a, b)\) is said to be \( \alpha_P \)-covered if every path \( f \in F^*_ab \) is covered by at least one AP.

Remark. (1) When \( F^*_ab \) is the set of shortest paths between \( a \) and \( b \), to provide \( \alpha_P \)-coverage, it suffices to consider those pair of points that are exactly \( \alpha \) distance apart. However, this property does not hold for general \( F^* \). (2) A deployment that provides \( \alpha_N \)-coverage also provides \( \alpha_P \)-coverage, but not vice versa as shown in Figure 2.4.

![Figure 2.4: A small road network with unit length edges where a single AP located at the midpoint of the middle edge provides \( \alpha_P \)-coverage but not \( \alpha_N \)-coverage for \( \alpha = 5 \).](image)

To simplify the presentation, we assume \( F^*_ab \) is the set of shortest paths in the following discussion. The solution can be easily extended to a large class of \( F^*_ab \) as discussed in Section 2.2.6.

### 2.2.2 Coverage Verification

In this section, the following problem is considered: given a graph \( G_r \), a set of points \( A_0 \) in \( G_r \), and \( \alpha \), determine whether \( A_0 \) provides \( \alpha \)-coverage (resp. \( \alpha_N \)-coverage and \( \alpha_P \)-coverage) to \( r \).

**\( \alpha \)-Coverage Verification is co-NPC**

**Lemma 2.2.1.** The Hamiltonian Path problem restricted to the class of graphs without degree 1 vertices is NP-complete.
Proof. The general Hamiltonian Path problem can be reduced to this subcase as follows. Let $G$ be an arbitrary graph. For any degree 1 vertex of $G$, say $v$, attach a triangle to $v$. That is, add two new vertices $v_1$ and $v_2$ and three new edges, $(v, v_1), (v_1, v_2), (v, v_2)$. Let $G'$ denote the new graph, which has no degree 1 vertex. We claim that $G$ has a Hamiltonian Path iff $G'$ has a Hamiltonian Path. First suppose $G$ has a Hamiltonian Path, say $f$. Then $G$ has at most two degree 1 vertices and each of them is an end point of $f$. For every such vertex, say $v$, extend $f$ by adding edges $(v, v_1), (v_1, v_2)$. We then obtain a Hamiltonian Path of $G'$. Conversely, suppose $G'$ has a Hamiltonian Path $f$. For each degree 1 vertex of $G$, say $v$, we must have $(v_1, v_2)$ and one of $(v, v_1)$ and $(v, v_2)$ in $f$. By removing $v_1$ and $v_2$ and the two edges from $f$ for every such $v$, we get a Hamiltonian Path of $G$. \hfill $\Box$

**Theorem 2.2.1.** The $\alpha$-coverage verification problem is co-NPC.

Proof. Given a deployment of a graph $G$, the complement of the $\alpha$-coverage verification problem is NP since a nondeterministic algorithm needs only guess a path of $G$, say $f$, and check in polynomial time if there is a general path of length $\alpha$ containing $f$ as a maximal subpath that is not covered.

We then reduce the Hamiltonian Path problem restricted to the class of graphs without degree 1 vertices to the complement of the $\alpha$-coverage verification problem as follows. Let $G(V, E)$ be a graph where each vertex has degree at least 2 and each edge has unit length. We claim that $G$ has a Hamiltonian path iff $G$ (with an empty deployment) is not $\alpha$ covered for $\alpha = |V|$. First, suppose $G$ has no Hamiltonian path. Then the length of a longest path in $G$ ending at vertices is at most $|V| - 2$. Hence the length of any general path in $G$ is less than $|V|$. Note that the equality does not hold since otherwise $G$ must have a Hamiltonian circuit and hence also a Hamiltonian path.
Therefore, $G$ is $\alpha$ covered. Conversely, suppose $G$ has a Hamiltonian path $f$. The length of $f$ is $|V| - 1$. Since every vertex of $G$ has degree at least 2, $f$ can be extended to a general path (which could be a Hamiltonian circuit) of length $|V|$. Hence, $G$ is not $\alpha$ covered. It follows that the complement of the $\alpha$-coverage verification problem is NP-complement. Hence the $\alpha$-coverage verification problem is co-NPC. 

\textbf{The Verification of $\alpha_N$-Coverage}

We show that $\alpha_N$-coverage can be verified in polynomial time. First, a new graph $G(V,E) = G_R \lor A_0$ is obtained. That is, we make each point in $A_0$ a vertex. Note that if there is an edge in $G$ with length larger than $\alpha$, then $A_0$ clearly fails to provide $\alpha_N$-coverage. Since this condition can be easily checked, we assume no such edge exists in the following discussion. Each vertex $v$ of $G$ is assigned a weight, denoted as $w(v)$, which equals to 1 if an AP is deployed at $v$, and is 0 otherwise. The following definitions are made with respect to $G$.

\textbf{Definition 2.2.5. Coverage weight:} The coverage weight of a path $f$, denoted as $c(f)$, equals to the sum of weight of the vertices on $f$.

\textbf{Definition 2.2.6. Coverage distance:} The coverage distance of a pair of points $(a,b)$ in $G$, denoted as $c(a,b)$, equals to the minimum coverage weight of all paths between $a$ and $b$.

We use the term $\alpha$-pair to refer to a pair of points in $G_R$ that are a distance of $\alpha$ apart. We observe that a deployment provides $\alpha_N$-coverage iff the coverage distance of each $\alpha$-pair is at least 1. We distinguish two types of $\alpha$-pairs: (1) At least one point in the pair is a vertex; (2) None of the two points in the pair is a vertex. Note that there are at most $2|V||E|$ number of $\alpha$-pairs in the first category since for any
vertex $v$ and an arbitrary edge $e$, there are at most two points on $e$ that are exactly $\alpha$ distance away from $v$. Furthermore, although there are infinite number of $\alpha$-pairs in the second category, they can be divided into equivalent classes as follows.

**Definition 2.2.7. N-equivalent pairs:** Two pair of points $(a, b)$ and $(c, d)$ in the second category are N-equivalent if $a$ and $c$ are on the same edge, and $b$ and $d$ are on the same edge.

We observe that all the pairs in the same N-equivalence class have the same coverage distance, and the number of equivalence classes is bounded by $|E|^2$ since for any pair of edges, there is at most one equivalence class. Furthermore, it suffices to only consider those pairs that are equivalent to an $\alpha$-pair, and every such class corresponds to a pair of edges $e_1$ and $e_2$ where there is point $a$ on $e_1$ and $b$ on $e_2$ such that $(a, b)$ is an $\alpha$-pair. Therefore, once all the equivalence classes are identified, $\alpha_N$-coverage can be determined by checking the coverage distance of $\alpha$-pairs in each class one by one. We then discuss the two steps in detail.

First, to identify all the N-equivalent pairs that are $\alpha$-distance apart, the following procedure is applied, which for a given pair of edges $e_1 = (u_1, u_2)$ and $e_2 = (v_1, v_2)$, determines if there are points $a$ on $e_1$ and $b$ on $e_2$ such that $\text{dist}(a, b) = \alpha$. Let $t_1 = |au_1|/|e_1|$, $t_2 = |bv_1|/|e_2|$. Let $d_{ij}(a, b) = \text{dist}(u_i, v_j) + |au_i| + |bv_j|, i, j \in \{1, 2\}$. Then $d_{ij}$ is a linear function of $t_1 \in [0, 1]$ and $t_2 \in [0, 1]$ for $i, j \in \{1, 2\}$. For given $i, j, i', j' \in \{1, 2\}$ and $(i, j) \neq (i', j')$, the solutions to the equation $d_{ij} - d_{i'j'} = 0$ divide the square $[0, 1] \times [0, 1]$ into two subregions. All the six equations partition $[0, 1] \times [0, 1]$ into convex subregions. Using the partition, the minimum and maximum distances between any pair of points on the two edges can be determined by only
studying the vertices of the convex regions. There is an $\alpha$-pair with respect to the
two edges iff $\alpha$ is between the two extremal values.

Second, to compute the coverage distance of an $\alpha$-pair, we first note that the
coverage distance of every pair of vertices of $G$ can be computed by extending the
Floyd's all-pairs shortest paths algorithm to the node weighted case, where the weight
of a vertex is its coverage weight. Now suppose $a$ is on edge $(u_1, u_2)$ and $b$ is on edge
$(v_1, v_2)$ and none of $a$ and $b$ is a vertex, then $c(a, b) = \min(c(u_1, v_1), c(u_1, v_2), c(u_2, v_1), c(u_2, v_2))$. The case where $a$ or $b$ is a vertex can be derived similarly.

The Verification of $\alpha_P$-Coverage

A polynomial time algorithm for $\alpha_P$-coverage verification can be derived in a
similar way as that for $\alpha_N$-coverage.

**Definition 2.2.8. The core of a general path:** The core of a general path is its
longest subpath ending at vertices of $G$.

For instance, in Figure 2.1, the core of $f_{st}$ is the path $(v_3, v_5, v_4, v_7)$.

**Definition 2.2.9. The core of a set of paths:** Let $F$ be a set of paths. The core
of $F$ is a set of paths, where each of them is the core of a path in $F$.

**Definition 2.2.10. P-equivalent pairs:** Two pairs of points $(a, b)$ and $(c, d)$ are
$P$-equivalent if 1) $a$ and $c$ are on the same edge, and $b$ and $d$ are on the same edge
and 2) the set of shortest paths between $a$ and $b$ and that between $c$ and $d$ have the
same core.

A polynomial time algorithm for verifying $\alpha_P$-coverage can then be derived by
noticing that 1) if $(a, b)$ and $(c, d)$ are $P$-equivalent pairs, then all the shortest paths
between $a$ and $b$ are covered iff all the shortest paths between $c$ and $d$ are covered; 2) each equivalence class can be verified in polynomial time since there are a small number of paths in the core shared by all the pairs in the same class; 3) the number of equivalence classes is bounded by $O(|E|^2)$ since for every pair of edges, there are only a small number of equivalence classes, and it suffices to consider only those pairs equivalent to an $\alpha$-pair. The procedure for identifying $N$-equivalent pairs can be extended to identify $P$-equivalent pairs with slight modification.

2.2.3 Proof of Hardness

We name the decision version of $\alpha_N$-coverage (resp. $\alpha_P$-coverage) optimization problem $\alpha_N$-COVER (resp. $\alpha_P$-COVER). In this section, we show that $\alpha_N$- and $\alpha_P$-COVER are NP-complete. By the existence of verification algorithms just presented, it suffices to show that there is a NP-complete problem that can be reduced to $\alpha_N$-COVER (resp. $\alpha_P$-COVER) in polynomial time. It is known that VERTEX COVER is NP-complete when restricted to triangle-free graphs\(^1\) without degree 1 vertices, since it remains NP-complete when restricted to triangle-free, 3-connected, cubic\(^2\) planar graphs [80]. We will reduce this subproblem of VERTEX COVER to a subproblem of $\alpha_N$-COVER with $\alpha = 2$ and $|A_0| = 0$, that is, there are no previously deployed APs.

**Lemma 2.2.2.** If $G$ is a triangle-free graph having no degree 1 vertices where each edge has unit length, a set of vertices form a vertex cover of $G$ iff it provides $\alpha_N$-coverage to $G$ for $\alpha = 2$.

\(^1\)A graph is triangle free if it has no cycles of length three.
\(^2\)A cubic graph is a graph where each vertex is incident to exactly three edges.
Proof. Suppose that a set of vertices, $A$, provides vertex cover to $G$, but does not provide $\alpha_N$-coverage for $\alpha = 2$. Then there is a path of length 2 that is disjoint from any vertices in $A$, which contains at least one edge not covered by any vertices in $A$, a contradiction. Conversely, suppose a set of vertices, $A$, provides $\alpha_N$-coverage to $G$ for $\alpha = 2$, but it is not a vertex cover, then there must be an uncovered edge $(u, v)$. Since $G$ has no degree 1 vertices and is triangle free, there must exist edges $(u_1, u)$ and $(v, v_1)$ with midpoints $a$ and $b$, respectively, such that $f_{ab} = auvb$ is a path disjoint from $A$ and $dist(a, b) = 2$, a contradiction. \hfill \sq

Although in general, a deployment that only uses vertices may be suboptimal as shown in Section 2.2.4, for the set of instances of $\alpha_N$-COVER we consider in this section, restricting APs to vertices actually gives optimal solutions as shown in the following lemma.

**Lemma 2.2.3.** Let $G$ be a triangle-free graph having no degree 1 vertices where every edge has unit length. If there is a set of $k$ points in $G$ that provides $\alpha_N$-coverage to $G$, then there is a set of $k$ vertices that also provides $\alpha_N$-coverage to $G$ for $\alpha = 2$.

*Proof.* Let $A$ be a set of $k$ points in $G$ that provides $\alpha_N$-coverage to $G$. We will construct a set of vertices, $B$, such that $|B| \leq k$ and $B$ is a vertex cover for $G$. The claim then follows from Lemma 2.2.2.

First, we note that for any vertex $v$, there is at most one edge incident to $v$ that is disjoint from $A$. Since, otherwise the two uncovered edges incident on $v$, say $(u, v)$ and $(v, w)$, form an uncovered path $(u, v, w)$ with $dist(u, w) = 2$, due to the fact that $G$ is triangle-free. This contradicts the fact that $A$ provides $\alpha_N$-coverage to $G$. 

30
All the points in $A$ that are also vertices are first added to $B$. Consider an edge $(x, y)$ that is disjoint from $A$. There must exist distinct edges $(w, x)$ and $(y, z)$ with $w \neq z$ and points $a$ on $(w, x)$ and $b$ on $(y, z)$ such that $a \in A$ and $b \in A$, and $|ax| \leq 0.5$ or $|by| \leq 0.5$. Without loss of generality, suppose $|ax| \leq 0.5$. Add $x$ to $B$. If there is an edge $(v, w)$ incident to $w$ that is disjoint from $A$, then there must be an edge $(u, v)$ and a point $c$ on $(u, v)$ such that $|cv| \leq 0.5$, add $v$ to $B$. Continue this process until either $y$ or a vertex where all the incident edges are covered by points in $A$ is reached. Then pick another edge that is disjoint from $A$. Repeat the process until all such edges have been considered. Then for each edge that contains a point in $A$ and has not been visited, add any one of its two ends to $B$. By its construction, $B$ covers all the edges and $|B| \leq |A|$.

\[ \square \]

**Theorem 2.2.2.** $\alpha_N$-COVER is NP-complete.

**Proof.** Given a triangle-free graph $G$ without degree 1 vertices, make an instance of $\alpha_N$-COVER with $G$ as the graph model of a road network where each edge has unit length, $\alpha = 2$, and $|A_0| = 0$. The theorem then follows from Lemma 2.2.2 and Lemma 2.2.3.

\[ \square \]

Similar argument as above can be applied to $\alpha_P$-COVER. Therefore, we also have

**Theorem 2.2.3.** $\alpha_P$-COVER is NP-complete.

### 2.2.4 Approximation Algorithms

In this section, we present approximation algorithms for the $\alpha_N$-coverage and $\alpha_P$-coverage optimization problems. Formally, given graph $G_R$ and a set of point $A_0$
in $G_R$, and $\alpha$, find a set of points $A$ such that $A_0 \cup A$ provides $\alpha_N$-coverage or $\alpha_P$-coverage to $\mathcal{R}$. We would like to minimize $|A|$. Let $G = G_R \cup A_0$. For any edge $e$ of $G$, if $|e| > \alpha$, $e$ is chopped into $\lceil |e|/\alpha \rceil$ pieces of equal length. In the following discussion, we will assume that the length of any edge of $G$ is no more than $\alpha$. Let $n$ and $m$ denote the number of vertices and edges of $G$, respectively.

We present two polynomial time algorithms in this section. The first algorithm reduces $\alpha_N$-coverage to the vertex multicut problem [31] and the second one reduces $\alpha_P$-coverage to the set cover problem [81]. Both algorithms have an $O(\log n)$ approximation factor. The second algorithm also works for $\alpha_N$-coverage. However, only for $\alpha_P$-coverage, the algorithm has polynomial time complexity. It should be noted that for a given road network, the first algorithm is much more time consuming than the second one, which is expected since $\alpha_N$-coverage provides higher coverage quality than $\alpha_P$-coverage.

Given $\alpha$, let $\text{OPT}$ denote the minimum $|A|$ in any deployment that provides $\alpha_N$-coverage where APs can be deployed at anywhere in $G$, and $\text{OPT}'$ denote the minimum $|A|$ for providing $\alpha_N$-coverage when APs can only be deployed at the vertices of $G$. Both of our algorithms use only the vertices of $G$ to construct $A$, which avoids an infinite search space so that approximation solutions can be found. The following lemma states that such a deployment decision doubles the number of APs used in the worst case.

**Lemma 2.2.4.** $\text{OPT} \leq \text{OPT}' \leq 2 \times \text{OPT}$.

**Proof.** $\text{OPT} \leq \text{OPT}'$ follows directly from the definition. Let $A$ be a set of $\text{OPT}$ points such that $A_0 \cup A$ provides $\alpha_N$-coverage. We apply the following two rules to construct a set of vertices, say $A'$, such that $A_0 \cup A'$ also provides $\alpha_N$-coverage: 1)
Add all the points in $A$ that are also vertices of $G$ to $A'$; 2) For each point in $A$ that is not a vertex, add the two ends of the edge where the point is on to $A'$. We have $|A'| \leq 2 \times |A|$. Let $F$ denote the set of paths required to be covered to ensure $\alpha_N$-coverage. For any $f \in F$, there exists a point, say $p$, in $A_0 \cup A$ that covers $f$. If $p$ is a vertex, $f$ is also covered by $p \in A_0 \cup A'$. Otherwise, suppose $p$ is on edge $(u, v)$. Then $u \in A'$ and $v \in A'$. Since $|f| \geq \alpha \geq |(u, v)|$, $f$ goes through at least one of $u$ and $v$, and thus is covered by $A'$. Therefore, $A_0 \cup A'$ also provides $\alpha_N$-coverage.

The same result also holds for $\alpha_P$-coverage. Notice that, an important advantage of restricting APs to vertices is that it increases the chance of data access since vehicles may stop or slow down near road intersections.

Figure 2.5 gives an example where using only vertices gives a suboptimal solution. In the figure, $|A_0| = 0$, $G$ is a single path with $n$ vertices, where each edge has unit length, and $\alpha \in (1, 2)$. To achieve $\alpha_N$-coverage, an optimal solution is a set of points uniformly spaced with $\alpha$ distance along the path. On the other hand, if only the vertices are allowed to be used, then all the vertices except the two ends of the path
have to be used to ensure \( \alpha_N \)-coverage. Therefore, \( \text{OPT} = \lceil (n - 1)/\alpha \rceil - 1 \), \( \text{OPT}' = n - 2 \), and \( \lim_{n \to \infty} \text{OPT}'/\text{OPT} = \alpha \). In particular, when \( n = 4 \), and \( \alpha \in [1.5, 2) \), a minimum cover contains only one point in the middle of the path, while two vertices are needed to ensure the coverage. In general, if \( G \) is a single path with \( n \) vertices and each edge has unit length, the factor 2 can be achieved when \( n = 2k + 2, \alpha \in [k + 0.5, k + 1) \) for any integer \( k \geq 1 \).

