AUTONOMOUS SPECTRUM ENFORCEMENT: A BLOCKCHAIN APPROACH

by

Maqsood Ahamed Abdul Careem

A Thesis

Submitted to the University at Albany, State University of New York

in Partial Fulfillment of

the Requirements for the Degree of

Masters of Science

College of Engineering and Applied Sciences

Department of Electrical and Computer Engineering

December, 2019

ABSTRACT

A core limitation in existing wireless technologies is the scarcity of spectrum, to support the exponential increase in Internet-connected and multimedia-capable mobile devices and the increasing demand for bandwidth-intensive services. As a solution, Dynamic Spectrum Access policies are being ratified to promote spectrum sharing for various spectrum bands and to improve the spectrum utilization. This poses an equally challenging problem of enforcing these spectrum policies. The distributed and dynamic nature of policy violations necessitates the use of autonomous agents to implement efficient and agile enforcement systems. The design of such a fully autonomous enforcement system is complicated due to the lack of trust in the agents and the requirement for agile scheduling schemes. We architect a deployable system, which leverages crowdsourced agents as eye-witnesses, to efficiently deploy mobile, multi-modal agents (unmanned land, sea or aerial vehicles) to potential spectrum infraction sites to collectively improve the enforcement accuracy. We leverage the distributed consensus mechanism employed in Blockchain networks to make distributed accurate and credible inferences even from trust-less agents. Collectively this leads to a highly reliable and feasible autonomous spectrum enforcement strategy, which outperforms static and purely crowdsourced enforcement paradigms.

To Mom and Dad

ACKNOWLEDGMENTS

I would like to begin by thanking my advisor, Prof. Aveek Dutta for his encouragement, advice and guidance through my career as a graduate student. I am also grateful for the insightful comments provided by Prof. Dola Saha, which have helped enrich my research. I would also like to thank Prof. Hany Elgala and Prof. Dola Saha, for serving as my committee members. Finally, yet most importantly, words cannot express my gratitude towards my parents and my siblings, for their perpetual support, motivation, sacrifices, and love. Thank you for making me the person I am today: this thesis is dedicated to you.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
Introduction

As sharing policies are being ratified by the Federal Communications Commission (FCC) for various spectrum bands for commercial broadband use, it poses an equally challenging problem of enforcing these policies. The distributed and dynamic nature of these policy violations necessitates the use of mobile autonomous agents (e.g., crowdsourcing, unmanned ground, sea or aerial vehicles) to implement cost-effective and efficient enforcement systems. The efficacy of such a distributed enforcement system greatly depends on the *accuracy or reliability* of evidential and inferred information and the *speed* of adjudication. Central to this problem is the lack of trust or reputation of the participating Agents, which often leads to incorrect and biased inferences. Additionally, leveraging heterogeneous mobile agents (with distinct sensing and hardware capabilities) for spectrum enforcement, requires the design of unique scheduling, detection and localization schemes that adapt to the hostility of the wireless environment. We define this problem as *Autonomous Spectrum Enforcement* and it involves two stages: 1. Autonomous Spectrum Sensing: We schedule multi-modal agents to detect, localize and adjudicate spectrum infractions and collectively improve the accuracy of these enforcement tasks in minimum possible time. 2. Distributed Fusion System: We leverage the distributed consensus mechanism employed in Blockchain networks to record and disseminate the sensing reports among agents and accrue their reputation, leading to credible inferences and a highly reliable and accurate enforcement system (without relying on centralized, explicitly trusted entity). Designed as a practical and deployable system, our solution leverages crowdsourced agents (as eye-witnesses) to deploy mobile agents to infraction sites depending on the hostility of the wireless environment. We evaluate the Autonomous Enforcement system using a novel integrated Spectrum Sensing and Blockchain simulator and show that such an Autonomous Enforcement strategy significantly improves the enforcement accuracy and feasibility of dynamic and distributed spectrum violations, over static paradigms.

1.1 Autonomous Spectrum Enforcement

Enforcement of spectrum policies is complementary to the well-studied problem of Dynamic Spectrum Access (DSA). However, the distributed nature of these policy violations (defined as

