ALGORITHMS FOR DISTRIBUTED MONITORING IN MULTI-CHANNEL AD HOC WIRELESS NETWORKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Donghoon Shin

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2012

Purdue University

West Lafayette, Indiana

Dedicated to my wife Misook Kim, my parents Sungjoon Shin and Bongsoon Lee,
and my brother Dongmin Shin for their love, support and encouragement

ACKNOWLEDGMENTS

I would like to first and foremost thank my advisor, Professor Saurabh Bagchi, for his guidance, support and encouragement during the entire of my Ph.D. years. His constant patience, motivation and enthusiasm as well as his insightful discussions and practical inputs tremendously helped my research. He also carefully taught me how to write good papers and give clear and impressive presentations. It has been my privilege and great pleasure to have him as my advisor and work with him.

I would like to thank Professor Ness B. Shroff, Professor Xiaojun Lin and Professor Chih-Chun Wang for serving my doctoral advisory committee. I am also privileged to be advised and taught by these wonderful professors. I am grateful to Professor Ness B. Shroff for his guidance and advice during the early stage of my thesis research. I was fascinated and motivated by his nice and elegant mathematical modeling of wireless communications and networking systems to pursue such kinds of research. I am thankful to Professor Xiaojun Lin for his guidance, advice, help and encouragement during the research projects that I have conducted under his guidance. Also, I learned a lot of mathematical modelings and theory on convex optimization through his courses. I am grateful to Professor Chih-Chun Wang for his guidance, advice and help during the senior years of my Ph.D. course. Without him, it would have been much more difficult for me to complete this thesis.

I am also grateful to the Dependable Computing Systems Laboratory (DCSL) and the Network Group lab-mates and friends. I have had many enthusiastic discussions and enjoyable moments with them. Finally, I would like to give my special thanks to Wallyholics who made my life at Purdue much more enjoyable.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Shin, Donghoon. Ph.D., Purdue University, August 2012. Algorithms for Distributed Monitoring in Multi-Channel Ad Hoc Wireless Networks. Major Professor: Saurabh Bagchi.

Ad hoc wireless networks are vulnerable to a wide range of security attacks, due to the ease of the nodes being compromised and the cooperative nature of these networks. A solution approach widely used for defending these networks is behavior-based detection. In this, nodes overhear communications in their neighborhood exploiting the open nature of the wireless medium, and determine if the behaviors of their neighbors are legitimate. An important issue with behavior-based detection that arises in multi-channel ad hoc wireless networks is on which channels monitoring nodes should overhear their neighbors' communications.

In this dissertation, we develop a framework for behavior-based detection in multi-channel ad hoc wireless networks. We are interested in the issue of how to optimally place monitoring nodes and to select channels to tune their radios to. We show that the problem is NP-hard, then develop approximation algorithms. We show that one of our algorithms attains the best approximation ratio achievable among all polynomial-time algorithms. Also, we develop distributed channel assignment algorithms for large-scale and dynamic networks. The distributed nature of the algorithm allows it to scale to large networks. Further, we allow for imperfect detection, where monitoring nodes may probabilistically fail to detect malicious behaviors. For this scenario, we consider providing multiple covers to each node, thereby still maintaining the detection accuracy above a certain level. We evaluate our algorithms for random and scale-free networks and consider optimizations for practical deployment scenarios,

such as when the network configuration is changing fast versus a relatively static network.

1. INTRODUCTION

Ad hoc wireless networks are vulnerable to a wide range of security attacks. An adversary can physically capture ad hoc nodes and tamper with them [1, 2]. This is because ad hoc nodes are often deployed in insecure locations, as the case for mesh routers deployed on rooftops or attached to streetlights [2], and for nodes deployed in a hostile environment (e.g., a battlefield) [3]. Further, the nodes often lack strong hardware protection. Once nodes are compromised, the adversary can launch a variety of attacks exploiting the cooperative nature of these networks. For example, the adversary can disrupt the network services by letting compromised nodes deny the network protocol such as the back-off rule at the MAC layer or the packet-relaying duty at the Network layer. Also, the compromised nodes can inject malicious traffic into the network (e.g., worm traffic into a sensor network [4]).

An approach widely used to detect this class of attacks is *behavior-based detection*. In this, nodes overhear communications in their neighborhood exploiting the open nature of wireless medium, and determine if the behaviors of their neighbors are legitimate. For instance, to detect the MAC-layer misbehavior, a node can verify if the back-off times of its neighbors follow a legitimate pattern. Also, to detect malicious traffic, a node can analyze the overheard packets to check if they contain any malicious data. In general, the behavior being monitored can be other than communication behavior, e.g., sensing behavior to see if sensed data is accurate. Upon detection, a remediation action can be taken, such as instructing intermediate nodes to drop malicious traffic by the detector nodes or isolating the misbehaving node by neighboring nodes.

Over the past few years, it has been extensively studied to use multiple channels in wireless networks, especially in wireless mesh networks (WMNs) [5–22]. It has been shown that equipping nodes with multiple radios tuned to different non-overlapping

channels can significantly increase the capacity of the network. In these multi-channel wireless networks, a key and challenging issue for accurate and timely behavior-based detection is to capture as large an amount of traffic or large a number of nodes as possible, ideally the entire, by judiciously placing a set of monitoring nodes in the network and also choosing channels to tune their radio(s) to.

In this dissertation, we develop a framework for behavior-based detection in multi-channel ad hoc wireless networks. This dissertation consists of three pieces of work on the optimal placement and channel assignment of monitoring nodes. We introduce them in the rest of this chapter.

1.1 OPTIMAL PLACEMENT AND CHANNEL SELECTION OF MONITORING NODES

In this first work, we study how to strategically deploy a given number of monitoring nodes in the network, and also which channels to tune their radios to. Here, we assume that a finite set of possible places (e.g., grid points) where the monitoring nodes will be deployed is given. Our goal is to maximize the number of nodes (or more generally, the amount of traffic) to be monitored by judiciously placing the monitoring nodes and assigning channels to them. We mathematically formulate this problem, and show that the problem is NP-hard with the computational complexity growing exponentially with the number of monitoring nodes. We then propose approximate solutions to solve the problem. In this work, our major contribution is to develop the best approximation algorithm that one can achieve for our problem. That is, our algorithm always achieves a factor of the optimal performance, i.e., the maximum detection coverage, and the approximation ratio is the best achievable for our problem among all polynomial-time algorithms. Also, we evaluate the proposed algorithm, in terms of the coverage and the execution time, through simulations in practical networks—random networks and scale-free networks.

1.2 DISTRIBUTED ONLINE CHANNEL ASSIGNMENT FOR MONITORING LARGE-SCALE NETWORKS

In this second work, we study an optimal channel assignment problem for passive monitoring in multi-channel wireless networks, where a set of sniffers capture and analyze the network traffic to monitor the network. The objective of this problem is to maximize the total amount of traffic captured by sniffers by judiciously assigning the radios of sniffers to a set of channels. This problem is NP-hard, with the computational complexity growing exponentially with the number of sniffers. We develop *distributed* and *online* solutions for large-scale and dynamic networks. Our algorithm preserves the same ratio while providing a distributed solution that is amenable to online implementation. Also, our algorithm is cost-effective, in terms of communication and computational overheads, due to the use of only local communication and the adaptation to incremental network changes. We present two operational modes of our algorithm for two types of networks that have different rates of network changes. One is a proactive mode for fast varying networks, while the other is a reactive mode for slowly varying networks. Simulation results demonstrate the effectiveness of the two modes of our algorithm.

1.3 OPTIMAL SNIFFER-CHANNEL ASSIGNMENT FOR RELIABLE MONITORING

In this third work, we study an optimal channel assignment problem for reliable monitoring in multi-channel wireless networks, where we allow for imperfect sniffers that may probabilistically generate errors on monitoring. In this scenario, we wish to still maintain the accuracy of the passive monitoring above a certain level. Our approach to this end is to provide multiple covers (i.e., sniffer redundancy) to each node. That is, each node is assigned a coverage requirement that is the minimum number of sniffers required for reliably monitoring the node. First, we are interested in a problem of how to assign a set of channels to sniffers' radios such that

the coverage requirements of all nodes are satisfied. We refer to this problem as the *Full-Coverage Reliable Monitoring* (FCRM). We, however, show that it is NP-hard to find any feasible solution to FCRM (i.e., any sniffer-channel assignment that satisfies all of the coverage requirements). Alternatively, we turn our attention to the corresponding optimization problem to FCRM, i.e., how to find a sniffer-channel assignment that maximizes the number (or the total weight) of nodes being reliably monitored. We refer to this problem as the *Maximum-Coverage Reliable Monitoring* (MCRM). However, MCRM is also NP-hard.

MCRM can be viewed as a generalization of the problems in the second work that assume perfect sniffers and thus do not need to consider the sniffer redundancy. However, we show that the generalized problem for reliable monitoring is different in nature from those of the previous works. As a results, in the generalized problem, the prior approximation algorithms no longer hold their performance guarantees. In this paper, we propose a variety of approximation algorithms based on two basic approachesgreedy approach and relaxation-and-rounding approach. We present a comparative analysis of the proposed algorithms through simulations. We evaluate the proposed algorithms in practical networks—random networks and scale-free networks—in terms of two metrics—detection coverage and running time.

2. OPTIMAL MONITORING IN MULTI-CHANNEL MULTI-RADIO WIRELESS MESH NETWORKS

2.1 INTRODUCTION

Wireless mesh networks (WMNs) are finding increasing usage in municipalities. Many cities (e.g., New Orleans, San Mateo, and Chaska) have already deployed WMNs for public service and safety personnel, and other cities, such as Philadelphia, Houston, and San Francisco, have planned city-wide WMN deployments for providing public broadband Internet access [23]. In WMNs, mobile devices connect to mesh routers, which are typically stationary devices, and mesh routers forward packets en route to the internet-conneted gateways.

WMNs are vulnerable to a wide range of security attacks that are more severe and easier to launch in these networks than in their wireline counterparts. An adversary can physically capture mesh routers and tamper with them. This is because mesh routers are often deployed in insecure locations (e.g., rooftops or streetlights), or even in a hostile environment (e.g., a battlefield). Also, they are typically low-cost devices, which lack strong hardware security protection [2]. Once mesh routers are compromised, the adversary can launch a variety of attacks with them exploiting the cooperative nature of WMNs among mesh routers. For example, the adversary can disrupt the network services by letting compromised mesh routers disobey the network protocols, such as the back-off rule for accessing channel at the MAC layer [24] and the packet-relaying duty at the Network layer.

An approach used to detect such attacks is *behavior-based detection*. In this, nodes overhear communications in their neighborhood via the open nature of wireless medium, and determine if the behaviors of their neighbors are legitimate. For instance, to detect the MAC-layer misbehavior, a node can verify if the back-off times

of its neighbors follow a legitimate pattern. Upon detection, a remediation action can be taken, such as isolation of the misbehaving node by neighboring nodes. A strategy proposed in the literature [25–27] to perform the behavior-based detection is to have *specialized monitoring nodes* deployed throughout the network. This takes the place of the more appealing architecture of every node participating in monitoring, because the latter is susceptible to framing of legitimate nodes due to erroneous reports by malicious nodes. Also, the quorum-based solution [28] only works well under relatively high network densities, which are unlikely in most WMN deployments.

Recently, the issue of use of multiple channels and also multiple radios in WMNs has been studied extensively (e.g., [6, 14, 17, 19, 29]). It has been shown that equipping nodes with multiple radios tuned to different non-overlapping channels can significantly increase the capacity of WMNs. An important issue that arises to defend these networks using behavior-based detection is how to strategically place a given number of monitoring nodes in the network and also which channels to tune their radios to, such that as large a fraction of normal nodes (i.e., the nodes that are not participating in monitoring) as possible are covered. One could have considered that, instead of tuning the radios of monitoring nodes to a fixed channel, we allow monitoring nodes to scan multiple channels by sensing multiple frequencies over time. However, the delay of switching the radio channel is non-negligible[1], and hence with this approach, monitoring nodes would waste their time switching channels.

Alternatively and also equivalently to the first problem formulation, one might be interested in a problem where, given a number of monitoring nodes deployed in the network, which monitoring nodes should be activated on which channels, in order to maximize the number of normal nodes covered. The latter problem is motivated by a desire to keep the resource consumption due to monitoring nodes at a low level. This is because the security analysis for behavior-based detection is computationally expensive and energy-intensive. Note that the former problem can be mapped to the

[1]Current estimate for switching delay between channels in the same frequency band with commodity IEEE 802.11 hardware is in the range of a few hundred microseconds [30] to a few milliseconds [31].

latter. To elaborate on this, assume that the network is arranged as a grid with a given number, say k, of monitoring nodes available for placement on any of m possible grid points. The former problem then becomes the following problem: how to choose k grid points on which to place the monitoring nodes and also the channels to which the radios of the monitoring nodes should be tuned, in order to attain the maximum coverage.

In this chapter, we first show that the maximum coverage problem in multi-channel networks, termed MCMC, is NP-hard with the computational cost growing exponentially with the number of monitoring radios in the network. We then present three approximation algorithms to solve MCMC. The first is a greedy algorithm, referred to as *GReedy Algorithm for MCMC* (GR-MCMC), and attains an approximation ratio of $\frac{1}{2}$. Here, the *approximation ratio* is defined as the minimum among all ratios of the number of normal nodes covered by an algorithm to the optimum, where the minimum is taken over all possible network instances. It is known that that the best possible approximation ratio achievable by any polynomial-time algorithm is $1 - \frac{1}{e} \approx 0.632$ (unless $P = NP$) [32]. Since the greedy algorithm cannot achieve the best approximation ratio, we explore further. The other two algorithms are based on Linear Program (LP) rounding technique (refer to Section 2.5). One called *Probabilistic Rounding Algorithm* (PRA) is a randomized algorithm, and achieves an *expected* approximation ratio of $1 - \frac{1}{e}$. Here, the expectation is taken over internal random coins of the algorithm. The other called *Deterministic Rounding Algorithm* (DRA) attains the best approximation ratio $1 - \frac{1}{e}$ in a *deterministic* manner, i.e., each time it runs, *regardless of the network topology and the channel assignment of normal nodes.* We conduct simulations in two kinds of networks—random networks and scale-free networks—and evaluate how the three algorithms—GR-MCMC, PRA, DRA—fare in these networks, in terms of detection coverage and execution time of the algorithms. A comparison of the three proposed algorithms is shown in Table 2.1.

The rest of the chapter is organized as follows. Section 2.2 describes the problem formulation. Section 2.3 discusses applications of the proposed algorithm. Section 2.4

Table 2.1: Performance comparison of the three proposed algorithms in this chapter.

	GR-MCMC	PRA	DRA
Approximation ratio	$\frac{1}{2}$	$1 - \frac{1}{e}$ (in expectation)	$1 - \frac{1}{e}$
Complexity	GR-MCMC < PRA < DRA		

shows NP hardness of our problem, and presents GR-MCMC. Section 2.5 introduces the LP rounding technique, and presents an overview of two LP rounding-based algorithms that we develop. The two LP rounding-based algorithms, PRA and DRA, are presented in Sections 2.6 and 2.7, respectively. Section 2.8 presents complexity analysis of the proposed algorithms. Section 2.9 presents performance evaluations of the proposed algorithms through simulation. Finally, Section 2.10 discusses conclusion.

2.2 PROBLEM FORMULATION

We are given a set of n normal nodes $u_1, \ldots, u_n$. Node u_i has a_i radios called *normal radios*. We define $U = \{u_1^1, \ldots, u_1^{a_1}, \ldots, u_n^1, \ldots, u_n^{a_n}\}$, where u_i^j denotes the radio j of normal node u_i. This set U defines the set of normal radios to be verified by monitoring nodes. Each normal radio is tuned to a specific wireless channel. Each normal radio u_i^j has a non-negative weight w_{ij}. These weights of normal radios can be used to capture various application-specific objectives of monitoring. For example, one can use the weights to capture transmission rates of normal radios, which can be estimated from historical data. In this scenario, we would assign higher weights to the nodes that transmit larger volumes of data, thereby biasing our algorithm to monitor such nodes more. Or, one can assign a weight to each normal node instead of each normal radio, taking into account their trustworthiness computed based on previous monitoring results. In this case, all radios of a given normal node will have the same weight, i.e., $w_{i1} = \cdots = w_{ia_i}$ for all i. A normal node that has been found to be compromised before (and repaired thereafter) will be assigned a higher

weight thereby indicating to our algorithm that it is more important such a node be monitored. This may be because the node is placed in a location where it is more apt to be compromised. We are given a set of m monitoring nodes $v_1, \ldots, v_m$. Monitoring node v_i has t_i radios called *monitoring radios*. Each monitoring radio can be tuned to a channel $j \in [c]$, where $[c] = \{1, \ldots, c\}$ and c is the number of available wireless channels. We say that a normal radio is *covered* by a monitoring radio if the latter can overhear the former's communication when it is tuned on the same channel. We are given a collection of subsets of U, $\mathcal{S} = \{S_{ij} : i \in [m], j \in [c]\}$, where a *coverage-set* $S_{ij} \subseteq U$ contains the normal radios that can be covered by any radio of monitoring node v_i tuned on channel j. We will use the term "set" as a shorthand for "coverage-set" whenever we can do so without loss of clarity. We denote $\mathcal{S}_i = \{S_{ij} : j \in [c]\}$ as a *group*, which is the set of normal nodes that can be covered by a monitoring node v_i if it had as many radios as the number of channels.

Our objective is to maximize the total weight of the normal radios covered by judiciously choosing at most k sets from $\mathcal{S}$ with at most t_i sets from group $\mathcal{S}_i$. The former constraint of at most k sets means that we can choose at most k monitoring radios for verifying normal radios. We call this constraint the *total budget constraint* (TBC). TBC is motivated by a desire to keep the resource consumption for verifying normal nodes at an appropriate level. The latter constraint of at most t_i sets from group $\mathcal{S}_i$ is due to the fact that monitoring node v_i has t_i radios and therefore t_i is the maximum number of sets that can be selected from the group $\mathcal{S}_i$. We call this constraint the *group budget constraint* (GBC). If k_i ($\leq t_i$) sets $S_{ij_1}, \ldots, S_{ij_{k_i}}$ in group $\mathcal{S}_i$ are selected for a solution by any one of the algorithms presented by us here, then k_i radios of v_i will be tuned to the channels $j_1, \ldots, j_{k_i}$, respectively. We refer to this problem as the *Maximum Coverage problem* with *Multiple Channels* (MCMC). We refer to a special case of MCMC where all nodes (normal and monitoring nodes) have a single channel and a single radio (i.e., MCMC with $c = 1$, $a_l = 1$ for all $l \in [n]$, and $t_i = 1$ for all $i \in [m]$) as the *Maximum Coverage problem* with *Single Channel*

Table 2.2: Summary of notation

Notation	Definition
U	Set of radios of normal nodes
n	Number of normal nodes
a_i	Number of radios that normal node u_i has
w_{ij}	Weight assigned to normal radio u_i^j (i.e., normal node u_i's radio j). A higher weight implies that it is more important for a monitoring node to cover this radio.
m	Number of monitoring nodes
t_i	Number of radios that monitoring node v_i has
c	Number of wireless channels
S_{ij}	Coverage-set of normal radios that can be covered by a radio of monitoring node v_i tuned to channel j
k	Maximum number of monitoring nodes that can be activated
x_{ld}	Indicator variable assigned to normal radio u_l^d. A value of one indicates that this normal radio is covered by at least one monitoring radio.
y_{ij}	Indicator variable assigned to coverage-set S_{ij}. A value of one indicates that monitoring node v_i has a radio tuned to channel j.

(MCSC). For convenience, the definitions of frequently used symbols are presented in Table 2.2.

We would like to point out that one could consider an alternative TBC on the number of monitoring nodes, i.e., we can choose at most k monitoring *nodes*. This alternative problem can, in fact, be formulated into MCMC by redefining the coverage-sets with each containing normal nodes covered by all radios of a monitoring node

tuned to a set of channels. Specifically, we redefine coverage-set S_{ij} as the set of normal nodes that can be covered by all of the t_i radios of monitoring node v_i with channel assignment j. Then, each v_i has $\binom{c}{t_i}$ coverage-sets, where $\binom{n}{c}$ denotes the number of ways in disregarding order that c objects can be chosen from among n objects, for all possible reasonable channel assignments for its radios, since it is inefficient to tune two radios of a monitoring node to the same channel. Although $\binom{c}{t_i}$ grows exponentially with t_i, in practice, it will not be large since t_i (the number of ratios that v_i has) is a small number, typically 2 or 3. With these redefined coverage-sets, we can formulate the alternative problem into the following problem: how to choose at most k coverage-sets from $\mathcal{S}$ with at most one from each group, which is an instance of MCMC.

2.3 APPLICATIONS

Although the mathematical problem that this chapter studies (i.e., MCMC in Section 2.2) is motivated by the issue of the deployment of security monitoring nodes in multi-channel multi-radio WMNs, it also captures the issue of the channel assignment for generic passive monitoring in multi-channel wireless networks. Passive monitoring is a widely-used and effective technique to monitor wireless networks, where a set of sniffers (i.e., software or hardware devices that intercept and log packets) are used to capture and analyze network traffic between other nodes to estimate network conditions and performance. Such estimates are utilized for efficient network operation, such as network resource management, network configuration, fault detection/diagnosis and network intrusion detection. A major challenge with passive monitoring in multi-channel wireless network is how to assign a set of channels to each sniffer's radios such that as large an amount of traffic, or large a number of nodes, as possible are captured. This sniffer-channel assignment problem is a special case of MCMC with all monitoring nodes being activated.

In practical applications, the algorithms proposed in this chapter can be utilized by employing a centralized network entity to determine the configuration of the monitoring nodes, i.e., the activation and the channel assignment of the monitoring nodes' radios. The centralized network entity first obtains the global knowledge by gathering from each sniffer the information of the channel usage of normal nodes, then runs the algorithm to determine the configuration, and distributes the configuration to each monitoring node. This mode of operation is particularly feasible for networks where the network configuration changes slowly with time.

2.4 NP-HARDNESS OF MCMC AND GREEDY APPROXIMATION ALGORITHM

2.4.1 NP-HARDNESS OF MCMC

Lemma 2.4.1 *MCMC is an NP-hard problem.*

Proof MCMC can be reduced to the *maximum coverage problem*[2] by setting $c = 1$, $t_i = 1$ for all $i \in [m]$. Hence, if the optimal solution to MCMC can be determined in polynomial time, then the maximum coverage problem can also be solved in polynomial time, which is a contradiction unless $P = NP$. ■

Hence, we will alternatively find an approximate solution that can be obtained in polynomial time.

Definition 2.4.1 *For a maximization problem, we say a polynomial-time algorithm to be a* δ-approximation algorithm *if for any instance of the problem, the algorithm yields a solution whose quality is at least* δ *times the optimum. Here,* δ *is referred to as the* approximation ratio.

[2]Given a set $U = \{1, \ldots, n\}$ with associated non-negative weights $\{w_1, \ldots, w_n\}$ and a collection of subsets of U, $\mathcal{C} = \{C_1, \ldots, C_m\}$, the *maximum coverage problem* is to select k of these subsets such that the total weight of the elements in the union of the selected subsets is maximized. This problem is known to be NP-hard [33].

Naturally, $\delta < 1$, and the closer δ is to 1, the better.

In the rest of this chapter, we seek to find answers to the following questions for MCMC:

1) What is the best approximation ratio attainable?

2) How can it be achieved through a realizable algorithm?

2.4.2 GREEDY APPROXIMATION ALGORITHM FOR MCMC

We first consider MCSC, which is a special case of MCMC but still an NP-hard problem since it is exactly the maximum coverage problem. It is known that a simple greedy algorithm solves MCSC within a factor of $1-(1-1/k)^k$ of the optimum [33], where k is the maximum number of monitoring radios that can be selected. We term this greedy algorithm as GR-MCSC. GR-MCSC selects k sets from $\mathcal{S}$ iteratively by picking the set of the maximum total weight of uncovered normal nodes at each iteration. It has been proven that no polynomial-time algorithm can achieve a higher approximation ratio than $1-\frac{1}{e}$ (≈ 0.632) provided that $P \neq NP$ [32]. Thus, the following lemma holds.

Lemma 2.4.2 *GR-MCSC is the* best *approximation algorithm for MCSC unless* $P = NP$.

Proof It follows from that $1-(1-1/k)^k > 1-\frac{1}{e}$ since $\lim_{k\to\infty}[1-(1-1/k)^k] = 1-\frac{1}{e}$ and $1-(1-1/k)^k$ is a decreasing function of k [33]. ■

We generalize the greedy approach of GR-MCSC to MCMC, and propose our first algorithm, *GReedy algorithm for MCMC* (GR-MCMC)[3]. GR-MCMC operates similarly to GR-MCSC, except that once t_i sets are selected from group $\mathcal{S}_i$, no other sets in $\mathcal{S}_i$ will be considered for further selection. We formally present GR-MCMC

[3]For the cardinality version of MCMC (i.e., MCMC with all weights being one), GR-MCMC and its performance results have previously appeared in [34]. Our results of GR-MCMC can be viewed as a generalization of those in [34].

Algorithm 1 `GR-MCMC`$\big(\mathcal{S}, k, \{a_i\}_{i=1}^{n}, \{w_{ij}\}_{i=1,\, j=1}^{n,\, a_i}, \{t_i\}_{i=1}^{m}\big)$

1: $I' \leftarrow \{1, \ldots, m\}$, $t'_i \leftarrow 0$ for all $i \in [m]$, $S'_{ij} \leftarrow S_{ij}$ for all $i \in [m]$, $j \in [c]$, and $\mathcal{G} \leftarrow \emptyset$
2: **for** $l \leftarrow 1$ to k **do**
3: Find i^*, j^* such that $w(S'_{i^*j^*}) = \max_{\forall i \in I', \forall j \in [c]} w(S'_{ij})$, where $w(S_{ij})$ denotes the total weight of normal nodes in S_{ij}
4: $G_l \leftarrow S_{i^*j^*}$, where G_l denotes the l-th element of $\mathcal{G}$
5: $t'_{i^*} \leftarrow t'_{i^*} + 1$
6: **if** $t'_{i^*} = t_{i^*}$ **then**
7: $I' \leftarrow I' \backslash \{i^*\}$
8: **end if**
9: **for** each $i \in I'$ **do**
10: **for** $j \leftarrow 1$ to c **do**
11: $S'_{ij} \leftarrow S'_{ij} \backslash S'_{i^*j^*}$
12: **end for**
13: **end for**
14: **end for**
15: **return** $\mathcal{G}$

in Alg. 1. `GR-MCMC` has k iterations, and at each iteration (i.e., `for` loop in line 2), `GR-MCMC` selects the set of the maximum total weight of uncovered normal nodes. Once the number of sets selected from a group reaches the budget assigned to the group, all the other sets in that group will be excluded for further selection, by removing the group index from the set I' of available group indices (lines 6–8). The normal nodes that are covered are removed from the available sets (lines 9–13).

We now show the performance of `GR-MCMC`.

Lemma 2.4.3 *`GR-MCMC` is a $\frac{1}{2}$-approximation algorithm.*

Proof Here, we reuse some notations in Alg. 1. Let $\mathcal{H} = \{H_1, \ldots, H_k\}$ be an optimal selection, where H_i is a coverage-set. Denote $w(G_i)$ as the total weight of the normal nodes in G_i (which is a coverage-set). To prove the lemma, we only need to show that

$$\sum_{i=1}^{k} w\left(G_i - \cup_{l=1}^{i-1} G_l\right) \geq \sum_{i=1}^{k} w\left(H_i - \cup_{l=1}^{k} G_l\right). \tag{2.1}$$

This is because, provided that Eq. (2.1) is true, it will follow that

$$\begin{aligned}
w\left(\cup_{i=1}^{k} G_i\right) &= \sum_{i=1}^{k} w\left(G_i - \cup_{l=1}^{i-1} G_l\right) \\
&\geq \sum_{i=1}^{k} w\left(H_i - \cup_{l=1}^{k} G_l\right) \quad \text{(due to Eq. (2.1))} \\
&\geq w\left(\cup_{i=1}^{k} (H_i - \cup_{l=1}^{k} G_l)\right) \quad (\text{since} \sum_i w(A_i) \geq w(\cup_i A_i)) \\
&= w\left(\cup_{i=1}^{k} H_i - \cup_{l=1}^{k} G_l\right) \\
&\geq w\left(\cup_{i=1}^{k} H_i\right) - w\left(\cup_{l=1}^{k} G_l\right).
\end{aligned}$$

This means that $w\left(\cup_{i=1}^{k} G_i\right) \geq \frac{1}{2} w\left(\cup_{i=1}^{k} H_i\right)$, i.e., the lemma follows.