\( \alpha_N \)-Coverage via Vertex MultiCut

Recall that we have distinguished two types of \( \alpha \)-pairs. Although for an arbitrary \( \alpha \), both types of \( \alpha \)-pairs have to be considered, it suffices to consider only the second type of \( \alpha \)-pairs in practice by the following observation. First, we can choose a small enough \( \epsilon \) such that for any pair of vertices \( (u, v) \) of \( G \), \( \text{dist}(u, v) \neq \alpha \). By making \( \epsilon \) small enough, \( (\alpha - \epsilon)_N \)-coverage can be viewed as equivalent to \( \alpha_N \)-coverage in any real settings. Second, by adjusting \( \epsilon \) and a careful analysis of the convex partition of \([0, 1] \times [0, 1]\) discussed in Section 2.2.2, we can ensure that if \( (a, b) \) is an \( \alpha \)-pair and \( a \) is a vertex and \( b \) is a non-vertex point, there always exists another \( \alpha \)-pair \( (c, d) \) such that none of \( c \) or \( d \) is a vertex, and for any deployment that only uses vertices, \( (a, b) \) is covered only if \( (c, d) \) is covered. This is possible because of the continuity of the distance function. Therefore, it suffices to only consider those \( \alpha \)-pairs consisting of non-vertex points.

Assuming only the vertices of \( G \) are used to construct \( A \), the \( \alpha_N \)-coverage optimization problem can be reduced to the minimum vertex multicut problem [31] defined as follows. Given a connected undirected graph \( G(V, E) \) with positive costs on its vertices, let \( \{(s_1, t_1), \ldots , (s_k, t_k)\} \) be a set of pairs of vertices, named as terminals, where each pair is distinct, but vertices in different pairs are not required to be
distinct. A vertex multicut is a set of non-terminal vertices whose removal separates each pair. The problem is to find a vertex multicut of minimum cost. We assume that all vertices have the same cost in this section. The main steps of our algorithm are summarized in Algorithm 2.2.1. For each pair of edges \((e_1, e_2)\), the algorithm determines whether there are points \(a\) in \(e_1\) and \(b\) in \(e_2\) such that \(\text{dist}(a, b) = \alpha\), which can be done in constant time after the preprocessing as shown in Section 2.2.2. If there is such a pair and the coverage distance of the pair is 0, the midpoints of \(e_1\) and \(e_2\) are treated as a terminal pair. The algorithm then resorts to the well known GVY algorithm [81] to find a minimum vertex multicut with \(A_0\) as a subset that separates every such midpoint pairs.

Algorithm 2.2.1 \(\alpha_N\)-coverage via vertex multicut (VC)

Input: \(G_R\), the graph model of a road network \(R\), \(A_0\), the set of vertices of \(G\) that represents APs previously deployed, and \(\alpha\)

Output: \(A \subseteq V(G)\) that provides \(\alpha_N\)-coverage

1: \(G \leftarrow G_R \lor A_0\);
2: if \(A_0\) provides \(\alpha_N\)-coverage to \(G\) then return;
3: \(\Gamma \leftarrow \emptyset\);
4: for each pair of edges \((e_1, e_2)\) of \(G\) do
5: if there are points \(a\) on \(e_1\) and \(b\) on \(e_2\) such that \(\text{dist}(a, b) = \alpha\) and \(c(a, b) = 0\) then
6: \(\Gamma \leftarrow \Gamma \cup (m_1, m_2)\); \(\triangleright m_i: \text{the midpoint of } e_i\)
7: \(A \leftarrow \text{GVY}(G, \Gamma, A_0)\) \(\triangleright\) Apply the GVY algorithm [81] to find a vertex multicut with \(A_0\) as a subset that separates each pair of midpoints in \(\Gamma\).

The following lemma states that Algorithm 2.2.1 finds a feasible solution to the \(\alpha_N\)-coverage problem.

**Lemma 2.2.5.** The set of vertices in the multicut found by Algorithm 2.2.1 provides \(\alpha_N\)-coverage.
Proof. We need to show that any path \( f_{ab} \) in \( G \) with \( \text{dist}(a, b) = \alpha \) is covered by at least one vertex in the multicut. By the assumptions made above, we can assume \( a \) and \( b \) are the interior points of two distinct edges of \( G \), say \( e_1 = (u_1, u_2) \) and \( e_2 = (v_1, v_2) \) respectively. Since \( \text{dist}(a, b) = \alpha \), the midpoints of \( e_1 \) and \( e_2 \), say \( m_1 \) and \( m_2 \), form a pair of terminals. Notice that \( f_{ab} \) contains a path \( f \) starting at \( u_i \) and ending at \( v_j \) for certain \( i, j \in \{1, 2\} \). Since by removing the multicut found, \( m_1 \) and \( m_2 \) will be disconnected, \( f \) goes through at least one vertex in the cut, so does \( f_{ab} \).

The algorithm stated in [81] actually solves the minimum edge multicut problem [30] defined similarly and achieves an \( O(\log k) \) approximation factor where \( k \) is the number of terminal pairs. However, it can be extended to the vertex version while preserving the approximation factor [31], by considering the following linear program formulation.

Let \( S = \{s_1, s_2, \ldots, s_k\} \) denote the set of sources and \( T = \{t_1, t_2, \ldots, t_k\} \) denote the set of destinations in the terminal pairs. For each vertex \( v \in V \setminus (S \cup T) \), let \( d_v \) be a non-negative variable called distance label. For each vertex \( v \in V \) and each terminal pair \( (s_i, t_i), i = 1, \ldots, k \), let \( y_{v,i} \) denote the shortest distance (in terms of the distance labels) from \( s_i \) to \( v \). When applied to our scenario, \( d_v \) corresponds to the coverage weight of \( v \), and \( y_{v,i} \) models \( c(s_i, v) \), the coverage distance from \( s_i \) to \( v \).

The linear program (after LP-relaxation) of the minimum vertex multicut problem is as follows, where \( c_v \) is the cost of vertex \( v \) and is fixed to 1 for each \( v \) in our scenario.
\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V \setminus (S \cup T)} c_v d_v \\
\text{subject to} & \quad y_{v,i} \leq y_{u,i} + d_v, \forall (u, v) \in E, \forall i = 1, \ldots, k, \\
& \quad y_{s,i,i} = 0, \forall i = 1, \ldots, k, \\
& \quad y_{t,i,i} \geq 1, \forall i = 1, \ldots, k, \\
& \quad d_v \geq 0, \forall v \in V \setminus (S \cup T), \\
& \quad d_v = 0, \forall v \in S \cup T.
\end{align*}
\]

The first constraint says that \(y_{v,i}\) satisfies triangle inequality. The second constraint says that the shortest distance from \(s_i\) to itself is 0. The third constraint requires that the shortest distance between each terminal pair is at least 1, which ensures that the set of vertices with positive distance labels form a multicut. The third and fourth constraints and the objective function together ensure that the distance label of each nonterminal vertex is between 0 and 1, and the last constraint says that the distance label of each terminal is 0. The dual of the above program models the vertex version of the well-known sum multicommodity flow problem [81].

The standard GVY algorithm first solves the above linear program to get a set of distance labels for all the vertices. Then from those vertices with positive distance labels, a subset of them are selected to form a multicut (LP-rounding). Although the first step is polynomial time solvable, it is very time consuming to find an accurate solution for a large road network, especially in our case where \(k\) – the number of terminal pairs (the number of commodities in the dual problem) – equals to \(m^2\) in the worst case, where \(m\) is the number of edges in the graph. To reduce time complexity, instead of solving the above linear program, we apply the combinatorial FPTAS algorithm proposed in [27], which computes a \((1 - 4\epsilon)\text{OPT}\) solution to the edge version sum multicommodity flow problem in \(O(\frac{1}{\epsilon^2}m(m + n \log m) \log n)\) time. It is important to notice that the running time is independent of \(k\). We adapt this algorithm to the vertex version and to find a close to optimal fractional multicut. The
second step introduces an extra $O(\log k)$ factor, which is the best known result for the minimum vertex multicut problem for a general graph. Considering Lemma 2.2.4, Algorithm 2.2.1 has an $O(\log n)$ approximation factor.

The standard GVY algorithm does not consider the case where a set of vertices are forbidden to be chosen as cut nodes. This can be solved by fixing the cost and the distance label of each vertex in $A_0$ to be 0 and 1, respectively. The analysis in [81] can still be applied to show that this modification does not impact the approximation factor.

**αₚ-Coverage via Set Cover**

Assuming only the vertices of $G$ are used to construct $A$, the $αₚ$-coverage optimization problem can be reduced to the set covering optimization problem [81] as shown in Algorithm 2.2.2. The ground set $U$ is the union of the set of uncovered paths in the cores (defined in Section 2.2.2) of all P-equivalent $α$-pairs, and each $S_v$ corresponds to a subset of paths in $U$ covered by a vertex $v$ not in $A_0$.

---

**Algorithm 2.2.2** $αₚ$-coverage via set cover (SC)

Input: $G_R$, the graph model of a road network $R$, $A_0$, the set of vertices of $G$ that represents APs previously deployed, and $α$

Output: $A \subseteq V(G)$ that provides $αₚ$-coverage

1: $G \leftarrow G_R \lor A_0$;
2: if $A_0$ provides $αₚ$-coverage to $G$ then return;
3: $U \leftarrow$ the union of cores with respect to all P-equivalent $α$-pairs except those paths have been covered by $A_0$;
4: $S \leftarrow \{S_v : v \in V \setminus A_0\}$ where $S_v \subseteq U$ is the set of paths covered by vertex $v \notin A_0$;
5: $A \leftarrow$ a subcollection of $S$ of minimum size that covers all the elements of $U$ using the greedy set cover algorithm [81]
This algorithm also applies to $\alpha_N$-coverage, where set $U$ has exponential size in the worst case. For $\alpha_P$-coverage, $|U| = O(|E|^2)$. Since the greedy set cover algorithm has an approximation factor $H_{U} = 1 + \frac{1}{2} + \cdots + \frac{1}{|U|} = \ln(|U|) + O(1)$ [81], considering Lemma 2.2.4, this algorithm has an $O(\log n)$ factor. Furthermore, this algorithm is much more efficient than the VC algorithm since the greedy set cover algorithm is much less time consuming than the GVY algorithm.

2.2.5 Budgeted Version

In this section, we show that the algorithms VC and SC can be used as subroutines to solve the following budgeted coverage problem: given a positive integer $B$, find a set of points $A$ in $G_R$ with $|A| \leq B$ such that the deployment $A_0 \cup A$ provides $\alpha_N$-coverage for a minimum possible $\alpha$ (budgeted deployment). For this purpose, a binary search technique is applied as shown in Algorithm 2.2.3 for budgeted $\alpha_N$-coverage, which applies to budgeted $\alpha_P$-coverage as well by replacing VC algorithm with SC algorithm. Staring at $\alpha$ equals to the diameter of $G$, that is the distance between two farthest points in $G$, a deployment providing $\alpha_N$-coverage is computed by invoking the VC algorithm. If the size of this deployment is larger than the budget, a larger $\alpha$ is selected. Otherwise, a smaller $\alpha$ is selected. The procedure continues until the difference between the upper and lower bounds of $\alpha$ is at most $\delta$, which can be adjusted to control the accuracy. Such a binary search provides a bicriteria approximation to the problem by relaxing both the requirement on the objective function and that on the budget. Specifically, it finds a subset $A \subseteq V(G)$ that is close to the optimum in the following sense.
**Theorem 2.2.4.** For a given budget $b$, let $\hat{\alpha}(b)$ denote the minimum $\alpha$ that can be achieved in an optimal solution to the budgeted $\alpha_N$-coverage problem, and let $\alpha(b)$ denote the solution of Algorithm 2.2.3. Then for a small enough $\delta$, $\alpha(\kappa b) \leq \hat{\alpha}(b)$, where $\kappa = O(\log n)$ is the approximation factor of the VC algorithm. A similar result holds for $\alpha_P$-coverage as well.

**Algorithm 2.2.3** Budgeted $\alpha_N$-coverage via binary search

| Input: $G_R$, the graph model of a road network $R$, $A_0$, the set of vertices of $G$ that represents APs previously deployed, and $B$, the number of extra APs can be used |
| Output: $\alpha$, $A \subseteq V(G)$ that provides $\alpha_N$-coverage |
| 1: $G \leftarrow G_R \lor A_0$; $A \leftarrow \emptyset$; |
| 2: $\alpha_{\text{min}} \leftarrow 0$; $\alpha_{\text{max}} \leftarrow$ the diameter of $G$; $\alpha \leftarrow \alpha_{\text{max}}$; |
| 3: while $(\alpha_{\text{max}} - \alpha_{\text{min}} > \delta)$ do |
| 4: $\bar{\alpha} \leftarrow (\alpha_{\text{max}} + \alpha_{\text{min}})/2$; |
| 5: $\bar{A} \leftarrow \text{VC}(G_R, A_0, \alpha)$; |
| 6: if $(|\bar{A}| > B)$ then |
| 7: $\alpha_{\text{min}} \leftarrow \bar{\alpha}$; |
| 8: else |
| 9: $\alpha_{\text{max}} \leftarrow \bar{\alpha}$; $\alpha \leftarrow \bar{\alpha}$; $A \leftarrow \bar{A}$ |

**2.2.6 Discussion**

In this section, we discuss several issues related to the modeling of road networks and APs including some real constraints ignored before, and an extension of $\alpha$-coverage.

**Modeling issues:** We have considered an undirected graph model of road networks so far to simplify the discussion. However, most of our results can be extended to a directed graph that models one-way roads as well. First, we can extend the definitions of general paths and distance on graph by taking direction into account. In this case,
dist\((a, b)\) and dist\((b, a)\) may be different. The definition of the three coverage metrics and the two types of equivalence classes can be extended as well. Then both the coverage verification algorithm and the set cover based optimization algorithm can be extended to directed graphs. On the other hand, it is known that the minimum multicut problem is much harder to approximate for directed graphs than undirected graphs. There are some recent results for finding the minimum edge multicut [5,38]. However, whether these results can be extended to vertex multicut while retaining the same approximation factor, needs further study.

**More about \(\alpha_P\)-Coverage:** In our definition of \(\alpha_P\)-coverage, the set of paths associated with a pair of points can be an arbitrary set of critical paths. Although we use shortest path as an example in the coverage verification and optimization algorithms, they can be extended to the general case as long as the following condition is satisfied: for each pair of edges in \(G\), there are only a constant number of \(P\)-equivalence classes. In general, one needs to consider all the pairs that are equivalent to an \(\alpha^*\)-pair for some \(\alpha^* \geq \alpha\). However, the total number of such equivalent classes is still bounded by \(O(|E|^2)\). Since the verification algorithm takes \(O(c|V|)\) time to check each equivalent class where \(c\) is the maximum number of paths in the cores associated with the pairs, the total running time is \(O(c|V||E|^2)\). The set cover based optimization algorithm takes two collections \(U\) and \(S\) as the input, where \(|S| = O(|V|)\). Since each equivalent class contributes at most \(c\) paths to \(U\), \(|U| = O(c|E|^2)\). Therefore, the optimization algorithm can also be done in polynomial time.
2.3 Connected Alpha Coverage

The notion of Alpha Coverage has been extended by taking connectivity into account and applied to an asset tracking system [64]. In this scenario, the asset under tracking is equipped with a tag mote, which sends beacons periodically, and a set of anchor nodes are deployed along a city-wide road network for tracking tag motes. The joint coverage and connectivity requirement is as follows: (1) a tag node that moves an absolute displacement beyond a certain distance (a parameter) along the road network starting at any location is guaranteed to be captured by at least one anchor node; (2) such events can be forwarded to a gateway node through a multi-hop mesh network of anchor nodes. We propose a two-stage solution to the problem of finding a deployment using minimum number of anchor nodes to provide the desired coverage and connectivity. Note that the coverage subproblem is exactly $\alpha_N$-coverage and the VC algorithm is applied. An approximation algorithm is derived for the connectivity subproblem, and an upper bound on the joint approximation factor is derived.

2.3.1 Problem Formulation

The same road network model as in Alpha Coverage is used here, and the trajectory of a moving vehicle is again modeled as a set of consecutive pathes in the road network graph $G$ starting and ending at any points (not necessary vertices) in $G$. In addition, we assume that anchor nodes are allowed to be deployed at all the the road intersections and possibly at some other given points in the road network graph $G$. By regarding these extra points as artificial road intersections, we may simply assume that every vertex of $G$ is a candidate location for deploying anchor nodes. Our solution can be easily extended to the case where only a subset of vertices of $G$
are candidate locations. We consider a homogeneous deployment in this work where each anchor node has the same communication range $R$, and the sensing range $r$ (for tracking beacons from the tag nodes) is small enough so that it is reasonable to model a deployed anchor node as a point. We say that a path $f$ is covered by an anchor node if it goes through the corresponding vertex of $G$ where the anchor node is deployed. For any two points $a, b$ in $G$, let $\text{dist}(a, b)$ denote their graph distance, that is, the length of a shortest path in $G$ connecting the two points, and let $F_{ab}$ denote the set of all possible paths connecting $a$ and $b$.

Again by introducing artificial intersections, we may safely assume that the length of each edge in $G$ is at most $\min(R, \alpha)$, where $\alpha$ is a given parameter. We further assume that the set of gateways with Internet backhaul are located at $B \subseteq V(G)$ with communication range $R$. Let $H$ denote the communication graph where $V(H) = V(G)$ and there is an edge between $a, b \in V(H)$ if their Euclidean distance $d(a, b) \leq R$. Note that $G$ is a spanning subgraph of $H$, and since $G$ is assumed to be connected, $H$ is also connected.

**Definition 2.3.1. Connected $\alpha_N$-Coverage:** A deployment of anchor nodes provides Connected $\alpha_N$-Coverage of distance $\alpha$, if (1) for any pair of points $(a, b)$ in $G$ with $\text{dist}(a, b) \geq \alpha$, any path $f \in F_{ab}$ is covered by at least one anchor node and (2) each anchor node is connected to at least one gateway (possibly via multi-hop wireless links).

We are looking for an optimal deployment that provides Connected $\alpha_N$-coverage for a given $\alpha$ while using minimum number of anchor nodes. Note that Connected $\alpha_P$-Coverage can be defined in a similar way and the two-stage algorithm discussed below can be applied as well, which will not be further discussed in this chapter.
2.3.2 Two-Stage Approximation

As a generalization of $\alpha_N$-coverage, the connected $\alpha_N$-coverage problem is also NP-hard. In this section, we provide an approximation algorithm by decomposing the problem into the coverage and connectivity subproblems. The entire solution is summarized in Algorithm 2.3.1. We then show how the approximation factor from the two subproblems can be combined to obtain an approximation factor for the joint problem.