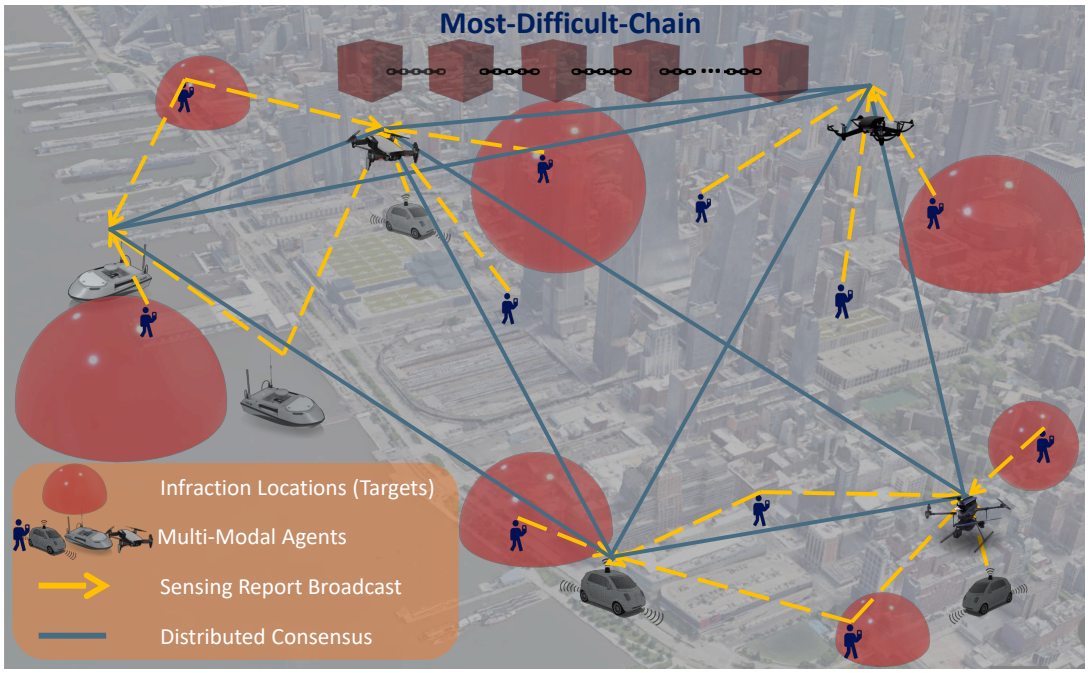


Figure 1.1: Distributed nature of Spectrum infractions require multi-modal agents. We leverage crowdsourced agents (as eye-witnesses) to deploy mobile agents (ground, sea or aerial vehicles) to infraction sites. Agents broadcast their reports and arrive at distributed consensus on the reputation of agents, enabling credible distributed spectrum enforcement.

"Targets") require accurate, cost-effective and mobile, autonomous entities (defined as *"Agents"*) to carry out all enforcement related tasks, as shown in figure 1.1. These tasks can be generalized as various levels of sensing (signal measurement) and decision making (waveform classification and localization in order to pin-point rogue sources) with very high accuracy. The balance between cost and accuracy of such a fully autonomous enforcement system critically depends on, 1. the appropriate amount of sensing resources (agents) mobilized to the potential infraction locations in the shortest possible time, and 2. the ability to make credible inferences from distributed fusion of sensing results from a set of spatially scattered trust-less agents, without relying on an explicitly trusted centralized entity. This is because wireless signal classification greatly benefits from proximity of agents to the potential source, heterogeneous sensing parameters (bandwidth, sample rate, battery constraints, etc) and the reliable aggregation of observations from multiple distributed agents.

Thus, we propose and architect an autonomous enforcement system where we rely on multi-modal agents (crowdsourced, land-based, sea-based or aerial vehicles) to autonomously sense, detect, localize and adjudicate spectrum policy violations with the highest possible accuracy. The

autonomous enforcement system is shown in figure 1.2 and involves two key components:

1. Autonomous Spectrum Sensing: Leverages crowdsourced measurements as eye-witness accounts to deploy mobile, multi-modal agents (unmanned ground, sea or aerial vehicles) to potential infraction sites for further sensing, depending on the veracity of crowd measurements.

2. Distributed Fusion System: Leverages the distributed consensus mechanism employed in Blockchain networks to record and disseminate the sensing reports and their veracity. This information is used to make reliable inferences among distributed trust-less agents.

Even though the purely crowdsourced paradigm [1] has been shown as a viable apparatus for distributed spectrum sensing, the lack of trust and incentives, limit the enforcement accuracy of such schemes. Hence, we envision a hybrid approach that leverages crowdsourced measurements as eye-witness accounts to deploy mobile agents to further improve the accuracy of detection and localization of violations. Specifically, we leverage the measurements from crowdsourced mobile users to determine the optimum routing schedule for the agents to achieve the desired level of accuracy of detection and localization at minimum possible cost. The design and analysis of this multi-modal autonomous sensing system is detailed in Chapter 3.