We now show Eq. (2.1). Let $\{I_1, I_2\}$ be a partition of $I = \{1, \ldots, m\}$ defined as: if all sets in group $\mathcal{S}_i$ are included in $\mathcal{G}$, then $i \in I_1$; otherwise, $i \in I_2$. Denote $g(G_i)$ as the group index of a coverage set G_i. Let $\{\mathcal{H}_1, \mathcal{H}_2\}$ be a partition of $\mathcal{H}$ defined as: if $g(H_i) \in I_1$, then $H_i \in \mathcal{H}_1$, and otherwise $H_i \in \mathcal{H}_2$. Observe that $I_2 \subseteq I'$ at every iteration of `GR-MCMC`. Hence, due to the greedy property of `GR-MCMC`, it follows that for all $G_i \in \mathcal{G}$ and $H_j \in \mathcal{H}_2$,

$$w\left(G_i - \cup_{l=1}^{i-1} G_l\right) \geq w\left(H_j - \cup_{l=1}^{i-1} G_l\right) \geq w\left(H_j - \cup_{l=1}^{k} G_l\right). \tag{2.2}$$

For $\mathcal{H}_1$, there are two possible cases as follows.

Case 1: $\mathcal{H}_1 = \emptyset$**.** In this case, $\mathcal{H} = \mathcal{H}_2$. Hence, Eq. (2.2) holds for all $G_i \in \mathcal{G}$ and $H_j \in \mathcal{H}$. By summing Eq. (2.2) over all $i \in [k]$, we can obtain Eq. (2.1).

Case 2: $\mathcal{H}_1 \neq \emptyset$**.** Let $\mathcal{H}_1 = \{H_{j_1}, \ldots, H_{j_{|\mathcal{H}_1|}}\}$. We pick $G_{i_1}, \ldots, G_{i_{|\mathcal{H}_1|}} \in \mathcal{G}$ so that we can define a set $P = \left\{(G_{i_l}, H_{j_l}) : g(G_{i_l}) = g(H_{j_l}) \text{ for all } l \in [|\mathcal{H}_1|], \text{ and } G_{i_x} \neq\right.$

Fig. 2.1.: Example where GR-MCMC achieves a half of the maximum coverage. There are two monitoring nodes v_1 and v_2, each with one radio, and $k = 2$. White and black circles denote normal nodes tuned to channels 1 and 2, respectively.

G_{i_y} if $x \neq y\}$. This is possible since for any $H_{j_l} \in \mathcal{H}_1$, there must exist $t_{g(H_{j_l})}$ sets in $\mathcal{G}$ whose group indices are $g(H_{j_l})$. Due to the greedy property of GR-MCMC, it follows that for any $(G_{i_l}, H_{j_l}) \in P$,

$$w\left(G_{i_l} - \cup_{d=1}^{i_l-1} G_d\right) \geq w\left(H_{j_l} - \cup_{d=1}^{i_l-1} G_d\right) \geq w\left(H_{j_l} - \cup_{d=1}^{k} G_d\right). \tag{2.3}$$

Let $\mathcal{G}_1 = \{G_{i_l} : l \in [|\mathcal{H}_1|]\}$. By summing Eq. (2.3), we can get the following:

$$\sum_{i:G_i \in \mathcal{G}_1} w\left(G_i - \cup_{l=1}^{i-1} G_l\right) \geq \sum_{j:H_j \in \mathcal{H}_1} w\left(H_j - \cup_{l=1}^{k} G_l\right). \tag{2.4}$$

Let $\mathcal{G}_2 = \mathcal{G} - \mathcal{G}_1$. Due to Eq. (2.2), we have the following:

$$\sum_{i:G_i \in \mathcal{G}_2} w\left(G_i - \cup_{l=1}^{i-1} G_l\right) \geq \sum_{j:H_j \in \mathcal{H}_2} w\left(H_j - \cup_{l=1}^{k} G_l\right). \tag{2.5}$$

By summing Eqs. (2.4) and (2.5), we obtain Eq. (2.1).

Hence, Eq. (2.1) holds in both of the cases. Therefore, Eq. (2.1) is true, and thus the lemma follows. ■

Lemma 2.4.4 *The approximation ratio of $\frac{1}{2}$ of GR-MCMC is tight.*

Proof Without loss of generality, we assume that GR-MCMC breaks a tie that occurs when choosing the maximum-improvement set by choosing the set of the smallest index. We show the lemma by constructing an instance of MCMC where GR-MCMC achieves $\frac{1}{2}$ of the maximum coverage. We construct such an instance as follows (see Fig. 2.1): there are 20 normal nodes, each with a single ratio and weight one (i.e., $U = \{u_1, \ldots, u_{20}\}$, $a_i = 1$ and $w_i = 1$ for all $i \in [20]$); there are two monitoring nodes, each with a single radio (i.e., $m = 2$, $t_i = 1$ for all $i \in [2]$), and two wireless channels

(i.e., $c = 2$); Coverage-sets are given by $S_{11} = \{u_1, \ldots, u_{10}\}$, $S_{12} = \{u_{11}, \ldots, u_{20}\}$, $S_{21} = \{u_1, \ldots, u_{10}\}$, $S_{22} = \emptyset$, and $k = 2$. In this instance, `GR-MCMC` chooses S_{11} and S_{21} while the optimal solution is S_{12} and S_{21}. Consequently, `GR-MCMC` achieves $\frac{1}{2}$ of the maximum coverage. Thus, the lemma follows. ■

Lemmas 2.4.3 and 2.4.4 lead to the following proposition.

Proposition 2.4.1 *`GR-MCMC` is a $\frac{1}{2}$-approximation algorithm, and this approximation ratio is tight.*

2.5 BACKGROUND OF LP ROUNDING AND OVERVIEW OF PROPOSED LP ROUNDING ALGORITHMS

In the previous section, we showed that `GR-MCMC` achieves an approximation ratio of $\frac{1}{2}$. From MCSC, we know that the best possible approximation ratio for MCMC is $1 - \frac{1}{e}$ since MCSC is a special case of MCMC and the best possible approximation ratio for MCSC is $1 - \frac{1}{e}$ [32]. Therefore, we have the remaining question: does there exist an approximation algorithm that can achieve the best possible approximation ratio $1 - \frac{1}{e}$? Towards answering this question, we develop two algorithms based on a technique called *linear program (LP) rounding* in the next two sections. For this, in this section, we introduce the LP rounding technique in Subsection 2.5.1 and present an overview of our LP rounding algorithms in Subsection 2.5.2.

2.5.1 LP ROUNDING

We often face optimization problems that can be formulated as integer linear programs (ILP). ILPs in many practical situations are NP-hard. Hence, instead of exact solutions, we seek to find *approximate* solutions achievable in polynomial time. There is a technique called *LP rounding*, which is a highly effective technique to design approximation algorithms with proven performance guarantees [35]. The typical steps of LP rounding are as follows:

1) Formulate a given optimization problem as an ILP

2) Transform the ILP to an LP by relaxing the integer constraints

3) Solve the LP relaxation exactly (using one of many existing polynomial-time LP solvers)

4) Round the optimal solution of the LP relaxation, i.e., convert any fractional values to integers to obtain a *feasible* solution to the original ILP.

In the fourth step, called *rounding*, there are two distinct approaches—randomized and deterministic.

2.5.2 OVERVIEW OF PROPOSED LP ROUNDING ALGORITHMS

In this subsection, we present an overview of the LP rounding algorithms that we will develop in the next two sections.

We first formulate MCMC into an ILP. We assign an indicator variable $x_{ld} \in \{0, 1\}$ to normal radio $u_l^d \in U$, and $x_{ld} = 1$ denotes that u_l^d is verified by at least one monitoring radio in the given solution. We assign an indicator variable $y_{ij} \in \{0, 1\}$

to coverage-set $S_{ij} \in \mathcal{S}$, and $y_{ij} = 1$ denotes that S_{ij} is chosen for a solution. An ILP formulation of MCMC, denoted by ILP-MC, is given by

$$\textbf{ILP-MC:} \quad \text{maximize} \quad \sum_{l=1}^{n} \sum_{d=1}^{a_l} w_{ld} x_{ld} \tag{2.6}$$

$$\text{subject to} \quad x_{ld} \leq \sum_{i,j:\, u_l^d \in S_{ij}} y_{ij} \quad \text{for all } l \in [n],\, d \in [a_l], \tag{2.7}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{c} y_{ij} \leq k, \tag{2.8}$$

$$\sum_{j=1}^{c} y_{ij} \leq t_i \quad \text{for all } i \in [m], \tag{2.9}$$

$$0 \leq y_{ij} \leq 1 \quad \text{for all } i \in [m],\, j \in [c], \tag{2.10}$$

$$0 \leq x_{ld} \leq 1 \quad \text{for all } l \in [n],\, d \in [a_l], \tag{2.11}$$

$$y_{ij} \in \{0, 1\} \quad \text{for all } i \in [m],\, j \in [c], \tag{2.12}$$

$$x_{ld} \in \{0, 1\} \quad \text{for all } l \in [n],\, d \in [a_l]. \tag{2.13}$$

The constraint (2.7) together with (2.12) makes $x_{ld} = 1$ if at least one coverage-sets that include u_l^d are chosen for a solution, and $x_{ld} = 0$ otherwise. The constraint (2.8) is due to TBC and says that the total number of monitoring radios to be chosen must be at most k. The constraint (2.9) is due to GBC and says that each monitoring node v_i can choose at most t_i channels for tuning its radios (since v_i has t_i radios). Although the constraints (2.10) and (2.11) are redundant due to the constraints (2.12) and (2.13), we keep them for LP relaxation since we need to still maintain (2.10) and (2.11) after relaxing the integer constraints. As expected due to the NP-hardness of MCMC (Lemma 2.4.1), ILP-MC cannot be solved in polynomial time.

We next transform ILP-MC into an LP relaxation (given as (2.6)–(2.11)), denoted by LP-MC, by relaxing the integer constraints (2.12) and (2.13). In LP-MC, the variables x_{ld}'s and y_{ij}'s can now take any value in $[0, 1]$, including fractional values. Consequently, the variables lose the physical significance that they originally have in ILP-MC. We can solve LP-MC exactly using one of existing polynomial-time LP solvers.

Fig. 2.2.: Overall procedure of our two proposed LP rounding algorithms: 1) Probabilistic Rounding Algorithm with Probabilistic Rounding Scheme; 2) Deterministic Rounding Algorithm with Deterministic Rounding Scheme.

The optimal solution of LP-MC may have fractional values since the integer constraints of ILP-MC are relaxed in LP-MC. Hence, in order to obtain a feasible solution to ILP-MC, we need to round the optimal solution of LP-MC to an integer solution. While rounding, a challenge is that we should keep TBC and GBC satisfied, and at the same time we should not degrade the solution quality too much so that the resulting integer solution has a good performance guarantee. In the next two sections (Sections 2.6 and 2.7), we develop two rounding schemes corresponding to the two distinct approaches, and thereby present two LP rounding algorithms. In the both rounding schemes, we first round fractional values of y_{ij}'s to an integer 0 or 1, and then determine x_{ld}'s as $x_{ld} = \min\left\{1, \sum_{i,j:u_l^d \in S_{ij}} y_{ij}^{\#}\right\}$, where $y_{ij}^{\#}$'s are the rounded integer values.

Figure 2.2 summarizes the overall procedure of our two LP rounding algorithms.

2.6 PROBABILISTIC ROUNDING ALGORITHM

In this section, we present our second algorithm, referred to as *Probabilistic Rounding Algorithm* (PRA), that uses *Probabilistic Rounding Scheme* (PRS) to round the optimal solution of LP-MC. We develop PRS by generalizing an existing algorithm called `SAMPLING` [36]. `SAMPLING` is a randomized rounding scheme and can be used to solve MCSC. However, `SAMPLING` does not apply to MCMC since it may violate GBC. We thus develop PRS by adapting `SAMPLING` to also satisfy GBC.

PRS takes the optimal solution of LP-MC as the input, and uses the optimal solution as the probability of rounding y_{ij} to 1. Let $\tilde{\vec{y}} = (\tilde{y}_{ij} : i \in [m],\, j \in [c])$ be an optimal solution of LP-MC and define a binary random variable $Y_{ij} \in \{0, 1\}$ to

denote the resulting integer value of $\tilde{y}_{ij}$ after rounding by PRS. As we will show later (Lemma 2.6.1), Y_{ij}'s satisfy the following properties:

(P1) $\Pr[Y_{ij} = 1] = \tilde{y}_{ij}$ for all $i \in [m]$, $j \in [c]$,

(P2) $\sum_{i=1}^{m} \sum_{j=1}^{c} Y_{ij} \leq k$,

(P3) $\sum_{j=1}^{c} Y_{ij} \leq t_i$ for all $i \in [m]$,

(P4) $\Pr\left[\cap_{(i,j)\in H}\{Y_{ij} = 0\}\right] \leq \prod_{(i,j)\in H} \Pr\left[Y_{ij} = 0\right]$ for all $H \subseteq \big\{(i,j) : i \in [m],\, j \in [c]\big\}$.

The properties (P2) and (P3) are TBC and GBC, respectively, which are necessary for the output of PRS to be a feasible solution to ILP-MC. The other two properties (P1) and (P4) enable PRS to have a good performance guarantee.

PRS has a basic ingredient called `SIMPLIFY` [36]. PRS rounds the optimal solution of LP-MC by invoking `SIMPLIFY` iteratively. `SIMPLIFY` takes two inputs α, $\beta \in [0, 1]$, and yields two outputs p_α, $p_\beta \in [0, 1]$ that are determined probabilistically as shown in Alg. 2. `SIMPLIFY` has two properties: 1) at least one of p_α and p_β take an integer value 0 or 1; 2) the sum of the input values is preserved, i.e., $p_\alpha + p_\beta = \alpha + \beta$. PRS uses p_α and p_β to round α and β in a probabilistic manner. Define two binary random variables X_α, $X_\beta \in \{0, 1\}$ to denote the resulting integer values of α and β, respectively, after rounding. For $i \in \{\alpha, \beta\}$, if p_i is 0 or 1, then we fix X_i to p_i, i.e., let X_i take a value of p_i. Otherwise, i.e., if p_i is not an integer, then we do not fix X_i at this iteration, but uses this p_i in the next invocation of `SIMPLIFY` to fix X_i by feeding p_i as the input. Note that, due to the first property of `SIMPLIFY`, PRS can fix at least one of X_α or X_β to either 0 or 1 at each invocation of `SIMPLIFY`.

We now describe how PRS rounds the optimal solution of LP-MC using `SIMPLIFY`. A formal description of PRS is presented in Alg. 3. `PRS` operates in two phases. In the first phase (lines 2–11), `PRS` has m iterations. At the i-th iteration (i.e., for a single monitoring node v_i), `PRS` rounds $\tilde{y}_{i1}, \ldots, \tilde{y}_{ic}$ by invoking `SIMPLIFY` repeatedly, until all of $Y_{i1}, \ldots, Y_{ic}$ except at most one are fixed, i.e., take an integer value 0 or 1.

Algorithm 2 SIMPLIFY($\alpha,\ \beta$) // $\alpha,\ \beta \in [0,1]$

1: // On termination, the following two invariants hold: 1) at least one of p_α and p_β has an integer value of either 0 or 1; 2) $p_\alpha + p_\beta = \alpha + \beta$
2: **if** $\alpha = 0$ & $\beta = 0$ **then**
3: $p_\alpha \leftarrow 0,\ p_\beta \leftarrow 0$
4: **else if** $\alpha = 1$ & $\beta = 1$ **then**
5: $p_\alpha \leftarrow 1,\ p_\beta \leftarrow 1$
6: **else if** $0 < \alpha + \beta < 1$ **then**
7: Toss a biased coin with probability of showing head being $\alpha/(\alpha+\beta)$
8: **if** the tossed coin shows head **then**
9: $p_\alpha \leftarrow \alpha + \beta,\ p_\beta \leftarrow 0$
10: **else**
11: $p_\alpha \leftarrow 0,\ p_\beta \leftarrow \alpha + \beta$
12: **end if**
13: **else if** $\alpha + \beta = 1$ **then**
14: Toss a biased coin with probability of showing head being α
15: **if** the tossed coin shows head **then**
16: $p_\alpha \leftarrow 1,\ p_\beta \leftarrow 0$
17: **else**
18: $p_\alpha \leftarrow 0,\ p_\beta \leftarrow 1$
19: **end if**
20: **else** $\{1 < \alpha + \beta < 2\}$
21: Toss a biased coin with probability of showing head being $(1-\beta)/(2-\alpha-\beta)$
22: **if** the tossed coin shows head **then**
23: $p_\alpha \leftarrow 1,\ p_\beta \leftarrow \alpha + \beta - 1$
24: **else**
25: $p_\alpha \leftarrow \alpha + \beta - 1,\ p_\beta \leftarrow 1$
26: **end if**
27: **end if**
28: **return** (p_α, p_β)

Algorithm 3 `Probabilistic Rounding Scheme: PRS`$(\tilde{\vec{y}})$ // $\tilde{\vec{y}}$ has a dimension $1 \times mc$

1: $H \leftarrow \{(i,j) : i \in [m],\ j \in [c]\}$, $p_{ij} \leftarrow \tilde{y}_{ij}$ for all $(i,j) \in H$
2: **for** $i \leftarrow 1$ to m **do**
3: **while** $\left|\left\{(i,j) : (i,j) \in H\right\}\right| > 1$ **do**
4: $(p_{ij_1}, p_{ij_2}) \leftarrow$ `SIMPLIFY`(p_{ij_1}, p_{ij_2}) for (i,j_1), $(i, j_2(\neq j_1)) \in H$
5: **for** $l \leftarrow 1$ to 2 **do**
6: **if** $p_{ij_l} = 0$ or 1 **then**
7: $Y_{ij_l} \leftarrow p_{ij_l}$, $H \leftarrow H \setminus \left\{(i,j_l)\right\}$
8: **end if**
9: **end for**
10: **end while**
11: **end for**
12: **if** $H \neq \emptyset$ and $\sum_{(i,j)\in H} p_{ij}$ is not an integer **then**
13: $p_{00} \leftarrow \left\lceil \sum_{(i,j)\in H} p_{ij} \right\rceil - \sum_{(i,j)\in H} p_{ij}$, $H \leftarrow H \cup \left\{(0,0)\right\}$
14: **end if**
15: **while** $H \neq \emptyset$ **do**
16: $(p_{i_1j_1}, p_{i_2j_2}) \leftarrow$ `SIMPLIFY`$(p_{i_1j_1}, p_{i_2j_2})$ for (i_1,j_1), $(i_2, j_2(\neq j_1)) \in H$
17: **for** $l \leftarrow 1$ to 2 **do**
18: **if** $p_{i_lj_l} = 0$ or 1 **then**
19: $Y_{i_lj_l} \leftarrow p_{i_lj_l}$, $H \leftarrow H \setminus \left\{(i_l,j_l)\right\}$
20: **end if**
21: **end for**
22: **end while**
23: **return** $\vec{Y} = (Y_{ij} : i \in [m],\ j \in [c])$

This is achieved since `PRS` fixes Y_{ij} to p_{ij} when p_{ij} is an integer (lines 5–9) and thus at least one of Y_{ij_1} and Y_{ij_2} get fixed at each invocation of `SIMPLIFY`. Consequently, after the first phase, all of $Y_{i1}, \ldots, Y_{ic}$ except at most one are fixed for all monitoring

Algorithm 4 `Probabilistic Rounding Algorithm (PRA)`

1: Formulate ILP-MC from a given MCMC
2: Transform ILP-MC into LP-MC by relaxing the integer constraints
3: Obtain an optimal solution $\tilde{\vec{y}} = (\tilde{y}_{ij} : i \in [m],\ j \in [c])$ of LP-MC (using an existing LP solver)
4: **if** $\tilde{\vec{y}}$ is an integer vector **then**
5: $\quad \vec{Y} \leftarrow \tilde{\vec{y}}$
6: **else**
7: $\quad \vec{Y} \leftarrow \texttt{PRS}(\tilde{\vec{y}})$
8: **end if**
9: **return** $\vec{Y}$

nodes. In the second phase (lines 15–22), the remaining unfixed variables get fixed so that all Y_{ij}'s have an integer value 0 or 1. Note that, due to introducing the dummy variable Y_{00} into H (lines 12–14), even if the sum of the optimal solution of LP-MC, i.e. $\sum_{i=1}^{m}\sum_{j=1}^{c}\tilde{y}_{ij}$, is not an integer, all Y_{ij}'s would still get fixed. This is because $\sum_{i=1}^{m}\sum_{j=1}^{c}p_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{c}\tilde{y}_{ij}$ due to the sum-preservation property of `SIMPLIFY`, and thus $\sum_{i=1}^{m}\sum_{j=1}^{c}p_{ij} + p_{00}$ is an integer. Once all Y_{ij}'s are fixed, the dummy variable Y_{00} is thrown away, leaving $m \cdot c$ integers in the output.

We also present PRA formally in Alg. 4.

We now show the feasibility of the solution of `PRA` and the performance of `PRA`. For this, we first show the following lemma.

Lemma 2.6.1 *The output vector $\vec{Y}$ of `PRA` satisfies the properties (P1)–(P4).*

Proof The properties (P1), (P2), and (P4) follow from Theorem 2.1 in [36] since `PRS` can be viewed as a specific way of implementing `SAMPLING` [36]. Therefore, we only need to show that (P3) is true, i.e., $\sum_{j=1}^{c}Y_{ij} \leq t_i$ for all i. During the first phase of `PRS`, it is true that $\sum_{j=1}^{c}p_{ij} = \sum_{j=1}^{c}\tilde{y}_{ij}$ for $i \in [m]$. This is because, in the first phase of `PRS`, `SIMPLIFY` always picks two fractional values from the same group and

preserves the sum of the input values after its execution. After the first phase of PRS, there are two possible cases for each group, depending on whether the group has an unfixed variable or not.

Case 1: All Y_{ij}'s of group $\mathcal{S}_i$ are fixed. In this case, it must be true that $\sum_{j=1}^{c} Y_{ij} = \sum_{j=1}^{c} p_{ij} = \sum_{j=1}^{c} \tilde{y}_{ij} \leq t_i$, where the last inequality holds since $\{\tilde{y}_{ij}\}$ is the optimal solution of LP-MC and hence must satisfy GBC. Thus, the property (P3) holds.

Case 2: Group $\mathcal{S}_i$ has only one unfixed variable. With loss of generality, we assume that the unfixed variable is Y_{ij_1}. Then, it must be true that $0 < p_{ij_1} < 1$, and $p_{ij} = 0$ or 1 for all $j\,(\neq j_1) \in [c]$. This implies that $\sum_{j=1}^{c} \tilde{y}_{ij} < t_i$, and thus $\sum_{\forall j(\neq j_1)} Y_{ij} = \sum_{\forall j(\neq j_1)} p_{ij} = \lfloor \sum_{j=1}^{c} \tilde{y}_{ij} \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer that does not exceeds x. After the second phase of PRS, all Y_{ij}'s of group $\mathcal{S}_i$ are fixed by Y_{ij_1} being fixed to 0 or 1. Hence, it follows that

$$\sum_{j=1}^{c} Y_{ij} = \sum_{\forall j(\neq j_1)} Y_{ij} + Y_{ij_1} \leq \left\lfloor \sum_{j=1}^{c} \tilde{y}_{ij} \right\rfloor + 1 \leq t_i.$$

Thus, the property (P3) holds, which concludes the proof. ■

We now show the feasibility of the solution of PRA.

Lemma 2.6.2 *PRA yields a feasible solution $\vec{Y}$ to ILP-MC.*

Proof If the condition in line 4 of PRA is true, then it is obvious that $\vec{Y}$ is a feasible solution to ILP-MC. Hence, it suffices to show that the output $\vec{Y}$ of PRS satisfies the constraints (2.8), (2.9), and (2.12). The constraints (2.8) and (2.9) are satisfied due to the properties (P2) and (P3), respectively. Also, the constraint (2.12) is satisfied since $Y_{ij} \in \{0, 1\}$ for all $i \in [m]$ and $j \in [c]$. Thus, the lemma follows. ■

We next show the performance of PRA. We first introduce the following lemma from [35]. As we will see later (in the proof of Lemma 2.6.4), this lemma plays an important role in obtaining the expected approximation ratio of PRA, i.e., the approximation ratio of the expected performance of PRA.

Lemma 2.6.3 *Let* $p = \left|\{(i,j) : u_l^d \in S_{ij}\}\right|$. *For any* u_l^d, *it follows that for* $0 \le y_{ij} \le 1$,

$$1 - \prod_{i,j:\, u_l^d \in S_{ij}} (1 - y_{ij}) \ge (1 - (1 - 1/p)^p) \cdot \min\left\{1, \sum_{i,j:\, u_l^d \in S_{ij}} y_{ij}\right\}.$$

Here, p means the number of monitoring radios that cover a given normal radio u_l^d. Note that each normal radio will be covered by at most one radio per monitoring node since it is inefficient to tune two radios of a monitoring node to the same channel. Hence, p is upper bounded by the number of monitoring nodes, i.e., $p \le m$.

In the following lemma, we show the performance of `PRA`.

Lemma 2.6.4 *The solution* $\vec{Y}$ *of* ***PRA*** *achieves,* in expectation, *at least* $1-(1-1/m)^m$ *of the optimum of ILP-MC, where* m *is the number of monitoring nodes.*

Proof Let $X_{ld} = \min\left\{1, \sum_{i,j:u_l^d \in S_{ij}} Y_{ij}\right\}$ and denote the optimal solution of LP-MC by $\{\tilde{x}_{ld}, \tilde{y}_{ij}\}$. Then, it follows that

$$\begin{aligned}
E[X_{ld}] &= \Pr[X_{ld} = 1] \\
&= 1 - \Pr\left[\cap_{i,j:\, u_l^d \in S_{ij}} \{Y_{ij} = 0\}\right] \\
&\ge 1 - \prod_{i,j:\, u_l^d \in S_{ij}} \Pr[Y_{ij} = 0] \quad \text{(due to (P4))} \\
&= 1 - \prod_{i,j:\, u_l^d \in S_{ij}} (1 - \tilde{y}_{ij}) \quad \text{(due to (P1))} \\
&\ge \left(1 - (1 - 1/m)^m\right) \cdot \min\left\{1, \sum_{i,j:\, u_l^d \in S_{ij}} \tilde{y}_{ij}\right\} \\
&\qquad \text{(due to Lemma 2.6.3 and } p \le m) \\
&= \left(1 - (1 - 1/m)^m\right) \cdot \tilde{x}_{ld}.
\end{aligned}$$

The last equality holds due to the following reason. Since $\{\tilde{x}_{ld},\ \tilde{y}_{ij}\}$ satisfies the constraints (2.7) and (2.11), it follows that $\tilde{x}_{ld} \le \min\{1, \sum_{i,j:\, u_l^d \in S_{ij}} \tilde{y}_{ij}\}$. Also, since

we would like to maximize $\tilde{x}_{ld}$, we have $\tilde{x}_{ld} = \min\{1, \sum_{i,j:\, u_l^d \in S_{ij}} \tilde{y}_{ij}\}$. Due to the linearity of expectation, it follows that

$$E\big[\sum_{l=1}^{n}\sum_{d=1}^{a_l} w_{ld}X_{ld}\big] \geq \big(1-(1-1/m)^m\big)\cdot\sum_{l=1}^{n}\sum_{d=1}^{a_l} w_{ld}\tilde{x}_{ld}.$$

Then, the lemma follows since the optimal value of LP-MC is an upper bound on the optimal value of ILP-MC. ■

Due to Lemmas 2.6.2 and 2.6.4, we have the following proposition.

Proposition 2.6.1 *The* expected *approximation ratio of* ***PRA*** *is* $1-\frac{1}{e}$*, which is the best possible approximation ratio unless* $P = NP$*.*

Proof The proposition follows since $1-(1-1/m)^m > 1-\frac{1}{e}$, which has been shown in the proof of Lemma 2.4.2. ■

Note that the ratio $1-\frac{1}{e}$ is reached asymptotically when m (i.e., the number of monitoring nodes) tends to infinity. Practically with a finite number of monitoring nodes, the expected ratio $1-(1-1/m)^m$ of `PRA` would be higher than the asymptotic ratio $1-\frac{1}{e}$. Also, for reasonable geographical spread of the network, the number of monitoring radios that cover a normal radio will be much smaller than the total number of monitoring nodes. This also contribute to the practical performance of `PRA` being better than $1-\frac{1}{e}$ times the optimum. Our simulation results (Figures 2.4–2.11(a) in Section 2.9) bear this reasoning out.

2.7 DETERMINISTIC ROUNDING ALGORITHM

In this section, we present our third algorithm, referred to as *Deterministic Rounding Algorithm* (DRA), that uses *Deterministic Rounding Scheme* (DRS) to round the optimal solution of LP-MC. We develop DRS based on an existing algorithm called `PIPAGE` [35]. `PIPAGE` is a deterministic rounding scheme and can be used to solve MCSC. However, `PIPAGE` does not apply to MCMC because it may violate GBC. Thus, we develop DRS by carefully employing `PIPAGE` in two phases.