Algorithm 2.3.1 Two-Stage Connected $\alpha_N$-Coverage (TSCC)

Input: $G$, $B$, $R$, $\alpha$.
Output: $A \subseteq V(G)$ that provides Connected $\alpha_N$-Coverage.

1: $A_1 \leftarrow$ a set of vertices providing $\alpha_N$-coverage found by the VC algorithm;
2: $H \leftarrow$ the communication graph for $V$ under range $R$;
3: $\tilde{H} \leftarrow H/B$ by replacing $B$ with a single vertex $b$;
4: $A_2 \leftarrow$ a close to optimal set of steiner nodes that connect $A_1 \setminus B \cup \{b\}$ in $\tilde{H}$;
5: $A = A_1 \cup A_2$

**Coverage Subproblem:** The coverage subproblem is exactly the $\alpha_N$-coverage optimization problem, and the vertex multicut based algorithm (VC) is applied.

**Connectivity Subproblem:** Given $H$, $B$, and $A_1$ computed by the coverage subproblem, the connectivity subproblem looks for a subset $A_2 \subseteq V(G) \setminus A_1$ such that for any $a \in A_1 \cup A_2$, there is a path in $H$ from $a$ to at least one $b \in B$, and $|A_2|$ is minimized. This problem can be reduced to the Node Weighted Steiner Tree Problem [36] with unit node weight as follows. First, given a connected undirected graph $G$ where each vertex has a positive weight, and a subset $T \subseteq V(G)$, the Node-Weighted Steiner Tree Problem (NSTP) asks for a subset $S \subseteq V(G) \setminus T$, such that the subgraph induced by $S \cup T$ is connected and the total weight of $S$ is minimized.
The vertices in $T$ are called terminals, and the vertices in $S$ are called Steiner points. Note that the weight of terminals does not count. Define $\tilde{H} = H/\mathcal{B}$, that is, $\tilde{H}$ is constructed from $H$ by replacing the vertices in $\mathcal{B}$ by a single vertex $b$ incident to all the edges which were incident in $H$ to at least one element in $\mathcal{B}$ [44]. Then we observe that the connectivity problem in $H$ is equivalent to NSTP with unit node weight and terminals $(A_1 \setminus \mathcal{B}) \cup \{b\}$ in $\tilde{H}$.

A constant factor approximation algorithm to the connectivity subproblem can then be derived as follows. First, we note that the general NSTP problem is harder than the (edge weighted) Steiner Tree Problem (STP). The latter allows a constant factor approximation while the best known lower bound on the approximation factor for NSTP is $O(\log k)$ where $k = |T|$ [48]. For unit disk graphs, however, a factor $2.5\rho$ approximation is obtained in [97] by reducing NSTP to STP and applying a factor $\rho$ algorithm to STP. The algorithm makes use of a key property proved in [18]: for a unit disk graph, there is an optimal node weighted Steiner tree such that the degree of each vertex in the tree is at most five. The same argument can be applied to all the vertices of $\tilde{H}$ except $b$, where the degree can be as large as $5|\mathcal{B}|$. However, since $b$ is a terminal, its weight does not count. Hence the algorithm can be applied to $\tilde{H}$ with the same factor retained, which can be as low as $2.5 \times (1 + \frac{\ln 3}{2}) \approx 3.88$ [73].

**Combined Approximation Factor:** Suppose the coverage subproblem and the connectivity subproblem can be approximated in a factor $\delta_1$ and $\delta_2$, respectively. We present the following lemma to show how these approximation factors can be combined to obtain an approximation factor for the joint problem.

**Lemma 2.3.1.** The two-stage algorithm yields a $(\delta_1 + \mu \delta_1 \delta_2)$ approximation for the Connected $\alpha_N$-Coverage problem, where $\mu = 2(\lceil \alpha/R \rceil - 1)$. 45
Proof. The following proof is similar to the analysis given in [76]. Let $A_1$ and $A_2$ denote the set of anchor nodes found by solving the coverage subproblem and the connectivity subproblem, respectively. Let $A = A_1 \cup A_2$. Since $A_1$ and $A_2$ are disjoint, we have $|A| = |A_1| + |A_2| \leq \delta_1 \text{OPT}_{\text{cov}} + \delta_2 \text{OPT}_{\text{con}}$, where $\text{OPT}_{\text{cov}}$ and $\text{OPT}_{\text{con}}$ denote the size of an optimal solution to the coverage subproblem and the connectivity subproblem given $A_1$ as input, respectively.

Given $A_1$, a (suboptimal) solution to the connectivity subproblem can be obtained by a growing process as follows. Initially, let $S = \emptyset$. At each step, find a vertex $a \in A_1 \setminus S$ that is closest (in terms of the graph distance in $G$) to $S$. Let $f$ denote a corresponding shortest path. Note that the length of $f$ is at most $\alpha$ since $A_1$ satisfies the coverage requirement. Hence by deploying at most $\mu = 2(\lceil \alpha/R \rceil - 1)$ extra anchor nodes along $f$, $a$ can be connected to an element in $\mathcal{B}$. Add $a$ and the vertices on $f$ corresponding to the extra anchor nodes to $S$. Repeat this process until all the elements in $A_1$ are connected to $\mathcal{B}$. This process gives a solution to the connectivity subproblem that uses at most $\mu|A_1|$ extra anchor nodes. Hence $\text{OPT}_{\text{con}} \leq \mu|A_1|$.

Let $OPT$ denote the size of an optimal solution to the Connected $\alpha_N$-Coverage problem. It is clear that $\text{OPT}_{\text{cov}} \leq OPT$. We then have $|A| \leq \delta_1 \text{OPT}_{\text{cov}} + \delta_2 (\mu|A_1|) \leq (\delta_1 + \mu \delta_1 \delta_2) \text{OPT}_{\text{cov}} \leq (\delta_1 + \mu \delta_1 \delta_2) \text{OPT}$.

From Lemma 2.3.1 and the approximation factors of the above algorithms for each subproblem, the two stage algorithm for the Connected $\alpha_N$-Coverage problem has an approximation factor $O(\frac{\alpha}{R} \log n)$ where $n = |V(G)|$.

\footnote{The factor 2 comes from the constraint that anchor nodes can only be deployed at vertices.}
2.4 Contact Opportunity

While Alpha Coverage provides a bound on worst-case interconnection gap, it only considers the number of contacts but ignores the quality of each contact. We propose to study metrics that can characterize the data quality that a mobile user experiences while traveling through a roadside WiFi network in a more explicit and accurate way while avoiding involving too many uncertain elements.

Ideally, we would like to have an economic deployment of APs that is able to serve mobile users with guaranteed performance in terms of some intuitive metric such as average throughput. Such an objective is complicated by various uncertainties in the system, such as unpredictable traffic conditions, unknown moving patterns of mobile users, and the dynamics involved in the performance of APs. To this end, we use an incremental approach. We first introduce a performance metric for roadside WiFi deployment that is closely related to average throughput while avoiding the above uncertainties such that an efficient solution can be obtained. In particular, we present a new metric, called Contact Opportunity [94], as a characterization of a roadside WiFi network. Informally, the contact opportunity for a given deployment measures the fraction of distance or time that a mobile user is in contact with some AP when moving through a certain path. Such a metric is closely related to the quality of data service that a mobile user might experience while driving on roads. We start with Contact Opportunity in distance (Section 2.4.1), which involves least amount of uncertainties, and present an efficient deployment method that maximizes the worst-case Contact Opportunity in distance under a budget constraint (Section 2.4.2). We then show how to extend this notion and the deployment techniques to Contact Opportunity in time by using an interval based road traffic modeling, and further
to the more intuitive metric – average throughput along a path – by taking various
dynamic elements into account (Section 2.4.3). In this section, we allow the coverage
region of an AP to have arbitrary shape.

2.4.1 Problem Formulation

We now define a performance metric for roadside deployment that does not require
any information about the dynamics of the system. Recall that $A$ denotes the set
of candidate locations in the 2-d region (not necessarily on the road network graph)
for deploying APs. Given a deployment $S \subseteq A$, the **Contact Opportunity in**
**Distance** of a path $p \in \mathbb{P}$, denoted as $\eta^d_p$, is defined as the fraction of distance on $p$
that is covered by some AP in $S$. Formally,

$$\eta^d_p(S) = \frac{\sum_{l \in L_p \cap L_S} d_l}{\sum_{l \in L_p} d_l}.$$  \hfill (2.1)

When a mobile user travels at a constant speed where each AP has the same data
rate, and there is only one user in the system, contact opportunity in distance can be
directly translated into average throughput that the user will experience. We show in
Section 2.4.3 how to extend this concept by taking care of various dynamic elements.
Given a budget $B$, we are looking for a deployment where the total cost of the APs
deployed does not exceed the budget, and the minimum contact opportunity over all
movements in $\mathbb{P}$ is maximized. Such a deployment provides a worst-case guarantee
and hence does not require statistics about the movement patterns of mobile users.
Furthermore, simulation results (in Section 2.5.1) show that even though our solution
is designed for the worst case, it also works well in the average case. Formally, if
we let $w(S)$ denote the cost of a deployment $S \subseteq A$, that is $w(S) = \sum_{a \in S} w_a$, the
optimization problem becomes

$$\max_{S \subseteq A} \min_{p \in P} \eta_p^d(S), \text{ subject to } w(S) \leq B.$$  \hfill (2.2)

### 2.4.2 Approximation via Submodular Set Cover

In this section, we first show that Problem (2.2) is an instance of a budgeted version of the submodular set covering problem first studied in [86] and recently extensively explored in [50] and hence allows an efficient bicriterion approximation. We then describe the detailed steps for solving the problem followed by various general and problem specific techniques for accelerating the computation. The following theorem summarizes our results in this section.

**Theorem 2.4.1.** Let $OPT(B)$ denote the minimum contact opportunity in distance of an optimal solution to (2.2). There is a polynomial time algorithm that finds a solution with the minimum contact opportunity to be at least $OPT(B/\gamma)$, where $\gamma$ is a logarithmic function of problem parameters defined below.

To show that (2.2) can be reduced to the budgeted submodular set covering problem, we note that the set function $\eta_p^d : 2^A \rightarrow [0,1]$ satisfies the following properties: (1) nondecreasing, i.e., $\eta_p^d(S) \leq \eta_p^d(T)$ whenever $S \subseteq T \subseteq A$, (2) normalized, i.e., $\eta_p^d(\emptyset) = 0$, and (3) submodular, i.e., for all $S \subseteq T \subseteq A$ and $a \in A \setminus T$, $\eta_p^d(S \cup \{a\}) - \eta_p^d(S) \geq \eta_p^d(T \cup \{a\}) - \eta_p^d(T)$. The last property is formally proved below, which essentially says that adding a new AP to a small set helps more than adding it to a large set. It captures our intuition that the total coverage that two APs can provide to a mobile user is reduced if their communication regions overlap with each other.
Lemma 2.4.1. $\eta_p^d$ is submodular.

Proof. For any $S \subseteq T \subseteq A$, $a \in A \setminus T$, $\eta_p^d(S \cup \{a\}) - \eta_p^d(S) = \frac{\sum_{l \in \mathcal{L}_p \cap (\mathcal{L}_{\{a\}} \setminus \mathcal{L}_S)} d_l}{\sum_{l \in \mathcal{L}_p} d_l}$, and $\eta_p^d(T \cup \{a\}) - \eta_p^d(T) = \frac{\sum_{l \in \mathcal{L}_p \cap (\mathcal{L}_{\{a\}} \setminus \mathcal{L}_T)} d_l}{\sum_{l \in \mathcal{L}_p} d_l}$. Since $S \subseteq T$, $\mathcal{L}_{\{a\}} \setminus \mathcal{L}_S \supseteq \mathcal{L}_{\{a\}} \setminus \mathcal{L}_T$. Therefore, $\eta_p^d(S \cup \{a\}) - \eta_p^d(S) \geq \eta_p^d(T \cup \{a\}) - \eta_p^d(T)$. □

It follows that (2.2) is an instance of the budgeted submodular set covering problem, which does not have a polynomial time approximation algorithm unless P=NP as shown in [50] using a reduction from the hitting set problem. The same argument can be applied to (2.2) with slight modification. Fortunately, an efficient bicriterion approximation can be achieved by relaxing both the requirement on the objective function and that on the budget. The solution framework proposed in [50] requires first solving the following variant of the problem, which is interesting by itself: given a required minimum contact opportunity $\lambda \in [0, 1]$ over all the movements, find a deployment of minimum cost. Formally,

$$\min_{S \subseteq A} w(S), \text{ subject to } \min_{p \in \mathcal{P}} \eta_p^d(S) \geq \lambda.$$    \hfill (2.3)

A binary search of $\lambda \in [0, 1]$ is then applied. For each $\lambda$, an instance of (2.3) is solved until a close to optimal solution to (2.2) is found. Although the budgeted version (2.2) is hard to approximate, the subproblem (2.3) allows an efficient approximation since it can be reduced to the submodular set covering problem as follows. Given $\lambda$, define:

$$\eta^d(S) = \sum_{p \in \mathcal{P}} \min\{\eta_p^d(S), \lambda\}$$ \hfill (2.4)

We note that $\eta^d$ is also a submodular function since (a) $\min\{\eta_p^d(S), \lambda\}$ as a set function on $A$ is submodular when $\eta_p^d$ is submodular [66] and (b) the sum of submodular
functions is submodular. Note that a subset $S \subseteq A$ is a feasible solution to (2.3) iff $\eta^d(S) = \eta^d(A) = |P|\lambda$. Therefore, (2.3) can be reformulated as a submodular set covering problem [86]:

$$\min_{S \subseteq A} w(S), \text{ subject to } \eta^d(S) = \eta^d(A)$$

(2.5)

Due to the submodularity of $\eta^d$, (2.5) allows an efficient greedy approximation. See Algorithm 2.4.1. The algorithm starts with an empty set and in each iteration picks and adds a new candidate location that is most cost effective until the required contact opportunity is achieved. This simple greedy procedure outputs a subset $S \subseteq A$, the cost of which never exceeds the cost of the optimal solution by more than a logarithmic factor [86]. In particular, by multiplying all $\eta^d_p$ by $10^n$ for some $n \geq 2$, $\eta^d$ can be made an integer valued function without loss of much accuracy, then an approximation factor $\gamma = O(1) + \ln(\max_{a \in A} \eta^d(a))$ can be achieved [86].

Algorithm 2.4.1 Minimum Cost Contact Opportunity

Input: $A, P, \lambda$
Output: A subset $S \subseteq A$

1: $S \leftarrow \emptyset$
2: while $\eta^d(S) < \eta^d(A)$ do
3: \hspace{1em} Find $a \in A \setminus S$ that maximizes $\eta^d(S \cup \{a\}) - \eta^d(S)$
4: \hspace{1em} $S \leftarrow S \cup \{a\}$

A binary search of $\lambda \in [0, 1]$ is then applied to solve (2.2). Let $B(\lambda)$ denote the total cost of a deployment that achieves $\lambda$ computed by Algorithm 2.4.1. Staring at $\lambda = \min_{p \in P} \eta^d_p(A)$, that is the minimum contact opportunity when all the candidate locations are used, if $B(\lambda) > B$, a lower $\lambda$ is selected; otherwise, a higher $\lambda$ is selected. The procedure continues until $B(\lambda) \leq B$ and $B(\lambda') > B$ for any $\lambda' : \lambda' - \lambda \geq \delta$, where
δ can be adjusted to control the accuracy. Given a budget $B$, such a binary search finds a subset $S \subseteq A$ that does not exceed the budget and has the minimum contact opportunity to be at least what an optimal solution with budget $B/\epsilon$ can achieve, that is, \( \min_{p \in \mathcal{P}} \eta^d_p(S) \geq \max_{T \subseteq A, w(T) \leq B/\epsilon} \min_{p \in \mathcal{P}} \eta^d_p(T) \). Furthermore, this result has shown to be hard to improve [50].

The above procedure can be naturally extended to the case of improving an existing deployment by adding new APs, by substituting all the evaluations of $\eta^d_p(S)$ with $\eta^d_p(S \cup A_0)$, where $A_0$ denotes the set of APs previously deployed.

**Techniques to Accelerate the Computation**

The number of different $\lambda$ values evaluated by the binary search procedure is $O(\log(\min_{p \in \mathcal{P}} \eta^d_p(A)/\delta))$, and for each $\lambda$, Algorithm 2.4.1 requires $O(|A|)$ iterations (line 2 to line 4) where each iteration involves $|A|$ evaluations of $\eta^d$. Hence the above solution requires $O(|A|^2 \log(\min_{p \in \mathcal{P}} \eta^d_p(A)/\delta))$ evaluations of $\eta^d$ and each evaluation involves computing $\eta^d_p$ for each $p \in \mathcal{P}$, which is very time consuming for a large road network, large $|A|$ and $|\mathcal{P}|$. We then describe several techniques that can be applied to accelerate the procedure.

- **Techniques to reduce the number of evaluations of $\eta^d$**: First, the accelerated greedy algorithm proposed in [63] is applied, which requires significantly fewer evaluations to find a candidate location that maximizes the marginal improvement in each iteration (line 3 in Algorithm 2.4.1) by making use of the submodularity of $\eta^d$. Second, whenever the subset found (line 4 in Algorithm 2.4.1) already violates the budget constraint, the procedure moves on to a new $\lambda$. 
Techniques to accelerate each evaluation of $\eta^d$: First, we note that for any path $p \in \mathbb{P}$, if $p$ can be divided at certain road intersection into two submovements $p_1, p_2 \in \mathbb{P}$ such that $\eta^d_{p_1}(S) \geq \lambda, \eta^d_{p_2}(S) \geq \lambda$, then $\eta^d_p(S) \geq \lambda$ as well. It follows that only a subset of $\mathbb{P}$, denoted as $\hat{\mathbb{P}}$, need to be considered in the evaluation of $\eta^d$. For instance, suppose $\mathbb{P}$ is the set of shortest paths in the road network graph of length at least $\alpha$, and $\kappa \ll \alpha$ where $\kappa$ denotes the maximum edge length in the graph, then $\hat{\mathbb{P}}$ contains mainly the shortest paths of length from $\alpha$ to $2\alpha$ since every shortest path of length greater than $2\alpha$ can be divided into shortest paths of length between $\alpha$ and $2\alpha$ with high chance. Second, we note that each candidate location only contributes to a small subset of $\hat{\mathbb{P}}$, and therefore an incremental calculation is more efficient, where $\eta^d(S \cup \{a\})$ is obtained from $\eta^d(S)$ by updating only $\eta^d_p$ for those $p$ covered by $C_a$.

We observe that these techniques improve the performance of our algorithm significantly in practice. For the $6 \times 6$ road network used in our simulations (in Section 2.5.1), the running time to find a solution to (2.2) is reduced from days to about half an hour under the same machine configuration.