Autonomous enforcement of spectrum policies requires the distributed fusion of sensing results from a set of spatially scattered agents to detect and localize spectrum violations with the highest possible accuracy, without relying on centralized, static, explicitly trusted infrastructure for data fusion and decision making. This problem is complicated, due to the lack of trust of the participating agents, which results in incorrect and biased inferences. In Chapter 4 we propose `SenseChain`, where we leverage the distributed consensus mechanism employed in Blockchain networks to create and disseminate an immutable record of the sensing reports and the behaviour of agents. The information disseminated via `SenseChain` enables the assessment of reputation of the participating agents and credible distributed fusion at the agents. This leads to highly reliable and accurate inferences even from distributed trust-less entities.

Figure 1.1 shows the distributed and heterogeneous nature of policy violations. These policy violations include greedy users who violate spectrum policies, and acts like denial of service and jamming attacks. Crowdsourced agents serve as eye-witnesses and provide initial estimates on the detection and location of the spectrum violations. These estimates are used to efficiently deploy

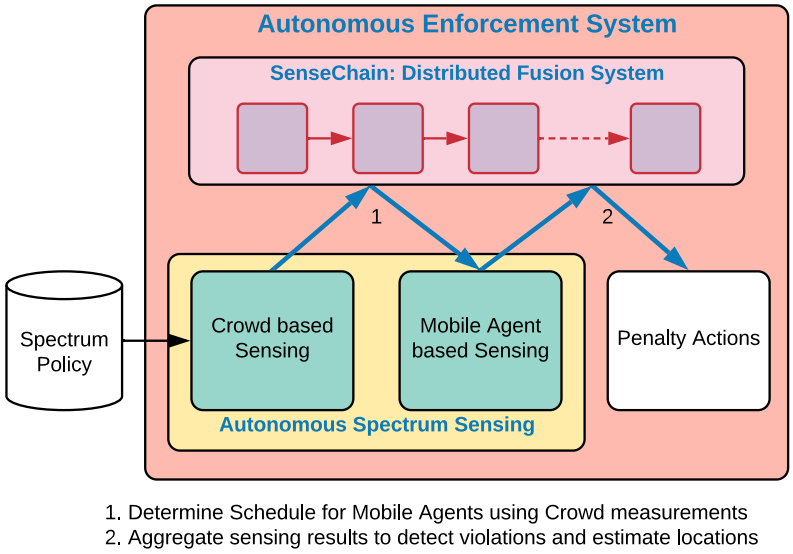


Figure 1.2: Autonomous Enforcement System: Crowdsourced measurements provide the basis for deploying mobile agents to detect and localize targets. The individual sensing reports and their veracity are recorded in `SenseChain` and disseminated to all agents. The information in `SenseChain` is used to make credible inferences even in the presence of trust-less agents.

mobile agents (unmanned ground, sea or aerial vehicles) to potential infraction sites to collectively improve the enforcement accuracy. All the sensing reports are broadcasted, validated and the agents arrive at distributed consensus (by leveraging the consensus mechanisms in blockchain networks) on the veracity of reports and the reputation of agents. This information is used to make credible inferences in a distributed manner that leads to an accurate and reliable enforcement system.

The autonomous enforcement system in figure 1.2 operates as follows: Certain agents assume the role of a Sensor while others assume the role of a Validator. Sensors are tasked primarily with sensing and reporting their measurements of Signal-to-Noise Ratio (SNR), Location (Loc), Probability of Detection (P_d) and Probability of False-positive (P_f)[1]. Validators are tasked with assessing the veracity of the sensing reports, recording and disseminating this information via `SenseChain` and serve as distributed fusion nodes. Thus, `SenseChain` represents an immutable record of the sensing reports that all the agents arrive at consensus upon, and can be used for credible distributed inferences. The initial assessment of the target from the crowd-

[1]Each Agent is provided with a set of rules to check for infraction and are equipped with the corresponding detectors either in hardware or software. [2, 3]

sourced agents (voluntary mobile users) is broadcasted for peer-validation by the Validators and are recorded in `SenseChain` along with their assessed veracity. This crowdsourced information is used by the validators to derive the optimal number (defined as "*Cardinality*") and routing schedule of mobile agents necessary to collectively improve the accuracy (of detection and localization of spectrum violations) at a bounded cost. This information is used to deploy mobile agents to perform the additional detection and localization tasks under the constraint of scheduling a fixed number of agents in minimum time. Each agent visits each target to collect measurements and broadcast these measurements. These sensing reports are validated by peer-validators and added to `SenseChain`. This ensures, the credible aggregation of multiple sensing results (using a reputation based weighted combination) at a very high SNR (due to the proximity of agents to targets), and hence lends to a highly reliable autonomous enforcement system[2].