We first introduce the PIPAGE algorithm. PIPAGE takes a binary program of a certain form, denoted by BP, and a vector $\vec{x}$ associated with the BP as the input. BP has the following form:

$$\begin{aligned}
(\mathbf{BP}) \quad \text{maximize} \quad & f(\vec{x}) && (2.14)\\
\text{subject to} \quad & \sum_{e \in E(v)} x_e \leq p(v) \quad \text{for all } v \in V, && (2.15)\\
& 0 \leq x_e \leq 1 \quad \text{for all } e \in E, && (2.16)\\
& x_e \in \{0,1\} \quad \text{for all } e \in E. && (2.17)
\end{aligned}$$

Here, $G = (V, E)$ is a bipartite graph, $\vec{x} = (x_e \in [0,1] : e \in E)$ is a vector in $|E|$-dimensional cube $[0,1]^{|E|}$, and the function $f : [0,1]^{|E|} \to \mathbb{R}^+$ maps a vector $\vec{x} \in [0,1]^{|E|}$ to a non-negative real number. $E(v)$ denotes the set of edges that are connected to a vertex $v \in V$, and the function $p : V \to \mathbb{Z}_+$ maps a vertex $v \in V$ to a positive integer. PIPAGE takes a *fractional* vector $\vec{x}$ that satisfies the constraints (2.15) and (2.16), and yields as the output a fractional solution $\vec{x}'$ to BP that has at least one more integer components than $\vec{x}$ unless all components of $\vec{x}$ are integers. Due to the properties of PIPAGE, one can convert a fractional solution that does not satisfy the constraint (2.17) to an *feasible* solution to the BP (that also satisfies the constraint (2.17)) within $|E|$ iterations of PIPAGE.

PIPAGE proceeds as follows. If $\vec{x}$ is an integer vector, then PIPAGE terminates and yields an output vector $\vec{x}' = \vec{x}$. Suppose now that $\vec{x}$ is a fractional solution to BP that does not satisfy the integer constraint (2.17). PIPAGE constructs a subgraph $H_{\vec{x}}$ of G with the same vertex set and the edge set $E_{\vec{x}}$ defined by the condition that $e \in E_{\vec{x}}$ if and only if x_e is fractional. If $H_{\vec{x}}$ contains cycles, then PIPAGE chooses one of the cycles and denotes it by R. Otherwise, i.e., if $H_{\vec{x}}$ is a forest, then PIPAGE chooses a path of $H_{\vec{x}}$ that has endpoints of degree one and denotes it by R. Since $H_{\vec{x}}$ is a bipartite graph, in both cases, R can be uniquely represented as the union of two matchings[4]. Let M_1 and M_2 denote those two matchings. Define $\vec{x}(\epsilon, R)$ as follows: if $e \in E - R$, $x_e(\epsilon, R) = x_e$; otherwise, $x_e(\epsilon, R) = x_e + \epsilon$ for $e \in M_1$ and

[4]In a graph, a *matching* is a set of edges without common vertices.

$x_e(\epsilon, R) = x_e - \epsilon$ for $e \in M_2$. Set $\epsilon_1 = \min\left\{ \min_{e\in M_1} x_e,\ \min_{e\in M_2}(1 - x_e)\right\}$ and $\epsilon_2 = \min\left\{ \min_{e\in M_1}(1 - x_e),\ \min_{e\in M_2} x_e\right\}$, and let $\vec{x}_1 = \vec{x}(-\epsilon_1, R)$ and $\vec{x}_2 = \vec{x}(\epsilon_2, R)$. Then, `PIPAGE` yields the output vector $\vec{x}'$ as follows: if $f(\vec{x}_1) > f(\vec{x}_2)$, $\vec{x}' = \vec{x}_1$; otherwise, $\vec{x}' = \vec{x}_2$.

We now describe how DRS rounds the optimal solution of LP-MC by employing `PIPAGE`. We define $f(\vec{y})$ as

$$f(\vec{y}) = \sum_{l=1}^{n} \sum_{d=1}^{a_l} w_{ld} \left(1 - \prod_{i,j:\, u_l^d \in S_{ij}} (1 - y_{ij}) \right). \tag{2.18}$$

Note that the problem of maximizing $f(\vec{y})$ under the constraints (2.8), (2.9), and (2.12) is equivalent to the problem of maximizing $\sum_{l=1}^{n} \sum_{d=1}^{a_l} w_{ld} x_{ld}$, where $x_{ld} = \min\{1, \sum_{i,j:\, u_l^d \in S_{ij}} y_{ij}\}$, under the same constraints, and the latter problem is ILP-MC. MCSC can be solved by formulating MCSC into BP with $f(\vec{y})$ as defined in (2.18) and the constraint (2.15) formulated as GBC. However, it is impossible to formulate MCMC into the form of BP due to the two constraints of MCMC, i.e., TBC and GBC. To address this issue, we create two different forms of BP, denoted by BP1 and BP2, that capture GBC and TBC respectively, and employ `PIPAGE` in two phases with BP1 and BP2.

An important goal in rounding in both phases is not to destroy TBC and GBC that are satisfied by the optimal solution of LP-MC. If the sum of the components in group $\mathcal{S}_i$, i.e. $\sum_{j=1}^{c} \tilde{y}_{ij}$, is an integer, then all of $\tilde{y}_{i1}, \ldots, \tilde{y}_{ic}$ can be rounded to either 0 or 1 by running `PIPAGE` repeatedly. After this process, the sum $\sum_{j=1}^{c} \tilde{y}_{ij}$ will be preserved, and consequently both TBC and GBC will be satisfied. However, a subtlety arises because, in general, $\sum_{j=1}^{c} \tilde{y}_{ij}$ is not an integer. To address this issue, we develop DRS, which in its first phase, invokes a modified version of `PIPAGE`. Due to this modification, after the first phase, every group has at most one fractional components, and both TBC and GBC are satisfied. In the second phase, DRS invokes the original `PIPAGE` iteratively until all the remaining fractional components are rounded to 0 or 1. Due to the properties of `PIPAGE`, both TBC and GBC are still satisfied after the second phase.

Fig. 2.3.: Two bipartite graphs

We now present a detailed description of DRS. As previously, we let $\tilde{\vec{y}} = (\tilde{y}_{ij} : i \in [m],\, j \in [c])$ be an optimal solution of LP-MC.

DRS-Phase 1. Define a bipartite graph $G_1 = (V_1, E_1)$ as shown in Fig. 2.3(a), where V_1 is partitioned into $P_1 = \{p_1, \ldots, p_m\}$ and $Q_1 = \{q_1, \ldots, q_{mc}\}$, and $E_1 = \{e_i = (p_{\lceil i/c \rceil}, q_i) : p_{\lceil i/c \rceil} \in P_1,\, q_i \in Q_1,\, i \in [mc]\}$. Assign variables y_{ij}'s to edges in E_1 such that y_{ij} is assigned to the edge $e_{(i-1)*c+j} = \big(p_i,\, q_{(i-1)c+j}\big)$. With G_1 and $\vec{y}$, formulate the following binary program:

$$
\begin{aligned}
(\textbf{BP1}) \quad \text{maximize} \quad & f(\vec{y}) && && (2.19)\\
\text{subject to} \quad & \sum_{j=1}^{c} y_{ij} \le t_i && \text{for all } i \in [m], && (2.20)\\
& 0 \le y_{ij} \le 1 && \text{for all } i \in [m] \text{ and } j \in [c], && (2.21)\\
& y_{ij} \in \{0, 1\} && \text{for all } i \in [m] \text{ and } j \in [c]. && (2.22)
\end{aligned}
$$

In this phase, DRS invokes a modified version of `PIPAGE`, denoted by `MOD-PIPAGE`, with BP1 (which includes G_1) and $\tilde{\vec{y}}$ as the input. `MOD-PIPAGE` operates similarly as `PIPAGE`. `MOD-PIPAGE` creates a subgraph $H_{\tilde{\vec{y}}}$ from G_1 as in `PIPAGE`. Since there is no cycle in G_1, R can be chosen as only a path in $H_{\tilde{\vec{y}}}$ that has endpoints of degree one. Consequently, R is constrained to be a path of length one or two. But, differently from `PIPAGE`, `MOD-PIPAGE` only chooses a path of length exactly two for R, and exits when no such a path exists. This modification enables DRS to keep both TBC and GBC satisfied. Hence, if there exists a path of length two in $H_{\tilde{\vec{y}}}$, `MOD-PIPAGE` will

produce an output vector that has at least one more integer components. In this first phase, DRS iteratively invokes `MOD-PIPAGE` and terminates when `MOD-PIPAGE` exits without making any change. At the end of this phase, each vertex in P_1 has at most one edges that have fractional values. We denote the resulting vector after the first phase by $\vec{y}'$.

To make this clear, we give an illustrative example. In this example, $m = 4$, $c = 3$, $t_i = 2$ for all $i \in [4]$, and $k = 5$. We are given an optimal solution of LP-MC as $\tilde{\vec{y}} = (\tilde{y}_{11}, \tilde{y}_{12}, \tilde{y}_{13}, \ldots, \tilde{y}_{41}, \tilde{y}_{42}, \tilde{y}_{43})$, where $(\tilde{y}_{11}, \tilde{y}_{12}, \tilde{y}_{13}) = (0.5, 0.8, 0.7)$, $(\tilde{y}_{21}, \tilde{y}_{22}, \tilde{y}_{23}) = (0.3, 0.6, 0)$, $(\tilde{y}_{31}, \tilde{y}_{32}, \tilde{y}_{33}) = (0.1, 0.2, 0.4)$, and $(\tilde{y}_{41}, \tilde{y}_{42}, \tilde{y}_{43}) = (0.5, 0.7, 0.2)$. We now show what can happen after the first phase of DRS. The resulting vector $\vec{y}'$ is given as $(y'_{11}, y'_{12}, y'_{13}) = (0, 1, 1)$, $(y'_{21}, y'_{22}, y'_{23}) = (0.9, 0, 0)$, $(y'_{31}, y'_{32}, y'_{33}) = (0, 0, 0.7)$, and $(y'_{41}, y'_{42}, y'_{43}) = (0.4, 1, 0)$. Each group now has at most one fractional components. Note that the sum of the components in each group is preserved (which we will show in Lemma 2.7.1), and consequently both TBC and GBC are satisfied.

DRS-Phase 2. Define a bipartite graph $G_2 = (V_2, E_2)$ as shown in Fig. 2.3(b), where V_2 is partitioned into $P_2 = \{p_1\}$ and $Q_2 = \{q_1, \ldots, q_{mc}\}$, and $E_2 = \{e_i = (p_1, q_i) : p_1 \in P_2,\ q_i \in Q_2,\ i \in [mc]\}$. Assign variables y_{ij}'s to edges in G_2 such that y_{ij} is assigned to the edge $e_{(i-1)*c+j} = \big(p_1, q_{(i-1)c+j}\big)$. With G_2 and $\vec{y}$, formulate the following binary program:

$$
\begin{aligned}
(\textbf{BP2}) \quad \text{maximize} \quad & f(\vec{y}) && (2.23)\\
\text{subject to} \quad & \sum_{i=1}^{m}\sum_{j=1}^{c} y_{ij} \le k, && (2.24)\\
& 0 \le y_{ij} \le 1 \quad \text{for all } i \in [m] \text{ and } j \in [c], && (2.25)\\
& y_{ij} \in \{0, 1\} \quad \text{for all } i \in [m] \text{ and } j \in [c]. && (2.26)
\end{aligned}
$$

In this phase, DRS invokes the original `PIPAGE` iteratively with BP2 and $\vec{y}'$ as the input, until an integer vector is obtained. We denote the resulting integer vector by $\vec{y}^{\#}$, which is a feasible solution to ILP-MC (which we will show in Lemma 2.7.3)—thereby if $y^{\#}_{ij} = 1$, monitoring node v_i verifies on channel j by tuning one of its radios to channel j.

Algorithm 5 `Deterministic Rounding Scheme: DRS`$(\tilde{\vec{y}})$ // $\tilde{\vec{y}}$ has a dimension $1 \times mc$

while (1) **do**
  $\vec{y}' \leftarrow$ `MOD-PIPAGE`(BP1, $\tilde{\vec{y}}$)
  if $\vec{y}' = \tilde{\vec{y}}$ **then**
    break;
  else
    $\tilde{\vec{y}} \leftarrow \vec{y}'$
  end if
end while
$\vec{y}^{\#} \leftarrow \vec{y}'$
while $\vec{y}^{\#}$ is not an integer vector **do**
  $\vec{y}^{\#} \leftarrow$ `PIPAGE`(BP2, $\vec{y}^{\#}$)
end while
return $\vec{y}^{\#}$

We continue the previous example. In this phase, the remaining fractional components in $\vec{y}'$, i.e. y'_{21}, y'_{33}, and y'_{41}, are rounded to 0 or 1. Suppose that they are rounded to 1, 1, and 0, respectively. This results in the integer solution $\vec{y}^{\#}$ given as $(y^{\#}_{11}, y^{\#}_{12}, y^{\#}_{13}) = (0, 1, 1)$, $(y^{\#}_{21}, y^{\#}_{22}, y^{\#}_{23}) = (1, 0, 0)$, $(y^{\#}_{31}, y^{\#}_{32}, y^{\#}_{33}) = (0, 0, 1)$, and $(y^{\#}_{41}, y^{\#}_{42}, y^{\#}_{43}) = (0, 1, 0)$. With this $y^{\#}$, DRS assigns the channels to monitoring radios as follows: v_1 tunes its two radios to channels 2 and 3, respectively; v_2, v_3, and v_4 tune one of their radios to channels 1, 3, and 2, respectively.

To summarize, we present a formal description of DRS in Alg. 5, and also present DRA in Alg. 6.

We now show the feasibility of the solution of `DRA` and the performance of `DRA`. We first prove the following two lemmas.

Lemma 2.7.1 *For all $i \in [m]$, it follows that $\sum_{j=1}^{c} y'_{ij} = \sum_{j=1}^{c} \tilde{y}_{ij}$.*

Algorithm 6 `Deterministic Rounding Algorithm (DRA)`

1: Formulate ILP-MC from a given MCMC
2: Transform ILP-MC into LP-MC by relaxing the integer constraints
3: Obtain an optimal solution $\tilde{\vec{y}} = (\tilde{y}_{ij} : i \in [m],\ j \in [c])$ of LP-MC (using an existing LP solver)
4: $\vec{y}^{\#} = \texttt{DRS}(\tilde{\vec{y}})$
5: **return** $\vec{y}^{\#}$

Proof `MOD-PIPAGE` chooses a path of two edges for R, and the two edges have a common vertex in P_1. If the value assigned to one edge is increased by ϵ, then the value assigned to the other edge is decreased by ϵ. Hence, for every vertex $i \in P_1$, the sum $\sum_{j=1}^{c} \tilde{y}_{ij}$ would remain the same after a single execution of `MOD-PIPAGE`. Consequently, at the end of the first phase of `DRS`, which is after multiple iterations of `MOD-PIPAGE`, we have $\sum_{j=1}^{c} y'_{ij} = \sum_{j=1}^{c} \tilde{y}_{ij}$ for all $i \in [m]$. ■

Lemma 2.7.2 *$\vec{y}'$ is a fractional solution to BP2.*

Proof Since $\tilde{\vec{y}}$ is an optimal solution of LP-MC, $\tilde{\vec{y}}$ must satisfy TBC (i.e., $\sum_{i=1}^{m} \sum_{j=1}^{c} \tilde{y}_{ij} \leq k$). Hence, due to Lemma 2.7.1, it follows that $\sum_{i=1}^{m} \sum_{j=1}^{c} y'_{ij} = \sum_{i=1}^{m} \sum_{j=1}^{c} \tilde{y}_{ij} \leq k$, which satisfies the constraint (2.24). Recall that `PIPAGE` yields a fractional solution to BP if the input vector is fractional. This also holds for `MOD-PIPAGE` since `MOD-PIPAGE` uses the same method to increment or decrement edge weights. Hence, $0 \leq y'_{ij} \leq 1$ for all $i \in [m]$ and $j \in [c]$, which satisfies the constraint (2.25). Since $\vec{y}'$ satisfies both the constraints (2.24) and (2.25), $\vec{y}'$ is a fractional solution to BP2. ■

In the following lemma, we show the feasibility of the solution of `DRA`.

Lemma 2.7.3 *The output $\vec{y}^{\#}$ of `DRA` is a feasible solution to ILP-MC.*

Proof Due to Lemma 2.7.2 and the property that `PIPAGE` gives a fractional solution to BP if the input vector is fractional, $\vec{y}^{\#}$ satisfies the constraint (2.24), thus TBC. Obviously, $\vec{y}^{\#}$ is an integer vector. Hence, to prove the lemma, we only need to show

that $\vec{y}^{\#}$ satisfies GBC, i.e. $\sum_{j=1}^{c} y_{ij}^{\#} \leq t_i$ for all $i \in [m]$. Recall that, for each $i \in [m]$, there are at most one $j \in [c]$ such that y'_{ij} is fractional. Hence, for each i, there are two possible cases depending whether all y'_{ij}'s are integers or not.

Case 1: All y'_{ij}'s are integers. In this case, $y_{ij}^{\#} = y'_{ij}$ for all $j \in [c]$ since integer components do not change. Hence, it follows that $\sum_{j=1}^{c} y_{ij}^{\#} = \sum_{j=1}^{c} y'_{ij}$. Due to Lemma 2.7.1, it follows that $\sum_{j=1}^{c} y'_{ij} = \sum_{j=1}^{c} \tilde{y}_{ij} \leq t_i$ for all $i \in [m]$. Thus, we have $\sum_{j=1}^{c} y_{ij}^{\#} \leq t_i$ for all $i \in [m]$.

Case 2: Only one of y'_{ij}'s is fractional. With loss of generality, we assume that y'_{ij_1} is fractional. Then, since $0 < y'_{ij_1} < 1$, it follows that $\sum_{\forall j(\neq j_1)} y_{ij}^{\#} = \sum_{\forall j(\neq j_1)} y'_{ij} = \left\lfloor \sum_{j=1}^{c} y'_{ij} \right\rfloor$. Also, $y_{ij_1}^{\#} = 0$ or 1. Hence, due to Lemma 2.7.1, it holds that

$$\sum_{j=1}^{c} y_{ij}^{\#} = \sum_{\forall j(\neq j_1)} y_{ij}^{\#} + y_{ij_1}^{\#} \leq \left\lfloor \sum_{j=1}^{c} y'_{ij} \right\rfloor + 1 = \left\lceil \sum_{j=1}^{c} y'_{ij} \right\rceil = \left\lceil \sum_{j=1}^{c} \tilde{y}_{ij} \right\rceil \leq \lceil t_i \rceil = t_i.$$

For both cases, it is true that $\sum_{j=1}^{c} y_{ij}^{\#} \leq t_i$ for all $i \in [m]$. Thus, $\vec{y}^{\#}$ satisfies GBC, which concludes the proof. ■

In order to show the performance of `DRA`, we first show the following lemma.

Lemma 2.7.4 *Let $\vec{y}_i$ and $\vec{y}_o$ be the input and the output vectors of `PIPAGE` (or `MOD-PIPAGE`), respectively. Then, the following holds: $f(\vec{y}_o) \geq f(\vec{y}_i)$.*

Proof Observe that for any fractional solution and any chosen path R, the function $f\big(\vec{y}_i(\epsilon, R)\big)$ is of the form $a_2\epsilon^2 + a_1\epsilon + a_0$, where $a_2 \geq 0$. Hence, $f\big(\vec{y}_i(\epsilon, R)\big)$ a convex function of ϵ and thus achieves the maximum at an endpoint of the interval $[-\epsilon_1, \epsilon_2]$. Since $\vec{y}_o = \max\{\vec{y}_i(-\epsilon_1, R),\ \vec{y}_i(\epsilon_2, R)\}$, $f(\vec{y}_o) \geq f(\vec{y}_i)$. This proof holds for both BP1 (`MOD-PIPAGE`) and BP2 (`PIPAGE`) since `PIPAGE` and `MOD-PIPAGE` differ only in the way that a path is chosen for R; they are identical in how the edge weights are updated. ■

In the following lemma, we show the performance of `DRA`.

Lemma 2.7.5 *The solution $\vec{y}^{\#}$ of `DRA` achieves at least $1-(1-1/m)^m$ of the optimum of ILP-MC.*

Proof Let $x_{ld}^{\#} = \min\left\{1, \sum_{i,j:u_l^d \in S_{ij}} y_{ij}^{\#}\right\}$ and denote the optimal solution of LP-MC by $\{\tilde{x}_{ld}, \tilde{y}_{ij}\}$. Then, it follows that

$$\begin{aligned}
\sum_{l=1}^{n}\sum_{d=1}^{a_l} w_{ld}x_{ld}^{\#} &= \sum_{l=1}^{n}\sum_{d=1}^{a_l} w_{ld}\left(1 - \prod_{i,j:\,u_l^d \in S_{ij}} (1 - y_{ij}^{\#})\right) \\
&\geq \sum_{l=1}^{n}\sum_{d=1}^{a_l} w_{ld}\left(1 - \prod_{i,j:\,u_l^d \in S_{ij}} (1 - y'_{ij})\right) \quad \text{(due to Lemma 2.7.4)} \\
&\geq \sum_{l=1}^{n}\sum_{d=1}^{a_l} w_{ld}\left(1 - \prod_{i,j:\,u_l^d \in S_{ij}} (1 - \tilde{y}_{ij})\right) \quad \text{(due to Lemma 2.7.4)} \\
&\geq \left(1 - (1 - 1/m)^m\right) \sum_{l=1}^{n}\sum_{d=1}^{a_l} w_{ld} \cdot \min\left\{1, \sum_{i,j:\,u_l^d \in S_{ij}} \tilde{y}_{ij}\right\} \\
&\qquad \text{(due to Lemma 2.6.3 and } p \leq m\text{)} \\
&= \left(1 - (1 - 1/m)^m\right) \sum_{l=1}^{n}\sum_{d=1}^{a_l} w_{ld}\tilde{x}_{ld}.
\end{aligned}$$

The last equality holds since $\tilde{x}_{ld} = \min\{1, \sum_{i,j:\,u_l^d \in S_{ij}} \tilde{y}_{ij}\}$, which has been shown in the proof of Lemma 2.6.4. Then, the lemma follows since the optimal value of LP-MC is an upper bound on the optimal value of ILP-MC. ■

Due to Lemmas 2.7.3 and 2.7.5, we have the following proposition.

Proposition 2.7.1 *DRA* deterministically *achieves the best approximation ratio* $1 - \frac{1}{e}$ *unless* $P = NP$.

Proof It follows since $1 - (1 - 1/m)^m > 1 - \frac{1}{e}$. ■

As in PRA, this ratio $1 - \frac{1}{e}$ of DRA is reached asymptotically when m (i.e., the number of monitoring nodes) tends to infinity. For practical deployments with a finite number of monitoring nodes, the worst-performance guarantee of DRA will be higher than $1 - \frac{1}{e}$.

2.8 COMPLEXITY ANALYSIS

We first compute time complexities of the three algorithms: GR-MCMC, PRA, and DRA.

Time complexity of GR-MCMC. `GR-MCMC` has k iterations, and at each iteration, selects the set that gives the maximum improvement. In each iteration, `GR-MCMC` needs to search $O(mc)$ number of sets to find the maximum-improvement set. The number of elements in a single coverage-set is upper bounded by the number of normal nodes since it is inefficient for a normal node to tune its two radios to the same channel. Hence, each iteration takes $O(nmc)$. Thus, `GR-MCMC` has time complexity of $O(knmc)$.

Time complexity of PRA. Recall that `PRA` comprises three steps: 1) formulate LP-MC; 2) solve LP-MC using an LP solver; 3) invoke `PRS`. At the first step, `PRA` formulates a given MCMC into an LP-MC of the matrix form: $\max(\vec{c}'\vec{z})$ subject to $A\vec{z} = \vec{b}$ and $\vec{z} \geq 0$, where $\vec{z}$ is a vector of the variables x_{ld}'s and y_{ij}'s. Building matrix A from the constraints (2.7)–(2.11) dominates the time complexity of the first step. Hence, we focus on the construction of A. We have $\sum_{l=1}^{n} a_l + mc$ variables in $\vec{z}$. The constraint (2.7) has $\sum_{l=1}^{n} a_l$ inequalities and thus it takes $O\Big(\big(\sum_{l=1}^{n} a_l\big) \cdot \big(\sum_{l=1}^{n} a_l + mc\big)\Big)$ to implement (2.7). Since it is not efficient for a normal node to tune its multiple radios to the same channel, the number of actively used radios for each node is upper bounded by the number of channels, i.e., $a_l \leq c$ for all $l \in [n]$. Therefore, implementing (2.7) takes $O(n(n+m)c^2)$. Similarly, we can calculate the time complexities for the other constraints (2.8)–(2.11), and they are given by $O\big((n+m)c\big)$, $O\big(m(n+m)c\big)$, $O\big(m(n+m)c^2\big)$, and $O\big(n(n+m)c^2\big)$, respectively. The number of normal nodes is most likely larger than that of monitoring nodes. Therefore, at the first step, setting up LP-MC takes $O(n^2c^2)$. At the second step, solving LP-MC takes $O\big((n^3c^6)/\log(n^3c^6)\big)$, which is obtained by using the complexity of LP solver in [37]. At the third step, `PRA` invokes `PRS`, which in turn invokes `SIMPLIFY` multiple times. `SIMPLIFY` takes $O(1)$, a constant time. In the first phase of `PRS` (lines 2–11 in Alg. 3), `SIMPLIFY` is invoked at most mc times. This follows since `SIMPLIFY` fixes at least one variables at each iteration and therefore `SIMPLIFY` is invoked at most c times in one iteration of the `for` loop in line 2. In the second phase of `PRS` (lines 15–22 in Alg. 3), `SIMPLIFY` is invoked at most m times since each

group has at most one unfixed variables. Thus, PRS takes $O(mc)$. Overall, the second step, i.e., solving LP-MC, dominates the time complexity of PRA. Thus, PRA has time complexity of $O\big((n^3c^6)/\log(n^3c^6)\big)$.

Time complexity of DRA. Since DRA and PRA have the first two steps in common, we can use the results that we have obtained for PRA for the first two steps of DRA. Then, we only need to compute the time complexity of DRS. In the first phase (lines 1–8 in Alg. 5), DRS invokes MOD-PIPAGE at most mc times since MOD-PIPAGE reduces the number of fractional components in the input vector by at least one. In the second phase (lines 10–12 in Alg. 5), DRS invokes PIPAGE at most m times since there are at most m unfixed variables at the end of the first phase and PIPAGE also decreases the number of fractional components in the input vector by at least one. For both MOD-PIPAGE and PIPAGE, evaluating the function value $f(\vec{y})$ (Eq. (2.18)) is dominant in time complexity, and thus MOD-PIPAGE and PIPAGE have the same time complexity of $O(nmc)$. Therefore, DRS has time complexity of $O(nm^2c^2)$. Also, in DRA, solving LP-MC is dominant in time complexity and hence has time complexity of $O\big((n^3c^6)/\log(n^3c^6)\big)$, which is same as PRA.

We next discuss communication costs of the three algorithms. All of the three algorithms must know the collection of coverage-sets $\mathcal{S}$, which is global information. A central entity first broadcasts a query to all monitoring nodes, then each monitoring node replies with its coverage-sets to the central entity, and finally the central entity distributes in a single broadcast to all monitoring nodes the determination of which monitoring nodes and channels were selected. Therefore, $m + 2$ network-wide communications are required in total. However, for GR-MCMC, we can lower the communication cost to three network-wide communications and local communications by employing the approach used in MUNEN-MC [25]. This approach has two phases—node-selection and node-retention phases. In the node-selection phase, through multiple iterations, monitoring nodes that can provide high coverage improvement are chosen as candidates for a solution, using only local communications. In the node-

retention phase, k candidates with highest coverage improvement are finally selected for a solution, which requires three broadcasts.

2.9 SIMULATION RESULTS

We evaluate the performance of the proposed algorithms through simulations in two kinds of networks: random networks and scale-free networks. In random networks, we randomly place normal nodes and monitoring nodes on an 1×1 square area, and set the receiving range of monitoring radios to 0.15. In scale-free networks, the distribution $f(d)$ of nodes with degree d follows a power law of the form d^{-r}, where $2 < r < 3$. That is, the number of nodes with high degree decreases exponentially. Many empirically observed networks, such as the world wide web and the Internet, appear to be scale-free. In scale-free networks, we pick m nodes with highest degrees as monitoring nodes so that we can have a higher detection coverage than picking them randomly.

In the first set of simulations (Figures 2.4–2.7), we set n, m, and c to 200, 50, and 4, respectively. Half the normal nodes have two radios, and the other half have three radios, leading to 500 normal radios in total. Each normal node's radios are tuned to different channels and these channels are chosen randomly. Every monitoring node has two radios, and thus the total number of monitoring radios is 100. As an input parameter, we vary k (i.e., the maximum number of monitoring radios that can be used to verify normal radios). This is expressed as a percentage of the total number of monitoring radios in the network. In the second and the third sets of simulations (Figures 2.8 and 2.9, and Figures 2.10 and 2.11), we fix k to 60%, and vary m and n, respectively, to see how the proposed algorithms perform as the network size grows, while using the same setting of the other parameters as in the first set of simulations. In all simulations, we evaluate the proposed algorithms in two metrics: coverage and execution time. Here, the coverage is defined as the sum of weights of normal radios

covered by a solution divided by the total weight. All the results are the averages over 30 iterations.