2.4.3 From Contact Opportunity to Average Throughput

The concept of contact opportunity in distance discussed above ignores several complexities involved in a real system and does not correspond directly to the real performance a mobile user might experience. Therefore, we seek to design performance metrics that are more intuitive to mobile application designers and end users. To this end, the various uncertainties involved in the system need to be taken into
account. The strategy we use is to consider the worst case, and design for it whenever possible.

**Contact Opportunity in Time**

The first natural extension to contact opportunity in distance is to replace distance with time, which informally is defined to be the fraction of time when a mobile user moving through a path is within the range of some AP. The main difficulty with such a definition is that both the contact time and the travel time are not fixed due to the uncertainties of traffic conditions such as traffic jams, accidents and stop signs. To model these uncertainties, we follow the interval based modeling approach [49] and assume that for each \( l \in \mathcal{L} \), the possible travel time \( t_l \) over \( l \) varies within an interval \([b_l, u_l], 0 < b_l \leq u_l\), which can be learned from historical traffic data. We define a scenario \( k \) to be an assignment of travel time \( t_l(k) \in [b_l, u_l] \) to each \( l \in \mathcal{L} \). Let \( K \) denote the set of all possible scenarios. Given a deployment \( S \subseteq \mathcal{A} \) and a scenario \( k \in K \), the **Contact Opportunity in Time** of a path \( p \in \mathbb{P} \) is defined to be:

\[
\eta^t_p(S, k) = \frac{\sum_{l \in \mathcal{L}_p \cap \mathcal{L}_S} t_l(k)}{\sum_{l \in \mathcal{L}_p} t_l(k)}.
\]  

(2.6)

Although there are infinitely many scenarios, we seek to find a deployment that performs well even in the worst case. Fortunately, a worst-case scenario for a given deployment \( S \subseteq \mathcal{A} \) can be easily determined: let \( k_S \) denote the scenario where \( t_l(k_S) = b_l \) if \( l \) is covered by some AP in \( S \), that is if \( l \in \mathcal{L}_S \), and \( t_l(k_S) = u_l \) otherwise. We claim that \( k_S \) is a worst-case scenario for \( S \), that is:

**Proposition 2.4.1.** Every \( \eta^t_p(S, k), p \in \mathbb{P} \) is minimized across all the scenarios if \( k = k_S \).
Proof. For every $p \in \mathcal{P}$, $\eta^t_p(S, k) = \frac{t_1}{t_2 + t_1}$ where $t_1$ denotes the travel time over the set of subsegments in $p$ that are covered by $S$ and $t_2$ denotes the travel time over the other subsegments in $p$. Note that $\eta^t_p(S, k)$ is minimized when $t_1$ is minimized and $t_2$ is maximized, which happens at $k_S$.

Given a budget $B$, we would like to find a deployment that does not exceed the budget while maximizing the minimum contact opportunity in time over all the movements in the worst-case scenario. Formally, we want to solve

$$\max_{S \subseteq A} \min_{p \in \mathcal{P}} \eta^t_p(S, k_S), \text{ subject to } w(S) \leq B.$$  

(2.7)

Note that $\eta^t_p(S, k_S)$ can be viewed as a set function since whenever a deployment $S \subseteq A$ is given, $\eta^t_p(S, k_S)$ can be evaluated. However,

**Proposition 2.4.2.** $\eta^t_p$ as a set function is nondecreasing and normalized, but not submodular.

*Proof.* $\eta^t_p$ is normalized by definition. For every $S \subseteq T \subseteq A$, we have $\mathcal{L}_p \cap \mathcal{L}_S \subseteq \mathcal{L}_p \cap \mathcal{L}_T$ and $\mathcal{L}_p \setminus \mathcal{L}_S \supseteq \mathcal{L}_p \setminus \mathcal{L}_T$. Therefore, $\sum_{l \in \mathcal{L}_p \cap \mathcal{L}_S} b_l \leq \sum_{l \in \mathcal{L}_p \cap \mathcal{L}_T} b_l$ and $\sum_{l \in \mathcal{L}_p \setminus \mathcal{L}_S} u_l + \sum_{l \in \mathcal{L}_p \cap \mathcal{L}_S} b_l = \sum_{l \in \mathcal{L}_p \setminus \mathcal{T}} u_l + \sum_{l \in \mathcal{L}_p \cap \mathcal{T}} b_l \geq \sum_{l \in \mathcal{L}_p \setminus \mathcal{T}} u_l + \sum_{l \in \mathcal{L}_p \cap \mathcal{T}} b_l$.

Hence $\eta^t_p(S) = \frac{\sum_{l \in \mathcal{L}_p \setminus \mathcal{L}_S} b_l}{\sum_{l \in \mathcal{L}_p \setminus \mathcal{L}_T} u_l + \sum_{l \in \mathcal{L}_p \cap \mathcal{L}_T} b_l} \leq \frac{\sum_{l \in \mathcal{L}_p \setminus \mathcal{L}_T} b_l}{\sum_{l \in \mathcal{L}_p \setminus \mathcal{T}} u_l + \sum_{l \in \mathcal{L}_p \cap \mathcal{T}} b_l} = \eta^t_p(T)$. So $\eta^t_p$ is nondecreasing.

To see that $\eta^t_p$ is not submodular, it suffices to give a counter-example. Let $p$ be a path from $(0, 0)$ to $(3, 0)$ in $\mathbb{R}^2$ and there is no other road intersection in $p$ other than the two endpoints. Assume that there are three candidate locations at points $(i + 0.5, 0)$, $i = 0, 1, 2$, where each of them covers an interval $[(i, 0), (i+1, 0)]$, $i = 0, 1, 2$ in $p$, and $p$ is partitioned into 3 subsegments by them. Suppose the travel time of
each subsegment varies over $[1, 2]$. Let $S = \{(0.5, 0)\}$, $T = \{(0.5, 0), (1.5, 0)\}$, and $a = (2.5, 0)$. Then $S \subseteq T$, $a \notin T$. $\eta^p_p(S) = 1/(2 \times 2 + 1) = 0.2$, $\eta^p_p(S \cup \{a\}) = \eta^p_p(T) = 2/(1 \times 2 + 2) = 0.5$, and $\eta^p_p(T \cup \{a\}) = 1$. Therefore, $\eta^p_p(S \cup \{a\}) - \eta^p_p(S) = 0.3$, while $\eta^p_p(T \cup \{a\}) - \eta^p_p(T) = 0.5$.

It follows that the solution framework for (2.2) when applied here (by replacing $\eta^{d}_p(S)$ by $\eta^p_p(S, k_S)$) does not guarantee the same approximation factor.

We note that for any fixed scenario $k$, $\eta^p_p(S, k)$ is submodular and hence the above framework can still be applied. The general problem can be viewed as a robust optimization problem and is more challenging. In fact, it has been observed that the robust counterparts of many polynomial time solvable optimization problems are NP-hard [49]. Although an efficient solution with guaranteed performance to the general problem remains open, we propose the following two approaches as first steps that work well in many practical cases.

First, we study the potential loss if we only design for a fixed scenario. Let $k_0$ denote the “mean” scenario where $t_l(k_0) = (b_l + u_l)/2$, $\forall l \in \mathcal{L}$. It turns out that, if $u_l/b_l$ is small for all $l \in \mathcal{L}$, $k_0$ can be used as a good approximation. More concretely, let $S$ denote an optimal deployment with respect to $k_0$ and let $T$ denote an optimal solution to (2.7), we have the following theorem, where to simplify the notion, we write $\tilde{\eta}(S, k)$ to denote the minimum contact opportunity in time across all the movements under the deployment $S$ and scenario $k$. The theorem says that if $u_l/b_l$ is bounded by $\beta \geq 1$, then the minimum contact opportunity in time (with respect to the worst-case scenario) achieved by an optimal deployment computed using the “mean” scenario is at most a factor $\beta$ worse than the optimal solution.

**Theorem 2.4.2.** If $u_l/b_l \leq \beta$ for all $l \in \mathcal{L}$, then $\tilde{\eta}(S, k_S) \leq \tilde{\eta}(T, k_T) \leq \beta \tilde{\eta}(S, k_S)$.
Proof. First, we have \( \tilde{\eta}(S, k_S) \leq \tilde{\eta}(T, k_T) \leq \tilde{\eta}(T, k_0) \leq \tilde{\eta}(S, k_0) \), where the first inequality follows from the optimality of \( T \) with respect to worst-case scenarios, and the second inequality follows from the fact that \( k_T \) is a worst-case scenario with respect to \( T \), and the third inequality follows from the optimality of \( T \) with respect to \( k_0 \).

To show that \( \tilde{\eta}(T, k_T) \leq \beta \tilde{\eta}(S, k_S) \), it suffices to show that \( \tilde{\eta}(S, k_0) \leq \beta \tilde{\eta}(S, k_S) \). For every path \( p \in \mathbb{P} \), let \( \mathcal{L}_0 \subseteq \mathcal{L}_p \) denote the set of subsegments covered by \( S \). Let \( t_1 \) and \( t_2 \) denote the travel time over covered subsegments in scenario \( k_0 \) and \( k_S \), respectively, that is, \( t_1 = \sum_{l \in \mathcal{L}_0} (b_l + u_l)/2 \) and \( t_2 = \sum_{l \in \mathcal{L}_0} b_l \). Let \( T_1 \) and \( T_2 \) denote the total travel time over \( p \) in scenario \( k_0 \) and \( k_S \), respectively, that is, \( T_1 = \sum_{l \in \mathcal{L}_p \setminus \mathcal{L}_0} (b_l + u_l)/2 \) and \( T_2 = \sum_{l \in \mathcal{L}_p \setminus \mathcal{L}_0} u_l + \sum_{l \in \mathcal{L}_0} b_l \leq \sum_{l \in \mathcal{L}_p} u_l \). Then \( \eta^p_{t_1}(S, k_0) = \frac{t_1}{T_1} \), \( \eta^p_{t_2}(S, k_S) = \frac{t_2}{T_2} \), and \( \frac{\eta^p_{t_1}(S, k_0)}{\eta^p_{t_2}(S, k_S)} = \frac{T_1}{T_2} \). When \( u_l/b_l \leq \beta \) for all \( l \in \mathcal{L} \), we have \( \frac{t_1}{t_2} \leq \frac{1+\beta}{2} \) and \( \frac{T_1}{T_2} \leq \frac{2\beta}{1+\beta} \). Hence \( \frac{\eta^p_{t_1}(S, k_0)}{\eta^p_{t_2}(S, k_S)} \leq \beta \). Since this inequality holds for every \( p \in \mathbb{P} \), \( \tilde{\eta}(S, k_0) \leq \beta \tilde{\eta}(S, k_S) \). \( \square \)

We then propose a solution to the general problem without making assumptions on \( u_l/b_l \), which works well if the set of candidate locations that cover \( p \), denoted as \( A_p \), has small cardinality, for every \( p \in \mathbb{P} \). For each pair of path \( p \) and scenario \( k \), write \( \eta^p_{t,k}(S) = \eta^p_{t,k}(S, k) \), where we move \( k \) to the subscript to emphasize that we are now considering \( \eta^p_{t,k} : 2^A \to [0, 1] \) as a set function for each \((p, k)\) pair. Notice that the objective in (2.7) is equivalent to

\[
\max_{S \subseteq A} \min_{p \in \mathbb{P}, k \in K} \eta^p_{t,k}(S), \text{ subject to } w(S) \leq B. \tag{2.8}
\]

This formulation essentially says that we are looking for a deployment that maximizes the worst-case contact opportunity across all the movements as well as all the scenarios. Since each \( \eta^p_{t,k} \) is nondecreasing and submodular, we can apply the framework discussed before. The only issue is that the possible scenarios could be...
a huge set. The good news is that for each $p$, we only need to consider the set of scenarios $\{k_S : S \subseteq A_p\}$. The remaining scenarios are either not distinguishable to $p$ or will never be a worst-case scenario. Hence the total number of relevant $(p, k)$ pairs is $\sum_{p \in P} 2^{|A_p|}$. When $|A_p|$ is small (e.g. $< 10$) for every $p$, this formulation gives an efficient solution with guaranteed performance.

**Average Throughput for a Single Mobile User**

We next extend the concept of contact opportunity in time to average throughput for a single mobile user. To this end, we need to consider a more complex system model. First, we assume each access point deployed at a candidate location $a \in A$ has a fixed worst-case data rate $r_a \in \mathbb{R}^+$. Second, let $A_l \subseteq A$ denote the set of candidate locations that cover $l \in \mathcal{L}$. We assume that a user moving through $l$ always selects the AP with the highest rate among $A_l$ to associate with. Third, we assume that connection setup time when a user starts to associate with an AP and the handoff time when a user switches between APs are small enough to be ignored. Then for a given deployment $S \subseteq A$ and a traffic condition scenario $k$, the average throughput for a user moving through a path $p$ can be stated as follows:

$$\eta^*_p(S, k) = \frac{\sum_{l \in \mathcal{L}_p \cap \mathcal{L}_S} t_l(k) \max_{a \in A_l \cap S} r_a}{\sum_{l \in \mathcal{L}_p} t_l(k)}.$$  \hspace{1cm} (2.9)

We again want to find a deployment that is within the budget while maximizing the minimum average throughput in the worst-case scenario across all the movements. We note that $\eta^*_p$ is not submodular in general, but is submodular for any fixed scenario, and the solutions proposed for contact opportunity in time can be applied as well.
Average Throughput for Multiple Mobile Users

Finally, we discuss an approach to take multiple mobile users into account by making two additional assumptions. First, for any subsegment \( l \in \mathcal{L} \), the maximum number of mobile users moving through \( l \) at any time instant is known, and is denoted as \( \nu_l \), which can be learned from historical traffic data. For any candidate location \( a \in A \), let \( \nu_a \) denote the maximum number of users within \( C_a \), that is \( \nu_a = \sum_{l \in \mathcal{L}(a)} \nu_l \).

Second, define the normalized data rate of an AP deployed at \( a \in A \) to be \( \hat{r}_a = r_a / \nu_a \). We assume that a user moving through \( l \) always selects the AP with the highest normalized rate among \( A_l \) to associate with. Note that this is consistent with the scheme defined in the previous section where \( \nu_a = 1 \). With these assumptions, for a given deployment \( S \subseteq A \) and a traffic condition scenario \( k \), the estimated average throughput in the worst-case traffic condition scenario for each user moving through a path \( p \) can be stated as follows:

\[
\eta_p^m(S, k) = \frac{\sum_{l \in \mathcal{L}_p \cap \mathcal{L}_S} \max_{a \in A_l \cap S} \hat{r}_a}{\sum_{l \in \mathcal{L}_p} t_l(k)}.
\]  

(2.10)

Similar optimization problem as before can then be studied with respect to \( \eta_p^m \), which is submodular for any fixed scenario. This problem formulation, however, has some limitations. First, the simplified association protocol does not fully model the reality and hence the estimated average throughput could be far from the real throughput for any particular user. Second, fairness among mobile users has been ignored. We propose to study more sophisticated joint deployment and association models that take fairness into account as part of the future work.
2.5 Evaluation

2.5.1 Simulations

In this section, we evaluate our solutions to the roadside AP deployment problem via simulations using real road network data retrieved from the 2008 Tiger/Line shapefiles [1] to understand the performance of the two Alpha Coverage algorithms, the Connected Alpha Coverage algorithm, the minimum and average contact opportunity in distance and time, and the distribution of contact opportunity and average throughput across a set of movements under various budget constraints.

Generating road networks

We obtain road network data from the 2008 Tiger/Line shapefiles [1]. We only use the All Lines shapefile since it contains all the information about a road network we need. The database is organized by counties. Our simulations are based on the road network of the Franklin county in State of Ohio. The database does not contain information about one-way roads, so we only consider the undirected graph model of road networks. In the database, each road segment is a polyline that contains two intersections and zero or more interior shape points. We ignore the shape points and connect the two ends of a road segment by a straight line to reduce the size of the graph. Figure 2.6 shows four road networks used in the simulations. We have mapped the road networks to a 2D plane by Mercator projection to facilitate simulations. In each case, the largest connected component of the corresponding graph is used. The $6 \times 6 \, \text{km}^2$ dense network has 1802 road intersections and a diameter of 10.84 km. The $6 \times 6 \, \text{km}^2$ sparse network has 737 road intersections and a diameter of 11.64 km. The $10 \times 10 \, \text{km}^2$ dense road network has 4774 road intersections and a diameter of 18.5
The 10 × 10 km\(^2\) sparse road network has 2113 road intersections and a diameter of 19.7 km.

**Baseline Algorithms**

Let \( A \) denote the set of candidate locations and \( B \) denote the budget. A unit cost is assumed for each location throughout the simulation section. The following baseline algorithms are used in simulations:
• **Random Sampling** (Rand for short): A subset of $B$ elements is selected from $A$ randomly.

• **Grid Sampling** (Grid for short): The region spanned by the road network graph is first divided into a $50 \times 50$ grid, and each AP is then deployed at the vertex closest (among all the unoccupied vertices) to the center of a randomly selected grid cell. No repetition of vertices or cells is allowed.

• **Max-Min Distance Sampling** [78] (DS for short), which starts at a randomly selected location in $A$, and at each step finds a new element from $A$ that maximizes the minimum graph distance (in terms of shortest paths) from the elements already selected, until $B$ elements have been found.

• **Connected Max-Min Distance Sampling** (CDS for short), which extends the Max-Min Distance Sampling algorithm by also considering connectivity. Given a budget of anchor nodes and a set of gateway locations, $B$, the algorithm tries to maximize the minimum mutual (graph) distance between anchor nodes while satisfying the connectivity requirement. See Algorithm 2.5.1, where $d(a, S)$ denotes the minimum Euclidean distance between $a$ and any element in set $S$. Note that once the first node is selected, the entire deployment is fixed.

**Evaluating $\alpha_N$-Coverage**

The two $6 \times 6$ km$^2$ road networks are used for simulations in this and next sections. The $\epsilon$ in the approximated fractional multicommodity flow algorithm is set to 0.1. We observe that the size of a fractional multicut found is typically very close to 1.1.
Algorithm 2.5.1 Connected Distance Sampling (CDS)

Output: a subset $A \subseteq V(G)$ with $|A| = B$.