1.2 Contributions

We design and analyze a *practical and deployable autonomous enforcement system*, making the following contributions:

1. Design of a multi-modal, autonomous sensing system that leverages crowdsourced measurements (as eye-witness accounts) to deploy mobile agents to potential infraction sites, to improve the enforcement accuracy in least possible time (Chapter 3).

2. Design of a fully distributed, credible fusion system by leveraging the distributed consensus mechanism in Blockchain networks to record sensing information and capture the reputation of sensors (Chapter 4).

3. Evaluation of the proposed Autonomous Enforcement system using a novel combined Spectrum Sensing and Blockchain simulator (Chapter 5).

1.3 Previously Published Content

The following publications were a direct result of the work presented in this thesis and collectively constitute the complete Autonomous Enforcement System. I am the primary researcher of these

[2]Penalty Actions include authoritative adjudication on the infraction and levying monetary or other forms of penalty to the Violator.

IEEE publications[3].

The Chapters on *Multi-Modal Autonomous Spectrum Sensing* (Chapter 3) and *Evaluation and Results* (Chapter 5) revise the following previous publications,

- "Spectrum Enforcement and Localization Using Autonomous Agents With Cardinality," [4], Maqsood Ahamed Abdul Careem, A. Dutta and W. Wang in IEEE Transactions on Cognitive Communications and Networking.

- "Multi-Agent Planning with Cardinality: Towards Autonomous Enforcement of Spectrum Policies," [5] Maqsood Ahamed Abdul Careem, Aveek Dutta and Weifu Wang in IEEE International Symposium on Dynamic Spectrum Access Networks 2018.

The Chapters on `SenseChain:` *Blockchain based Distributed Fusion System* (Chapter 4) and *Evaluation and Results* (Chapter 5) revise a previous publication,

- "SenseChain: Blockchain based Reputation System for Distributed Spectrum Enforcement," [6] Maqsood Ahamed Abdul Careem and Aveek Dutta in IEEE International Symposium on Dynamic Spectrum Access Networks 2019.

[3]**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:** Requirements to be followed when using an entire IEEE copyrighted paper in a thesis: 1) The following IEEE copyright/ credit notice *should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]* 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line. 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

CHAPTER 2
Related Work

We categorize the related literature on Autonomous Spectrum Enforcement into three categories: Spectrum Enforcement strategies, Sensing using Autonomous Agents and Distributed Fusion Systems.

2.1 Spectrum Enforcement strategies

The literature on spectrum sensing and enforcement of spectrum policies have leveraged diverse mechanisms that range from static sensor deployments and crowdsourced paradigms using a set of mobile crowd users. However, a majority of this literature rely on centralized fusion entities or static deployments of sensors. We discuss the most relevant literature on Spectrum enforcement and the related literature on the use of autonomous agents for spectrum sensing in this section.

Static Enforcement paradigms: In practice, networks comprised of opportunistic users have limited communication range and transmit power [7],[8], to avoid interference to neighboring networks sharing the same frequency band. This increases the difficulty of detecting spectrum infractions with the desired accuracy unless the detector is in the vicinity of the infraction. So, dense deployment of static infrastructure with a wide coverage can be prohibitively expensive for practical purposes. Even with mesh deployments of static enforcers unless the policy violation events occur in the vicinity of the enforcers, it would lead to poor localization and false alarms [1]. In contrast we leverage autonomous agents which are routed to the vicinity of the potential infraction locations as determined from initial crowdsourced estimates to achieve the desired performance.

Crowdsourced Paradigms: Crowd-sourcing has been made a reality by commodity user devices such as Smartphones. A study on the incentives of a generic crowd-sourced paradigm is shown in [9]. The ubiquity of mobile devices to aggregate information have been embraced in traffic and civilian law enforcement agencies, which have long relied on eye-witness accounts [10, 11]. Enforcement of Spectrum policies using crowdsourced mobile users lies at the crossroads of many different research areas. A description on ex-ante and ex-post paradigms for spectrum enforcement is presented in [12]. The need and design criterion for federal and non-federal frequency bands has been presented in [13], which are crucial for practical implementation. [14] outlines the common

threats in Dynamic Spectrum Access. [15] presents a specific form of enforcement that leverages and depends on advanced coding and detection theory. Authors in [1] present a crowdsourced method that harnesses the collective power of the "crowd" as opposed to individual capabilities for any form of infraction as long as the Receiver operating characteristic (ROC) is available. They have shown that crowdsourced approaches outperform static enforcement schemes. However, in addition to the drawbacks of the purely crowdsourced paradigms presented in literature (lack of trust, controlability and incentive strategies), they also rely on a centralized, explicitly trusted fusion center for reliable enforcement. In contrast, we present a fully distributed enforcement system where the fusion of information is performed by the scattered set of agents.