Figures 2.4(a) and (b) show the coverage and execution time, respectively, in the random networks for the case when the weights of normal radios are all identical. In Fig. 2.4(a), LP-OPT denotes the optimal value of LP-MC, which is used as an upper bound on the optimal value of ILP-MC. Figure 2.4(a) shows that `DRA` achieves the highest coverage, `GR-MCMC` follows `DRA` with only a small gap, and `PRA` shows an inferior performance. `DRA`, `GR-MCMC`, and `PRA` have coverage at least 99.1%, 97.4%, and 91.4% of LP-OPT, respectively. Figure 2.4(b) shows the execution times of the three algorithms, with two y axes. The y axis on the left denotes the execution times of `GR-MCMC` and `PRA` while the y axis on the right denotes the execution time of `DRA`. We observe that the execution time of `GR-MCMC` increases almost linearly with k, as expected from the asymptotic analysis in Section 2.8. On the other hand, the results of `PRA` and `DRA` are surprising since Fig. 2.4(b) (and Figures 2.5–2.11(b), also) shows quite a different result from their asymptotic time complexities. Recall that `PRA` and `DRA` have in common their first two steps, which are formulating and solving LP-MC, and that both algorithms have the same asymptotic time complexity since the second step dominates their time complexities. However, in Fig. 2.4(b), we observe that `PRA` runs much faster than `DRA`. This result implies that, in practice, solving LP-MC takes a small amount of time, and the time complexities of `PRA` and `DRA` are determined by their rounding schemes, `PRS` and `DRS` (whose asymptotic time complexities are $O(mc)$ and $O(nm^2c^2)$, respectively). Also, note that the execution time of `DRA` increases with k, even though the asymptotic time complexity of `DRA` does not depend on k. This can be explained as follows. As k increases, the number of fractional components in the input of `DRS` is likely to increase. Consequently, `DRS` needs more invocations of `PIPAGE`/`MOD-PIPAGE`, and this makes the execution time of `DRA` increase.

Figure 2.5(a) and (b) show the results in random networks for the case when each normal radio's weight is randomly assigned to 1, 2, or 3. Comparing Fig. 2.5(a) with Fig. 2.4(a), we observe that they have little difference. `DRA`, `GR-MCMC`, and `PRA` have

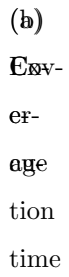

Fig. 2.4.: Random networks for different values of k, where $n = 200$, $m = 50$, and every normal radio has the identical weight.

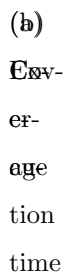

Fig. 2.5.: Random networks for different values of k, where $n = 200$, $m = 50$, and the weight of each normal radio is randomly assigned to 1, 2, or 3.

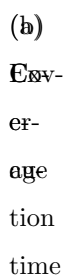

Fig. 2.6.: Scale-free networks for different values of k, where $n = 200$, $m = 50$, and every normal radio has the identical weight.

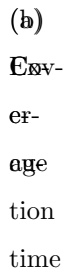

Fig. 2.7.: Scale-free networks for different values of k, where $n = 200$, $m = 50$, and the weight of each normal radio is randomly assigned to 1, 2, or 3.

coverage at least 99.3%, 97.6%, and 92.2% of LP-OPT, respectively, which is slightly better than the coverage results for the identical-weight case. On the other hand, comparing Fig. 2.5(b) (Fig. 2.7(b)) with Fig. 2.4(b) (Fig. 2.6(b)), we observe that the execution times of all the three algorithms for the different-weight case are less than those for the identical-weight case. This result suggests that, in terms of time complexity, the networks with different weights are more favorable inputs to all of the three algorithms, especially `DRA`, than networks with the identical weight. For both cases of the weight assignment, the achievable coverage levels off at around 90%. This is due to a few unfortunately placed normal nodes whose radios cannot be covered by any monitoring node, since these nodes are not within the receiving range of any monitoring node.

We next see the performance of the three algorithms in scale-free networks. Figures 2.6(a) and (b) show the coverage and execution time, respectively, for the identical-weight case, and Fig. 2.7(a) and (b) show the results for the different-weight case. We observe similar results to those for random networks. `DRA`, `GR-MCMC`, and `PRA` have coverage at least 98.2%, 97.3%, and 90.6% of LP-OPT, respectively, for the identical-weight case, and at least 98.9%, 97.8%, and 92.8% of LP-OPT, respectively, for the different-weight case. A notable result is that `DRA` runs much faster in scale-free networks than in random networks, for both cases of weight assignment. This result

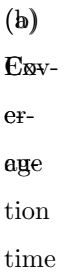

Fig. 2.8.: Random networks for different values of m, where $n = 200$, $k = 60\%$, and every normal radio has the identical weight.

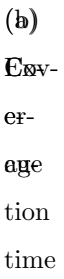

Fig. 2.9.: Scale-free networks for different values of m, where $n = 200$, $k = 60\%$, and every normal radio has the identical weight.

implies that, in terms of time complexity, scale-free networks provide more favorable inputs to DRA than random networks.

We now see how the proposed algorithms perform as the network size grows. We present only the results for the identical-weight case since those for the different-weight case show similar trends. Figures 2.8 and 2.9 show the results in the random and the scale-free networks, respectively, for different values of m. For the coverage, DRA achieves the highest coverage very close to LP-OPT, GR-MCMC attains coverage comparable to that of DRA, and PRA has an inferior coverage, similar to the results for the different values of k (i.e., Fig. 2.4–2.7(a)). For execution time, we also see similar trends—GR-MCMC, PRA, DRA in increasing order of execution time. We observe

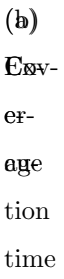

Fig. 2.10.: Random networks for different values of n, where $m = 50$, $k = 60\%$, and every normal radio has the identical weight.

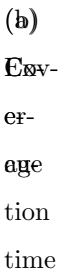

Fig. 2.11.: Scale-free networks for different values of n, where $m = 50$, $k = 60\%$, and every normal radio has the identical weight.

that the execution times of `GR-MCMC` and `PRA` increase almost linearly with m. The result for `GR-MCMC` is expected from its asymptotic time complexity, i.e., $O(knmc)$. The result for `PRA` is explained by the aforementioned reasoning that, in practice, the time complexity of `PRA` is determined by its rounding scheme, `PRS`, whose asymptotic time complexity is $O(mc)$. For `DRA`, the execution times for the random and the scale-free networks seem to increase quadratically (as the asymptotic time complexity of `DRS`, i.e., $O(nm^2c^2)$) and linearly, respectively, with m. We again observe that the scale-free networks give more favorable inputs to all of the three algorithms than the random networks, in terms of time complexity.

Figures 2.10 and 2.11 show the results in the random and the scale-free networks, respectively, for different values of n. We also see similar trends in the performance comparison of the proposed algorithms. We observe that the coverages of the three algorithms decrease as n grows. This is due to the decreasing ratio of the number of monitoring nodes to the number of normal nodes. A notable observation is that the execution time of `GR-MCMC` increases slowly with n whereas the execution time of `PRA` and `DRA` grow much faster.

We also observe similar results for other values of c, and therefore they are not included here. In addition, we would like to point out that it is not fair to compare the coverage results in random networks with those in scale-free networks, even for the same settings. This is because, in these two kinds of networks, different parameters are used to determine the coverage-sets, which are receiving range for random networks and the parameter r (in the distribution d^{-r}) for scale-free networks.

Summarizing our results, `DRA` shows the highest coverage close to the maximum coverage but has high time complexity, `PRA` shows an inferior coverage but has reasonable time complexity, and `GR-MCMC` shows a good coverage comparable to `DRA` and has low time complexity. Hence, `GR-MCMC` can be a good compromise between coverage and time complexity. However, for critical security deployments, the network administrator needs to guarantee the worst-case performance, in which `DRA` and `PRA` would be favored.

2.10 CONCLUSIONS

In this chapter, we study the problem of the optimal selection of monitoring nodes and channels in multi-channel multi-radio WMNs for verifying the behavior of normal network nodes. We mathematically formulate this problem, and show that obtaining the exact optimal solution is NP-hard. We then present three algorithms to approximate the optimal solution—GR-MCMC, PRA, and DRA. GR-MCMC is an intuitive extension from an existing greedy algorithm for single-channel networks,

and achieves an approximation ratio of $\frac{1}{2}$, which is inferior to the best possible approximation ratio $1 - \frac{1}{e}$ for our problem. The other two algorithms are based on the LP rounding technique, and achieve the best approximation ratio $1 - \frac{1}{e}$. PRA attains this ratio probabilistically while DRA achieves it deterministically. Our simulations results show that GR-MCMC is a good compromise between coverage and execution time. However, for critical security deployments, PRA and DRA are favored since they provide a superior performance guarantee in the worst case.

3. DISTRIBUTED ONLINE CHANNEL ASSIGNMENT TOWARD OPTIMAL MONITORING IN MULTI-CHANNEL WIRELESS NETWORKS

3.1 INTRODUCTION

We consider a channel assignment problem for passive monitoring in multi-channel wireless networks. Passive monitoring is a widely-used and effective technique to monitor wireless networks, where a set of sniffers (i.e., software or hardware devices that intercept and log packets) are used to capture and analyze network traffic between other nodes to estimate network conditions and performance. Such estimates are utilized for efficient network operation, such as network resource management, network configuration, fault detection/diagnosis and network intrusion detection. Recently, it has been extensively studied to use multiple channels in wireless networks, especially in wireless mesh networks (WMNs) [6, 14, 17, 19, 29]. It has been shown that equipping nodes with multiple radios tuned to different non-overlapping channels can significantly increase the capacity of the network. In multi-channel wireless networks, a major challenge with passive monitoring is how to assign a set of channels to each sniffer's radios such that as large an amount of traffic or large a number of nodes as possible are captured. We call this the *optimal sniffer-channel assignment* (OSCA) problem.

We can employ the algorithms proposed in Chapter 2 to solve OSCA, since OSCA is a special case of MCMC with all monitoring nodes being activated (refer to Section 2.2). But, they are centralized and offline algorithms. That is, the algorithms requires a central authority that first gathers, from all sniffers, either a prior knowledge of the network topology and the channel usages of all nodes to be monitored, or

primitive information to estimate the prior knowledge, then runs the algorithm and distributes the solution to all sniffers.

These centralized algorithms are not suitable for large-scale and dynamic networks, due to several reasons. The centralized algorithms require an efficient and cost-effective two-way global communication mechanism between the central authority and all sniffers, i.e., the communications from all sniffers to the central authority for the delivery of the prior knowledge, and also the communication from the central authority to all sniffers for the distribution of the solution. However, this is difficult to achieve in large-scale networks, especially in multi-hop wireless networks. Also, such a two-way global communication needs to be achieved without too much delay, otherwise the centralized algorithms are not agile to frequent network changes, such as channel-usage changes of nodes and network topology changes due to mobility of nodes and arrivals/departures of sniffers. In addition, the centralized algorithms are difficult to deploy in ad hoc wireless networks, which lack the central authority or a powerful node that has a high computational power, a large memory, and no significant energy constraint. Moreover, the powerful node needs to be fault-tolerant or easily replaceable when it fails, since otherwise the entire monitoring system may fail due to a single-point failure.

In this chapter, we develop *distributed* and *online* solutions to OSCA for large-scale and dynamic networks. The distributed algorithm, called DA-OSCA, achieves a *provably good* performance. DA-OSCA can always achieve at least $1 - \frac{1}{e}$ (≈ 0.632) of the maximum monitoring coverage, regardless of the network topology and the channel assignment of nodes to be monitored. Thus, DA-OSCA preserves the approximation ratio that the centralized algorithm DRA achieves, while providing a distributed solution that is amenable to online implementation. Also, DA-OSCA is *cost-effective*, in terms of communication and computational overheads, since DA-OSCA requires only local communication among neighboring nodes and also adapts incrementally to network changes. DA-OSCA solves OSCA in two steps. At the first step, DA-OSCA solves distributedly an LP relaxation of OSCA, which is obtained

by removing the integer constraints from integer linear program (ILP) formulation of OSCA. At the second step, DA-OSCA rounds distributedly the fractional solution of the LP relaxation to an integer solution, while obtaining a feasible solution to the original ILP. Moreover, the decentralized and adaptive structure of DA-OSCA allows us to operate DA-OSCA in two different modes that are suitable for fast-varying and slow-varying networks, respectively. Specifically, one is a proactive mode for fast-varying network, while the other is a reactive mode for slow-varying networks. With these two operational modes, DA-OSCA can adapt to two different rates of network changes in a cost-effective manner. To demonstrate the effectiveness of DA-OSCA in these modes, we conduct simulations in two kinds of network—random networks and scale-free networks.

The rest of the chapter is organized as follows. Section 3.2 presents the problem formulation and existing results. Section 3.3 presents the distributed algorithm. Section 3.4 describes the online implementation of the distributed algorithm. Section 3.5 discusss notes. Section 3.6 presents simulation results. Section 3.7 concludes this chapter and discusses future works.

3.2 PROBLEM FORMULATION

3.2.1 OPTIMAL SNIFFER-CHANNEL ASSIGNMENT (OSCA) PROBLEM

We are given a set N of nodes to be monitored, and each node $n \in N$ is tuned to a wireless channel chosen from a set C of available wireless channels, where $|C| \geq 2$. The channels are chosen according to one of many available channel assignment algorithms (e.g., [15, 17, 19]). Each node n is given a non-negative weight w_n. These weights of nodes can be used to capture various application-specific objectives of monitoring. For example, one can use the weights to capture transmission rates of nodes. In this scenario, we would assign higher weights to the nodes that transmit larger volumes of data, thereby biasing our algorithm to monitor such nodes more. Or, for security

monitoring, one can assign the weights by taking into account nodes' trustworthiness computed based on previous monitoring results. Here, a node that has been found to be compromised before (and repaired thereafter) will be assigned a higher weight.

We are given a set S of sniffers, each of which needs to determine a wireless channel from C to tune its radio to. We say that a sniffer and a node are *neighbors* if the sniffer can overhear the node, and also that two sniffers are *neighbors* if there exists a node that can be overheard by both the sniffers. We say that a node is *covered* if the node is overheard by at least one sniffer being tuned to the same channel as the node. We are given a collection of coverage-sets, $\mathcal{K} = \{K_{s,c} \subseteq N : s \in S, c \in C\}$, where a *coverage-set* $K_{s,c}$ contains the nodes that can be covered by sniffer s being tuned to channel c. We define a *group* as a collection of coverage-sets of a sniffer over all channels, i.e. $\mathcal{K}_s = \{K_{s,c} : c \in C\}$. Our objective is to maximize the total weight of the nodes covered by judiciously choosing one coverage-set from each group. Here, the constraint that only one coverage-set can be chosen from each group arises since each sniffer can tune its radio to only one channel at a time, since it has a single radio. We call this constraint the *group budget constraint*, and refer to the optimization problem as the ***optimal sniffer-channel assignment*** (OSCA) problem.

For ease of exposition, we assume that all of the nodes and the sniffers have only one radio. However, the multi-radio case, where nodes and sniffers are equipped with multiple radios, can be easily mapped to this single-radio case (refer to Section 3.5).

3.2.2 HARDNESS OF OSCA

We present existing results on the hardness of OSCA.

Theorem 3.2.1 (Theorem 1 [38]) *OSCA is NP-hard.*

This means that the computational complexity to solve OSCA grows exponentially with the number of sniffers, unless $P = NP$.

Also, we have an inapproximability result for OSCA.

Fig. 3.1.: Distributed Algorithm for OSCA (DA-OSCA).

Theorem 3.2.2 (Corollary 2 [38]) *For any $\epsilon > 0$, it is NP-hard to approximate OSCA within a factor of $\frac{7}{8} + \epsilon$ of the optimum.*

Thus, the best achievable approximation ratio for OSCA is at most $\frac{7}{8}$.

3.3 THE DISTRIBUTED ALGORITHM FOR OSCA

We develop a distributed algorithm to solve OSCA, referred to as DA-OSCA. The basic structure of DA-OSCA is based on the Linear Program (LP) rounding technique, where we first solve the LP relaxation of OSCA and then round the (fractional) solution of the LP relaxation to an feasible integer solution to the original OSCA problem. Figure 3.1 shows an overview of how DA-OSCA yields an approximate solution to OSCA. DA-OSCA consists of two components: 1) the Distributed Algorithm to solve the LP relaxation of OSCA (DA-LP$_{\text{OSCA}}$); 2) Opportunistic Channel Assignment Algorithm (OCAA) to perform distributed rounding of the fractional solution yielded by DA-LP$_{\text{OSCA}}$.

3.3.1 DISTRIBUTED ALGORITHM FOR SOLVING LP RELAXATION OF OSCA

LP relaxation of OSCA. We first formulate an integer linear program (ILP) of OSCA. We assign an indicator variable $x_n \in \{0, 1\}$ to each node $n \in N$, where $x_n = 1$ indicates that node n is covered by the given solution. We assign an indicator

variable $y_{s,c} \in \{0,1\}$ to a coverage-set $K_{s,c} \in \mathcal{K}$, and $y_{s,c} = 1$ indicates that sniffer s will be tuned to channel c. The ILP of OSCA, denoted by ILP_{OSCA}, is given by:

$$\text{maximize} \quad \sum_{n \in N} w_n x_n \tag{3.1}$$

$$\text{subject to} \quad x_n \leq \sum_{s,c:\, n \in K_{s,c}} y_{s,c} \qquad \forall n \in N, \tag{3.2}$$

$$\sum_{c \in C} y_{s,c} \leq 1 \qquad \forall s \in S, \tag{3.3}$$

$$0 \leq x_n, y_{s,c} \leq 1 \qquad \forall n \in N, s \in S, c \in C, \tag{3.4}$$

$$x_n, y_{s,c} \in \{0,1\} \qquad \forall n \in N, s \in S, c \in C. \tag{3.5}$$

The objective function (3.1) together with the constraints (3.2) and (3.5) makes $x_n = 1$ if at least one coverage-set that includes the node n is chosen for a solution, and $x_n = 0$ otherwise. Eq. (3.3) is due to the group budget constraint.

Since ILP_{OSCA} cannot be solved in polynomial time, we relax the integer constraint (3.5) to obtain the LP relaxation of OSCA, i.e., Eqs. (3.1)–(3.4), denoted by LP_{OSCA}. In LP_{OSCA}, the variables x_l's and y_{ij}'s can now take any value in $[0,1]$, including fractional values.

Solving LP_{OSCA}. We use the *Proximal Optimization Algorithm* (POA) [39, Ch. 3.4.3] combined with a dual approach to solve LP_{OSCA}. POA introduces a set of auxiliary variables and adds quadratic terms to the objective function (3.1) of LP_{OSCA} to transform LP_{OSCA} into a quadratic program (QP) (as given in Eq. (3.6)), and then solves the QP by sequentially updating the values of the two kinds of variables, i.e. first the original variables and then the auxiliary variables. The rationale behind the transformation is to resolve a difficulty due to the linearity of the objective function (3.1) when we solve the dual problem of LP_{OSCA}. Specifically, the objective function (3.1) of LP_{OSCA} is linear, and hence it is not strictly concave. As a result, the dual problem of LP_{OSCA} may not be differentiable at every point. This leads to a difficulty when we use the Gradient Projection Algorithm [39, Ch. 3.3.2] to solve the dual problem. However, such a difficulty will be resolved with the QP, since the objective

function of the $\mathrm{QP}_{\mathrm{OSCA}}$ is strictly concave due to the added quadratic terms and thus is differentiable.

We now apply POA to $\mathrm{LP}_{\mathrm{OSCA}}$. We introduce a set of auxiliary variables $\{x_n^{\mathrm{aux}}, y_{s,c}^{\mathrm{aux}} : n \in N, s \in S, c \in C\}$, and transform $\mathrm{LP}_{\mathrm{OSCA}}$ into the following equivalent quadratic program, denoted by $\mathrm{QP}_{\mathrm{OSCA}}$:

$$\begin{aligned} \text{maximize} \quad & \sum_{n\in N} w_n x_n - \frac{1}{2d}\Bigg(\sum_{n\in N} (x_n - x_n^{\mathrm{aux}})^2 \\ & \qquad + \sum_{\forall(s,c)} \left(y_{s,c} - y_{s,c}^{\mathrm{aux}}\right)^2 \Bigg) \qquad (3.6) \\ \text{subject to} \quad & \text{Eqs. (3.2)–(3.4).} \end{aligned}$$

Here, d is a positive constant. It can be shown that solving $\mathrm{QP}_{\mathrm{OSCA}}$ is equivalent to solving $\mathrm{LP}_{\mathrm{OSCA}}$ (refer to Appendix A.1 for the proof of this claim). For notational simplicity, we define $\vec{x} = (x_n : n \in N)$ and $\vec{y} = (y_{s,c} : s \in S, c \in C)$, and define $\vec{x}^{\mathrm{aux}}$ and $\vec{y}^{\mathrm{aux}}$ similarly as $\vec{x}$ and $\vec{y}$. The POA to solve $\mathrm{QP}_{\mathrm{OSCA}}$, referred to as POA-$\mathrm{QP}_{\mathrm{OSCA}}$, proceeds as follows. At t-th iteration, $t = 1, 2, 3, \ldots$, POA-$\mathrm{QP}_{\mathrm{OSCA}}$ executes the following two steps:

S1: Fixing $\vec{x}^{\mathrm{aux}} = \vec{x}^{\mathrm{aux}}(t)$ and $\vec{y}^{\mathrm{aux}} = \vec{y}^{\mathrm{aux}}(t)$, solve $\mathrm{QP}_{\mathrm{OSCA}}$ with respect to $\vec{x}$ and $\vec{y}$. Let the solution obtained be $\vec{x}(t), \vec{y}(t)$.

S2: Let $\vec{x}^{\mathrm{aux}}(t+1) = \vec{x}(t)$ and $\vec{y}^{\mathrm{aux}}(t+1) = \vec{y}(t)$.

POA-$\mathrm{QP}_{\mathrm{OSCA}}$ can start with any initial values, i.e. any $\vec{x}^{\mathrm{aux}}(1)$ and $\vec{y}^{\mathrm{aux}}(1)$. As the number t of iterations tends to infinity, a sequence of vectors generated by POA-$\mathrm{QP}_{\mathrm{OSCA}}$ converges to the optimal solution of $\mathrm{QP}_{\mathrm{OSCA}}$ [39, Ch. 3.4.3].

Note that, at Step S1 in each iteration of POA-$\mathrm{QP}_{\mathrm{OSCA}}$, we still have an optimization problem to be solved. We solve the optimization problem given at Step S1 by solving its dual problem instead. The reason why we solve the dual problem instead of the primal problem is that the dual problem has a simple form of constraints and is easily decomposable, and these features enable us to design a distributed algorithm to solve the problem.

We derive the dual problem of the optimization problem given by Step S1 of POA-QP_{OSCA}, i.e., the QP_{OSCA} with $\vec{x}^{\text{aux}}$ and $\vec{y}^{\text{aux}}$ being fixed. For notational simplicity, we let $\vec{z} = (\vec{x}, \vec{y})$ and $\vec{z}^{\text{aux}} = (\vec{x}^{\text{aux}}, \vec{y}^{\text{aux}})$. We define a set Z that contains all of $(\vec{x}, \vec{y})$'s satisfying Eqs. (3.3) and (3.4). We define a set of Lagrange Multipliers $\vec{p} = (p_n : n \in N)$ for the $|N|$ constraints in Eq. (3.2). We define the Lagrangian function of the QP_{OSCA} with fixed $\vec{x}^{\text{aux}}$ and $\vec{y}^{\text{aux}}$ as

$$L(\vec{z}, \vec{p}; \vec{z}^{\text{aux}}) = \sum_{n \in N} w_n x_n + \sum_{n \in N} p_n \left(\sum_{(s,c): n \in K_{s,c}} y_{s,c} - x_n \right) - \frac{1}{2d} \left(\sum_{n \in N} (x_n - x_n^{\text{aux}})^2 + \sum_{\forall (s,c)} \left(y_{s,c} - y_{s,c}^{\text{aux}} \right)^2 \right). \tag{3.7}$$

The dual problem is then given by

$$\begin{aligned} \text{minimize} \quad & D(\vec{p}; \vec{z}^{\text{aux}}) \triangleq \max_{\vec{z} \in Z} L(\vec{z}, \vec{p}; \vec{z}^{\text{aux}}) \\ \text{subject to} \quad & \vec{p} \geq 0. \end{aligned} \tag{3.8}$$

Since the dual objective function D in (3.8) is now differentiable due to the quadratic terms in Eq. (3.7), we can use the Gradient Projection Algorithm (GPA) (refer to [39, Ch. 3.3.2]) to solve the dual problem. The GPA to solve the dual problem has the following iterations: for $i = 0, 1, 2, \ldots,$

$$\begin{aligned} & p_n(i+1) = \left[p_n(i) + \beta \cdot g_n(i) \right]^+_{[0, +\infty)}, \\ & \text{where } g_n(i) \triangleq \left. \frac{\partial D}{\partial p_n} \right|_{p_n = p_n(i)} = x_n^*(i) - \sum_{(s,c): n \in K_{s,c}} y_{s,c}^*(i). \end{aligned} \tag{3.9}$$

Here, $\beta > 0$ is the step size, $[\vec{p}]^+_A$ denotes the projection to a set A, which maps $\vec{p}$ to the point in A that is closest to $\vec{p}$, and $(\vec{x}^*(i), \vec{y}^*(i)) \in Z$ is the optimal solution that maximizes $L(\vec{z}, \vec{p}(i); \vec{z}^{\text{aux}})$ for given $\vec{p}(i)$. To compute the iterations in Eq. (3.9), at each iteration, we need to solve the following maximization problem : for given $\vec{p}(i)$,

$$\begin{aligned} \text{maximize} \quad & L(\vec{z}, \vec{p}(i); \vec{z}^{\text{aux}}) \\ \text{subject to} \quad & \vec{z} \in Z. \end{aligned} \tag{3.10}$$

To solve Eq. (3.10), we rearrange the terms in Eq. (3.7) and rewrite Eq. (3.7) as the following:

$$L(\vec{z},\vec{p};\vec{z}^{\text{aux}}) = \sum_{n\in N}\left(-\frac{1}{2d}\left(x_n - x_n^{\text{aux}}\right)^2 + (w_n - p_n)x_n\right) + \sum_{\forall(s,c)}\left(-\frac{1}{2d}\left(y_{s,c} - y_{s,c}^{\text{aux}}\right)^2 + y_{s,c}\sum_{n\in K_{s,c}} p_n\right). \tag{3.11}$$

Using Eq. (3.11), we can decompose the problem in Eq. (3.10) into the following sets of independent subproblems:

1) for each $n \in N$,

$$\begin{aligned} \text{maximize} \quad & -\frac{1}{2d}\left(x_n - x_n^{\text{aux}}\right)^2 + \left(w_n - p_n(i)\right)x_n \\ \text{subject to} \quad & 0 \le x_n \le 1 \end{aligned} \tag{3.12}$$

2) for each $s \in S$,

$$\begin{aligned} \text{maximize} \quad & \sum_{c\in C}\left(-\frac{1}{2d}\left(y_{s,c} - y_{s,c}^{\text{aux}}\right)^2 + y_{s,c}\sum_{n\in K_{s,c}} p_n(i)\right) \\ \text{subject to} \quad & \sum_{c\in C} y_{s,c} \le 1 \text{ and } y_{s,c} \ge 0 \quad \forall c \in C. \end{aligned} \tag{3.13}$$

Note that each sub-problem can be solved independently at each node and at each sniffer, using purely local communication. By solving each subproblem independently, we can obtain the solutions to Eqs. (3.12) and (3.13) as the following:

$$x_n^*(i) = [x_n^{\text{aux}} + d(w_n - p_n(i))]_{[0,1]}^+ \tag{3.14}$$

$$\vec{y}_s^*(i) = \left[\left(y_{s,c}^{\text{aux}} + d\sum_{n\in K_{s,c}} p_n(i) : c \in C\right)\right]_{Y_s}^+, \text{ where}$$

$$Y_s = \left\{\vec{y}_s \triangleq (y_{s,c} : c \in C) : \sum_{c\in C} y_{s,c} \le 1,\ y_{s,c} \ge 0\ \forall c\right\}. \tag{3.15}$$

Here, the projection $[\cdot]_{Y_s}^+$ in (3.15) can be easily done, e.g., with Alg. 16 in Appendix A.2. Thus, we now have the solution to the dual problem (3.8). To solve the dual problem, we iteratively update the dual variables $\vec{p}$ according to Eq. (3.9). Here,

at each iteration, we need to compute $g_n(i)$, and this requires to solve the independent problems in Eqs. (3.12) and (3.13). To solve them, we update the primal variables $\vec{x}$ and $\vec{y}$ according to Eqs. (3.14) and (3.15).

Consequently, we finally have the solution to the Step S1 of POA-QP$_{\text{OSCA}}$. We obtain the solution by alternately updating the dual and the primal variables, according to Eq. (3.9) and Eqs. (3.14), (3.15), respectively. As the number i of iterations tends to infinity, a sequence of vectors given by Eq. (3.9) converges to the optimal solution of the dual problem [39, Proposition 3.4]. Once the optimal solution of the dual problem is obtained, we can find the optimal solution of the primal problem (i.e. the optimization problem given by Step S1 of POA-QP$_{\text{OSCA}}$) using (3.14) and (3.15) [40, Ch. 5.5.3].