1: Find an arbitrary $a \in V(G)$ such that $d(a, B) \leq R$.
2: $A \leftarrow \{a\};$
3: for $i = 2 : B$ do
4:     $a \leftarrow \arg\max_{a' \in V(G) \setminus A \text{ and } d(a', A \cup B) \leq R} \min_{s \in A} \text{dist}(a', s);$ 
5:     $A \leftarrow A \cup \{a\};$

The percentage of the paths containing $\alpha$-pairs among all the selected paths is then counted. Figure 2.7 plots the results for various $\alpha$ values and for $\gamma = 1.5\alpha$ and $\gamma = 2\alpha$. For the dense network, the percentage decreases when $\alpha$ increases, which is consistent times the optimal one and is much better than the worst-case guarantee. The $\delta$ is Algorithm 2.2.3 is set to $0.1$ km. We first study how $\alpha_N$-coverage performances as an approximation to $\alpha$-coverage in real road networks. For each road network, a set of paths of length at least $\gamma$ is randomly selected in the following way, where $\gamma$ is a parameter. First a random source node is picked and a path of length 0 is initialized. At each step, the path is extended by one edge by selecting uniformly a neighbor of the current end of the path from all its neighbors not in the path. This process continues until either the length of the path is at least $\gamma$ and the path is added to the path set or all the neighbors of the current end of the path have been used and the path is discarded. Then another random source node is picked. The process is repeated $10^7$ times unless $10^6$ paths of length at least $\gamma$ have been found (repeating paths are allowed). For every such path, it is determined if there is an $\alpha$-pair on the path. Note that every path containing an $\alpha$-pair is covered by at least one AP in a deployment that provides $\alpha_N$-coverage.
with our analysis for grid graphs in Section 2.1.1. Such a trend does not exist in the sparse road network, where one can always find a large fraction of paths containing α-pairs, which implies that in this case $\alpha_N$-coverage provides a good approximation to $\alpha$-coverage if $\alpha$ is allowed to be relaxed by a factor of 1.5 or 2.

![Diagram](a) Dense network  (b) Sparse network

Figure 2.7: The percentage of the paths containing $\alpha$-pairs among all the selected paths of length at least $\gamma$.

We then evaluate the performance of the algorithms under various budget constraints. For a given budget, we first apply Algorithm 2.2.3 to find the minimum $\alpha$, denoted by $\alpha_{VC}$, that can be achieved for providing $\alpha_N$-coverage. We then generate 20 random deployments for each of the first three baseline algorithms and compute for each deployment (1) the minimum $\alpha$ that can be achieved by that deployment to provide $\alpha_N$-coverage by doing a search; and, (2) the percentage of uncovered N-equivalent $\alpha_{VC}$-pairs. We know all these pairs are covered in a deployment computed by our algorithm, but many of them may not be covered in a deployment computed using a baseline algorithm. Figure 2.8 shows the minimum $\alpha$ that is achievable by the VC algorithm and that by random deployments to provide $\alpha_N$-coverage. Figure 2.9
shows the percentage of uncovered N-equivalent $\alpha_{VC}$-pairs. For each of the three baseline algorithms, the average values over the 20 deployments are shown. Note that the $\alpha$ values are bounded by the network diameter. We observe that (1) our algorithm achieves much smaller $\alpha$ values than random deployments using the same budget; (2) the improvement of our algorithm decreases for larger budget, which is expected; (3) The grid based random sampling performs worse than the random sampling over road intersections for large budgets. This is because the underlying graph structure is ignored to a great extent in the former; and, (4) The max-min distance sampling performs worst in most scenarios under evaluation, especially when the budget is small (with respect to the graph size), which is expected as it tends to distribute APs in a more uniform way and therefore misses many short movements when the budget is relatively small.

![Figure 2.8](image)

(a) Dense network  
(b) Sparse network

Figure 2.8: The minimum $\alpha$ that is achievable for each budget to provide $\alpha_N$-coverage.
Figure 2.9: The percentage of N-equivalent $\alpha_{VC}$-pairs that are not covered by random deployment where $\alpha_{VC}$ corresponds to the values achieved by VC in Figure 2.8.

Evaluating $\alpha_P$-Coverage

We evaluate $\alpha_P$-coverage using similar metrics as for $\alpha_N$-coverage: (1) the minimum $\alpha$ that can be achieved by a random deployment for a given budget to provide $\alpha_P$-coverage by doing a search; (2) the percentage of uncovered P-equivalent $\alpha_{SC}$-pairs, where $\alpha_{SC}$ denotes the minimum $\alpha$ computed by Algorithm 2.2.3 to provide $\alpha_P$-coverage for a given budget. Figure 2.10 and 2.11 plot the results for the two metrics, respectively. We again see that our algorithm achieves a significantly smaller $\alpha$ using the same budget. We also observe that to achieve a similar $\alpha$ value using our algorithms, $\alpha_P$-coverage requires significantly fewer APs than $\alpha_N$-coverage since the latter has a more strict requirement on coverage.

For $\alpha_P$-coverage, we further compute the distribution of interconnection gap under a restricted random waypoint mobility model. For this purpose, we first generate a movement file, which is a sequence of 5000 randomly selected vertices, where the distance between two consecutive vertices are at least $\alpha_{SC}$, and then an imaginary move is carried out on the sequence of vertices selected where between two consecutive
vertices in the sequence, a shortest path is followed. The gap between any two consecutive contacts with the vertices where APs are deployed is then computed. We did not start simulations in steady state (perfect simulation) [14] since it is too time consuming for large graphs. Figure 2.12 shows the cdf of the interconnection gap under the random waypoint model. For the dense network, the case where budget equals to 150 ($\alpha_{SC} \approx 1.61$ km) is plotted. For the sparse deployment, the case
Figure 2.12: The cumulative distribution of the interconnection gap under a random waypoint mobility model. The maximum gap for our algorithm is bounded by twice of $\alpha_{SC}$ (see the text for an explanation.) The maximum gap for a random deployment (not shown in the figure) can be as large as 23 km for Rand, 37 km for Grid, and 67 km for DS in the dense network, and 33 km for Rand, 51 km for Grid, and 32 km for DS in the sparse network.

Figure 2.13: The standard deviation of the interconnection gap under a random waypoint mobility model. In the dense network, when budget is 50, the standard deviation of the max-min distance sampling is more than 48 (not shown).

where budget equals to 60 ($\alpha_{SC} \approx 1.55$ km) is plotted. For each baseline algorithm, the values are accumulated over all the 20 deployments. For the dense network, the maximum gap for our algorithm is about 3.1 km, which is larger than $\alpha_{SC}$. 
This is because of the moving pattern we use. Since each move follows a shortest path of length at least $\alpha_{SC}$, $\alpha_P$-coverage guarantees that at least one AP will be encountered within a move. However, an interconnection gap may span two adjacent moves, and hence the worst-case gap is bounded by twice of $\alpha_{SC}$. On the other hand, the maximum gap for a random deployment can be larger than the network diameter. Similar observations are also made with respect to the sparse network. Figure 2.13 shows the standard deviation of the interconnection gap under various budget constraints. We can clearly see that the standard deviation of our algorithm is much smaller than that of the baseline algorithms especially for a small budget. In all these simulations, we again see that the max-min distance sampling performs worst for most scenarios under evaluation.

**Evaluating Connected Alpha Coverage**

The two $10 \times 10$ km$^2$ road networks are used in simulations and the Connected Max-Min Distance Sampling Algorithm is used as baseline. The $\epsilon$ in the fractional cut algorithm is set to 0.1. We fix the communication range $R$ of anchor nodes to be 500m, and vary the value of $\alpha$ and the number and distribution of gateways. For a given number of gateways, 10 random gateway deployments are generated. For a given set of gateways, we first apply our algorithm to compute a deployment of anchor nodes. We then apply the CDS algorithm to compute 10 different deployments using the same number of anchor nodes.

We first study the performance of our algorithm. Figures 2.14(a) and 2.14(b) show the number of anchor nodes required to provide connected $\alpha_N$-coverage under various $\alpha$ values and gateway deployments. For a given number of gateway, the results averaged over all the 10 gateway deployments are plotted. We note that the
number of gateways has little impact on the number of anchor nodes used especially for small $\alpha$. This number is dominated by the coverage part of our algorithm, and the connectivity stage only introduces few extra anchor nodes. This is mainly because of the strict coverage requirement and the approximation factors in the algorithm. However, the number of gateways has a big impact on the average hops of each anchor node away from the closest gateway as shown in Figure 2.14(c), which in turn affects the communication delay significantly. We observe that when $R = 500$m and $\alpha = 2$km, 40 gateways are required to achieve an average hop distance of less than 3 in both networks.

We then compare the performance of the two algorithms. For a given value of $\alpha$, the CDS algorithm uses the same number of anchor nodes used by our algorithm to compute a deployment, which guarantees connectivity but not necessarily $\alpha_N$-coverage. The percentage of uncovered $\alpha$-pairs for a CDS deployment is shown in Figure 2.15(a). The results are averaged over all the gateway deployments. We then apply a binary search to find the minimum value of $\alpha$ that can be achieved by a CDS deployment. The results are shown in Figures 2.15(b) and 2.15(c). We observe that our algorithm achieves much smaller $\alpha$ compared with CDS using the same number of anchor nodes.

**Evaluating Contact Opportunity**

In this set of simulations, we use the dense $6 \times 6$ km$^2$ road network shown in Figure 2.6(a). Each edge of the road network graph is associated with an interval of travel speed: $[\tau - 5 \text{ m/s}, \tau \text{ m/s}]$, where $\tau$ is randomly selected from [10,20]. The coverage region at each candidate location is modeled using a simplified version of a sector based approach recently proposed in [74], where each region is composed
Figure 2.14: Connected $\alpha_N$-Coverage with various diameters and number of gateways. In (a) and (b), * denotes the number of anchor nodes required to meet the coverage requirement only, and the number of gateways are 1, 5, 10, 20, and 40, respectively. In (c), $\alpha$ is fixed to 2 km.

Figure 2.15: Comparing the Two-Stage Connected $\alpha_N$-Coverage (TSCC) algorithm with Connected Distance Sampling. The average results over all the gateway deployments are shown.

of 4 sectors of $90^\circ$ with radius randomly selected from [100m,200m], as shown in Figure 2.16. The more general model involving multiple sectors of different degrees and orientations can be easily taken into account. Each candidate location is assigned a unit cost. The set of movements considered are the set of all the shortest paths of length at least 2km between any two road intersections (there are about $1.4 \times 10^6$ such paths in the road network).
The following variants of our algorithms are used in the evaluation. For contact opportunity in distance, the algorithm presented in Section 2.4.2 is used. The parameter $\delta$ used in the binary search is set to 0.5%. For contact opportunity in time, the same algorithm is extended to consider time instead of distance and uses the “mean” traffic scenario based approach. We also extend the same algorithm to the average throughput case by again using the “mean” scenario, where the data rate of each AP is set to 1Mbps and a large movement file is generated (see the next paragraph) to estimate user density for multiple user case. For each of the baseline algorithms, 100 different deployments are evaluated and the average or cumulative results across all the deployments are plotted.

We further carry out $ns$-2 [68] based packet level simulations to evaluate the performance in terms of average throughput for both a single mobile user and multiple user scenarios. Each user moves in the network following a restricted random waypoint model composed of randomly selected shortest paths of length at least 2km, and the travel speed over each edge is randomly selected from the corresponding interval. We did not start simulations at steady-state since it is too time consuming for large
graphs. The physical layer data rate of each AP is set to 1Mbps. A mobile user selects one of the APs in range (if there is one) with minimum load to associate and then downloads CBR data (1Mbps) from that AP, where the load of an AP is the number of users currently associated with it. To avoid frequent switching between APs, a node switches from AP $a$ to $b$ only if the current load of $a$ is higher than the current load of $b$ plus 1, that is the new load of $b$ if the user switches to $b$. For the single user case, a movement file describes a continuous move for $10^4$ minutes (more than 1500 paths are traveled) is generated, and the deployment computed by our algorithm and the first 10 of the 100 deployments computed by each baseline algorithm are evaluated. For the multiple user case, a large movement file is first generated as the training data to estimate the worst-case user density in the coverage region of each candidate location. Due to the time required to simulate multiple user scenarios, we use a smaller road network ($2\, \text{km} \times 2\, \text{km}$ subregion in the same area) and simulate 5 mobile users where each user moves continuously for 1000 minutes.

Figure 2.17 shows the minimum and average (across all the movements) contact opportunity in distance and time achieved by our solution (Opp for short) and the two baseline algorithms under various budgets, where a budget is simply the number of APs allowed to use since each candidate location is assigned a unit cost. For contact opportunity in time, the value for the worst-case traffic condition is plotted. Our algorithm performs significantly better than the baselines in all the cases. The minimum contact opportunity is more than 200% higher. In fact, when the budget is low, the minimum contact opportunity of the two baseline algorithms is very close to 0. Even if our solution is designed for the worst case, it still achieves 30%-100% higher average contact opportunity. Furthermore, we observe that (not shown in the
Figure 2.17: The minimum and average contact opportunity (CO) across all the movements. For (c) and (d), the contact opportunity in time corresponding to the worst-case traffic condition is plotted. For the two baseline algorithms, the results are averaged over all the deployments.

Figure 2.17: The minimum and average contact opportunity (CO) across all the movements. For (c) and (d), the contact opportunity in time corresponding to the worst-case traffic condition is plotted. For the two baseline algorithms, the results are averaged over all the deployments.

We also observe that the max-min sampling performs better than random sampling in terms of minimum contact opportunity but worse in terms of average performance unless the budget is very high. One explanation is that the max-min sampling distributes APs in a more uniform way and when the budget is low, it does not provide enough coverage to short movements.
Figure 2.18: The complementary cumulative distribution of contact opportunity (for all the shortest paths $\geq 2$km) in distance and time. For (c) and (d), the contact opportunity in time corresponds to the worst-case traffic condition is plotted.

(a) in distance, budget = 100  (b) in distance, budget = 200  
(c) in time, budget = 100  (d) in time, budget = 200

Figure 2.19: The complementary cumulative distribution of average throughput (over every 400s interval). (a) and (b): a single mobile user in the large network; (c): 5 mobile users in the small network. The results are cumulative across all the users.

(a) a single user, budget = 100  (b) a single user, budget = 200  (c) 5 users, budget = 20
Figure 2.18 shows the complementary cumulative distribution of contact opportunity in distance and time for the three algorithms, and our algorithm performs clearly better. We then perform $ns$-2 simulations over the set of deployments used in Figures 2.18(c) and 2.18(d), and compute the average throughput that a single mobile user experiences for every 400 seconds interval, as shown in Figures 2.19(a) and 2.19(b). Even if the user only visits a small subset of all the shortest paths, our solution still performs much better than the baselines. Figure 2.19(c) shows the result for the 5 user case. We observe that the performance of different users are similar and hence the cumulative results across all the users are plotted.

2.5.2 A Small Scale Controlled Experiment

We set up a small scale controlled experiment to better understand the performance of our approach for optimizing average throughput. The experiment was carried out in a 180m × 120m parking lot located at the west campus of OSU and is free of potential interference from other WiFi networks. The experiment was usually carried out at night when the parking lot was empty. We artificially divided the parking lot area into a 6 by 4 grid and use it as a small road network. All the 24 intersections are treated as candidate locations for deploying APs.

A single mobile node carried by a car and 4 APs are used in the experiment. Each AP is a laptop equipped with an Orinoco 802.11b/g PC card and an external antenna mounted on a 1.7m high tripod so that the signal will not be blocked by the car in the test. The single mobile node is a laptop equipped with a Ubiquiti Networks SRC 802.11a/b/g PC card and two external antennas fixed at the two sides of the car. The transmission power of each AP is set to 6 dBm, which is tested to give an effective
transmission distance of no more than 50 meters. Each node runs Ubuntu Linux with Linux 2.6.24 kernel and madwifi device driver for the 802.11 interface. The physical layer data rate of each node is fixed at 54Mbps.

A total of 5 random deployments are evaluated and compared with a deployment computed by our algorithm that maximizes the minimum contact opportunity in distance across the set of shortest paths between intersections of length at least 200m (there are 30 such paths in total). The algorithm assumes that each AP has a unit cost and the coverage region of each AP is a disk of a radius 50m.

![Figure 2.20: The average throughput of the 6 paths under evaluation, where Rand represents the average of 5 random deployments.](image)

Because of the large volume of driving work and limited availability of that place, we picked 6 representative shortest paths that go through different parts and directions of the parking lot, and drove through each of them 3 times for each deployment. The moving speed is kept at about 10mph. When moving through a path, the mobile node attempts to associate with an AP with the strongest signal. Once associated, it
downloads UDP packets from the AP until it is disconnected from that AP. The mobile node then finds another AP with the strongest signal to associate. Figure 2.20 shows the average throughput of each of the 6 paths. For random sampling, the average results across the five deployments are plotted. We observe that our solution achieves up to 66.7% higher throughput, and across all the 6 paths the average improvement in throughput is 26.4%.
CHAPTER 3

SPARSE SENSOR DEPLOYMENT FOR TRACKING 2-D MOVEMENTS

3.1 Model, Contributions, and Related Work

3.1.1 Model

In this section, we list the key definitions and notations that we use throughout this chapter.

Let $\mathcal{A}$ denote the target region in the plane $\mathbb{R}^2$ and $S$ denote the set of sensor nodes deployed over $\mathcal{A}$. Let $C_S$ denote the region covered by $S$, that is, the union of the sensing regions of the nodes in $S$, and $U_S$ denote the uncovered region in $\mathcal{A}$, that is, $U_S = \mathcal{A} \setminus C_S$.

**Definition 3.1.1. Coverage hole, hole loop:** A coverage hole is a connected component of $U_S$. A hole loop is the outermost boundary cycle of a coverage hole (see Figure 3.1). Note that a hole loop may include part of the boundary of $\mathcal{A}$.

A sensor network providing Trap Coverage [9] guarantees that any moving object or phenomena can move only a (known) bounded displacement before it is guaranteed to be detected by the network for any trajectory and speed. The formal definition is as follows.
Figure 3.1: A coverage hole with its hole loop along the outermost boundary highlighted.

Figure 3.2: In this deployment, $d$ is the diameter of the largest hole. Notice that although the diameter line intersects with a covered section, it still represents the largest displacement that a moving object can travel without being detected.

**Definition 3.1.2. Trap Coverage:** A sensor network is said to provide Trap Coverage with diameter $d$ to $A$ if the diameter of any Coverage Hole in $A$ is at most $d$.

Figure 3.2 shows an example deployment region where the size of the largest uncovered region is $d$ units. Tracking applications aside, the model of trap coverage generalizes the de-facto model of full coverage itself, by allowing for holes of a given maximum diameter.
Figure 3.3: $s_2$ and $s_3$ are the sensing neighbors of $s_1$. The four sensing segments $ab$, $bc$, $cd$, and $da$ of $s_1$ are highlighted, where $da$ is a 1-covered sensing segment.

**Definition 3.1.3. Sensing neighbor, sensing segment:** A sensing neighbor of node $s$ is a node whose sensing region overlaps with the sensing region of $s$. Consider the perimeter of the sensing region of $s$, which is divided into one or more segments by the sensing perimeters of its sensing neighbors. Every such segment is called a sensing segment of $s$ (see Figure 3.3).

A sensing segment $w$ is said to be $k$-covered if every interior point of $w$ is covered by $k$ nodes for some $k \geq 1$. Note that a sensing segment is part of a hole loop if and only if it is 1-covered [42], and each hole loop is composed of contiguous 1-covered sensing segments.