To summarize, we present a formal description of the overall procedure to solve LP$_{\text{OSCA}}$ in Alg. 7, which we refer to as the Distributed Algorithm for solving LP$_{\text{OSCA}}$ (DA-LP$_{\text{OSCA}}$). Note that DA-LP$_{\text{OSCA}}$ requires *only local communications* among neighboring nodes. In many monitoring applications, it would be desirable that DA-LP$_{\text{OSCA}}$ should be run by only sniffers since DA-LP$_{\text{OSCA}}$ is for sniffers to determine their channels. In such cases, we can let one of neighboring sniffers of node n act as a proxy and take over the node n's duty of updating values of the variables x_n, x_n^{aux} and p_n. Hence, each sniffer s needs to update values of its own variables $\vec{y}_s$, $\vec{y}_s^{\text{aux}}$, and also variables x_n's, x_n^{aux}'s and p_n's for some of its neighboring nodes. Since now sniffers update also the variables of nodes, each sniffer only needs to communicate with its neighboring sniffers to obtain the required values for the update of its variables.

DA-LP$_{\text{OSCA}}$ with $I = 1$. The standard POA [39, Ch. 3.4.3], which is the DA-LP$_{\text{OSCA}}$ when $I \to \infty$, requires a two-level convergence structure. That is, the inner-level iterations (i.e., the `for` loop in lines 3–8) must converge before the next outer-level iteration (i.e., the `while` loop in lines 1–11) begins. However, such a two-level convergence structure is not suitable for distributed algorithms because it increases the running time of DA-LP$_{\text{OSCA}}$ and also incurs substantial communication overheads, due to a mechanism required to determine when to stop inner-level iterations. This

Algorithm 7 DA-LP$_{\text{OSCA}}$

while TRUE **do**
  // Step 1 of POA-QP$_{\text{OSCA}}$
  for $i = 0$ to $I \to \infty$ **do**
    Each node n and each sniffer s compute $x_n(i)$ and $\vec{y}_s(i)$ according to Eqs. (3.14) and (3.15), respectively. Then, sniffer s sends the updated values $\vec{y}_s(i)$ to its neighboring nodes.
    if $i \neq I$ **then**
      Each node n computes $p_n(i+1)$ according to Eq. (3.9), then sends $p_n(i+1)$ to its neighboring nodes and sniffers.
    end if
  end for
  // Step 2 of POA-QP$_{\text{OSCA}}$
  Each node n and each sniffer s set initial values of their variables for the next iteration as

$$x_n^{\text{aux}} \leftarrow x_n(I) \text{ and } p_n(0) \leftarrow p_n(I) \qquad \text{(node } n\text{)}$$
$$\vec{y}_s^{\text{aux}} \leftarrow \vec{y}_s(I) \qquad \text{(sniffer } s\text{)}.$$

end while

intuition is that, as the number of inner-level iterations increases, the improvement of the solution quality at each iteration would decrease. Hence, such later iterations that give a small improvement would be wasteful, since solving the problem given by Step S1 is only an intermediate step to solve the ultimate problem. For these reasons, we fix the number of inner-level iterations of DA-LP$_{\text{OSCA}}$ to 2 (i.e. $I = 1$), and find a good approximate solution.

We now show that, even with $I = 1$, DA-LP$_{\text{OSCA}}$ can converge to the optimal solution. We let $\vec{z}^{\text{aux},t}$ and $\vec{p}^{\,t}$ be the values of $\vec{z}^{\text{aux}}(I)$ and $\vec{p}(I)$, respectively, at the t-th outer-level iteration. Also, we let $\vec{z}^{\text{aux},*}$ and $\vec{p}^{\,*}$ be the primal optimal solution and

the dual optimal solution, respectively, of QP_{OSCA}. The following theorem[1] provides a sufficient condition of the step size β (to solve the dual problem Eq. (3.9)) for DA-LP_{OSCA} with $I = 1$ to converge.

Theorem 3.3.1 *As $t \to \infty$, a sequence of vectors $(\vec{z}^{aux,t}, \vec{p}^{t})$ given by DA-LP_{OSCA} with $I = 1$ converges to $(\vec{z}^{aux,*}, \vec{p}^{*})$, provided that*

$$\beta < \frac{1}{2dB_1B_2}, \text{ where}$$

$$B_1 = \max\{1, |K_{s,c}| : s \in S, c \in C\} + 1,$$
$$B_2 = \max\{|C|, M + 1\}, \text{ and } M = \max_{n \in N} |\{K_{s,c} : n \in K_{s,c}\}|.$$

The proof is given in Appendix A.3. Here, the upper bound $\frac{1}{2dB_2B_2}$ can be obtained by computing the two pieces of information: the maximum number of node that can be covered by any sniffer operating on any channel, and the maximum number of neighboring sniffers that a normal node has.

3.3.2 OPPORTUNISTIC CHANNEL ASSIGNMENT ALGORITHM

We develop a distributed rounding algorithm that determines the channel assignment of sniffers based on the optimal solution $\vec{y}^{*}$ given by DA-LP_{OSCA}. We refer to this as the *Opportunistic Channel Assignment Algorithm* (OCAA). OCAA can be viewed as a distributed generalization of a centralized rounding scheme called `PIPAGE` [35]. `PIPAGE` guarantees that, for a given LP-relaxation solution that achieves a constant factor α of the optimal value of the LP relaxation, the integer solution yielded by `PIPAGE` always achieves at least $\alpha \cdot (1 - \frac{1}{e})$ of the optimal value of the original ILP. However, `PIPAGE` is not suitable for distributed solutions because `PIPAGE` rounds the

[1]Our result in Theorem 3.3.1 can be viewed as a parallel version of the improved POA scheme [41], which has studied a cross-layer transmission scheduling problem in wireless networks. This work has previously used the idea of fixing the number of inner-level iterations. But, the results in [41] are based on the assumption that the coefficients in the constraints of the underlying LP problem must be non-negative. Hence, the results in [41] cannot be directly applied to our problem, i.e., LP_{OSCA} that have negative coefficients in the constraints.

LP-relaxation solution through a number of iterations and each iteration requires a global communication to evaluate the quality of the intermediate solution. On the other hand, our OCAA can achieve the same ratio $1-\frac{1}{e}$ in a distributed manner that requires only local communications among neighboring sniffers. In this subsection, we first describe OCAA and then present the guarantee of OCAA.

We first introduce a metric called *coverage improvement* that guides each sniffer to make a good decision on selecting its channel. For a given set of values $\vec{y}^*_{N(s)} = \{y^*_{s',c} : s' \in N(s), c \in C\}$, where $N(s)$ denotes the set of neighboring sniffers of sniffer s, the *coverage improvement* of coverage-set $K_{s,c}$ is defined as

$$I\left(K_{s,c}; \vec{y}^*_{N(s)}\right) = \sum_{n \in K_{s,c}} w_n \left(\prod_{(s',c): s' \neq s, n \in K_{s',c}} (1 - y^*_{s',c}) \right). \tag{3.16}$$

Intuitively, by viewing $y^*_{s',c}$ as the probability that sniffer s' tunes its radio to channel c, we can interpret $I(K_{s,c}; \vec{y}^*_{N(s)})$ as an expected coverage improvement, in terms of the total weight of the nodes in $K(s,c)$, that can be achieved by sniffer s tuning its ratio to channel c. Specifically, when $y^*_{s',c}$ is viewed as such a probability, $I(K_{s,c}; \vec{y}^*_{N(s)})$ means the expected total weight of the *uncovered* nodes in $K(s,c)$, provided that all the neighboring sniffers of s (i.e., all s') do not tune their channels to c. In other words, $I(K_{s,c}; \vec{y}^*_{N(s)})$ is the expected total weight improvement that sniffer s can achieve by tuning its radio to channel c. Note that sniffer s can compute its coverage improvements over all the channels by communicating only with its neighbors.

We formally present OCAA in Alg. 8. OCAA determines the channels of sniffers through several iterations, in the order according $\mathcal{P}$. In each iteration, the sniffers in P_i determine theirs channels in parallel such that each sniffer s selects the channel that achieves the maximum coverage improvement in terms of $I(K_{s,c^*}; \vec{y}^*_{N(s)})$ for a fixed set of values $\vec{y}^*_{N(s)}$ for its neighbors (line 4). Thereafter, the sniffers that have determined their channels send the determination to their neighbors (line 5), so that, in the next iteration, some of the neighbors (in P_{i+1}) can use the determination to compute their coverage improvements. Here, the sequence $\mathcal{P}$ can be determined a priori or through

Algorithm 8 Opportunistic Channel Assignment Algorithm

1: // Assume a partition $\mathcal{P} = \{P_i\}$ of the set S of all sniffers such that no two sniffers in any P_i are neighbors.
2: **for** $i = 1$ to $|\mathcal{P}|$ **do**
3: // All sniffers in P_i can choose their channels in parallel.
4: Each sniffer $s \in P_i$ tunes its radio to a channel $c^* \in C$ such that

$$I(K_{s,c^*}; \vec{y}^*_{N(s)}) = \max_{c \in C} I(K_{s,c}; \vec{y}^*_{N(s)}).$$

5: After determining its channel, the sniffer s sends the determination to its neighboring sniffers.
6: **end for**

an ad hoc coordination among sniffers, e.g., employing one of existing scheduling algorithms at the Medium Access Control (MAC) layer.

Theorem 3.3.2 *Given an solution to LP_{OSCA} that attains a constant factor α of the optimal value of LP_{OSCA}, OCAA guarantees to achieve at least $\alpha \cdot (1 - \frac{1}{e})$ ($\approx 0.632\alpha$) of the maximum monitoring coverage of OSCA.*

The proof is given in Appendix A.4. Here, the factor α comes from the approximate solution of $\mathrm{LP}_{\mathrm{OSCA}}$. However, note that we can make the approximate solution arbitrarily close to the optimal solution of $\mathrm{LP}_{\mathrm{OSCA}}$ as we increase the number of outer-level iterations of DA-$\mathrm{LP}_{\mathrm{OSCA}}$. Hence, due to Theorems 3.3.1 and 3.3.2, we finally have the following theorem.

Theorem 3.3.3 *DA-OSCA can always achieve at least $1 - \frac{1}{e}$ (≈ 0.632) of the maximum monitoring coverage of OSCA, regardless of the network topology and the channel assignment of nodes.*

3.4 ONLINE IMPLEMENTATION OF DA-OSCA

In this section, we present how to implement DA-OSCA to operate online so that DA-OSCA is agile and adapts incrementally to network changes, such as, changes to the channels assigned to nodes, changes in the usage of its channel by a node, and network topology changes due to mobility of nodes or arrivals/departures of sniffers. We present two operational modes of DA-OSCA—Mode-I and Mode-II, that are suitable for fast-varying and slow-varying networks, respectively. By developing the two operational modes, we enable DA-OSCA to operate in a more cost-effective manner for the two types of dynamic networks.

We first describe the procedure that sniffers need to perform, commonly for both operational modes, when they find arrivals/departures of their neighboring nodes/sniffers. Note that failures and recoveries of nodes/sniffers can be viewed as their departures and arrivals, respectively.

3.4.1 BASIC INFORMATION UPDATE

When sniffer s finds arrivals or departures of its neighboring nodes, it first updates its coverage-sets (i.e. $\mathcal{K}_s$). For the arrival of a new neighboring node n, the sniffer s that acts as a proxy for node n (for updating values of the node n's variables) introduces a set of new variables for node n, i.e., x_n, x_n^{aux} and p_n, and sets their initial values as follows: $x_n = 1$ if node n is covered (by any of its neighboring sniffers), and otherwise $x_n = 0$; $x_n^{\text{aux}} = x_n$; $p_n = 0$. For the departure of its neighboring node n, the sniffer s removes the set of the variables for node n. When new sniffer s arrives, it first creates its coverage-sets and its variables, i.e., $\vec{y}_s$ and $\vec{y}_s^{\text{aux}}$, and then sets their initial values as follows: $y_{s,c^*} = 1$ for $c^* \in C$ such that K_{s,c^*} achieves the maximum coverage improvement (according to Eq. (3.16)), and $y_{s,c} = 0$ for all $c \neq c^* \in C$; $\vec{y}_s^{\text{aux}} = \vec{y}_s$. When sniffer s leaves, one of its neighboring sniffers takes over the proxy duty that sniffer s had been doing.

Algorithm 9 DA-OSCA in Mode-I

1: **if** $t = k \cdot T_1$, $\forall k = 1, 2, \cdots$ **then**
2: Perform one outer-iteration of DA-LP$_{\text{OSCA}}$ (i.e., lines 3–11 of Alg. 7)
3: **if** $t = k \cdot (lT_1)$, $\forall k = 1, 2, \cdots$ **then**
4: Invoke OCAA
5: **end if**
6: **end if**

3.4.2 MODE-I: DA-OSCA FOR FAST-VARYING NETWORKS

In this mode, DA-OSCA operates *proactively* to adapt to frequent network changes. The rationale behind this proactive mode is that, when the network changes frequently, it is cost-effective to run DA-OSCA continuously, rather than running it on demand. This is because, as we will see in Mode-II, such a reactive operation of DA-OSCA will require global communications to evaluate the quality of the current monitoring coverage to determine when to start and also when to terminate. This process is costly.

The operation of DA-OSCA in Mode-I is presented in Alg 9. DA-OSCA executes one outer-level iteration of DA-LP$_{\text{OSCA}}$ every T_1 time (line 2), and invokes OCAA every lT_1, i.e., every l outer-level iterations of DA-LP$_{\text{OSCA}}$ (line 4). Intuitively, DA-OSCA keeps updating the primal and the dual variables (using DA-LP$_{\text{OSCA}}$) and periodically change the channel assignment of sniffers based on the updated values of $\vec{y}$.

3.4.3 MODE-II: DA-OSCA FOR SLOW-VARYING NETWORKS

In this mode, DA-OSCA operates *on demand*, i.e., only when it needs to change the channel assignment of sniffers to improve the degraded monitoring coverage. For this reactive operational mode, DA-OSCA needs a mechanism to evaluate the quality of monitoring coverage to determine whether the invocation of DA-OSCA is needed, and

Algorithm 10 An efficient information-aggregation procedure to evaluate the quality of monitoring coverage

1: // A pre-constructed spanning tree of sniffers is assumed.

2: **Aggregation of information.** This step is initiated by leaf sniffers and is executed sequentially along the levels of the spanning tree upwards before the root sniffer. At a level of the spanning tree, sniffer s computes:

$$C_s = \sum_{s' \in \mathrm{CS}(s)} C_{s'} + \sum_{n \in L(s)} w_n \cdot \min\left\{1, \sum_{(s,c): n \in K_{s,c}} y_{s,c}\right\}$$
$$D_s = \sum_{s' \in \mathrm{CS}(s)} D_{s'} + \sum_{n \in K_{s,c^*}} p_n + \sum_{n \in L(s)} [w_n - p_n]^+, \tag{3.17}$$

where $c^* \in \mathrm{argmax}_{c \in C} \sum_{n \in K_{s,c}} p_n$, $[x]^+ = \max\{x, 0\}$, and $\mathrm{CS}(s)$ and $L(s)$ denote the set of the child sniffers of sniffer s and the set of neighboring nodes of sniffer s, respectively. Thereafter, sniffer s sends G_s to its parent sniffer.

3: **Determination of solution quality.** The root sniffer computes C_{root} and D_{root} according to Eq. (3.17), and makes a decision of the termination of DA-LP$_{\mathrm{OSCA}}$ as follows: if $C_{\mathrm{root}} \geq \gamma \cdot D_{\mathrm{root}}$, then determines that the current channel assignment achieves the desired monitoring coverage. Thereafter, the root sniffer sends to its child sniffers a message to inform this determination.

4: **Distribution of determination.** The determination made by the root sniffer is delivered to all sniffers along the spanning tree.

also to check whether the iterations of DA-LP$_{\mathrm{OSCA}}$ are sufficiently close to the optimal solution so that DA-OSCA should terminate DA-LP$_{\mathrm{OSCA}}$ and round the solution with OCAA. Hence, in this subsection, we first develop a procedure to evaluate the quality of monitoring coverage, and then present how DA-OSCA employs the procedure to operate in the reactive mode.

We present an efficient information-aggregation procedure to evaluate the quality of monitoring coverage in Alg. 10. Basically, Alg. 10 estimates the gap between the current monitoring coverage and the maximum monitoring coverage, and then

Algorithm 11 DA-OSCA in Mode-II

1: **if** $t = k \cdot T_2$, $\forall k = 1, 2, \cdots$ **then**
2: **if** $r_{\mathrm{MC}} \leq \gamma_1$ (by invoking Alg. 10) **then**
3: // i.e., when the ratio of the current monitoring coverage to the maximum possible monitoring coverage is below a desired level γ_1
4: **while** $r_{\mathrm{LP}} \leq \gamma_2$ (by invoking Alg. 10) **do**
5: Perform N_o outer-iterations of DA-LP$_{\mathrm{OSCA}}$ (i.e., lines 3–11 of Alg. 7)
6: **end while**
7: Invoke OCAA
8: **end if**
9: **end if**

determines whether the estimate is above a desired level (that is specified by a predetermined value of γ). Here, the gap is defined as the ratio of the current monitoring coverage to the maximum monitoring coverage. To estimate the gap, Alg. 10 computes the current monitoring coverage (i.e., C_{root}) and the dual objective function value (i.e., D_{root}) since it follows from the duality theory [40, Ch. 5.1.3] that any dual objective function is an upper bound on the primal optimal value, which is the optimal value of LP$_{\mathrm{OSCA}}$, and thus is an upper bound on the maximum monitoring coverage. To compute them, Alg. 10 efficiently aggregates information through the spanning tree of sniffers (line 2), and then determines whether the current monitoring coverage is above the desired level by checking $C_{\mathrm{root}} \geq \gamma \cdot D_{\mathrm{root}}$ (line 3). Thus, this process does require collection of information in a hierarchical manner from all the sniffer nodes. Finally, the determination is distributed to all sniffers through the spanning tree. The proof of the correctness of Alg. 10 is given in Appendix A.5.

We now describe how DA-OSCA operates on demand by employing Alg. 10. We formally present the Mode-II of DA-OSCA in Alg. 11. In this mode, DA-OSCA evaluates the quality of the current monitoring coverage periodically, i.e., every T_2 time, by employing Alg. 10 (i.e., line 2 in Alg. 11). If the estimate (i.e., r_{MC}) of the gap

between the current monitoring coverage and the maximum monitoring coverage is above a desired level, DA-OSCA terminates doing nothing (i.e., when the condition line 2 is not met). Otherwise, DA-OSCA starts to solve the new OSCA that has resulted from the network changes (lines 4–7). To solve the problem, DA-OSCA runs N_o outer-level iterations of DA-LP$_{\mathrm{OSCA}}$. Here, N_o gives a trade-off between the cost due to checking the stopping criterion and the cost due to running more number of outer-level iterations of DA-LP$_{\mathrm{OSCA}}$ than required to reach the solution quality. Hence, N_o needs to be carefully chosen taking into account the convergence speed of DA-LP$_{\mathrm{OSCA}}$. DA-OSCA checks whether the ratio r_{LP} of the solution of DA-LP$_{\mathrm{OSCA}}$ at the current iteration is sufficiently close to the optimal solution of LP$_{\mathrm{OSCA}}$ by employing Alg. 10 with a pre-specified precision of γ_2 (line 4). Once a near-optimal solution to LP$_{\mathrm{OSCA}}$ is obtained, DA-OSCA terminates DA-LP$_{\mathrm{OSCA}}$ and then rounds the solution of LP$_{\mathrm{OSCA}}$ with OCAA to obtain an integer solution. Then, DA-OSCA terminates.

3.5 NOTES

In OSCA, we assume that all of the nodes and the sniffers have only one radio. However, the case, where nodes and sniffers are equipped with multiple radios, can be easily mapped to this single-radio case by regarding radios of a node (or a sniffer) as different nodes (or sniffers) with a single radio. One might think that, the single-radio case, which is mapped from the multi-radio case, needs an additional constraint that ensures each sniffer to tune its radios to different channels. However, even without the additional constraint, our algorithm will automatically determine a set of distinct channels for each sniffer's radios. This is because tuning two radios of a sniffer to the same channel in the multi-radio case implies choosing two coverage-sets that contain the same nodes, and this always gives a lower coverage than choosing either of the two coverage-sets and any other coverage-set.

For OSCA, one could consider a simple randomized rounding scheme that views a channel assignment of a sniffer as a random experiment, where a random variable is assigned to each sniffer, and each random variable is realized to one of the available channels with a probability of its fractional value obtained by solving $\mathrm{LP}_{\mathrm{OSCA}}$ (i.e. the LP relaxation of OSCA). It is easy to show (as in the proof of PRA in Section 2.6) that this randomized rounding scheme guarantees to achieve at least $1 - \frac{1}{e}$ (≈ 0.632) of the optimum of OSCA, in expectation. However, in order to achieve the expected guarantee of $1 - \frac{1}{e}$, the randomized rounding scheme requires sniffers to switch their channels a large number of times by repeatedly realizing their random variables with the same probability distribution. However, the delay of switching the radio channel is non-negligible[2]. Hence, with this randomized rounding scheme, sniffers would waste their time switching channels. Thus, we use a deterministic rounding scheme, which does not require sniffers to switch their channels but can achieve the same approximation ratio $1 - \frac{1}{e}$ *deterministically*.

Theorem 3.3.1 suggests that the value of d (which is the coefficient of the quadratic term in the objective function (3.6) of $\mathrm{QP}_{\mathrm{OSCA}}$) should be small so that the step size β can be chosen to a large value, thus leading to a larger improvement at each inner-level iteration. On the other hand, a small value of d will cause the objective function (3.6) of $\mathrm{QP}_{\mathrm{OSCA}}$ to be different from the objective function (3.1) of the original problem $\mathrm{LP}_{\mathrm{OSCA}}$, and hence require more outer-level iterations, thus potentially leading to slow convergence of DA-$\mathrm{LP}_{\mathrm{OSCA}}$. Therefore, the value of d should be tuned carefully.

3.6 SIMULATION

We conduct simulations to demonstrate the efficacy of the two modes of DA-OSCA for two kinds of networks—random networks and scale-free networks. In random networks, nodes are randomly deployed with a uniform distribution. In scale-free networks, nodes are deployed such that the distribution $f(d)$ of nodes with degree d

[2]Current estimate for switching delay between channels in the same frequency band with commodity IEEE 802.11 hardware is in the range of a few milliseconds [31] to a few hundred microseconds [30].

follows a power law in a form of d^{-r}. The performance of DA-OSCA largely depends on the network topology, and these two kinds of networks have a significant difference in their topologies. Also, their topologies are observed in many practical networks[3].

We choose the settings of the network and the parameters of DA-OSCA as follows. There are 500 nodes of identical weight and 50 sniffers in the network. The number of available wireless channels is three (i.e., $|C| = 3$), same as the number of non-overlapping wireless channels in IEEE 802.11. For random networks, we randomly place nodes and sniffers on a 1×1 square area, and set the receiving range of sniffers to 0.15. For scale-free networks, the parameter r of the distribution $f(d) = O(d^{-r})$ is chosen as $2 < r < 3$. In scale-free networks, we pick nodes with highest degrees as sniffers. This is reasonable because thereby we can achieve a higher monitoring coverage than picking them randomly. The parameters of DA-OSCA are set as $S = 1$ (i.e., the number of inner-level iterations is 2), $d = 0.5$, and $\beta = 1/(B_1 B_2)$.

We conduct two experiments in each network. In one experiment, we evaluate the Mode-I of DA-OSCA in fast-varying networks, and in the other experiment, we evaluate the Mode-II of DA-OSCA in slow-varying networks. In all experiments, we demonstrate how monitoring coverage evolves as DA-OSCA adapts to the changes to the channels assigned to nodes. The channel of each node is assigned randomly to channel 1, 2, or, 3 with probabilities 0.2, 0.3, and 0.5, respectively. The channel assignment of a fraction of nodes (randomly chosen between 10% and 40%) changes every 5 time units and every 100 time units in the fast-varying and slow-varying networks, respectively. Here, we one time unit as the time that DA-OSCA takes to run one outer-level iteration of DA-LP$_{\text{OSCA}}$. In Mode-I, we set the parameters as $T_1 = 1$ and $l = 3$. In Mode-II, we set the parameters as $T_2 = 30$, $\gamma_1 = 0.8$, $\gamma_2 = 0.8$, and $N_o = 1$. Here, we set the values of γ_1 and γ_2 taking into account that Alg. 10 underestimates the quality of monitoring coverage since its uses an upper

[3]Wireless networks where mobile users move randomly can be viewed as random networks, and many empirically observed networks, such as the world wide web and the Internet, have been found to be scale-free.

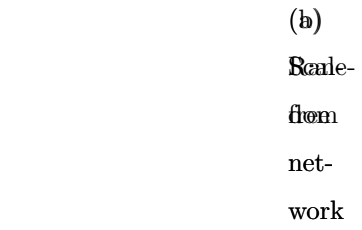

Fig. 3.2.: Mode-I: DA-OSCA for fast-varying networks where the LP rounding executes continuously with updated coverage information.

bound on the maximum coverage. In all experiments, the results are the averages over 10 different network realizations.

Figure 3.2(a) and (b) show how the monitoring coverage evolves as DA-OSCA in Mode-I runs in a random networks and in a scale-free network, respectively. Here, the monitoring coverage is normalized by the optimal value of $\mathrm{LP}_{\mathrm{OSCA}}$, which is an upper bound on the maximum monitoring coverage. In this experiment, DA-OSCA adjusts the channel assignment of sniffers after 10 time units since the simulation begins. For both networks, we observe that the fractional monitoring coverage due to the solution of DA-$\mathrm{LP}_{\mathrm{OSCA}}$ converges rapidly (within 10 time units) until it reaches about 90% of the maximum coverage, and it flattens out after it goes above 90% of the maximum coverage. We also observe that DA-$\mathrm{LP}_{\mathrm{OSCA}}$ quickly recovers the degraded fractional monitoring coverage, due to the changes of the channels assigned to nodes. Within only a few time units, the new channel assignment of sniffers by OCAA attains a high monitoring coverage, maintained above 95% of the maximum coverage. A notable difference between these results (also observed in Fig. 3.3(a), (b)) is that, in random networks, the channel changes of nodes incur less degradation of the monitoring coverage than in scale-free networks, and DA-OSCA achieves a higher monitoring coverage in random networks. This is, possibly, because in random networks sniffers are uniformly distributed and this makes sniffers have a better topological coverage than in scale-free networks.

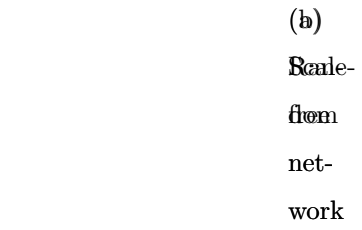

Fig. 3.3.: Mode-II: DA-OSCA for slow-varying networks where the algorithm is executed on demand when a change is detected in the network.

Figure 3.3(a) and (b) demonstrate the on-demand operation of DA-OSCA in Mode-II for slow-varying networks. In both figures, we see observe large intervals of time where the monitoring coverage is flat. This means that, through Alg. 10, DA-OSCA determined that the monitoring coverage meets the desired level, and then terminates without any processing, thereby saving unnecessary cost. We notice that when the network changes, the monitoring coverage suffers (note the dips) but quickly recovers (always within 20 time units) as OCAA is executed on demand. Also, we observe that the improved monitoring coverage after the execution of DA-OSCA is higher than required (recall that $\gamma_2 = 0.8$). This can be explained by the following two facts. The first is that OCAA often improves the fractional solution while rounding it, which can be observed from Fig. 3.2(a) and (b). The second is that since Alg. 10 underestimates the quality of monitoring coverage, DA-OSCA may run the outer-iterations of DA-LP$_{\text{OSCA}}$ more than required.

Both experiments show that DA-OSCA is able to adapt to different kinds of networks, fast-varying and slow-varying, and is able to operate incrementally with respect to network changes. By setting the values of γ, the system owner can control how close she wants the normalized monitoring coverage to get to the value of one.

3.7 CONCLUSION

In this chapter, we presented a distributed online algorithm for the optimal channel assignment problem for passive monitoring in multi-channel wireless networks. Our algorithm preserves the approximation ratio $1 - \frac{1}{e}$ that the existing centralized algorithms have previously attained, while providing a distributed solution that is amenable to online implementation. We present two operational modes of our algorithm for cost-effective operation in two types of networks that have different rates of network changes. Simulation results demonstrate the effectiveness of the two modes of our algorithm.

4. OPTIMAL SNIFFER-CHANNEL ASSIGNMENT FOR RELIABLE MONITORING IN MULTI-CHANNEL WIRELESS NETWORKS

4.1 INTRODUCTION

In the previous chapters, we assumed that sniffers are perfect, i.e., do not fail. This implies that once a node has at least one sniffer within its transmission range operating on the same channel, the node's activity will always be monitored without any error. However, in practice, sniffers may intermittently/periodically/permanently stop functioning and/or generate erroneous reports on monitoring results. There are various reasons for this including operational failure, poor reception due to packet collisions or poor channel conditions, sleep mode for energy saving, and compromise by an adversary. The failure and malfunctions of sniffers decrease the quality of monitoring, and consequently degrade the network performance.

In this chapter, we allow for imperfect sniffers that may probabilistically generate errors on monitoring. In this scenario, we wish to still maintain the accuracy of the passive monitoring above a certain level. Our approach to this end is to provide multiple covers (i.e., sniffer redundancy) to each node. That is, each node is assigned a coverage requirement that is the minimum number of sniffers required for reliably monitoring the node. In this approach, a problem that naturally arises is how to assign a set of channels to sniffers' radios such that the coverage requirements of all nodes are satisfied. We refer to this problem as the *Full-Coverage Reliable Monitoring* (FCRM). We, however, show that it is NP-hard to find any feasible solution to FCRM (i.e., any sniffer-channel assignment that satisfies all of the coverage requirements). Alternatively, we turn our attention to the corresponding optimization problem to FCRM, i.e., how to find a sniffer-channel assignment that maximizes the number

(or the total weight) of nodes being reliably monitored. We call this problem the *Maximum-Coverage Reliable Monitoring* (MCRM). Note that, by solving MCRM, we can determine the maximum coverage achievable by the given set of sniffers, and also obtain the answer to FCRM by verifying if the maximum coverage obtained meets the full coverage. However, MCRM is also NP-hard, as implied by the reduction from FCRM to MCRM.

MCRM can be viewed as a generalization of the maximum-coverage monitoring problem with perfect sniffers studied in the previous chapters. In other words, the maximum-coverage monitoring problem with perfect sniffers is a special case of MCRM with every node requiring a *single* cover, i.e., only one sniffer, to be reliably monitored. However, we find out that the general MCRM, i.e., the MCRM with at least one node requiring multiple covers (MCRM-MC), is different in nature from the MCRM with single cover (MCRM-SC). We show this by proving that the objective functions generated by MCRM-MC do not preserve the *submodular* property (refer to Section 4.2.2) that holds for those generated by MCRM-SC. Due to the loss of the submodularity in MCRM-MC, the performance guarantees of the approximation algorithms for solving MCRM-SC no longer hold in MCRM-MC.

In this chapter, we propose a variety of approximation algorithms to solve MCRM-MC based on two basic approaches—one is greedy approach and the other is relaxation-and-rounding (RaR) approach. First, we develop two variants of a look-ahead greedy algorithm, which are different from the naive greedy algorithms in that they make a greedy decision at each step by considering not only the current step but also future steps. We next develop two relaxation schemes based on Linear Program (LP) and SemiDefinite Programing (SDP), and two rounding algorithms—Randomized Rounding Algorithm (RRA) and Greedy Rounding Algorithm (GRA), leading to four variants of an RaR algorithm. We present a comparative study of the proposed algorithms through simulations. We evaluate the proposed algorithms in two different topologies of networks—random and scale-free networks—in terms of two metrics—coverage and running time.

The rest of the chapter is organized as follows. Section 4.2 describes the problem formulation. Section 4.3 presents the two look-ahead greedy algorithms, and Section 4.4 presents the four variants of the RaR algorithm. Section 4.5 provides asymptotic time-complexity analysis of the proposed algorithms. Section 4.6 presents performance evaluation of the proposed algorithms through simulation. Finally, Section 4.7 discusses conclusion.

4.2 PROBLEM FORMULATION

We are given a set N of nodes to be monitored, and each node $n \in N$ is tuned to a wireless channel chosen from a set C of available wireless channels, where $|C| \geq 2$. The channels of nodes are chosen according to one of existing channel assignment algorithms in the literature (e.g., [15, 17, 19]). Each node n is given a coverage requirement r_n that is a positive integer and denotes the minimum number of sniffers required to reliably monitor node n. The value of the coverage requirements will depend on the failure model of sniffers (e.g., false negatives/positives), the desired accuracy of monitoring, and monitoring applications. We say that node n is *covered* if it is overheard by at least r_n sniffers operating on the same channel as the node. Also, each node n is given a non-negative weight w_n. These weights of nodes can be used to capture various application-specific objectives of monitoring. For example, one can assign higher weights to the nodes that transmit larger volumes of data, thereby biasing our algorithm to monitor such nodes more. Or, for security monitoring, one can assign the weights by taking into account nodes trustworthiness computed based on previous monitoring results. Here, a node that has been found to be compromised before (and repaired thereafter) will be assigned a higher weight.

We are given a set S of sniffers, each of which needs to determine a wireless channel from C to tune its radio to. We are given a collection of coverage-sets $\mathcal{K} = \{K_{s,c} \subseteq N : s \in S, c \in C\}$, where a *coverage-set* $K_{s,c}$ includes the nodes that can be overheard by sniffer s being tuned to channel c. We define a *sniffer-channel*

assignment as a subset of $\mathcal{K}$ that includes only one coverage-set for each sniffer. Here, the constraint that only one coverage-set for each sniffer can be included in a coverage-set is due to the fact that each sniffer has only one radio and hence can tune its radio to only one channel at a time.

4.2.1 FULL-COVERAGE RELIABLE MONITORING

We first consider a decision problem to determine whether or not there exists a sniffer-channel assignment that achieves the full coverage, i.e., covers all nodes in N. We refer to this problem as the *Full-Coverage Reliable Monitoring* (FCRM). We denote the FCRM with k channels and a set $\vec{r} = (r_n : n \in N)$ of coverage requirements of nodes by FCRM$(k, \vec{r})$.

Theorem 4.2.1 *For fixed $k \geq 2$ and $\vec{r}$, FCRM$(k, \vec{r})$ is NP-hard.*

Proof We let $[m] = \{1, \ldots, m\}$ and $\vec{1} = (1, \ldots, 1)$. To show the theorem, we use a reduction from FCRM$(2, \vec{1})$, which is an NP-hard problem [38, Theorem 1], to FCRM$(k, \vec{r})$ for any fixed k and $\vec{r}$. Given a collection of coverage-sets $\mathcal{K} = \{K_{s,c} : s \in [m], c \in [2]\}$, we augment $\mathcal{K}$ to $\mathcal{K}^{\text{aug}} = \{K^{\text{aug}}_{s,c} : s \in [m + R]\}, c \in [k]\}$, where $R = \max\{r_n : n \in N\} - 1$, as follows: $\forall s \in [m]$, $K^{\text{aug}}_{s,c} = K_{s,c}$ for $c \in [2]$ and $K^{\text{aug}}_{s,c} = \emptyset$ for $c \in \{3, \ldots, k\}$; $\forall s \in \{m + 1, \ldots, m + R\}$, $K^{\text{aug}}_{s,1} = \{n \in N : r_n \geq s - m + 1\}$ and $K^{\text{aug}}_{s,c} = \emptyset$ for $c \in \{2, \ldots, k\}$. Note that, with the additional sniffers, i.e., the sniffers $m + 1, \ldots, m + R$, we can achieve only a partial coverage of at most $r_n - 1$ for every node n, and also that the additional channels give only zero coverage. The reduction can be done in polynomial time. Given a sniffer-channel assignment $\mathcal{A}$ for an instance of FCRM$(2, \vec{1})$ with $\mathcal{K}$, we define a sniffer-channel assignment $\mathcal{A}^{\text{aux}} = \mathcal{A} \cup \{K_{s,1} : s \in \{m + 1, \ldots, m + R\}\}$. It is easy to see that $\mathcal{A}$ achieves the full coverage for an instance of FCRM$(2, \vec{1})$ with $\mathcal{K}$ if and only if $\mathcal{A}^{\text{aux}}$ attains the full coverage for the corresponding instance of FCRM$(k, \vec{r})$ with $\mathcal{K}^{\text{aug}}$. This means that if FCRM$(k, \vec{r})$ can be solved in polynomial time, then so can FCRM$(2, \vec{1})$ be. However,

since FCRM$(2, \vec{1})$ is NP-hard, this is a contradiction if P $\neq$ NP. Thus, the theorem follows. ■

Hence, we cannot find the answer to FCRM in polynomial time.

4.2.2 MAXIMUM-COVERAGE RELIABLE MONITORING

Alternatively, we consider a coverage maximization problem where we wish to maximize the total weights of nodes being covered by judiciously assigning channels to sniffers. We refer to this problem as the *Maximum-Coverage Reliable Monitoring* (MCRM). We denote the MCRM with k channels and a set $\vec{r} = (r_n : n \in N)$ of coverage requirements of nodes by MCRM$(k, \vec{r})$. Also, we denote MCRM$(k, \vec{1})$ with $k \geq 2$ by MCRM-SC (i.e., MCRM with single cover) and MCRM$(k, \vec{r})$ with $k \geq 2$ and $r_n \geq 2$ for some nodes $n \in N$ by MCRM-MC (i.e., MCRM with multiple covers).

The corollary below follows from Theorem 4.2.1, since we can find the answer to FCRM by solving MCRM and then verifying whether the full coverage is achieved.

Corollary 4.2.1 *For fixed $k \geq 2$ and $\vec{r}$, MCRM$(k, \vec{r})$ is NP-hard.*

This means that the computational complexity to obtain an optimal solution to MCRM grows exponentially with the number of sniffers, unless P = NP.

Corollary 4.2.2 *For fixed $k \geq 2$ and $\vec{r}$, it is NP-hard to approximate MCRM$(k, \vec{r})$ within a factor of $\frac{7}{8} + \epsilon$ of the maximum coverage for any $\epsilon > 0$,*

Proof In the proof of Theorem 4.2.1, we have shown that FCRM$(2, \vec{1})$ can be reduced to FCRM$(k, \vec{r})$ for any $k \geq 2$ and any $\vec{r}$. Also, as mentioned above, FCRM$(k, \vec{r})$ can be reduced to MCRM$(k, \vec{r})$. Hence, FCRM$(2, \vec{1})$ can be reduced to MCRM$(k, \vec{r})$. On the other hand, it is NP-hard to approximate FCRM$(2, \vec{1})$ within a factor of $\frac{7}{8} + \epsilon$ for any $\epsilon > 0$ [38, Corollary 2]. Thus, the corollary follows. ■

This implies that the best approximation ratio attainable for MCRM is at most $\frac{7}{8}$.

Non-submodularity of MCRM-MC. Submodularity is an important property in discrete optimization which allows to efficiently find provably (near-)optimal solutions, similarly to convexity in continuous optimization [42]. A real-valued function $f : 2^S \to \mathbb{R}$, defined on the subsets of a finite set S, is said *submodular* if the following inequality holds for any two subsets X and Y of S:

$$f(X \cap Y) + f(X \cup Y) \leq f(X) + f(Y).$$

The submodularity is better characterized by the definition: f is submodular if and only if, for any $X \subseteq S - \{a\}$, the derived set function $\Delta f(a|X) \triangleq f(X \cup \{a\}) - f(X)$ is monotonically increasing, i.e., $\Delta f(a|X) \geq \Delta f(a|Y)$ for $X \subseteq Y$. Intuitively, submodularity is a *diminishing-return* property.

On the other hand, non-submodular functions are known to be difficult to deal with. In the the literature of theoretical computer science, there are little results on the provable performance guarantees for non-submodular functions. Also, many greedy heuristics with good performance demonstrated in computational experiments cannot receive a theoretical analysis due to the difficulty on dealing with non-submodular functions [43].

For MCRM, we can define the objective function as a (real-valued) weight function $w : 2^{\mathcal{K}} \to \mathbb{R}$, defined on collections of coverage-sets in $\mathcal{K}$, which computes the total weights of the nodes covered by a collection of coverage-sets. In MCRM-SC, a node is covered if it is monitored by only one sniffer. Hence, adding a coverage-set to a smaller collection $\mathcal{C}$ of coverage-sets earns more increment on the total weight than adding it to a larger collection $\mathcal{C}'$ including $\mathcal{C}$. Thus, we have the following theorem.

Theorem 4.2.2 *For MCRM*$(k, \vec{1})$ *where* $k \geq 2$*, the weight function* w *is submodular.*

Due to the submodularity, MCRM-SC can be approximated within a factor of $1 - \frac{1}{e}$ (≈ 0.632) of the maximum coverage.

On the other hand, in MCRM-MC, the weight function w is no longer submodular.

Theorem 4.2.3 *For MCRM$(k, \vec{r})$ where $k \geq 2$ and $r_n \geq 2$ for some nodes $n \in N$, the weight function w is not submodular.*

Proof We show the theorem by a counter example. Assume that there exists a node $n \in N$ such that $r_n \geq 2$. We construct an instance of MCRM$(k, \vec{r})$ where $w_n = 1$ (i.e., the weight of node n is 1) and $K_{1,1} = \cdots = K_{r_n,1} = \{n\}$ (i.e., sniffers $1, \ldots, r_n$ can overhear only node n by tuning their radios to channel 1). Consider two collections of coverage sets, $\mathcal{C} = \emptyset$ and $\mathcal{C}' = \{K_{1,1}, \ldots, K_{r_n-1,1}\}$. Then, it is follow that $\Delta w(K_{r_n,1}|\,\mathcal{C}) = 0$ and $\Delta w(K_{r_n,1}|\,\mathcal{C}') = 1$. Hence, we have $\Delta w(K_{r_n,1}|\,\mathcal{C}) < \Delta w(K_{r_n,1}|\,\mathcal{C}')$ for $\mathcal{C} \subset \mathcal{C}'$. Thus, the theorem holds. ■

4.3 LOOK-AHEAD GREEDY ALGORITHMS

We first consider a greedy strategy to solve MCRM. We can employ GR-MCMC in Section 2.4.2 to solve MCRM, which picks at each step the coverage-set that maximizes the coverage improvement, i.e., the total weight of uncovered nodes, among all coverage-sets of the sniffers whose channel assignment is not determined yet. GR-MCMC can approximate MCRM-SC within a factor of $\frac{1}{2}$ of the maximum coverage. However, due to the non-submodularity of MCRM-MC, the performance guarantee of the greedy algorithm no longer holds for MCRM-MC.

One may consider a straightforward extension of GR-MCMC to solve MCRM-MC. In MCRM-MC, two extensions of the greedy algorithm can be considered; at each step, one picks the coverage-set that achieves the maximum coverage improvement, while the other picks the coverage-set that maximizes the total weight of uncovered nodes. Note that these two greedy extensions result in different solutions. To see this, observe that, in MCRM-MC, uncovered nodes can have a partial coverage of

$1, \ldots, r_n - 2$ or $r_n - 1$, other than zero coverage. Hence, when a coverage-set is picked at a step, only the uncovered nodes of the partial coverage of $r_n - 1$ in the coverage-set can be covered. However, the two greedy extensions both show a poor performance due to their myopic nature, which is illustrated by the following examples (and is also shown by the simulation results in Section 4.6).

First, for the former greedy extension, consider the following example: $K_{1,1} = \{1, 2, 3, 4\}, K_{1,2} = \{5, 6, 7\}, K_{2,1} = \{1\}, K_{2,2} = \{5, 6, 7\}, K_{3,1} = \{2\}, K_{3,2} = \{8, 9, 10\}, K_{4,1} = \{3\}, K_{4,2} = \{8, 9, 10\}$; $w_n = 1$ and $r_n = 2$ for all $n \in \{1, \ldots, 10\}$. Provided that ties are broken by choosing the coverage-set that maximizes the total weight of uncovered nodes, the former greedy extension will yield a solution $\{K_{1,1}, K_{2,1}, K_{3,1}, K_{4,1}\}$ leading to a coverage of 3, while the optimal solution is $\{K_{1,2}, K_{2,2}, K_{3,2}, K_{4,2}\}$ leading to a coverage of 6. In this example, the former greedy extension makes myopic decisions at the steps 2, 3 and 4 to maximize the coverage improvement at each step. Next, for the latter greedy extension, consider the following example: $K_{1,1} = \{1, 2, 3, 4\}, K_{1,2} = \{5, 6, 7\}, K_{2,1} = \{1, 2\}, K_{2,2} = \{5, 6\}$; $w_n = 1$ and $r_n = 2$ for all $n \in \{1, \ldots, 7\}$. The latter greedy extension will yields a solution $\{K_{1,1}, K_{2,1}\}$ leading to zero coverage, while the optimal solution is $\{K_{1,2}, K_{2,2}\}$ leading to a coverage of 2. In this example, the latter greedy extension chooses at each step the coverage-set of the maximum total weight of uncovered nodes, without verifying if such uncovered nodes can be indeed covered at later steps. As shown in these two examples, both of the naive greedy extensions make poor decisions due to their myopic nature.

Inspired by the observation through the illustrative examples above, we design two look-ahead greedy algorithms to solve MCRM-MC, shown in Alg. 12 and Alg. 13. Both of the look-ahead greedy algorithms have a parameter $t \in \{0, \ldots, |S| - 1\}$, which determines how far the algorithm looks ahead. It is reasonable to set $t = \max_{n \in N} r_n - 1$, because it requires at least r_n sniffers to cover node n. If we set $t = |S| - 1$, both of the look-ahead greedy algorithms will solve MCRM exactly, i.e., always yield an optimal solution to MCRM. However, the computational complexity will grow exponentially with $|S|$ (i.e., the number of sniffers).

Algorithm 12 Look-t-Steps-Ahead Greedy Algorithm

1: $\mathcal{G} \leftarrow \emptyset$, $S' \leftarrow S$

2: **while** $|S'| \neq 0$ **do**

3: $t' \leftarrow \min\{t+1, |S'|\}$

4: $\mathcal{P} \leftarrow \left\{\{K_{s_1,c_1}, \ldots, K_{s_{t'},c_{t'}}\} : s_i \in S',\ c_i \in C\ \forall i, \text{ and } s_i \neq s_j \text{ if } i \neq j\right\}$

// i.e., $\mathcal{P}$ is the set of all possible channel assignments for any t' sniffers in S'

5: Find $\mathcal{C}^* \in \mathcal{P}$ such that

$$\Delta w\left(\mathcal{C}^* | \mathcal{G}\right) = \max_{\forall \mathcal{C} \in \mathcal{P}} \Delta w\left(\mathcal{C} | \mathcal{G}\right)$$

// i.e., $\mathcal{C}^*$ achieves the maximum coverage improvement for any t' sniffers whose channels are not yet determined and any channel assignment for them

6: Find $K_{s^*,c^*} \in \mathcal{C}^*$ such that

$$\Delta w\left(\{K_{s^*,c^*}\} | \mathcal{G}\right) = \max_{\forall (s,c) \in P} \Delta w\left(\{K_{s,c}\} | \mathcal{G}\right)$$

// i.e., K_{s^*,c^*} achieves the maximum coverage improvement for any pair of sniffer and channel in $\mathcal{C}^*$

// where the ties are broken by choosing a coverage-set that maximizes the total weight of uncovered nodes

7: $\mathcal{G} \leftarrow \mathcal{G} \cup \{K_{s^*,c^*}\}$

8: $S' \leftarrow S' - \{s^*\}$

9: **end while**

10: **return** $\mathcal{G}$

Alg. 12 has a fixed number $|S|$ of steps. At each step, the algorithm looks $t'-1$ steps ahead to find a coverage-set that is best for the current step and the next $t'-1$ steps. Here, t' is the minimum of the parameter t and the number $|S'|$ of the remaining steps of the algorithm. To find the best coverage-set, it first finds a collection $\mathcal{C}^*$ of t' coverage-sets that achieve the maximum coverage improvement for the current step and the next $t'-1$ steps (line 5). Then, among the coverage-sets in $\mathcal{C}^*$, it chooses a

Algorithm 13 t-Sniffers-at-One-Step Greedy Algorithm

1: $\mathcal{G} \leftarrow \emptyset$

2: **while** $|S'| \neq 0$ **do**

3: $\mathcal{Q} \leftarrow \big\{\{K_{s_1,c_1}, \ldots, K_{s_{t'},c_{t'}}\} : t' \leq \min\{t+1, |S'|\},\ s_i \in S',\ c_i \in C\ \forall i,$ and $s_i \neq s_j$ if $i \neq j\big\}$

// i.e., $\mathcal{Q}$ is the set of all possible channel assignments for any t' ($\leq t+1$) sniffers in S'

4: Find $\mathcal{C}^* \in \mathcal{Q}$ such that

$$\frac{\Delta w\,(\mathcal{C}^*|\mathcal{G})}{|\mathcal{C}^*|} = \max_{\mathcal{C}\in\mathcal{Q}} \frac{\Delta w\,(\mathcal{C}|\mathcal{G})}{|\mathcal{C}|}$$

// i.e., $\mathcal{C}^*$ achieves the maximum *per-sniffer* coverage improvement for any $t' \leq t+1$ sniffers whose channels are not yet determined and any channel assignment for them

5: $\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{C}^*$

6: $S' \leftarrow S' - s(\mathcal{C}^*)$, where $s(\mathcal{C}^*)$ denotes the set of the sniffers chosen by $\mathcal{C}^*$

7: **end while**

8: **return** $\mathcal{G}$

coverage-set K_{s^*,c^*} that achieves the maximum coverage improvement at the current step (lines 6 and 7).

Alg. 13 has a variable number of steps, depending on the number of coverage-sets chosen at the steps. At each step, the algorithm chooses the collection $\mathcal{C}^*$ of coverage-sets with $|\mathcal{C}^*| \leq t+1$ that maximizes the *per-sniffer* coverage improvement among all possible channel assignments for any t' sniffers whose channel assignment is not yet determined (lines 4 and 5).

4.4 RELAXATION-AND-ROUNDING ALGORITHMS

In this section, we design relaxation-and-rounding (RaR) algorithms to solve MCRM. RaR is a highly effective technique to solve NP-hard optimization problems. Intuitively, RaR algorithms first solve a relaxed problem to the given optimization problem that is solvable in polynomial time, thereby gaining information about the optimal solution, and then finds a good approximate solution based on the information gained. The formal steps involved in RaR algorithms are:

- Step 1: Formulate the given optimization problem into an integer program (IP)
- Step 2: Transform the IP into a relaxed program where the integer constraints are relaxed and that is solvable in polynomial time
- Step 3: Solve the relaxed program and thereby obtain the optimal solution to the relaxed program
- Step 4: Round the non-integer values of the optimal solution to an integer value in order to obtain a feasible solution to the original IP

At Step 2, an important issue is to find as strong a relaxed program as possible while keeping the relaxed program solvable in polynomial time. The benefit of a stronger relaxed program (i.e., that has a smaller set of constraints including the optimal IP solution) lie in two folds. First, it often leads to a better approximate solution to the IP, since a stronger relaxed program will likely yield a non-integer solution closer to the optimal IP solution. Second, it will possibly provide a better estimate (i.e., upper bound) of the maximum achievable coverage. At Step 4, a challenging goal is to minimize the degradation of the quality of the resulting integer solution so as to obtain an integer solution that is as close to the optimal IP solution as possible.

4.4.1 LP-BASED AND SDP-BASED RELAXATIONS

We present two relaxations. One is linear program (LP) based relaxation, and the other is semidefinite program (SDP) based relaxation. We define a set of indicator variables for the formulation of the two relaxations. We assign an indicator variable $x_n \in \{0,1\}$ to each node $n \in N$, and $x_n = 1$ indicates that node n is covered by the given solution. We assign an indicator variable $y_{s,c} \in \{0,1\}$ to a coverage-set $K_{s,c} \in \mathcal{K}$, and $y_{s,c} = 1$ indicates that sniffer s will be tuned to channel c.

LP-based relaxation. We first formulate MCRM into the following integer linear program (ILP), denoted by ILP_{MCRM}:

$$\text{maximize} \quad \sum_{n \in N} w_n x_n \tag{4.1}$$

$$\text{subject to} \quad \sum_{c \in C} y_{s,c} = 1 \qquad \forall s \in S, \tag{4.2}$$

$$x_n \leq \frac{1}{r_n} \sum_{s,c:\, n \in K_{s,c}} y_{s,c} \qquad \forall n \in N, \tag{4.3}$$

$$x_n,\, y_{s,c} \in \{0,1\} \qquad \forall n \in N,\, s \in S,\, c \in C. \tag{4.4}$$

The constraint (4.2) is due to the fact that each sniffer's radio can be tuned to only one channel at a time. The objective function (4.1) together with the constraints (4.3) and (4.4) makes $x_n = 1$ if at least r_n coverage-sets that includes the node n are chosen for a solution, and otherwise makes $x_n = 0$.

We transform ILP_{MCRM} into the following LP relaxation, denoted by LP_{MCRM}:

$$\text{maximize} \quad \sum_{n \in N} w_n x_n \tag{4.5}$$

$$\text{subject to} \quad \sum_{c \in C} y_{s,c} = 1 \qquad \forall s \in S, \tag{4.6}$$

$$x_n \leq \frac{1}{r_n} \sum_{s,c:\, n \in K_{s,c}} y_{s,c} \qquad \forall n \in N, \tag{4.7}$$

$$0 \leq x_n,\, y_{s,c} \leq 1 \qquad \forall n \in N,\, s \in S,\, c \in C, \tag{4.8}$$

$$x_n \left(|\{(s,c) : n \in K_{s,c}\}| - r_n \right) \geq 0 \qquad \forall n \in N. \tag{4.9}$$

The integer constraint (4.4) in ILP_{MCRM} is relaxed to the fractional constraint (4.8). Also, we add the constraint (4.9) to make a stronger LP relaxation. Note that the objective function (4.1) together with the constraints (4.8) and (4.9) makes $x_n = 0$ if the number of sniffers overhearing node n is smaller than the node n's coverage requirement r_n. Thus, the constraint (4.9) guides the algorithm that solves LP_{MCRM} to make a right decision, so that it allocates no monitoring resources to the nodes that are impossible to cover due to the lack of the sufficient number of sniffers neighboring to them.

SDP-based relaxation. We first formulate MCRM into the following quadratically constrained linear program, denoted by $\text{QCLP}_{\text{MCRM}}$:

$$\text{maximize} \quad \sum_{n \in N} w_n x_n \tag{4.10}$$

$$\text{subject to} \quad \sum_{c \in C} y_{s,c} = 1 \qquad \forall s \in S, \tag{4.11}$$

$$x_n \left(\frac{1}{r_n} \sum_{s,c:\, n \in K_{s,c}} y_{s,c} - 1 \right) \geq 0 \qquad \forall n \in N, \tag{4.12}$$

$$y_{s,c}(y_{s,c} - 1) = 0 \qquad \forall s \in S,\, c \in C, \tag{4.13}$$

$$x_n(x_n - 1) = 0 \qquad \forall n \in N. \tag{4.14}$$

The constraints (4.13) and (4.14) represent the integer constraints of x_n and $y_{s,c}$. The objective function (4.10) together with the constraint (4.12) makes $x_n = 1$ if node n is overheard by at least r_n sniffers, and otherwise makes $x_n = 0$.

We now add the constraints (4.7)–(4.9) to $\text{QCLP}_{\text{MCRM}}$, and transform the $\text{QCLP}_{\text{MCRM}}$ with the additional constraints (4.7)–(4.9) into a SDP relaxation. Although the additional constraints (4.7)–(4.9) are redundant in $\text{QCLP}_{\text{MCRM}}$, as we will see later, they are needed in the SDP relaxation to be obtained. We define $\vec{z} = (x_1, \ldots, x_{|N|}, y_{1,1}, \ldots, y_{|S|,|C|}) \in \mathbb{R}^{|N|+|S|\cdot|C|}$, and denote the i-th entry of $\vec{z}$ by z_i. We define a symmetric square matrix M of order $|N| + |S| \cdot |C| + 1$ as

$$M = \begin{pmatrix} 1 & \vec{z} \\ \vec{z}^T & Z \end{pmatrix},$$

where Z is a symmetric square matrix of order $|N| + |S| \cdot |C|$ whose entry in the i-th row and the j-th column is denoted by $Z_{i,j}$. We can rewrite the $\text{QCLP}_{\text{MCRM}}$ with the additional constraints (4.7)–(4.9) into the following form:

$$\begin{aligned} \text{maximize} \quad & W \bullet M && && (4.15)\\ \text{subject to} \quad & A_i \bullet M \leq b_i, && i \in I && (4.16)\\ & Z = \vec{z}^T \vec{z}. && && (4.17) \end{aligned}$$

Here, W and A_i are symmetric square matrices of order $|N| + |S| \cdot |C|$, b_i is a real number, and I is an index set. The notation $\bullet$ denotes the Frobenius inner product, i.e., $W \bullet M = \sum_{i,j} W_{i,j} M_{i,j}$, and $(\cdot)^T$ denotes the matrix transpose. Note that, due to the constraint (4.17), the entry $Z_{i,j}$ of the matrix Z is equal to the quadratic term $z_i z_j$.

We transform Eqs. (4.15)–(4.17) into the following SDP relaxation, denoted by SDP_{MCRM}:

$$\begin{aligned} \text{maximize} \quad & W \bullet M && && (4.18)\\ \text{subject to} \quad & A_i \bullet M \leq b_i, && i \in I && (4.19)\\ & M \succeq 0 \quad (\Leftrightarrow Z - \vec{z}^T \vec{z} \succeq 0). && && (4.20) \end{aligned}$$

Here, $M \succeq 0$ means that the matrix M must be positive semidefinite, i.e., satisfy $\vec{v} M \vec{v}^T \geq 0$ for any real vector $\vec{v}$. SDP_{MCRM} is a relaxed program of $\text{QCLP}_{\text{MCRM}}$ since a zero matrix is positive semidefinite and hence $Z - \vec{z}^T \vec{z} = 0$ implies $Z - \vec{z}^T \vec{z} \succeq 0$. In SDP_{MCRM}, $Z_{i,j}$ is no longer equal to $z_i z_j$ and is now an independent variable. We can thus view SDP_{MCRM} as an LP, defined over the variables in M, with the non-linear constraint (4.20). Note that although SDP_{MCRM} is defined in a higher dimensional space than LP_{MCRM}, the objective function of SDP_{MCRM} is still defined over only the variables $x_1, \ldots, x_n$. Also, note that the value that x_n can take is constrained by the constraints of LP_{MCRM} (i.e., Eqs. (4.6)–(4.9)). Hence, SDP_{MCRM} is at least as strong as LP_{MCRM}. We thus have the following theorem.