**Definition 3.1.4. Sensing graph:** The sensing graph $G$ of a deployment $S$ has vertex set $S$ and edges between sensing neighbors, i.e., two sensors are joined iff their sensing regions intersect.
3.1.2 Summary of Contributions

In this chapter, we make the following contributions on the issue of deploying sensor networks in a large 2-d region to track movements and efficient routing for pairwise communication in sensor networks with holes.

- **Trap Coverage Verification**: We propose efficient algorithms for determining the level of trap coverage that an arbitrarily deployed sensor network provides [9]. The algorithms not only work for non-convex models of sensing regions, but they also work when sensing regions are uncertain. Further, they take into consideration the complications that may arise due to the boundary of the deployment region.

- **Hole Approximation**: We propose a core-set based approach for summarizing the boundary of a 2-d hole of roughly convex shape, where the size of the core-set is independent of the size of the original hole [93].

- **Hole Bypassing Routing**: By applying the hole approximation technique to communication holes in a large sensor network, we establish a distributed shortest-path roadmap based routing paradigm, which provides a bounded route-stretch and can make desired tradeoff between route-stretch and control overhead [93].

3.1.3 Related Work

Coverage: Coverage problem has been intensively studied for sensor networks [16, 42,53,54,89,92] and other types of wireless networks [76]. Various coverage models and deployment strategies (random versus deterministic) have been proposed. Full (area)
coverage requires every point in a continuous area to be covered \cite{42,54,89,92}, while point coverage only requires a set of discrete points to be covered \cite{61,76}. \cite{42} provides an efficient algorithm to determine the coverage level of a deployed sensor network. By covering every crossing path in a belt region, barrier coverage \cite{16,17,53} provides a scalable solution to intruder detection and border surveillance. One advantage of barrier coverage over full coverage is that localized algorithms can be designed \cite{16,17}. To extend the network lifetime while maintaining certain coverage quality, sleep-wakeup algorithms have been studied for various types of sensing coverage models \cite{37,54,55,89,92}, which are usually applied to large scale random deployment.

Work on full coverage that does consider holes focuses on the fraction of region that is (un)covered, see \cite{91}. They attempt to asymptotically minimize the area of vacant region and do not provide a closed form expression for the density needed in a random deployment to achieve a desired fraction of uncovered region. Perhaps, the work closest to trap coverage are \cite{23,37} that allow holes for surveillance applications. Here the quality of surveillance metric is based on the distance that a moving target, starting at a random location, moving in a random direction can travel in a *straight line* before it is detected by a sensor. In \cite{23}, distance to detection by a giant connected component is also studied. There are several issues with such metric. For one, they do not provide any worst case guarantee on how far a target can move before being detected, unlike trap coverage. For example, if the density chosen is just large enough that a giant component exists almost surely, as in \cite{23}, the hole diameters are not bounded by any constant; they grow as a function of \( \log(n) \) where \( n \) is the number of sensors deployed. Further, even though the average distance may be bounded (even close to zero), the worst case distance could be arbitrarily large.
Hole Discovery: Tools from both algebraic topology and computational geometry have been used for detecting coverage holes. Most focus on coverage verification and boundary node detection without computing the exact boundary of holes [11, 21, 33, 42, 82, 90], and several of them assume a disk sensing model and an open target field [11, 21, 33, 82, 90].

In topology based approaches, certain criteria to detect holes [33] or verify coverage [21] are derived from the topology of the covered region without using the positions of nodes. However, these criteria are computed in a centralized way and the complexity is not well studied yet. In contrast, geometry based approaches assume the positions of nodes are known [42, 82, 90] or at least the accurate distances among neighboring nodes are known [11] and use certain locally computable geometric objects to detect nodes on the coverage boundary. The first localized approach is proposed in [42] where every node can locally determine whether it is on the boundary of a $k$-coverage hole by counting the coverage levels of its sensing perimeter, which is simplified in the case of 1-coverage in [92]. The location free version of [42] is proposed in [11]. Another geometric approach uses Voronoi diagram [26, 82, 90]. The intuition is that if a point in the Voronoi cell of a node is not covered by that node, it is not covered by others, as well, which may not hold for non-convex sensing regions.

Based on [42], [79] proposes an algorithm to determine exact boundaries of coverage holes. However, it can only find those boundary with at most one piece from $\partial A$, such as $H_5$ and $H_6$ in Figure 3.4, and it assumes a disk sensing model. An algorithm to find the boundary of routing holes is proposed in [26], while [84] proposes a
method to determine the boundary of communication holes using only the communication graph and a general sensing model. However, $\partial A$ is not considered in either approach.

**Geographic Routing:** Geographic routing protocols typically only require nodes to maintain location information of the neighboring nodes, and are therefore highly scalable. Packets are forwarded toward the destination greedily which is efficient, however, may fail if it encounters routing holes (communication voids) where packets reach a node with no neighbors closer to the destination (also referred to as local minima). Various extensions have been proposed to guarantee delivery in presence of holes. GFG [13] and GPSR [47] guarantee delivery by using perimeter routing, or face routing, to recover from local minima. In GFG and GPSR, nodes forward their packets in greedy mode in the beginning. When a local minimum is encountered, it switches to recovery mode by routing around the hole. GOAFR+ [51] uses similar greedy forwarding and face routing concepts in geographic routing. Unlike GFG and GPSR, GOAFR+ bounds the search on the boundary of holes and therefore achieves asymptotic optimality. GPVFR [57] routes packet greedily to the node whose adjacent edge is closest to the destination on the same face as the sender. GPVFR maintains partial information about the nodes on the same face and uses this information to route packets around the face. However, maintaining complete face information is not a scalable approach.

There do have several recent works that deal with holes explicitly so that packets can bypass holes in advance without getting trapped [6, 45, 98], by filling the nodes in the concave regions of routing holes with higher cost metrics [98], or repositioning
these nodes by virtual coordinates [6], or propagating the local hole information to
neighboring nodes [45]. However, none of these works ensure a bound on route-stretch
and some of them [45, 98] only support “many to one” communications.

Our approach is different from all the above work in two key aspects. First, we
view routing holes as obstacles and propagate them proactively in a controlled way.
In contrast, most geographic routing protocols treat holes in a reactive way and only
try to bypass them when greedy forwarding fails, leading to high stretch. PAGER [98]
and NEAR [6] are among the few works that deal with holes explicitly in order to
reduce or eliminate the incidence of local minima, but they fail to provide bounds on
the route-stretch.

Second, all the above work focus on dealing with local minima or the non-convex
regions of routing holes, and apply greedy forwarding whenever possible. However,
we point out that with the existence of holes, even if there is no local minima in the
network, greedy forwarding can still be suboptimal. Our approach can be used to
improve even the greedy forwarding phase of existing protocols.

3.2 Trap Coverage Verification

In order to determine whether a deployed network continues to provide trap cov-
erage over time, efficient algorithms are needed to determine 1) whether a deployed
wireless sensor network provides trap coverage, and if so 2) what is the largest diam-
eter. In this work, we propose such algorithms.

3.2.1 Challenges

Figure 3.4 shows a target field with several sensing coverage holes. Although the
sensors are plotted as disks in the figure, we are not assuming a disk sensing model.
Further, the sensing regions of different sensors may be different. Except in Section 3.2.4, where sensing regions are assumed to be star convex, the only assumptions we make are: 1) Two sensor nodes are within the transmission range of each other if their sensing regions overlap; 2) The accurate positions of nodes can be learned; 3) The boundary of the target field is a simple polygon. Let $\mathcal{A}$ denote the target field and $\partial \mathcal{A}$ be its boundary.

To determine the largest diameter of coverage holes, the following two steps are applied. First, the boundary of each hole is found. Second, the diameters of these holes are computed based on their boundaries to obtain the largest diameter. The good news is that several ideas from existing work on discovering exact hole boundaries [11,42,79,82,90] can be applied here. However, the following challenges, which are critical to the trap coverage model, are not addressed there.

1. The boundary of a coverage hole may involve part of $\partial \mathcal{A}$, such as hole $H_7$ in Figure 3.4, so that it is hard to discover the entire boundary. This is especially true if our approach of overdeployment at the boundary is used [9].

2. In a realistic sensing model, the boundary of a coverage hole may have an arbitrary shape, which makes the computation of the accurate diameter non-trivial.

3. When the shapes of sensing regions are unknown or uncertain (as in probabilistic sensing models), the boundary of individual coverage holes may not be accurately determined.

We describe in Sections 3.2.2 and 3.2.3 a modification to existing algorithms that computes an accurate diameter for convex sensing regions and approximate diameter
for non-convex but known sensing regions. In Section 3.2.4, we describe an outline of a simpler algorithm that computes an approximate diameter for both known and unknown (uncertain) sensing regions.

3.2.2 Discovering Hole Boundary

Our algorithm first applies the perimeter coverage based approach [42] to detect nodes on the boundary of coverage holes. The idea is that the sensing perimeter of one node is divided into one or more sensing segments by the sensing perimeters of the neighboring nodes, see Figure 3.3. According to [42], a node is on the boundary of a coverage hole iff it has a 1-covered sensing segment – a sensing segment where each point on it (except the two ends of the segment) is within the sensing region of a single node, the node where the segment belongs to.

Recall that a coverage hole is a connected components of $U_S$ – the uncovered portion within the target region. A connected component of $C_S$ – the region that is covered – is called a coverage component. The key idea of our solution is that all the
coverage components are first discovered, which are then used along with $\partial A$ to find all coverage holes.

A coverage component may or may not contain hole(s). For instance, in Figure 3.4, $D_1$ has two holes while $D_2$ has no hole. The boundary of a coverage component is composed of its outermost boundary curve and the boundary of each hole in it.

**Definition 3.2.1. Coverage cycle:** A negative coverage cycle or negative cycle is the outermost boundary cycle of a coverage component. A positive coverage cycle or positive cycle is the boundary cycle of a hole in a coverage component. A coverage cycle is either a negative cycle or a positive cycle.

In Figure 3.4, $C_1, C_3, C_5, C_6, C_7, C_{11}$ are negative cycles, and the remaining ones are positive cycles. According to the observation made in [42], each coverage cycle is composed of contiguous 1-covered sensing segments.

To derive coverage holes from coverage components, we first notice that there is a partial order on the set of coverage cycles.

**Definition 3.2.2. Coverage cycle group:** For two coverage cycles $C_i$ and $C_j$, $C_i \preceq C_j$ if the region bounded by $C_i$ is included in the region bounded by $C_j$. $C_i$ is said to be directly enclosed in $C_j$ if $C_i \preceq C_j$ and there is no $k \neq j$ such that $C_i \preceq C_k \preceq C_j$. A coverage cycle group is a set of coverage cycles, where each group is either composed of one positive cycle and zero or more negative coverage cycles directly enclosed in the positive cycle (type 1), or composed of all the negative cycles not enclosed in any coverage cycles (type 2).
Figure 3.5: The partial order among all coverage cycles in Figure 3.4. The cycles at the same level are either all negative(-) or all positive(+). Each box represents a coverage cycle group. \( \{C_1, C_7\} \) is type 2, and rest are type 1.

Notice that there is only one group of type 2 in any deployment. Figure 3.5 shows the partial order defined on the coverage cycles in Figure 3.4. By enumerating all the possibilities, we have the following proposition.

**Proposition 3.2.1.** *Every coverage hole can be derived from a coverage cycle group and \( \partial A \).*

Actually, there are only four possibilities as shown in Figure 3.4. That is, a coverage hole can be derived from (i) a positive cycle, e.g. hole \( \mathcal{H}_5 \) and \( \mathcal{H}_8 \); (ii) a positive cycle and \( \partial A \), e.g. hole \( \mathcal{H}_6 \); (iii) one or more negative cycles and \( \partial A \), e.g. hole \( \mathcal{H}_4 \); (iv) a positive cycle, one or more negative cycles, and \( \partial A \), e.g. hole \( \mathcal{H}_1-\mathcal{H}_3 \), and \( \mathcal{H}_7 \).

It follows that once all the coverage cycles are discovered and their types are determined, all the coverage holes can be derived. The boundary of coverage holes needed for diameter computation is then derived based on the following observations, which can be verified in Figure 3.4. First, the boundary of a coverage hole is composed of one or more cycles, but its diameter is only determined by the outermost cycle,
called hole loop. For instance, the boundary of $H_3$ in Figure 3.4 has two cycles, but the inner cycle – the perimeter of the two overlapped sensing regions – can be safely ignored. Second, if a hole is completely contained in another hole, it can be ignored, such as $H_8$ in Figure 3.4. Third, each hole cycle is composed of sensing segments and (possibly) parts of $\partial A$. If it is composed of only sensing segments, the entire cycle can be found by traversing the nodes on it. Otherwise, each piece that is composed of only sensing segments on the cycle can be found. Once all the pieces of hole boundaries are known, a polygon clipper algorithm [58] can be extended to find the hole loops by also taking $\partial A$ into account.

### 3.2.3 Diameter Computation

Let $H$ denote a hole loop of a hole $H$, and $X_H$ denote the set of crossings on that loop, where a crossing is defined as an intersection point of either two sensing perimeters, or a sensing perimeter with $\partial A$, or a vertex of the simple polygon $\partial A$. The following lemma states that $X_H$ is a good approximation of $H$ in terms of the diameters, even if sensing region is not convex.

**Lemma 3.2.1.** $\text{diam } X_H \leq \text{diam } H \leq \text{diam } X_H + 2D$, where $D$ is the maximum diameter of all sensing regions. Moreover, if the sensing regions are convex, then $\text{diam } X_H = \text{diam } H$.

**Proof.** $\text{diam } X_H \leq \text{diam } H$ follows from the definitions directly. Let $x$ and $y$ be two points on $H$ with $\|x - y\| = \text{diam } H$, where $\| \cdot \|$ denotes the Euclidean distance. If $x$ is a crossing, let $x' = x$. Otherwise, let $x'$ be the crossing on $H$ closest to $x$. Choose a crossing $y'$ relative to $y$ in a similar way. Then we have $\|x - y\| \leq$
\[ \|x' - y'\| + \|x' - x\| + \|y' - y\| \leq \|x' - y'\| + 2D. \] Since \( \|x' - y'\| \leq \text{diam } X_H \), we have \( \text{diam } H = \|x - y\| \leq \text{diam } X_H + 2D. \)

If the sensing regions are convex, then \( H \) is contained within the convex hull of \( X_H \). Since a set and its convex hull have the same diameter, the result follows. \( \square \)

According to Lemma 3.2.1, when the sensing regions are all convex, it suffices to maintain the set of crossings on each hole loop instead of their accurate shapes in order to find the largest diameter. For arbitrary sensing regions, this also gives a good approximation when \( d \gg 2D \).

### 3.2.4 Coping with Sensing Region Uncertainty

Sensing regions show irregularity due to hardware calibration and obstacles and therefore are hard to characterize deterministically [43]. A more realistic way to characterize sensing regions is to use a sampling based approach, where the sensing region of a node is approximated by the discrete points corresponding to the events detected by this node [43]. In this section, we consider how to compute the largest diameter of coverage holes if only a limited number of samples are known. To this end, we first construct a planar graph based on the samples observed. This graph is used to approximate the real covered region, that is, the union of all the sensing regions. We then show that under certain assumptions, the largest diameter of coverage holes can by estimated by the largest diameter of the faces of this graph.

Let \( B_s \) denote the sensing region of node \( s \). We also use \( s \) to denote its position and \( e \) to denote the position where event \( e \) happened. We make the following assumptions.

1. The positions of nodes and events observed are known.
Figure 3.6: The approximation of covered region by a planar graph. The dashed lines show part of \( \partial A \). The dashed curves show the real sensing regions (unknown) of node \( s_1 \) and \( s_2 \). \( e_1 \) and \( e_2 \) are two events detected by both of them. \( a, c, \) and \( d \) are points on the edges of \( \partial A \) intersecting the two sensing regions, and \( b \) is a vertex of \( \partial A \). Three faces, \( as_1e_1ba, s_1e_1s_2e_2s_1, \) and \( s_1e_2s_2dcs_1 \) are shown.

2. Each \( B_s \) is a star convex subset of \( R^2 \) with respect to \( s \), that is, any line segment joining \( s \) to a point \( t \) in \( B_s \), denoted as \( st \), lies in \( B_s \). Figure 3.6 shows an example of two overlapped star-convex sensing regions.

3. For every connected component \( C_i \) of \( B_{s_1} \cap B_{s_2}, s_1 \neq s_2 \), there is at least one event detected in each \( C_i \), i.e., there is a point \( e_i \in C_i \) known such that \( s_1e_i \) lies in \( B_{s_1} \) and \( s_2e_i \) lies in \( B_{s_2} \). For instance, the two sensing regions in Figure 3.6 intersect at two connected subregions, with one common event detected in each subregion.

4. For each node \( s \), it is known whether \( B_s \) is completely in \( \partial A \), or completely outside of \( \partial A \), or intersects \( \partial A \). In the last case, for each edge of \( \partial A \) that intersects \( B_s \), at least one point on the part of the edge within \( B_s \) is known.

Let \( S \) denote the set of nodes whose sensing regions are within or intersect \( \partial A \), and \( E \) denote the set of events observed by nodes in \( S \). Let \( A \) denote the set of vertices of \( \partial A \). For each node \( s \in S \) and each edge of \( \partial A \) that intersects \( B_s \), pick
an arbitrary point on that edge that is within $B_s$, such as points $a$, $c$, and $d$ in Figure 3.6. Name the set of such points $I$. We construct a geometric graph $G(V, E)$, where $V = S \cup E \cup A \cup I$, and each edge in $E$ corresponds to either a line segment joining a node $s$ and an event $e$ detected by $s$, or a line segment on $\partial A$ joining points in $A$ and $I$, or a line segment joining a node $s$ and a point $a$ on an edge of $\partial A$ intersecting $B_s$. See Figure 3.6 for reference. Notice that, the edges of $G$ may intersect at points other than vertices. We make $G$ planar by treating these intersections as vertices as well. We then observe that $G$ is a planar graph without open faces. Let $D$ and $D'$ denote the largest diameter of coverage holes and that of the faces of $G$, respectively. Then under the assumptions made above, we have the following lemma.

**Lemma 3.2.2.** $D \leq D' \leq D + 2D$, where $D$ is the maximum sensing diameter.

**Proof.** $D \leq D'$ follows directly from the fact that $G$ is completely contained in the real covered region according to the second assumption made above. Consider a face of $G$, say $f$. $f$ either contains a coverage hole or is fully covered. First suppose there is a hole $H$ within $f$ and let $T$ denote the set of boundary nodes on the hole loop $H$ of $H$. Let $x$ and $y$ be two points on the face with $\|x - y\| = \text{diam } f$. Then $x$ and $y$ must be vertices of $f$. We argue that if $x$ is not on the hole loop, then it is covered by a node in $T$. Suppose this is not true. Then there is a subgraph of $G$ induced from the nodes in $T$ and a set of common events detected by them together with part of $\partial A$ that forms a polygon enclosing $H$ with $x$ outside of the polygon. However, this contradicts the fact that $x$ is vertex of a face that contains the hole. Similar argument applies to $y$ as well. It follows that there exist points $x'$ and $y'$ on the hole loop such that $\|x - x'\| \leq D$ and $\|y - y'\| \leq D$. Therefore, $\|x - y\| \leq 2D + \|x' - y'\| \leq 2D +$
diam $H$. That is, diam $f \leq 2D + \text{diam } H$. The case where $f$ is fully covered can be viewed as the degenerate case with diam $H = 0$.