Theorem 4.4.1 *SDP_{MCRM} is a relaxed program of ILP_{MCRM} that is at least as strong as LP_{MCRM}.*

Intuitively, we can interpret $\mathrm{SDP}_{\mathrm{MCRM}}$ as a polynomial-time complexity emulation of $\mathrm{QCLP}_{\mathrm{MCRM}}$ by introducing the auxiliary variables $Z_{i,j}$'s and aiming $Z_{i,j} = z_i z_j$ with the constraints (4.12)–(4.14) and (4.20).

4.4.2 ROUNDING ALGORITHMS

We present two distinct rounding algorithms. One is randomized, while the other is deterministic.

Randomized Rounding Algorithm. We present the Randomized Rounding Algorithm (RRA) in Alg. 14. RRA has $|S|$ iterations. At the s-th iteration, RRA probabilistically selects a channel for sniffer s based on the optimal values $y^*_{s,1}, \ldots, y^*_{s,|C|}$ of sniffer s. RRA rounds $y^*_{s,c}$ to 1 such that the probability of rounding $y^*_{s,c}$ to 1 is equal to $y^*_{s,c}$. That is, $P(y^{\#}_{s,c} = 1) = y^*_{s,c}$, where $y^{\#}_{s,c}$ denotes the resulting integer value of $y^*_{s,c}$ after rounding by RRA.

Greedy Rounding Algorithm. We present the Greedy Rounding Algorithm (GRA) in Alg. 15. GRA rounds $\vec{y}^*$ by choosing at each iteration a sniffer-channel pair whose value will be rounded to 0. In an iteration (lines 4–16), for each sniffer-channel pair $p = (s, c)$ whose value is not yet rounded to an integer, GRA adjusts the values of $y^p_{s,1}, \ldots, y^p_{s,|C|}$ according to Eq. (4.21). Here, the sniffer s allocates no radio-resource to the channel c, and distributes the radio-resource $y^p_{s,c}$ assigned to the channel c to the other channels proportionally to the radio resources assigned to the other channels. For the sniffer-channel pair to be rounded to 0 at the iteration, GRA selects the one

Algorithm 14 Randomized Rounding Algorithm

1: Let $\vec{y}^*$ be the optimal solution to $\text{LP}_{\text{MCRM}}/\text{SDP}_{\text{MCRM}}$
2: **for** $s \leftarrow 1$ to $|S|$ **do**
3: $(y^{\#}_{s,1}, \ldots, y^{\#}_{s,|C|}) \leftarrow (0, \ldots, 0)$
4: **for** $c \leftarrow 1$ to $|C|$ **do**
5: Toss a biased coin with probability of head being $y^*_{s,c} \,/\, \sum_{i=c}^{|C|} y^*_{s,i}$
6: **if** the tossed coin shows head **then**
7: $y^{\#}_{s,c} \leftarrow 1$
8: Break (i.e., start the next iteration of the for loop in line 2)
9: **end if**
10: **end for**
11: **end for**
12: **return** $\vec{y}^{\#}$

that achieves the maximum coverage improvement (or the minimum coverage loss) (lines 9 and 10). Here, the coverage improvement gained by $\vec{y}^p$ is defined as

$$\Delta w(\vec{y}^p, \vec{y}^{\#}) = \sum_{n \in N(s)} \left(w_n(\vec{y}^p) - w_n(\vec{y}^{\#}) \right), \text{ where}$$

$$w_n(\vec{y}) = \left\lfloor \min \left\{ 1, \frac{1}{r_n} \sum_{(s,c):\, n \in K_{s,c}} y_{s,c} \right\} \right\rfloor, \tag{4.22}$$

$N(s)$ denotes the set of the neighboring nodes of sniffer s, and $\lfloor x \rfloor$ denotes the largest integer that is not greater than x. At each iteration, GRA rounds one or two non-integer values of a sniffer depending on whether the sniffer has at least two non-integer values or only one (lines 11–15).

4.5 TIME COMPLEXITY ANALYSIS

In this section, we present asymptotic analysis of the time complexities of the proposed algorithms.

Algorithm 15 Greedy Rounding Algorithm

Let $\vec{y}^*$ be the optimal solution to $\text{LP}_{\text{MCRM}}/\text{SDP}_{\text{MCRM}}$
$\vec{y}^{\#} \leftarrow \vec{y}^*$
$P \leftarrow \{p = (s, c) : 0 < y^{\#}_{s,c} < 1 \; \forall s \in S, \, c \in C\}$
while $P \neq \emptyset$ **do**
  for each $p = (s, c) \in P$ **do**
    $\vec{y}^p \leftarrow \vec{y}^{\#}$
    Adjust the values of the entries $y^p_{s,1}, \ldots, y^p_{s,|C|}$ of $\vec{y}^p$ according to:

$$y^p_{s,c} \leftarrow 0, \quad y^p_{s,c'} \leftarrow \frac{y^p_{s,c'}}{\sum_{\forall c \in C} y^p_{s,c}} \; \forall c' \neq c \tag{4.21}$$

  end for
  Find $\tilde{p} = (\tilde{s}, \tilde{c}) \in P$ that maximizes the coverage improvement gained by $\vec{y}^p$ (which is defined in Eq. (4.22))
  $\vec{y}^{\#} \leftarrow \vec{y}^{\tilde{p}}$
  if $y^{\tilde{p}}_{\tilde{s},c} \in \{0, 1\}$ for all $c \in C$ **then**
    $P \leftarrow P - \{(\tilde{s}, 1), \ldots, (\tilde{s}, |C|)\}$
  else
    $P \leftarrow P - \{(\tilde{s}, \tilde{c})\}$
  end if
end while
return $\vec{y}^{\#}$

4.5.1 LOOK-AHEAD GREEDY ALGORITHMS

The look-ahead greedy algorithms both have at most $|S|$ iterations of the while loop. At each iteration, they need to consider at most $O(|S|^{t+1}|C|^{t+1})$ possible channel assignments in $\mathcal{P}$ (or $\mathcal{Q}$). Here, t, i.e., the look-ahead capability, is assumed to be less than a half of $|S|$, which is true for almost all cases. Also, any channel assignment has at most $O(|N|)$ nodes whose coverage needs to be verified to compute the coverage

improvement. Thus, both of the look-ahead greedy algorithms have the same time complexity of $O(|S|^{t+2}|C|^{t+1}|N|)$.

4.5.2 RELAXATION-AND-ROUNDING ALGORITHMS

To compute the time complexities of the RaR algorithms, we first compute the time complexity to formulate and solve the LP/SDP relaxation, and then the time complexity to run GRA/RRA.

Formulating and solving $\mathbf{LP_{MCRM}}$. To formulate $\mathrm{LP_{MCRM}}$, we need to build an LP in the following matrix form: maximize $\vec{c} \cdot \vec{x}$ subject to $A\vec{x} = \vec{b}$ and $\vec{x} \geq 0$. In the formulation of $\mathrm{LP_{MCRM}}$, building matrix A with the constraints (4.6)–(4.9) dominates the complexity, which will take $O\big((|N| + |S| \cdot |C|)^2\big)$ time since we have $|N| + |S| \cdot |C|$ variables and $O(|N| + |S| \cdot |C|)$ constraints in $\mathrm{LP_{MCRM}}$. To solve $\mathrm{LP_{MCRM}}$, one can employ one of many existing LP solvers, e.g., the one in [37], which will take $O\big((|N| + |S| \cdot |C|)^3 / \log(|N| + |S| \cdot |C|)\big)$ time. Thus, in total, it takes $O\big((|N| + |S| \cdot |C|)^3 / \log(|N| + |S| \cdot |C|)\big)$ time to formulate and solve $\mathrm{LP_{MCRM}}$.

Formulating and solving $\mathbf{SDP_{MCRM}}$. To formulate $\mathrm{SDP_{MCRM}}$, constructing the matrices A_i's in the constraint (4.19) dominates the complexity. This will take $O\big((|N|+|S|\cdot|C|)^3\big)$ time, since each A_i has $(|N|+|S|\cdot|C|+1)^2$ entries and $\mathrm{SDP_{MCRM}}$ has $O(|N| + |S| \cdot |C|)$ constraints. To solve $\mathrm{SDP_{MCRM}}$, one can use one of various SDP solvers available, which will take $O\big((|N| + |S| \cdot |C|)^3\big)$ time [44]. Thus, in total, it takes $O\big((|N| + |S| \cdot |C|)^3\big)$ time to formulate and solve $\mathrm{SDP_{MCRM}}$.

RRA. It has $|S|$ iterations. In each iteration, it performs at most $|C|$ random experiments (i.e., tossing a coin), each of which takes a constant time. Thus, RRA has a time complexity of $O(|S| \cdot |C|)$.

GRA. It has at most $|S| \cdot |C|$ iterations of the while loop, and in each iteration P has at most $O(|S| \cdot |C|)$ sniffer-channel pairs. For each pair $p = (s, c)$, there are at most $O(|N|)$ nodes in $N(s)$ whose coverage improvement need to be computed. For each

Table 4.1: Time complexity of proposed algorithms

Algorithm	Time complexity
Look-t-Steps-Ahead Greedy	$O(\|S\|^{t+2}\|C\|^{t+1}\|N\|)$
t-Sniffers-at-One-Step Greedy	$O(\|S\|^{t+2}\|C\|^{t+1}\|N\|)$
LP_{MCRM} + RRA/GRA	$O\left(\frac{(\|N\|+\|S\|\cdot\|C\|)^3}{\log(\|N\|+\|S\|\cdot\|C\|)}\right)$
SDP_{MCRM} + RRA/GRA	$O\left((\|N\|+\|S\|\cdot\|C\|)^3\right)$
RRA	$O(\|S\|\cdot\|C\|)$
GRA	$O(\|S\|^2\cdot\|C\|^2\cdot\|N\|)$

node, it takes a constant time to compute the coverage improvement because, among the sniffer-channel pairs that can cover the node, only one has an adjustment in its value. Hence, it takes at most $O(|N|)$ time to compute the coverage improvement for a sniffer-channel pair. Thus, GRA has a time complexity of $O(|S|^2 \cdot |C|^2 \cdot |N|)$.

Based on these results, we summarize the time complexities of the proposed algorithms in Table 4.1.

4.6 NUMERICAL EXPERIMENTS

We evaluate the performance of the proposed algorithms—the two look-ahead greedy algorithms and the four RaR algorithms (i.e., the four combinations of the two relaxations and the two rounding algorithms)—through simulations in two kinds of networks: random networks and scale-free networks. In random networks, nodes are randomly deployed in a 1×1 square area with a uniform distribution, and the receiving range is 0.25. In scale-free networks, nodes are deployed such that the distribution $f(d)$ of the nodes with degree d follows a power law in a form of d^{-p}, i.e., the number of nodes with high degree decreases exponentially. We set $2 < p < 3$, and pick the nodes with highest degrees as sniffers so that we can achieve a higher monitoring coverage

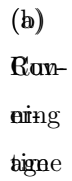

Fig. 4.1.: Random networks for varying number of sniffers

than picking them randomly. We choose these networks, because the performance of the proposed algorithms will largely depends on the network topology and these two kinds of networks show significantly different topologies. Also, they are observed in many practical networks[1].

We evaluate the proposed algorithms in two metrics: coverage and running time. We compare the coverage of the proposed algorithms with the maximum achievable coverage (i.e., the optimum of ILP_{MCRM}) and also with the coverage of the naive greedy extensions. In all simulations, we use the same value for $|N|$ (i.e., the number of nodes), w_n (i.e., the weight of node n), r_n (i.e., the coverage requirement of node n), and t (i.e., the look-ahead capability): $|N| = 40$, $w_n = 1$ and $r_n = 2$ for all n, and $t = \max_{\forall} r_n - 1 = 1$. In the first set of simulations, we fix $|C|$ (i.e., the number of wireless channels) to 3, and see how the proposed algorithms perform as $|S|$ (i.e., the number of sniffers) varies from 10 to 40. In the second set of simulations, we fix $|S| = 30$, and see the performance of the proposed algorithms as $|C|$ varies from 2 to 6. All of the results shown below are the averages over more than 30 iterations.

Figure 4.1(a), (b) show the coverage and the running time of the proposed algorithms, respectively, in random networks for varying number of sniffers. In Fig. 4.1(a), we observe that the coverage of the SDP-and-GRA and the LP-and-GRA are comparable to the maximum achievable coverage (i.e., the ILP optimum), followed by the look-ahead greedy algorithms with a small gap. We can see that, after rounding,

[1]Wireless networks where mobile users move randomly can be viewed as random networks, and many empirically observed networks such as the world wide web and the Internet have been found to be scale-free.

GRA maintains the solution quality of the optimal fractional solution closer to the maximum coverage, while RRA results in the degradation of the solution quality. In the figure, GRD-Ext1 denotes the first naive greedy extension that chooses the coverage-set maximizing the coverage improvement, and GRD-Ext2 denotes the second naive greedy extension that selects the coverage-set maximizing the total weight of the uncovered nodes that it contains. We observe that the naive greedy extensions both show poor performance. LP-UP and SDP-UP denote the optimal values of the $\mathrm{LP}_{\mathrm{MCRM}}$ and $\mathrm{SDP}_{\mathrm{MCRM}}$, respectively, which are shown as upper bounds on the maximum coverage. We can see that the SDP relaxation provides only a slightly tighter upper bound than the LP relaxation, and accordingly that its corresponding (i.e., the SDP-based) RaR algorithms perform little better than the LP-based RaR algorithms.

In Fig. 4.1(b), we have two different y axes; the y axis on the right represents the running time of the look-ahead greedy algorithms, while the y axis on the left represents the running time of the other proposed algorithms. We observe a (relatively) large gap between the running times of the LP-based RaR algorithms and the SDP-based RaR algorithms, not expected from their asymptotic time complexity results. Also, we observe that the running time of the look-ahead greedy algorithms is much larger than that of the other algorithms, and they grow rapidly as the number $|S|$ of sniffers increases, as expected from its asymptotic time complexity of order 3 in $|S|$. A notable observation is that the running time of the t-sniffers-at-one-step greedy algorithm is almost half of that of the look-t-steps-ahead greedy algorithm. This implies that, at each iteration of the while loop, the t-sniffers-at-one-step greedy algorithm performed channel assignment mostly for two sniffers, while the look-t-steps-ahead greedy algorithm did for only one sniffer. But, the t-sniffers-at-one-step greedy algorithm still shows the coverage comparable to that of the look-t-steps-ahead greedy algorithm.

Figure 4.2(a), (b) show the coverage and the running time of the proposed algorithms, respectively, in scale-free networks for varying number of sniffers. We observe similar results to those in random networks in the both metrics. But, a notable

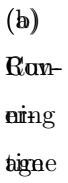

Fig. 4.2.: Scale-free networks for varying number of sniffers

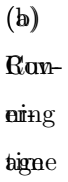

Fig. 4.3.: Random networks for varying the number of available wireless channels

observation in coverage is that, in scale-free networks, the SDP relaxation shows a substantial improvement on the upper bound on the maximum achievable coverage, thus implying that it provides a better fractional solution to rounding algorithms. Accordingly, we can see that the SDP-based RaR algorithms show a noticeable coverage improvement, compared to the LP-based RaR algorithms. Also, we observe that, in scale-free networks, the gap between the running times of the two RaR algorithms with a different rounding algorithm is smaller than that in random networks. This implies that scale-free networks are likely to yield the problem instances for which the fractional optimal solution has less number of fractional values, so that RRA/GRA runs less number of iterations.

Figures 4.3 and 4.4 show the performance of the proposed algorithms in random networks and scale-free networks, respectively, for varying number of available wireless channels. In the comparison among the proposed algorithms, we observe similar trends to those for varying number of sniffers. In Figs. 4.3, 4.4(a), as the number of available wireless channels increases, the coverage decreases because the channel as-

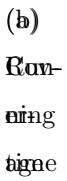

Fig. 4.4.: Scale-free networks for varying the number of available wireless channels

signment of nodes is distributed over more number of channels and hence the number of nodes that each coverage-set of a sniffer contains decreases.

To summarize the simulation results, the SDP-and-GRA achieves the highest coverage close to the maximum coverage, but shows a (relatively) long running time. Hence, the SDP-and-GRA will be favored, especially, for monitoring applications where a higher coverage is more emphasized, such as security monitoring. On the other hand, the LP-and-GRA attains the coverage comparable to that of the SDP-and-GRA, and also shows a fast running time. Thus, LP-and-GRA can be considered as a good compromise between the coverage and the running-time, and will be favored for monitoring applications requiring fast running-time, such as monitoring in dynamic network environments where the channel assignment of sniffers needs to changed rapidly.

4.7 CONCLUSION

In this chapter, we studied the optimal sniffer-channel assignment problem for reliable monitoring in multi-channel wireless networks, where each node is given sniffer redundancy to maintain a certain level of monitoring accuracy. This problem can be viewed as a generalization of the problems studied in the previous chapters that assume perfect sniffers and thus do not need to consider the sniffer redundancy. However, we showed that the generalized problem no longer holds the submodular property, unlike the special case studied in the previous chapters. As a results, in

the generalized problem, the prior approximation algorithms lose their performance guarantees. To solve the generalized problem, we proposed a variety of approximation algorithms based on two basic approaches—greedy approach and the relaxation-and-rounding approach. We present a comparative analysis of the proposed algorithms through simulations.

Our conclusion is that SDP-and-GRA (i.e., the combination of the SDP relaxation and the GRA) achieves the highest coverage close to the maximum achievable coverage, but shows a (relatively) long running time. On the other hand, LP-and-GRA (i.e., the combination of the LP relaxation and the GRA) attains the coverage comparable to the coverage of the SDP-and-GRA, and also shows a fast running time. Hence, LP-and-GRA can be considered as a good compromise between the coverage and the running-time. Thus, the SDP-and-GRA will be favored, especially, for monitoring applications where a higher coverage is more emphasized (e.g., security monitoring), while LP-and-GRA will be favored for monitoring applications requiring fast running-time (e.g., monitoring dynamic network environments).

5. RELATED WORK

5.1 OPTIMAL PLACEMENT OF MONITORING NODES IN SINGLE-CHANNEL WIRELESS NETWORKS

Subhadrabandhu *et al.* [25–27] have studied the optimal placement of monitoring nodes in *single-channel* wireless networks. The work [25] studies a problem of how to optimally select a subset of monitoring nodes to execute intrusion detection modules (IDSs), given a budget on the number of monitoring nodes. The goal is to maximize the number of normal nodes covered by the selected monitoring nodes. The work presents a greedy approximation algorithm to the coverage maximization problem (which is NP-hard), which achieves the best possible approximation ratio.

The work [26] allows for IDSs that periodically stop functioning due to operational failure or compromise by intruders. It develops a framework to counter the failure of IDSs, and studies a problem of how to find an optimal set of monitoring nodes that minimize the resource consumption, i.e., the number of monitoring nodes selected to execute IDSs, while covering all normal nodes in the network. The work presents a distributed approximation algorithm to the resource minimization problem, which attains the best possible approximation ratio.

The work [27] allows for IDSs that periodically fail to detect attacks and also generate false positives, and develops a similar framework to that of [26]. In all of the works [25–27], it is assumed that the network uses only one channel, and hence there is no issue of channel assignment of monitoring nodes. On the other hand, the problem that we study in this dissertation (i.e., MCMC in Section 2.2) deals with the optimal placement and channel assignment of monitoring nodes, which is a generalization of the coverage maximization problem in [25].

5.2 CHANNEL ASSIGNMENT OF SNIFFERS IN MULTI-CHANNEL WIRELESS NETWORKS

Some works [38,45,46] have also studied the optimal monitoring problem in multi-channel wireless networks, but their focus or performance guarantee is different from that of this dissertation. Chhetri *et al.* [38] have studied two models of sniffers that assume different capabilities of sniffers capturing traffic. The first, called user-centric model, assumes that frame-level information can be captured so that the activities of different users are distinguishable. The problem in the user-centric model is a special case of MCMC where all monitoring nodes are activated, and all of monitoring nodes and normal nodes have a single radio. The second, called sniffer-centric model, assumes that only binary information is available regarding channel activities, i.e., whether some user is active in a specific channel near a sniffer. The authors show that the sniffer-centric model can be mapped to the user-centric model to solve the problem in two models.

The work [38] and our works in this dissertation all assume that the global knowledge of the topology and the channel usages of normal nodes is given to, or can be inferred by, sniffers. On the other hand, Arora *et al.* [45] have studied a trade-off between assigning the radios of sniffers to channels known to be busiest based on the current knowledge, versus exploring channels that are under observed.

Also, the work [46] has proposed a distributed algorithm to solve OSCA (in Section 3.2) based on a Gibbs sampler approach. However, unlike our DA-OSCA (in Chapter 3), the algorithm does not provide a performance guarantee.

APPENDICES

A. SUPPORTING RESULTS FOR CHAPTER 3

A.1 PROOF OF THE CLAIM IN SECTION 3.3.1

We show the claim in Section 3.3.1 that solving $\mathrm{QP}_{\mathrm{OSCA}}$ is equivalent to solving $\mathrm{LP}_{\mathrm{OSCA}}$. Let $\{x_n^*, y_{s,c}^*, x_n^{\mathrm{aux},*}, y_{s,c}^{\mathrm{aux},*}\}$ be the optimal solution of $\mathrm{QP}_{\mathrm{OSCA}}$. Note that all of the quadratic terms in the objective function (3.6) of $\mathrm{QP}_{\mathrm{OSCA}}$ are non-positive, and also that there is no constraint on the variables $x_n^{\mathrm{aux},*}$'s and $y_{s,c}^{\mathrm{aux},*}$'s. Hence, in order to maximize the objective function (3.6), it must be true that $x_n^{\mathrm{aux},*} = x_n^*$ and $y_{s,c}^{\mathrm{aux},*} = y_{s,c}^*$. This means that $\{x_n^*, y_{s,c}^*\}$ maximizes $\sum_{n\in N} w_n x_n$ subject to Eqs. (3.2)–(3.4) and thus is an optimal solution to $\mathrm{LP}_{\mathrm{OSCA}}$. Therefore, we can find an optimal solution to $\mathrm{LP}_{\mathrm{OSCA}}$ by solving $\mathrm{QP}_{\mathrm{OSCA}}$. Thus, the claim is true.

A.2 DERIVATION OF ALGORITHM 16

Let $\vec{v}^{+V}$ be the projection of $\vec{v}$ to V. With definition of projection, i.e., $\vec{v}^{+V} = \mathrm{argmin}_{\vec{x}\in V}\, d(\vec{v},\vec{x})$ where $d(\vec{v},\vec{x})$ denotes the Euclidean distance between $\vec{v}$ and $\vec{x}$, it is easy to verify that if $v_j \le 0$, then $v_j^{+V} = 0$. In order to obtain v_j^{+V} for $v_j > 0$, we redefine $\vec{v}$ by removing the negative and zero components from $\vec{v}$. We assume that the dimension of the redefined vector $\vec{v}$ is $d \le c$. We also redefine $V = \{\vec{x} = (x_1, \ldots, x_d) : x_j \ge 0 \text{ for all } j \in \{1, \ldots, d\} \text{ and } \sum_{j=1}^{d} x_j \le 1\}$. The problem then becomes to find the projection of the redefined vector $\vec{v} > 0$ to V.

Obviously, if $\vec{v} \in V$, $\vec{v}^{+V} = \vec{v}$. Hence, we only need to consider the case when $\vec{v} \notin V$. In this case, $\vec{v}$ must be included in the set $U = \{\vec{x} : \sum_{j=1}^{d} x_j > 1 \text{ and } x_j > 0 \text{ for all } j \in \{1, \ldots, d\}\}$. We define a bounded hyperplane $F = \{\vec{x} : \sum_{j=1}^{d} x_j = 1 \text{ and } x_j \ge 0 \text{ for all } j \in \{1, \ldots, d\}\}$, and define $H = \{\vec{x} : \sum_{j=1}^{d} x_j = 1\}$ to be the

hyperplane that includes F. Due to the following lemma, we only need to find $[\vec{v}^{\perp_H}]^+_F$ in order to obtain $\vec{v}^{+_V}$.

Lemma A.2.1 *For any $\vec{v} \in U$, $\vec{v}^{+_V} = [\vec{v}^{\perp_H}]^+_F$, where $\vec{v}^{\perp_H}$ denotes the perpendicular foot of $\vec{v}$ onto the hyperplane H.*

Proof To prove the lemma, we first show that $\vec{v}^{+_V}$ is a point on the bounded hyperplane F. To show this claim, we only need to show that the line segment that connects any $\vec{v} \in U$ and any $\vec{x} \in V$, denoted by $\overline{vx}$, intersects with F. It is because if there exists a point at which $\overline{vx}$ intersects with F, denoted by $\vec{y}$, the distance between $\vec{v}$ and $\vec{y}$ would be smaller than or equal to the distance between $\vec{v}$ and $\vec{x}$, which implies that $\vec{v}^{+_V} \in F$. In order to show the claim, we consider the line that passes through the points $\vec{v}$ and $\vec{x}$, denoted by $\overleftrightarrow{vx}$. The line $\overleftrightarrow{vx}$ is a set of points $\{\vec{x} + t(\vec{v} - \vec{x}) : t \text{ is a real number}\}$. This line intersects with the hyperplane H at the point $\vec{p} = \vec{x} + t(\vec{v} - \vec{x})$, where $t = \frac{1 - \sum_{j=1}^{d} x_j}{\sum_{j=1}^{d} v_j - \sum_{j=1}^{d} x_j}$. Since $\vec{v} \in U$ and $\vec{x} \in V$, it is true that $0 \leq t < 1$. This implies that $\vec{p} \in \overline{vx}$ and also that $\vec{p} > 0$. Also, due to the facts that $\vec{p} \in H$ and that $\vec{p} > 0$, it follows that $\vec{p} \in F$. Hence, $\overline{vx}$ intersects with F at the point $\vec{p}$, and thus the claim is true, i.e., $\vec{v}^{+_V} \in F$. Then, $\vec{v}^{+_V} = \text{argmin}_{\vec{x} \in F}\, d(\vec{v}, \vec{x})$. By Pythagorean theorem, it follows that $d(\vec{v}, \vec{x})^2 = d(\vec{v}, \vec{v}^{\perp_H})^2 + d(\vec{v}^{\perp_H}, \vec{x})^2$ for any $\vec{x} \in F$. Here, $d(\vec{v}, \vec{v}^{\perp_H})$ is a constant. Hence, $\vec{v}^{+_V} = \text{argmin}_{\vec{x} \in F}\, d(\vec{v}^{\perp_H}, \vec{x})$, i.e., $\vec{v}^{+_V} = [\vec{v}^{\perp_H}]^+_F$. ■

We find $[\vec{v}^{\perp_H}]^+_F$ in a recursive manner. Let $\vec{v}^{+,(0)} = [\vec{v}^{\perp_H}]^+_F$. A simple calculation gives $\vec{v}^{\perp_H} = (v_1 + t, \ldots, v_d + t)$ where $t = \frac{1}{d}(1 - \sum_{j=1}^{d} v_j)$. If $\vec{v}^{\perp_H} \in \mathcal{F}$, $\vec{v}^{+,(0)} = \vec{v}^{\perp_H}$. Otherwise, i.e., if $\vec{v}^{\perp_H} \notin \mathcal{F}$, at least one component of $\vec{v}^{\perp_H}$ must have a negative value since $\vec{v}^{\perp_H} \in H$. It is easy to verify that the components of $\vec{v}^{+,(0)}$ corresponding to those of $\vec{v}^{\perp_H}$ that have a negative value or zero must be zero. Without loss of generality, we assume that the positive components of $\vec{v}^{\perp_H}$ are $v_1^{\perp_H}, \ldots, v_e^{\perp_H}$ where $e \leq d-1$. Since $\sum_{j=1}^{d} v_j^{\perp_H} = 1$ and $\vec{v}^{\perp_H}$ has at least one negative component, it follows that $\sum_{j=1}^{e} v_j^{\perp_H} > 1$. Let $\vec{v}^{(1)} = (v_1^{\perp_H}, \ldots, v_e^{\perp_H})$ and $U^{(1)} = \{(x_1, \ldots, x_e) : \sum_{j=1}^{e} x_j >$

Algorithm 16 Projection Algorithm

// Algorithm projects $\vec{v}$ to $V = \{(x_1, \ldots, x_c) : x_j \geq 0 \text{ for all } j \in \{1, \ldots, c\} \text{ and } \sum_{j=1}^{c} x_j \leq 1\}$.
$J \leftarrow \{1, \ldots, c\}$
while (1) **do**
  for $j \leftarrow 1$ to $|J|$ **do**
    if $v_{J_j} \leq 0$ (where J_j denotes the j-th element of J) **then**
      $v_{J_j} \leftarrow 0$
      $J \leftarrow J \setminus \{J_j\}$
    end if
  end for
  // Here, it is invariant that $v_j > 0$ for all $j \in J$, and also that $v_j = 0$ for all $j \notin J$.
  if $|J| = 0$ or $\sum_{j=1}^{|J|} v_{J_j} \leq 1$ **then**
    Terminate the algorithm
  else
    for $j \leftarrow 1$ to $|J|$ **do**
      $v_{J_j} \leftarrow v_{J_j} + \frac{1}{|J|}\left(1 - \sum_{j=1}^{|J|} v_{J_j}\right)$
    end for
    // Here, it is invariant that $\sum_{i=1}^{c} v_i = 1$.
  end if
end while
return $\vec{v}$

1 and $x_j > 0$ for all $j \in \{1, \ldots, e\}\}$, then $\vec{v}^{(1)} \in U^{(1)}$. Define $F^{(1)} = \{(x_1, \ldots, x_e) : \sum_{j=1}^{e} x_j = 1 \text{ and } x_j > 0 \text{ for all } j \in \{1, \ldots, e\}\}$ and $H^{(1)} = \{(x_1, \ldots, x_e) : \sum_{j=1}^{e} x_j = 1\}$. We then have $(v_1^{+,(0)}, \ldots, v_e^{+,(0)}) = [\vec{v}^{(1)}]^{+}_{F^{(1)}}$ since $v_{e+1}^{+,(0)}, \ldots, v_d^{+,(0)}$ are all zeros. Using Pythagorean theorem, we get $(v_1^{+,(0)}, \ldots, v_e^{+,(0)}) = [\vec{v}^{(1)\perp_{H^{(1)}}}]^{+}_{F^{(1)}}$, where $\vec{v}^{(1)\perp_{H^{(1)}}}$ denotes the perpendicular foot of $\vec{v}^{(1)}$ onto the hyperplane $H^{(1)}$. The problem of

finding $[\vec{v}^{\perp_H}]^+_F$ then becomes to find $[\vec{v}^{(1)\perp_{H^{(1)}}}]^+_{F^{(1)}}$. Note that both the problems differ only in the dimension of the vector. Also, the dimension of the vector in the former problem is at least one less than that in the latter problem. Hence, in order to find $[\vec{v}^{(1)\perp_{H^{(1)}}}]^+_{F^{(1)}}$, we can repeat the process that we have done to find $[\vec{v}^{\perp_H}]^+_F$. At the n-th iteration of this process, we would be able to obtain $[\vec{v}^{(n-1)\perp_{H^{(n-1)}}}]^+_{F^{(n-1)}}$, equivalently $[\vec{v}^{\perp_H}]^+_F$, or reduce the dimension of the vector by at least one. Since we start with the dimension $d \leq c$, the number of these iterations to obtain $[\vec{v}^{\perp_H}]^+_F$ is at most c.