Notice that, the above approximation can also be applied to the case where all the sensing regions are known. It is not as accurate as the approach discussed in Section 3.2.2, but more efficient since the faces of $G$ and their diameters can be easily computed. If $d \gg 2D$, the approximation may be desirable. In addition, if more events than required are detected, they can be used to improve the accuracy of the approximation.

3.3 Hole Bypassing Routing

3.3.1 Motivation

With emerging sensor applications where packets may originate from anywhere in the network and may be destined to any node, such as pursuer-evader tracking [22] and battlefield monitoring [88], pairwise communication between arbitrary sensors becomes an essential requirement in sensor networks. New functionalities such as in-network storage [71] also require communication between arbitrary nodes. Stateful routing protocols designed for multi-hop wireless networks can incur high communication and storage overhead (up to $O(n)$ per node in the worst case where $n$ is the number of nodes in the network), and therefore are not suitable for resource constrained sensor networks. Although stateless routing protocols based on geographic information have been proposed [13, 47, 51, 57], and perform well in networks with dense deployments, their performance can be severely impacted in presence of holes in the network. Large holes may exist in a network due to the presence of large
obstacles such as buildings or lakes, or a deployment strategy that allows holes as in the case of Trap Coverage.

Geographic routing protocols typically operate in two phases. A packet is greedily forwarded towards the destination until a “local minimum” is reached, where the forwarding node has no neighbor closer to the destination due to the incidence of a routing hole. A recovery phase is then followed to bypass the hole until the greedy phase can be continued. Due to the reactive routing decisions upon encountering a hole, the discovered path may be substantially longer than the shortest path in terms of the number of hops, especially when the hole is large. Figure 3.7 shows a scenario from our simulations. The path found by GPSR [47], a classic geographic routing protocol, has 66 hops. However if the hole is known by the nodes in its neighborhood and bypassed in advance, the path length can be lowered to 26 hops.

One of the critical metrics for routing in sensor networks is the route-stretch, which is the ratio of the number of hops on the computed route to the number of hops on the shortest route. The route-stretch is closely related to the end-to-end latency as well as the energy consumption. In Figure 3.7, as the “concave” regions of the hole can be arbitrarily deep, the stretch is unbounded for GPSR. Some recent works have dealt with holes explicitly so that packets can bypass holes in advance without getting trapped [6,45,98]. However, none of these works ensure a bound on route-stretch and some of them [45,98] only support “many to one” communications, where all the data packets are forwarded towards a single sink. The key observation is that the presence of a hole has to be made known to at least the nodes in the vicinity of the hole to bound the stretch.
Figure 3.7: Geographic routing can result in sub-optimal routes and high route-stretch. The dashed curve is the routing path of GPSR, and the solid curve is the routing path if the hole can be bypassed before its boundary is touched. The boundary nodes of the hole are highlighted.

We propose a distributed shortest-path roadmap based routing technique where each routing hole is treated as a polygonal obstacle, and explore two ideas, routing hole approximation and controlled advertisement, to support low-overhead critical information propagation for holes to assist in computing routes with bounded stretch. Instead of handling holes passively, we proactively advertise information on holes, but within a controlled region where route-stretch is most affected by the holes. Our approach is composed of two components:

- **Hole approximation and advertisement**: Each routing hole is approximated by its core, a simple polygon enclosed by the hole, controlled by a single parameter $\alpha$. Each routing core is then flooded to the $k$-hop neighborhood of the hole, where $k$ is proportional to the size of the hole.
• **Hole bypassing routing**: Each node sets up a shortest-path roadmap [56] locally by treating the cores it knows as obstacles, and makes routing decisions based on that map. The real path mimics the planned one and is realized by greedy forwarding and hole traversing, and is further optimized by a local strategy.

This is the first protocol that explicitly uses the information about holes to bound route-stretch. Furthermore, by controlling the level of approximation and advertisement, our protocol can make desired tradeoff between route-stretch and control overhead, which is rarely considered in previous works.

### 3.3.2 Roadmap Aided Geographic Routing

We assume in this work that each node knows the positions of itself and its 1-hop neighbors, and each routing hole \( H \) is a closed region bounded by a simple polygon \( \partial H \), where there is a node at each vertex, and two adjacent boundary nodes are within the transmission range of each other. Notice that, routing holes are not necessarily disjoint from each other and may share boundary nodes or edges. In the continuous domain where the network density is so high that we can assume that there is a node at every point in the target field, it is well known from motion planning [56] and computational geometry [20] that if every node \( s \) knows the complete boundaries of all routing holes, \( s \) can build a complete shortest-path roadmap locally by viewing routing holes as polygonal obstacles, and then for a given destination \( t \), \( s \) can find an optimal path to \( t \). Furthermore, an optimal path is composed of line segments connecting convex boundary vertices defined as follows.
Definition 3.3.1. **Convex vertex, concave vertex:** A convex (resp. concave) vertex of a polygon $P$ is a vertex of $P$ for which the interior angle is less (resp. greater) than $\pi$.

The main problem of applying this approach to a sensor network directly is its high message overhead due to the flooding of the complete hole boundaries to the entire network and the high storage overhead at each node to maintain a complete shortest-path map. The main idea of our approach is to approximate each routing hole $H$ with a core, a simple polygon $H_c$ enclosed by $H$ that satisfies special requirements defined precisely in Section 3.3.3. One of these requirements relevant now is that every convex vertex of $H_c$ must be a boundary node of $H$.

Cores are then advertised to its $k$-hop neighborhood where $k$ is determined by the size of the hole. Every node can build a shortest-path roadmap locally by viewing the cores it knows as obstacles. Since each core is contained in a routing hole, cores do not intersect with each other except possibly at boundaries. Furthermore, a path computed is composed of segments ending at the convex vertices of cores, which are boundary nodes by definition. Therefore, by substituting every such segment with a subpath implemented by either greedy forwarding or hole traversing, a realistic path can be constructed. We will show that such a hole approximation and advertising approach can significantly reduce the control overhead while still ensuring a desired route-stretch.

**Definition 3.3.2. Route-stretch:** For a given routing protocol $\mathcal{R}$, its route-stretch with respect to a source destination pair $(s, t)$ is $\rho(s, t) = |f_\mathcal{R}| / |f_{opt}|$, where $f_\mathcal{R}$ and $f_{opt}$ are the paths from $s$ to $t$ found by $\mathcal{R}$ and the shortest path, respectively, and $|f|$
is the length of $f$ in continuous domain or the number of hops of the path in discrete domain.

In this section, we present the main idea of the routing protocol, and assume that all the routing holes have been discovered, approximated, and advertised. These mechanisms will be presented in Section 3.3.3.

![Figure 3.8: A shortest-path roadmap and a path from $s$ to $t$ computed by HBR in the continuous domain. The three polygons are routing holes and the dashed polygons are their cores. The dotted segments are the bitangent edges of the map. The path computed using cores is $(s, v_0, v_2, v_4, t)$. The real path is highlighted. $x$ and $y$ are the intersections of line $v_2v_4$ with the boundary of the hole in the middle.]

Building Roadmaps

Once a node $s$ learns about a set of cores, it builds a shortest-path roadmap locally, which is defined as follows [56]. See Figure 3.8 for reference.

**Definition 3.3.3. Shortest-path roadmap:** A shortest-path roadmap at node $s$ consists of the set of the convex vertices $V_s$ of the cores that $s$ knows and the set of edges $E_s$. For any two vertices $a, b \in V_s$, $(a, b) \in E_s$ if $a$ and $b$ are visible to each
other, and either \((a, b)\) is an edge of a core or the line going through \(a\) and \(b\) is a bitangent line, that is, a tangent line at \(a\) and at \(b\) with respect to the cores they belong to.

The shortest-path roadmap at node \(s\) can be built in \(O(|V|^2 \log |V|)\) time, where \(V\) is the set of core vertices that \(s\) knows [56].

**Routing Protocol**

Consider an arbitrary source-destination pair \((s, t)\). Algorithm 3.3.1 shows how a packet \(p\) is forwarded from \(s\) to \(t\). See Figure 3.8 for reference.

---

**Algorithm 3.3.1 Hole bypassing routing**

1. Initialize: \(v \leftarrow s\).

2. \(v\) makes a routing plan locally to select the next convex vertex \(v'\) on the shortest path from \(v\) to \(t\).

3. \(p\) is forwarded towards \(v'\) as follows.

   (a) If \(v\) and \(v'\) are consecutive vertices of a core \(H_c\) for some hole \(H\), \(\partial H\) is traversed to reach \(v'\).

   (b) Otherwise, \(p\) is forwarded greedily towards \(v'\). If a hole \(H\) is touched before reaching \(v'\), \(\partial H\) is traversed to reach the node that is closest to the intersection of \(\partial H\) and \(vv'\) where \(v'\) is visible, then greedy forwarding continues.

4. Let \(v \leftarrow v'\) and repeat steps 2 and 3 until \(t\) is reached.

---

At step 2, the routing plan at node \(v\) is made as follows. First, the shortest-path roadmap is extended by connecting \(v\) and \(t\) to all the visible roadmap vertices. Second, Dijkstra’s algorithm is applied to find the first convex vertex on a shortest path from \(v\) to \(t\). At step 3(b), there are two possible directions when traversing \(\partial H\) is required. Suppose \(p\) touches \(\partial H\) at node \(x\). \(x\) will make a routing plan towards \(v'\).
using only the core of $H$ to determine which direction to go. If $xv'$ is disjoint from the core, the packet will follow the direction so that the real path is also disjoint from the core. For instance, in Figure 3.8, a packet forwarded by $x$ towards $y$ traverses the hole boundary in clockwise order, which is planned by $x$. To assist the routing protocol, each data packet header carries the locations and node IDs of the source, the destination, and the next route planning node (a convex vertex).

In the above protocol, a new routing plan is made at each convex vertex on a path from $s$ to $t$. This can be optimized as follows. First, when a planned path includes a sequence of contiguous convex vertices on the same hole, routing plans could be made only at the first and last convex vertices, and let the packet header carry the last one so that the intermediate convex vertices can simply forward the packet towards the last vertex. Second, a packet header could carry part of the routing plan (a sequence of convex vertices) made at the source or an intermediate node. Third, a node could cache the routing plans made for certain destinations. In this case, a protocol that handles outdated plans due to the changes of network topology is needed, which is left for future work.

### 3.3.3 Hole Approximation and Advertisement

In this section, we discuss how a routing hole is discovered, approximated by its core, and advertised in a controlled way.

**Hole Discovery**

We apply the approach proposed in [26] to discover routing holes in a network, which involves a local rule called TENT that finds nodes where packets may get stuck in greedy forwarding and the BoundHole algorithm that discovers routing holes
with stuck nodes on their boundaries. The nodes on the same hole boundary then cooperate to elect the node with the smallest node ID as the hole coordinator. That ID is also used as the hole ID. Each boundary node then sends a message containing its position to the coordinator. Furthermore, every boundary node keeps the positions of the two neighboring boundary nodes in each direction. The coordinator then approximates the hole boundary as discussed below.

**Hole Approximation**

In this section, we discuss how to approximate a routing hole by its core. Our approach can be viewed as a polygon simplification approach since each routing hole is bounded by a polygon. Although many work on polygon simplification have been done in computational geometry, they cannot be directly applied to our scenario because the approximation has to satisfy the following two properties to simplify the routing protocol and bound route-stretch:

1. A core is contained in the original routing hole, and its convex vertices must be the boundary nodes of the hole.

2. A core should be a good approximation of the original routing hole so that the route-stretch of our hole bypassing protocol can be bounded.

We will first consider how to approximate a routing hole bounded by a convex polygon, and then extend the approach to a general polygon. In both cases, we assume a routing hole has at least 4 boundary nodes since a polygon with less than 4 vertices cannot be simplified any further.
**Holes with Convex Boundary:** Consider a routing hole $H$ bounded by a convex polygon $\partial H$ with vertices $v_0, v_1, v_2, \ldots, v_{n-1}$ ($n \geq 4$) sorted in counterclockwise order.

The idea is to divide $\partial H$ into chains and replace each chain with a line segment. The approximation has only one parameter $\alpha$, which determines how vertices are grouped.

Let $\beta_{kk'} \in [0, \pi)$, $k' \geq k + 1^4$ denote the angle from line $v_kv_{k+1}$ to line $v_{k'-1}v_{k'}$ (see Figure 3.9). Algorithm 3.3.2 is named as $\alpha$-approximation, which begins at a vertex $v_0$ and traverses the hole boundary in counterclockwise order. See Figure 3.9 for reference.

![](image.png)

**Algorithm 3.3.2 $\alpha$-approximation of a convex hole $H$**

1. $H_c \leftarrow \{v_0\}$, $k \leftarrow 0$
2. repeat
3. find the largest $k'$ s.t. $k < k' \leq n$ and $\beta_{kk'} \leq \alpha$
4. $H_c \leftarrow H_c \cup \{v_{k'}\}$, $k \leftarrow k'$
5. until $k = n$

---

$^4$In this section, all the arithmetic operations on subscripts are modulo $n$ operations, and $v_n := v_0$. 

104
The time complexity of the algorithm is $O(n)$. The following proposition states that the size of a core in terms of the number of vertices on it is independent of the size of the hole, and is only determined by $\alpha$, which directly follows from the fact that the sum of the exterior angles of any convex polygon is $2\pi$.

**Proposition 3.3.1.** The size of the core of a convex hole obtained by $\alpha$-approximation is bounded by $\lfloor 2\pi/\alpha \rfloor$.

**Holes with non-Convex Boundary:** Consider a routing hole $H$ bounded by a non-convex polygon $\partial H$ with vertices $v_0, v_1, v_2, ..., v_{n-1} (n \geq 4)$ sorted in counterclockwise order. $\partial H$ can be divided into multiple disjoint interleaving convex and concave chains defined as:

**Definition 3.3.4. Convex chain, concave chain:** A sequence of vertices $C_{ij} = (v_i, v_{i+1}, ..., v_j)$ (without repetition) of a polygon $P$ is a convex (resp. concave) chain if every vertex in the chain is a convex (resp. concave) vertex, and the chain cannot be extended further to maintain the property.

The approximation of a non-convex $\partial H$ works as follows.

1. Apply $\alpha$-approximation to every convex chain of $\partial H$ starting at one end of the chain, with the additional requirement that the line segments used to replace the chain must lie in the core computed so far. Name the resulting polygon $P$.
2. Simplify every concave chain of $P$ (discussed below) to get $H_c$.

There are two things to be noted. First, a line segment $v_iv_j$ lies in a polygon $P$ iff (1) $v_iv_j$ is disjoint from any edges of $P$ except possibly at $v_i$ and $v_j$ and (2) for any point $x$ in the segment other than $v_i$ and $v_j$, $x$ lies in $P$. Both conditions can be
checked in $O(n)$ time. Therefore, the running time of the above procedure is $O(n^2)$.

Second, step 1 and step 2 may be applied alternately more than once to reduce the size of $H_c$.

To simplify concave chains, we recall that the concave vertices of a core are not part of the shortest-path map, and therefore a concave chain could be simplified without considering the error criterion – the worst case route-stretch in our scenario. Consider a concave chain $C_{ij}$. Let $w_{k,k'}, k' \geq k + 1$ denote the intersection of line $v_kv_{k+1}$ and line $v_{k'-1}v_{k'}$ (see Figure 3.10). Algorithm 3.3.3 embodies the similar idea of $\alpha$-approximation and outputs $C$, the simplification of $C_{ij}$. See Figure 3.10 for reference. The running time of the algorithm is $O(|C_{ij}|n)$ and the total time complexity of the approximation of a routing hole with non-convex boundary of size $n$ is therefore $O(n^2)$.

Figure 3.10: The approximation of a routing hole with one convex chain and one concave chain. $\alpha = \pi/2$ for the convex chain. The hole is simplified to the polygon with vertices $v_1, v_2, v_5, v_8, v_{10}, w_{10,1}$. 

106
Algorithm 3.3.3 Simplification of a concave chain \((v_i, ..., v_j)\)

1: \(C \leftarrow \{v_{i-1}\}, k \leftarrow i - 1\)
2: repeat
3:    find the largest \(k'\) s.t. \(k < k' \leq j + 1\) and both segments \(w_{kk'}v_k\) and \(w_{kk'}v_{k'}\) lie in the polygon computed so far
4:    \(C \leftarrow C \cup \{w_{kk'}, v_{k'}\}, k \leftarrow k'\)
5: until \(k = j + 1\)

Notice that for any concave chain \(C_{ij}, v_{i-1}\) and \(v_{j+1}\) must exist and must be convex vertices. Further more, \(v_{i-1}\) and \(v_{j+1}\) will not be removed by the approximation algorithm, which implies the following two properties about cores.

**Property 3.3.1.** The vertex set of a convex (resp. concave) chain of \(H_c\) is a subset of the vertex set of a convex (resp. concave) chain of \(H\).

**Property 3.3.2.** For any shortest path \(f\) computed using cores, if \(f\) is disjoint from a core \(H_c\), then either \(f\) is disjoint from the corresponding hole \(H\), or \(f\) intersects a single convex chain of \(H\).

**Controlled Hole Advertisement**

After \(H_c\) is computed, the coordinator of \(H\) computes \(C_H\), the minimum bounding circle of \(H\), which can be done in linear time [62]. Suppose \(C_H\) is centered at \(x\).

The coordinator then sends a message containing all the vertices of \(H_c\) sorted in counterclockwise order. The message is flooded to all the nodes within a big circle \(C'_H\) centered at \(x\) with radius \(R_H = \lambda p_H\), where \(\lambda \geq 1\) is a constant and \(p_H\) is the perimeter of \(H\), which equals to the size of \(H\) in the discrete domain. The impact of different choices of \(\lambda\) on the worst case route-stretch will be discussed in Section 3.3.4.

To reach obstructed regions within \(C'_H\), the flooding messages also hug the boundary of holes intersecting with \(C'_H\).
3.3.4 Worst-Case Stretch in Continuous Domain

In this section, we analyze the worst case route-stretch of HBR in the continuous domain, assuming routing holes are approximated using Algorithms 3.3.2 and 3.3.3.

**Hole Approximation:** we first assume that all the cores are advertised to the entire network, and show that the worst case route-stretch is only determined by $\alpha$. Consider an arbitrary source-destination pair $(s, t)$. Let $f$ denote the path from $s$ to $t$ found by HBR with $\alpha$-approximation and $f_{\text{opt}}$ denote the shortest path. We have the following theorem.