Alg. 16 implements this procedure to obtain the projection $[\vec{v}]^+_V$.

A.3 PROOF OF THEOREM 3.3.1

To show the theorem, we use the proof of Proposition 4 in [41]. For this, we first formulate the constraints (3.2)–(3.4) of QP-MC into the matrix form: $A\vec{z} \leq \vec{0}, \vec{z} \in Z$, where the matrix A is defined as

$$A = \left(\begin{array}{c|c} I_{|N|} & A_{\text{sub1}} \\ \hline O_{|S|,|N|} & A_{\text{sub2}}, \end{array}\right) \in \mathbb{R}^{(|N|+|S|)\times(|N|+|S|\cdot|C|)},$$

where

$$A_{\text{sub1}} = \begin{pmatrix} -\mathbf{1}_{K_{S_1,C_1}}(N_1) & \dots & -\mathbf{1}_{K_{S_{|S|},C_{|C|}}}(N_1) \\ \vdots & \ddots & \vdots \\ -\mathbf{1}_{K_{S_1,C_1}}(N_{|N|}) & \dots & -\mathbf{1}_{K_{S_{|S|},C_{|C|}}}(N_{|N|}) \end{pmatrix} \in \mathbb{R}^{|N|\times(|S|\cdot|C|)},$$

$$A_{\text{sub2}} = \begin{pmatrix} 1 & \dots & 1 & \dots & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots & & \\ 0 & \dots & 0 & \dots & 1 & \dots & 1 \end{pmatrix} \in \mathbb{R}^{|S|\times(|S|\cdot|C|)}.$$

Here, $I_{|N|}$ is $|N| \times |N|$ identity matrix, $O_{|S|,|N|}$ is $|S| \times |N|$ zero matrix, S_i denotes the i-th element of the set S, and $\mathbf{1}_S(s)$ is an indicator function defined as: $\mathbf{1}_S(s) = 1$ if $s \in S$; otherwise, $\mathbf{1}_S(s) = 0$.

Using the proof of Proposition 4 in [41], it can be shown that a sufficient condition for DA-LP$_{\text{OSCA}}$ to converge is that $\frac{1}{\beta}I_{|N|+|S|} - 2dAA^T$ must be positive definite. The matrix $\frac{1}{\beta}I_{|N|+|S|} - 2dAA^T$ is positive definite if and only if for any non-zero vector $\vec{s}$,

$$\begin{aligned} &\vec{s}^T \left(\frac{1}{\beta}I_{|N|+|S|} - 2dAA^T\right)\vec{s} > 0 \\ &\quad\iff \frac{1}{\beta}\sum_{i=1}^{|N|+|S|} s_i^2 > 2d\left(A^T\vec{s}\right)^2. \end{aligned} \tag{A.1}$$

It follows that

$$\begin{aligned} \left(A^T\vec{s}\right)^2 &= \sum_{j=1}^{|N|+|S|\cdot|C|}\left(\sum_{i=1}^{|N|+|S|} A_{i,j}s_i\right)^2 \\ &\le \sum_{j=1}^{|N|+|S|\cdot|C|}\left(\sum_{i=1}^{|N|+|S|}|A_{i,j}|\right)\left(\sum_{i=1}^{|N|+|S|}|A_{i,j}|s_i^2\right) \\ &\qquad\text{(by Cauchy-Schwartz inequality)} \\ &\le \max_{\forall j}\left\{\sum_{i=1}^{|N|+|S|}|A_{i,j}|\right\}\sum_{i=1}^{|N|+|S|}s_i^2\sum_{j=1}^{|N|+|S|\cdot|C|}|A_{i,j}| \\ &\le \max_{\forall j}\left\{\sum_{i=1}^{|N|+|S|}|A_{i,j}|\right\}\cdot\max_{\forall i}\left\{\sum_{j=1}^{|N|+|S|\cdot|C|}|A_{i,j}|\right\} \\ &\quad\times\sum_{i=1}^{|N|+|S|}s_i^2. \end{aligned} \tag{A.2}$$

Hence, $\frac{1}{\beta}I_{|N|+|S|} - 2dAA^T$ is positive definite if the following holds:

$$\beta < \frac{1}{2dB_1B_2},$$

where

$$\begin{aligned} B_1 &= \max\left\{\sum_{i=1}^{|N|+|S|}|A_{i,j}| : j \in [|N|+|S|\cdot|C|]\right\}, \\ B_2 &= \max\left\{\sum_{j=1}^{|N|+|S|\cdot|C|}|A_{i,j}| : i \in [|N|+|S|]\right\}, \end{aligned}$$

where $[n]$ denotes an index set $\{1, \ldots, n\}$. It follows that

$$B_1 = \max\left\{1, 1 + \sum_{l=1}^{|N|} \mathbf{1}_{K_{S_i,C_j}}(N_l) : i \in [|S|], j \in [|C|]\right\}$$
$$= \max\left\{1, |K_{s,c}| + 1 : s \in S, c \in C\right\},$$

and also that

$$B_2 = \max\left\{|C|, 1 + \sum_{i=1}^{|S|}\sum_{j=1}^{|C|} \mathbf{1}_{K_{S_i,C_j}}(N_l) : l \in [|N|]\right\}$$
$$= \max\left\{|C|, M + 1\right\},$$

where $M = \max_{n \in N} |\{K_{s,c} : n \in K_{s,c}\}|$. Thus, the theorem follows.

A.4 PROOF OF THEOREM 3.3.2

To prove the theorem, we show that OCAA is a distributed generalization of `PIPAGE` [35] that achieves the guarantee in the theorem in a centralized manner. For this, we first explain how `PIPAGE` solves OSCA. The `PIPAGE` applied to solve OSCA rounds a (fractional) solution of $\mathrm{LP}_{\mathrm{OSCA}}$ to a feasible integer solution to $\mathrm{ILP}_{\mathrm{OSCA}}$ in an iterative manner. Since each sniffer can assign only one channel to its radio, each sniffer has more than two non-integer values if it has non-integer values. At each iteration, `PIPAGE` adjusts two non-integer values of a sniffer such that at least one of them becomes an integer of 0 or 1, and the sum of them are preserved. Hence, when a sniffer has only two non-integer values, both of them will become an integer value of 0 or 1 after the adjustment by `PIPAGE`. At each iteration, `PIPAGE` adjusts two non-integer values of a sniffer as follows. Let $0 < y_{s,c_1}, y_{s,c_2} < 1$ be the two non-integer values of a sniffer to be adjusted at an iteration, and define $\epsilon_1 = \min\{y_{s,c_1}, 1 - y_{s,c_2}\}$ and $\epsilon_2 = \min\{1 - y_{s,c_1}, y_{s,c_2}\}$. At the iteration, `PIPAGE` adjusts the fractional solution $\vec{y}$ including y_{s,c_1} and y_{s,c_2} to a new solution of either $\vec{y}^{(1)}$ or $\vec{y}^{(2)}$, which have the same values for all components except ones whose indices are (s, c_1) and (s, c_2). In $\vec{y}^{(1)}$, the two components are $y^{(1)}_{s,c_1} = y_{s,c_1} - \epsilon_1$ and $y^{(1)}_{s,c_2} = y_{s,c_2} + \epsilon_1$, and in $\vec{y}^{(2)}$,

they are $y^{(2)}_{s,c_1} = y_{s,c_1} + \epsilon_2$ and $y^{(2)}_{s,c_2} = y_{s,c_2} - \epsilon_2$ in $\vec{y}^{(2)}$. `PIPAGE` adjusts $\vec{y}$ to $\vec{y}^{(1)}$ if $F(\vec{y}^{(1)}) \geq F(\vec{y}^{(2)})$, where $F(\vec{y}) = \sum_{n\in N} w_n \left(1 - \prod_{(s,c):n\in K_{s,c}}(1 - y_{s,c})\right)$. Otherwise, `PIPAGE` adjusts $\vec{y}$ to $\vec{y}^{(2)}$.

We now show OCAA accomplishes the procedure that the `PIPAGE` applied to solve OSCA performs. To show this, we first derive an efficient way of evaluating the criterion $F(\vec{y}^{(1)}) \geq F(\vec{y}^{(2)})$ that `PIPAGE` uses to adjust the fractional solution at each iteration. Since $y_{s,c_1} + y_{s,c_2} \leq 1$ due to the group budget constraint, we have $\epsilon_1 = y_{s,c_1}$ and $\epsilon_2 = y_{s,c_2}$, and consequently we have

$$\begin{aligned} y^{(1)}_{s,c_1} &= 0, & y^{(1)}_{s,c_2} &= y_{s,c_1} + y_{s,c_2}, \\ y^{(2)}_{s,c_1} &= y_{s,c_1} + y_{s,c_2}, & y^{(2)}_{s,c_2} &= 0. \end{aligned}$$

It follows that

$$\begin{aligned} F(\vec{y}) &= \sum_{n\in N} w_n \left(1 - \prod_{(s,c):n\in K_{s,c}} (1 - y_{s,c})\right) \\ &= \sum_{n\in N} w_n - \sum_{n\in N} w_n \left(\prod_{(s,c):n\in K_{s,c}} (1 - y_{s,c})\right), \end{aligned}$$

and also that

$$\begin{aligned} &\sum_{n\in N} w_n \left(\prod_{(s,c):n\in K_{s,c}} (1 - y_{s,c})\right) \\ &= \sum_{n\in K_{s,c_1}} w_n \left(\prod_{s'\neq s:n\in K_{s',c_1}} (1 - y_{s',c_1})\right)(1 - y_{s,c_1}) \\ &+ \sum_{n\in K_{s,c_2}} w_n \left(\prod_{s'\neq s:n\in K_{s',c_2}} (1 - y_{s',c_2})\right)(1 - y_{s,c_2}) \\ &+ \sum_{n\in N:n\notin K_{s,c_1},n\notin K_{s,c_2}} w_n \left(\prod_{(s,c):n\in K_{s,c}} (1 - y_{s,c})\right). \end{aligned}$$

Since $y^{(1)}_{s',c} = y^{(2)}_{s',c} = y_{s',c}$ for all $(s',c) \neq (s,c_1),(s,c_2)$ and $(y^{(1)}_{s,c_1} - y^{(2)}_{s,c_1}) = -(y^{(1)}_{s,c_2} - y^{(2)}_{s,c_2})$, it follows that

$$\begin{aligned}
&F(\vec{y}^{(1)}) - F(\vec{y}^{(2)}) \\
&= \sum_{n \in K_{s,c_1}} w_n \left(\prod_{s' \neq s : n \in K_{s',c_1}} (1 - y_{s',c_1}) \right) \times \left(y^{(1)}_{s,c_1} - y^{(2)}_{s,c_1} \right) \\
&+ \sum_{n \in K_{s,c_2}} w_n \left(\prod_{s' \neq s : n \in K_{s',c_2}} (1 - y_{s',c_2}) \right) \times \left(y^{(1)}_{s,c_2} - y^{(2)}_{s,c_2} \right) \\
&= \left(I(K_{s,c_1}, \vec{y}_{N(s)}) - I(K_{s,c_2}, \vec{y}_{N(s)}) \right) \times \left(y^{(1)}_{s,c_1} - y^{(2)}_{s,c_1} \right).
\end{aligned}$$

Hence, since $y^{(1)}_{s,c_1} < y^{(2)}_{s,c_1}$, $F(\vec{y}^{(1)}) \geq F(\vec{y}^{(2)})$ if $I(K_{s,c_1}, \vec{y}_{N(s)}) \leq I(K_{s,c_2}, \vec{y}_{N(s)})$. This means that `PIPAGE` adjusts $\vec{y}$ to $\vec{y}^{(1)}$ if $I(K_{s,c_1}, \vec{y}_{N(s)}) \leq I(K_{s,c_2}, \vec{y}_{N(s)})$. Otherwise, `PIPAGE` adjusts $\vec{y}$ to $\vec{y}^{(2)}$.

Recall that when `PIPAGE` rounds non-integer values of the variables $\vec{y}_s = (y_{s,c} : c \in C)$ of sniffer s through multiple iterations, the values that are not in $\vec{y}_s$, i.e., $\tilde{y}_{s',c}$'s for all (s',c) such that $s' \neq s$, will remain the same. Hence, while the non-integer values of $\vec{y}_s$ are rounded, the values of $I(K_{s,c}, \vec{y}_{N(s)})$'s for all $c \in C$ will remain the same. Therefore, after the multiple iterations to round the non-integer values of $\vec{y}_s$, all of the non-integer values except one that has the maximum coverage improvement among all non-integer values, say y_{s,c^*}, will be rounded to 0, and y_{s,c^*} will be adjusted to the sum of all the non-integer values, which is equal to 1. This is the rounding procedure that OCAA performs. Thus, the theorem follows.

A.5 PROOF OF THE CORRECTNESS OF ALG. 10

To show the correctness of Alg. 10, we use the duality theory [40, Ch. 5.1.3], which states that, for any maximization problem, the maximum of the given primal problem is upper bounded by the dual objective value of any feasible dual solution.

To derive the dual problem of $\mathrm{LP}_{\mathrm{OSCA}}$, we define the Lagrangian function of $\mathrm{LP}_{\mathrm{OSCA}}$ as

$$L_{\mathrm{LP}}(\vec{z},\vec{p}) = \sum_{n\in N} w_n x_n + \sum_{n\in N} p_n \left(\sum_{(s,c):n\in K_{s,c}} y_{s,c} - x_n \right). \tag{A.3}$$

The dual problem of $\mathrm{LP}_{\mathrm{OSCA}}$ is then given as

$$\text{minimize} \quad D_{\mathrm{LP}}(\vec{p}) \triangleq \max_{\vec{z}\in Z} L_{\mathrm{LP}}(\vec{z},\vec{p}), \tag{A.4}$$

where Z is the set that contains all of $(\vec{x},\vec{y})$'s satisfying Eqs. (3.3) and (3.4). Let $F_{\mathrm{LP}}(\vec{z}) = \sum_{n\in N} w_n x_n$, and $\tilde{\vec{z}}$, $\tilde{\vec{p}}$ be any feasible primal and dual solutions, respectively. Due to the duality theory [40, Ch. 5.1.3], it follows that for $0 < \gamma < 1$,

$$F_{\mathrm{LP}}(\tilde{\vec{z}}) \geq \gamma \cdot D_{\mathrm{LP}}(\tilde{\vec{p}}) \Longrightarrow F_{\mathrm{LP}}(\tilde{\vec{z}}) \geq \gamma \cdot F^*_{\mathrm{LP}}, \tag{A.5}$$

where F^*_{LP} denotes the maximum of $\mathrm{LP}_{\mathrm{OSCA}}$.

We show the correctness of Alg. 10 using Eq. (A.5). For a given channel assignment of sniffers, which we denote by an integer vector $\vec{y}^{\mathrm{int}}$, the monitoring coverage due to $\vec{y}^{\mathrm{int}}$ is given as $\sum_{n\in N} w_n x_n^{\mathrm{int}}$, where $x_n^{\mathrm{int}} = \min\left\{1, \sum_{(s,c):n\in K_{s,c}} y_{s,c}^{\mathrm{int}}\right\}$, which is equal to C_{root} in Alg. 10. It is easy to see that $\vec{z}^{\mathrm{int}} = (\vec{x}^{\mathrm{int}}, \vec{y}^{\mathrm{int}})$ is a feasible solution to $\mathrm{LP}_{\mathrm{OSCA}}$. We next compute $D_{\mathrm{LP}}(\tilde{\vec{p}})$ for any given $\tilde{\vec{p}} \geq 0$. We rewrite Eq. (A.3) as

$$L_{\mathrm{LP}}(\vec{z},\tilde{\vec{p}}) = \sum_{n\in N} (w_n - \tilde{p}_n) x_n + \sum_{s\in S}\sum_{c\in C} \left(\sum_{n\in K_{s,c}} \tilde{p}_n \right) y_{s,c}. \tag{A.6}$$

For the given $\tilde{\vec{p}}$, we can obtain $\vec{z}^* \in Z$ that maximizes $L_{\mathrm{LP}}(\vec{z},\tilde{\vec{p}})$ subject to $\vec{z} \in Z$ as

$$\begin{aligned} x_n^* &= \begin{cases} 1 & \text{if } w_n \geq \tilde{p}_n, \\ 0 & \text{otherwise,} \end{cases} \\ y_{s,c}^* &= \begin{cases} 1 & \text{for } c^* \in \operatorname{argmax}_{c\in C} \sum_{n\in K_{s,c}} \tilde{p}_n, \\ 0 & \text{for all } c \neq c^*. \end{cases} \end{aligned} \tag{A.7}$$

Using Eqs. (A.4), (A.6), and (A.7), we can obtain $D_{\mathrm{LP}}(\tilde{\vec{p}})$ for the given $\tilde{\vec{p}}$ as

$$D_{\mathrm{LP}}(\tilde{\vec{p}}) = \sum_{n\in N} [w_n - \tilde{p}_n]^+ + \sum_{s\in S} \sum_{n\in K_{s,c^*}} \tilde{p}_n,$$

where $c^* \in \operatorname{argmax}_{c \in C} \sum_{n \in K_{s,c}} \tilde{p}_n$. Hence, $D_{\mathrm{LP}}(\tilde{\vec{p}})$ is equal to D_{root} in Alg. 10. Therefore, due to Eq. (A.5), if $C_{\mathrm{root}} \geq \gamma \cdot D_{\mathrm{root}}$, then $C_{\mathrm{root}} \geq \gamma \cdot F^*_{\mathrm{LP}}$, which concludes the proof.

LIST OF REFERENCES

LIST OF REFERENCES

[1] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang, "Security in Mobile Ad Hoc Networks:Challenges and Solutions," *IEEE Wireless Communications*, vol. 11, pp. 34–47, February 2004.

[2] N. B. Salem and J.-P. Hubaux, "Securing Wireless Mesh Networks," *IEEE Wireless Communications*, vol. 13, no. 2, pp. 50–55, 2006.

[3] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network Magazine*, vol. 13, no. 6, pp. 24–30, 1999.

[4] Y. Yang, S. Zhu, and G. Cao, "Improving Sensor Network Immunity under Worm Attacks: a Software Diversity Approach," in *Proc. of ACM MobiHoc*, pp. 149–158, May 2008.

[5] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks," in *Proc. of IEEE Broadnets, San Jose, CA*, October 2004.

[6] M. Alicherry, R. Bhatia, and L. Li, "Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Networks," in *Proc. of ACM MobiCom, Cologne, Germany*, August 2005.

[7] V. Bhandari and N. H. Vaidya, "Capacity of Multi-Channel Wireless Networks with Random (c, f) Assignment," in *Proc. of ACM MobiHoc, Montreal, QC, Canada*, September 2007.

[8] V. Bhandari and N. H. Vaidya, "Connectivity and Capacity of Multi-Channel Wireless Networks with Channel Switching Constraints," in *Proc. of IEEE INFOCOM, Anchorage, Alaska, USA*, May 2007.

[9] C. Chereddi, P. Kyasanur, and N. H. Vaidya, "Design and Implementation of a Multi-Channel Multi-Interface Network," in *REALMAN*, May 2006.

[10] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *Proc. of ACM MobiCom, Philadelphia, Pennsylvania, USA*, September 2004.

[11] C. Kim, Y.-B. Ko, and N. H. Vaidya, "Link-State Routing Protocol for Multi-Channel Multi-Interface Wireless Networks," in *Proc. of IEEE MILCOM, San Diego, CA, USA*, November 2008.

[12] M. Kodialam and T. Nandagopal, "Characterizing the Capacity Region in Multi-Radio Multi-Channel Wireless Mesh Networks," in *Proc. of ACM MobiCom, Cologne, Germany*, August 2005.

[13] P. Kyasanur and N. H. Vaidya, "Capacity of Multi-Channel Wireless Networks: Impact of Number of Channels and Interfaces," in *Proc. of ACM MobiCom, Cologne, Germany*, August 2005.

[14] P. Kyasanur, J. So, C. Chereddi, and N. H. Vaidya, "Multi-Channel Mesh Networks: Challenges and Protocols," *IEEE Wireless Communications*, April 2006.

[15] P. Kyasanur and N. H. Vaidya, "Routing and Link-layer Protocols for Multi-Channel Multi-Interface Ad Hoc Wireless Networks," *SIGMOBILE Mobile Computing and Communications Review*, vol. 10, pp. 31–43, January 2006.

[16] S.-H. Lim, C. Kim, Y.-B. Ko, and N. H. Vaidya, "An Efficient Multicasting for Multi-Channel Multi-Interface Wireless Mesh Networks," in *Proc. of IEEE MILCOM, Boston, USA*, October 2009.

[17] X. Lin and S. Rasool, "A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad Hoc Wireless Networks," in *Proc. of IEEE INFOCOM, Anchorage, Alaska, USA*, May 2007.

[18] S. Merlin, N. H. Vaidya, and M. Zorzi, "Resource Allocation in Multi-Radio Multi-Channel Multi-Hop Wireless Networks," in *Proc. of IEEE INFOCOM, Phoenix, AZ, USA*, April 2008.

[19] A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," in *Proc. of IEEE INFOCOM, Miami, FL, USA*, March 2005.

[20] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized Algorithms for Multi-channel Wireless Mesh Networks," *ACM Mobile Computing and Communications Review*, April 2004.

[21] J. So and N. H. Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Tranceiver," in *Proc. of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2004.

[22] J. So and N. H. Vaidya, "Routing and Channel Assignment in Multi-Channel Multi-Hop Wireless Networks with Single Network Interface," in *Proc. of The Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine)*, August 2005.

[23] J. D. Camp and E. W. Knightly, "The IEEE 802.11s Extended Service Set Mesh Networking Standard," *IEEE Communications Magazine*, vol. 46, pp. 120–126, August 2008.

[24] P. Kyasanur and N. H. Vaidya, "Detection and Handling of MAC Layer Misbehavior in Wireless Networks," in *Proc. of DSN*, pp. 173–182, Jun. 2003.

[25] D. Subhadrabandhu, S. Sarkar, and F. Anjum, "A Framework for Misuse Detection in Ad Hoc Networks—Part I," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 274–289, February 2006.

[26] D. Subhadrabandhu, S. Sarkar, and F. Anjum, "A Framework for Misuse Detection in Ad Hoc Networks—Part II," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 290–304, February 2006.

[27] D. Subhadrabandhu, S. Sarkar, and F. Anjum, "A Statistical Framework for Intrusion Detection in Ad Hoc Networks," in *Proc. of the 25th IEEE International Conference on Computer Communications (INFOCOM'06), Barcelona Spain*, April 2006.

[28] I. Khalil, S. Bagchi, and N. B. Shroff, "A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks," in *Proc. of IEEE/IFIP DSN*, pp. 612–621, Jun.-Jul. 2005.

[29] A. Dhananjay, H. Zhang, J. Li, and L. Subramanian, "Practical, Distributed Channel Assignment and Routing in Dual-radio Mesh Networks," in *Proceedings of ACM SIGCOMM*, pp. 99–110, 2009.

[30] "Maxim 2.4 GHz 802.11b Zero-IF Transceivers," in *http://pdfserv.maxim-ic.com/en/ds/MAX2820-MAX2821.pdf*.

[31] R. Chandra, P. Bahl, and P. Bahl, "MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card," in *Proceedings of IEEE INFOCOM*, pp. 882–893, 2004.

[32] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.

[33] D. S. Hochbaum, *Approximation Algorithm for NP-Hard Problems*. PWS Publishing Company, Massachusetts, 1997.

[34] C. Chekuri and A. Kumar, "Maximum Coverage Problem with Group Budget Constraints and Applications," in *Proc. of Approximation, Randomization, and Combinatorial Optimization (APPROX)*, pp. 72–83, Springer LNCS, 2004.

[35] A. Ageev and M. Sviridenko, "A New Method of Constructing Algorithms with Proven Performance Guarantee," *Journal of Combinatorial Optimization*, vol. 8, pp. 307–328, 2004.

[36] A. Srinivasan, "Distributions on Level-Sets with Applications to Approximation Algorithms," in *Proc. of IEEE FOCS*, pp. 588–597, Oct. 2001.

[37] K. M. Anstreicher, "Linear Programming in O([n3/ln n]L) Operations," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 803–812, 1999.

[38] A. Chhetri, H. Nguyen, G. Scalosub, and R. Zheng, "On quality of monitoring for multi-channel wireless infrastructure networks," in *Proc. of ACM MobiHoc*, September 2010.

[39] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, New Jersey, 1989.

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[41] X. Lin and N. B. Shroff, "Utility Maximization for Communication Networks with Multi-path Routing," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 766–781, 2006.

[42] "Tutorials, references, activities and tools for submodular optimization."

[43] D.-Z. Du, R. L. Graham, P. M. Pardalos, P.-J. Wan, W. Wu, and W. Zhao, "Analysis of Greedy Approximations with Nonsubmodular Potential Functions," in *Proc. of ACM-SIAM SODA*, January 2008.

[44] B. Borchers and J. G. Young, "Implementation of a primal-dual method for SDP on a shared memory parallel architecture," *Computational Optimization and Applications*, vol. 37, pp. 355–369, July 2007.

[45] P. Arora, C. Szepesvari, and R. Zheng, "Sequential Learning for Optimal Monitoring of Multi-channel Wireless Networks," in *Proc. of IEEE INFOCOM*, April 2011.

[46] P. Arora, N. Xia, and R. Zheng, "A Gibbs Sampler Approach for Optimal Distributed Monitoring of Multi-Channel Wireless Networks," in *Proc. of IEEE GLOBECOM*, 2011.

VITA

VITA

Donghoon Shin was born in Seoul, South Korea. He received his B.E. and M.S. degrees in Electrical Engineering from Korea University in 2003 and from Korea Advanced Institute of Science and Technology (KAIST) in 2006, respectively. Donghoon started to pursue his Ph.D. in the School of Electrical and Computer Engineering at Purdue University, West Lafayette, Indiana, in fall 2006. He has been conducting research on network security under the guidance of Professor Bagchi, and also has been working with Professor Ness B. Shroff, Professor Xiaojun Lin, and Professor Chih-Chun Wang for his dissertation research and other research projects. His research interests lie broadly in the areas of wireless networks, mobile systems, and smart grids with emphasis on mathematical modeling, algorithm/protocol design and performance analysis for security of these systems. He has worked as an intern at Intel in Hillsboro, Oregon, in Standards and Advanced Technology Group from August 2011 to June 2012.