**Theorem 3.3.1.** $|f| \leq |f_{\text{opt}}|/\cos(\alpha/2)$.

*Proof.* Let $f'$ denote the shortest path planned using cores. First, we have $|f'| \leq |f_{\text{opt}}|$ since every core is contained in the corresponding routing hole. We show that $|f| \leq |f'|/\cos(\alpha/2)$, which implies the theorem.

It is known that $f'$ is composed of contiguous line segments ending at convex vertices of cores (we can view $s$ and $t$ as cores with only one vertex). By Algorithm 3.3.1 and Properties 3.3.1 and 3.3.2, $f$ is also composed of a sequence of segments, and every such segment $\omega$ corresponds to a line segment $\omega'$ in $f'$. Furthermore, there are only three cases where $\omega$ and $\omega'$ are different, as shown in Figure 3.11. It is sufficient to show that $|\omega| \leq |\omega'|/\cos(\alpha/2)$ for each $\omega$.

Without loss of generality, consider the case (c) in Figure 3.11, where $\omega = (x, v_1, v_2, v_3, y)$ is part of a convex chain and $\omega' = xy$. By geometric argument, we have $|\omega| \leq |xw| + |yw| \leq |\omega'|/\cos(\beta/2) \leq |\omega'|/\cos(\alpha/2)$. \qed

108
Controlled Hole Advertisement: In this section, we consider the impact of controlled advertisement on route-stretch. For an arbitrary source-destination pair \((s, t)\), we still use \(f\) to denote the path from \(s\) to \(t\) found by HBR, with both \(\alpha\)-approximation and controlled advertisement applied, and use \(f_{opt}\) to denote the shortest path. Let \(h\) denote the number of routing holes whose core intersects \(f\) but is not used in route planning. We have the following theorem.

**Theorem 3.3.2.** \(|f| \leq |f_{opt}|(1/cos(\alpha/2) + h/(\lambda - 1))\).

**Proof.** Let \(f'\) denote the shortest path planned using cores. First, we have \(|f'| \leq |f_{opt}|\) since only a subset of cores are used to make routing plans and every core is contained in the corresponding routing hole. Therefore, it is sufficient to show \(|f| \leq |f'|(1/cos(\alpha/2) + h/(\lambda - 1))\). Let \(\mathcal{H}\) denote the set of routing holes whose cores intersect \(f'\) but not used in route planning. \(|\mathcal{H}| = h\). Since each \(H \in \mathcal{H}\) increases the route by at most \(p_H\), combining with Theorem 3.3.1, we have \(|f| \leq |f'|/cos(\alpha/2) + \sum_{H \in \mathcal{H}} p_H\). For every \(H \in \mathcal{H}\), since \(f'\) intersects \(H\) and \(H\) is not known by the last route planning node before \(\partial H\) is touched, we have \(|f'| > \lambda p_H - r_H\) according to the advertising
Figure 3.12: Convex chain \((v_0, v_1, ..., v_4)\) is approximated by \(v_0v_4\). A packet is forwarded by \(v\) towards \(v_4\). (a) The optimization of HBR by early bypassing. \((v, x, v_1, v_2, v_3, v_4)\) and \((v, v_2, v_3, v_4)\) are paths found by HBR without and with optimization, respectively. (b) The optimization cannot be applied if the extreme node is not visible.

protocol, where \(r_H\) is the radius of the minimum bounding circle of \(H\). Since \(r_H \leq p_H\), we have \(|f'| > (\lambda - 1)p_H\). It follows that \(|f| \leq |f'|((1/\cos(\alpha/2)) + h/(\lambda - 1))\).

### 3.3.5 An Optimization of HBR

In this section, we consider an optimization of HBR motivated by the following observation. See Figure 3.12 for reference. Suppose a packet is forwarded by node \(v\) towards \(v_4\) on \(\partial H\). By Algorithm 3.3.1 proposed in Section 3.3.2, the packet will follow the path \((v, x, v_1, v_2, v_3, v_4)\) where \(x\) is the intersection of line \(vv_4\) with \(\partial H\). However, if \(v_2\) is visible to \(v\), a shorter path would be \((v, v_2, v_3, v_4)\).

The example also reveals a typical drawback of most geographic routing protocols that treat holes in a reactive way. It shows that even if there is no “local minima”, the simple greedy forwarding strategy may lead to a suboptimal path. To optimize this scenario, we first give the following definitions.
Figure 3.13: EVRs of a convex chain of hole $H$. For instance, regions I, II, and III are EVRs of $(v_2, v_3)$, $(v_2, v_4)$, and $(v_3, v_4)$, respectively.

Definition 3.3.5. Visible region of a convex chain: Given a convex chain $C = (v_i, v_{i+1}, ..., v_j)$, each line $v_kv_{k+1}(i \leq k < j)$ induces a half plane $A_k$ that is disjoint from the chain except at $v_kv_{k+1}$. The visible region $V_C$ of $C$ is defined as $\bigcup_{i \leq k < j} A_k$.

Definition 3.3.6. Extreme nodes: Given a convex chain $C = (v_i, v_{i+1}, ..., v_j)$ and a point $s \in V_C$, there are two vertices $v_k, v_{k'}, i \leq k < k' \leq j$ such that $v_k$ and $v_{k'}$ are visible to $s$ while $v_{k+1}$ and $v_{k'-1}$ are not, with respect to $C$. $v_k$ and $v_{k'}$ are called extreme nodes of $s$, with respect to $C$.

For instance, in Figure 3.12(a), $v_0$ and $v_2$ are extreme nodes of $v$; in Figure 3.13, $v_2$ and $v_3$ are extreme nodes of $a$. By the above definition, we make the following observation.

Property 3.3.3. Given a convex chain $C$, the two extreme nodes of a point $s \in V_C$ are separated by any straight line that goes through $s$ and intersects $C$.

For instance, in Figure 3.12(a), $v_0$ and $v_2$ are separated by line $vv_4$. 

111
Definition 3.3.7. EVR: Given a pair of vertices $v_i$ and $v_j$ of a convex chain, the connected subregion in the network in which every point has $v_i$ and $v_j$ as the extreme nodes is named as an extremely visible region (EVR) relative to $v_i$ and $v_j$.

The above optimization can be more precisely described as follows. Suppose the positions of each pair of vertices in any convex chain have been advertised to its EVR. How this is implemented will be discussed later. Suppose node $v$ needs to forward a packet to convex vertex $v'$ of hole $H$, the step 3 of Algorithm 3.3.1 is modified as follows.

(a) If $v$ and $v'$ are consecutive vertices of a core $H_c$ for some hole $H$, $\partial H$ is traversed to reach $v'$.

(b) Let $v_i, v_j$ denote the extreme nodes of $v$ with respect to the convex chain where $v'$ is on. If no such $v_i$ and $v_j$ exist, goto (c). Suppose in the chain, $v_j$ is closer to $v'$ than $v_i$. If $v_j$ is visible to $v$, then $p$ is forwarded greedily towards $v_j$.

(c) Otherwise, $p$ is forwarded greedily towards $v'$. If a hole $H$ is touched before reaching $v'$, $\partial H$ is traversed to reach the node that is closest to the intersection of $\partial H$ and $\overline{vv'}$ where $v'$ is visible, then greedy forwarding continues.

In step (b), $v$ needs to check whether $v_j$ is visible. Figure 3.12(b) explains the reason. If $\overline{vv_2}$ intersects another hole $H'$, then using $v_2$ does not necessarily give a shorter path. The following proposition states a way for node $v$ to check whether a convex vertex $v_j$ is visible.

**Proposition 3.3.2.** Suppose node $v$ knows the cores of all routing holes and all the extreme nodes. Then for a given convex vertex $v_j$ of hole $H$ and another routing hole $H'$, $v$ can locally check whether $\overline{vv_j}$ intersects $H'$ or not.
Proof. First if \( \overrightarrow{vv_j} \) intersects \( H'_c \), \( v \) can learn that since \( v \) knows \( H'_c \). Suppose \( \overrightarrow{vv_j} \) is disjoint from \( H'_c \) but intersects \( H' \). By Property 3.3.2, \( \overrightarrow{vv_j} \) must intersect a single convex chain of \( H' \). Then \( v \) knows the extreme nodes \( a \) and \( b \) on that chain. Furthermore, \( \overrightarrow{vv_j} \) must intersect \( \overline{ab} \) by Property 3.3.3. Therefore, \( \overrightarrow{vv_j} \) is disjoint from \( H' \) iff \( \overrightarrow{vv_j} \) is disjoint from both \( H'_c \) and \( \overline{ab} \). \( \square \)

We then describe a protocol that advertises extreme nodes to their EVRs. We first make the following observation.

**Property 3.3.4.** An EVR is a convex region with at most 4 edges without counting the boundary of the network.

According to this property, an EVR of a pair of vertex \((v_i, v_j)\) are completely defined by the four edges going through \(v_i\) and \(v_j\). See Figure 3.13 for reference. Therefore, a node can determine whether it is in the EVR of \((v_i, v_j)\) once it knows the two tuples \(\langle v_{i-1}, v_i, v_{i+1} \rangle\) and \(\langle v_{j-1}, v_j, v_{j+1} \rangle\).

Algorithm 3.3.3 outlines the advertisement protocol. Each advertisement contains two tuples defining the EVR where it should be received. Initially, every convex node broadcasts three messages (line 2). For instance, in Figure 3.13, \( v_3 \) will broadcast 3 messages to the subregions \( I \), \( II \), and \( III \), respectively. When a node receives an advertisement, it will first check whether it is within the specific EVR by examining the two tuples in the message. If it is not, the message is simply dropped (line 5). Otherwise, it stores the two tuples locally and then forwards the advertisement (line 9). A node at the common vertex of three or more EVRs may receive two advertisements. For instance, node \( a \) in Figure 3.13 will receive the advertisements.
from both $v_2$ and $v_3$. It can then figure out from the four tuples received that $v_1$ and $v_4$ are the extreme nodes of region $IV$, and broadcasts a new advertisement (line 8).

Algorithm 3.3.4 Local advertisement protocol at node $v$

1: if $v = v_i$ is vertex of a convex chain $C$ then
2: Advertise 3 messages: $m_1$: $(\langle v_{i-2}, v_{i-1}, v_i \rangle, \langle v_{i-1}, v_i, v_{i+1} \rangle)$, $m_2$: $(\langle v_{i-2}, v_{i-1}, v_i \rangle, \langle v_i, v_{i+1}, v_{i+2} \rangle)$, $m_3$: $(\langle v_{i-1}, v_i, v_{i+1} \rangle, \langle v_i, v_{i+1}, v_{i+2} \rangle)$
3: [v received a new advertisement $m$ from a chain $C$]
4: if $v$ is outside the EVR defined by the two tuples in $m$ then
5: drop $m$
6: else
7: if four tuples received from $C$ then
8: Advertise a new message (suppose the extreme nodes of the target EVR are $v_i$ and $v_j$): $m'$: $(\langle v_{i-1}, v_i, v_{i+1} \rangle, \langle v_{j-1}, v_j, v_{j+1} \rangle)$
9: Advertise $m$

The algorithm may fail if there is no node at the common vertex of multiple EVRs. For instance, in Figure 3.13, if there is no node at position $a$, then region $IV$ and all the other regions depending on the advertisement from $a$ will not be covered. To address this problem, the protocol allows each message to be sent to a small number of nodes outside the desired region by a small margin $\delta$. The desired value of $\delta$ depends on the network density. In addition, if a message touches the boundary of a routing hole, it also needs to be forwarded along that boundary.

3.3.6 Evaluation

Simulation Settings: We evaluated the HBR protocol using ns-2. 2000 nodes are randomly deployed in a $1000m \times 1000m$ area. The transmission range of each node is 40m and the average node degree is around 15. Besides the small holes formed randomly in a network, two types of big holes are artificially introduced: elliptical holes and rectangular holes with concave regions, as illustrated in Fig.3.14.
The semimajor and semiminor axes of an elliptical hole are uniformly distributed on $[a/2, a]$, and the length and width of a rectangular hole are uniformly distributed on $[3b/4, b]$ and $[9b/16, 3b/4]$, respectively, where $a$ and $b$ are parameters. For each type of holes, we first fixed the number of holes to be 2 and varied the size ($a$ or $b$) of holes. Then we fixed the size of all holes to be 300, and varied the number of holes. The positions of holes are randomly selected. Given the number, size, and shape of holes, 5 network scenarios are generated randomly, and the routes between all pairs of nodes are computed in each scenario. The results are averaged over all these scenarios. In all simulations, $\alpha$ is set to $\pi/2$, $\lambda$ is set so that the cores of the artificially introduced big holes are advertised to the entire network.

HBR is compared with GPSR [47] and GLDR [67]. GLDR is a virtual coordinates based routing protocol including a landmark selection algorithm and a greedy routing protocol based on landmarks, which ensures delivery in the continuous domain. The IDs of 8 historical extreme nodes are maintained in the packet header to detect loops in the discrete domain. When local minimum happens or a loop is detected, $L_1$-norm
and then $L_\infty$-norm are tried. If the destination is still not reachable, scoped flooding is performed. In our simulation, 25 landmarks are used on average. The routing success rate without flooding is about 95%. The average hop distance from the node where flooding is issued to the destination is about 3 hops.

**Route-Stretch:** Figure 3.15 shows the average route-stretches for the three protocols evaluated. We can see that in all the cases, HBR performs much better than GPSR. For elliptic holes, HBR performs better than both GPSR and GLDR. For rectangular holes, GLDR has a relatively stable stretch and performs better than HBR. This is because the performance of GLDR mainly depends on the distribution of landmarks and network density instead of the shapes of holes. However, GLDR achieves the low stretch by paying a relatively high overhead as discussed below. Furthermore, although Figure 3.15(b) shows that HBR can lead to an increasing stretch with increasing hole size, the stretch is bounded by $1/\cos(\alpha/2) = \sqrt{2}$ since $\alpha = \pi/2$.

**Message Overhead:** Figure 3.16 shows the normalized number of packet transmissions, which is defined as the number of transmissions for routing a packet using a specific routing protocol divided by that in shortest path routing, averaged over all source-destination pairs. For both GPSR and HBR, the values are the same as their stretches. For GLDR, the value is much higher because scoped flooding is used when destination is not reachable by greedy forwarding. Although this does not happen frequently, the impact is huge, especially for a network with high density.

Besides the high transmission overhead, GLDR also suffers from high overhead for each data packet because a big packet header is used to save the distances to the 10 addressing landmarks of the destination, and IDs of the last 8 extreme nodes
visited. In contrast, the packet header size of HBR can be much smaller as discussed in Section 3.3.2.

Figure 3.16: Normalized number of packet transmissions in routing procedure
4.1 The Thesis

In this dissertation, we have developed the foundation for Sparse Coverage to address the problem of deploying large scale wireless networks for mobile targets. We proposed three sparse coverage models, Alpha Coverage, Contact Opportunity, and Trap Coverage to enable the deployment of both city-wide roadside WiFi networks and sensor networks in large 2-d regions. These models share the common theme that the worst-case quality of service is guaranteed and is tunable against the cost of network deployment. For the first two models, we developed efficient approximation algorithms for optimizing the worst-case performance for a given budget on the available access points, and evaluated the algorithms by comprehensive simulations using real road network data and a small scale experiment. For the third model, we developed an algorithm for determining the exact coverage level for a given deployment and more efficient approximation algorithms that work for non-convex and uncertain sensing shapes. We also proposed a novel hole approximation algorithm and showed its application in deriving a geographic routing protocol that can trade off route-stretch and control overhead.
4.2 Future Work

4.2.1 Better Approximation for Alpha Coverage

Our algorithms for the two Alpha Coverage metrics have logarithmic factors. To achieve better approximation, we plan to proceed in the following two directions.

First, we would like to improve the approximation factors for achieving the two Alpha Coverage metrics for a large class of road networks by exploring the special properties of road network graphs, either by showing a tighter bound of our algorithms or designing better algorithms. One such property is *highway dimension*, an empirical characterization of road network graphs proposed very recently [4] (similar to the small-world models for social networks in some sense). It has been observed that many road networks have low highway dimension, and for such road networks, a sublinear query bound can be proved for several heuristics for answering shortest path queries. In fact, the definition of highway dimension is closely related to Alpha Coverage. A graph has low highway dimension if for every $r > 0$, there is a sparse set $S_r$ of vertices that touch every shortest path of length greater than $r$, and a set of vertices $S_r$ is called sparse if every ball of radius $O(r)$ (in terms of graph distance) contains a small number of elements from $S_r$. It seems that a graph with low highway dimension would allow $\alpha_p$-coverage of small cardinality for every $\alpha$.

Second, we will explore other approximations of the general notion of Alpha Coverage and other sparse coverage notions that are appropriate for roadside WiFi deployment. For instance, we can define the Maximum $\alpha_p$-coverage problem as follows: given a budget of APs and $\alpha$, find a deployment that covers as many shortest paths of length at least $\alpha$ as possible. This formulation satisfies submodularity and allows
a constant factor approximation. We are looking for more realistic problem formulations that can be approximated within a constant factor.

4.2.2 Network Design with Uncertainty

Design for the worst-case scenario, as we have done for roadside WiFi deployment, is probably the best that one can do if there is no prior information about the uncertainties involved in the system. Such a conservative approach, however, is often suboptimal with respect to real traffic. When we do have some information either before a new deployment or when an incremental deployment is needed, a better approach that can incorporate such information is needed. We hence propose to study the general network design problem under uncertainty. One promising approach is a two-stage stochastic formulation. In such a two-stage approach, the deployment cost involves two components, the total cost for initial deployment and the expected cost for incremental deployment which happens when the current service cannot meet the request anymore. For the deployment of a large scale wireless network that provides data service to mobile users, the main challenge is to model the user-AP association by taking the uncertainty of user mobility and signal state into account.

4.2.3 Open Problems for Trap Coverage

Although we have addressed the problems of algorithmic determination of the status of trap coverage as well as the estimate of critical density for achieving trap coverage in a random deployment [9], several fundamental deployment and topology control problems remain open. First, the problem of optimal deterministic deployment patterns for achieving trap coverage under various ranges of $d$ and $r$ largely
remains open, with or without connectivity requirement. Second, given a parameter $d$, an efficient algorithm is needed to select a minimum subset of nodes from a given deployment that achieves trap coverage with diameter $d$ to a target region of arbitrary shape. More generally, the problem of sleep-wakeup, which has been intensively studied for full coverage model and barrier coverage model, also needs to be reinvestigated for trap coverage.
BIBLIOGRAPHY


[34] Hector Gonzalez, Jiawei Han, Xiaolei Li, Margaret Myslinska, and John Paul Sondag. Adaptive Fastest Path Computation on a Road Network: a Traffic Mining Approach. In *Proc. of VLDB*, September 2007.