2.2 Sensing using Autonomous Agents:

Recent research has shown growing interest in collaborative autonomous agents (Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs)) for applications ranging from multi-agent cooperation [16], planning [17, 18] and sensing [19, 20]. A core issue in leveraging teams of mobile, autonomous agents, is the determination of an optimal schedule of the agents. Most applications of hybrid UAV-UGV planning have limited scope to small sets of agents & targets. The Scheduling problem has its roots in Multiagent planning (MP) [21] which is the NP-hard problem [22] of finding the shortest paths of agents with targets visited at least once. In general, these methods do not enforce unique visits at targets. In the context of Multiagent Planning with uniqueness (MPU), an intriguing class of problems is the Multiple Traveling Salesmen Problem (MTSP), which finds closed tours for agents, while enforcing uniqueness. MTSP is challenging due to its combinatorial nature and NP-hardness [23]), and it does not solve the MPU directly as it yields tours (not paths) for agents. [24] proposes a heuristic search method to solve the Multiagent Path Finding problem, which is similar to MPU, except that endpoints of tours are also fixed. [23] presents a genetic algorithm based method for solving the MPU.

For detecting and localizing infractions we require multiple, unique agents visiting each infraction to address its hostility (Cardinality). An exact method for heterogeneous MTSP (some targets can be visited only by a specific agent) is provided in [25]. The class of MTSP does not addresses the notion of multiple visits. An interesting class of problems here, is the Vehicular Routing Problems (VRP) [26], which might facilitate multiple visits [27], however does not ensure uniqueness or constraints on number of visits (i.e., the cardinality). Multi-Depot VRP (MDVRP),

introduce some notion of heterogeneity (some targets can be visited only by a specific agent), for which a 8-approximation algorithm is presented in [28]. However, these approaches are based on graph partitioning, where imposing cardinality constraint is challenging. Multiagent patrolling problems [29] enable targets to be visited multiple times, however by the same agent. Under the constraint of Cardinality, the MP problem evolves to Multiagent Planning with Cardinality (MPC) problem. To the authors best knowledge, the challenging problem of Multiagent planning with Cardinality and the notion of using the crowd as eyewitnesses to efficiently deploy agents, to improve the enforcement of spectrum policies, is unprecedented in literature.

2.3 Distributed Fusion Systems

We categorize the related literature on Distributed Fusion Systems into three groups:

Centralized vs Distributed Inferences: A majority of the spectrum sensing and spectrum enforcement literature rely on centralized or hierarchical fusion of sensing information by an explicitly trusted dedicated entity [30, 31, 1]. While the centralized architecture is theoretically optimal, it requires high communication bandwidth to send sensed and inferred data to and from the fusion node, which should have enough computational resources to process the data. In a fully autonomous and distributed architecture, there is no fixed superior-subordinate relationship among entities and the assumption of such dedicated infrastructure may be impractical and restrictive. Each agent may communicate with other agents subject to connectivity constraints and each agent may have its own processor to fuse the local sensing data. Distributed fusion architectures have the following advantages [32]: lighter processing load at each distributed fusion node, lower communication load, faster access to fusion results due to lower communication delay, and higher survivability since there is no single point of failure associated with a central fusion node. A core challenge in distributed fusion is the lack of trust in the participating entities. There exist distribute fusion schemes where the fusion task is distributed to a multiple scattered nodes, however they mostly assume that the fusion nodes are trustworthy [33]. In contrast, `SenseChain` achieves credible inferences even in the presence of trust-less entities.

Anomalous behaviour Detection: Spectrum enforcement systems is analogous to intrusion detection systems in wired and wireless networks [34, 35]. Generally speaking, intrusion detection systems are limited because all detectors observe the infringement or infringement event at similar granularity and thus yield the same detection results. Detection of infringements that spread over

radio waves, however, is *challenging* due to propagation-related losses that decrease the signal integrity. This serves as one of the most significant and motivating factors for detection and inference using autonomous and crowdsourced approaches, which far outperform static enforcement schemes.

Trust and reputation based models for malicious sensor identification have been widely studied in the context of wireless sensor networks [36]. [37] uses a neighbor weight trust algorithm, in the problem of malicious node detection. [38] proposed a new trust management scheme based on D-S (Dempster-Shafer) evidence theory, by considering the spatio-temporal correlation of data collected by neighbouring sensors. These models rely on local inferences from neighbours, which needs to be disseminated throughout the network of trustless entities. In our work we achieve distributed consensus among nodes by sharing information on a blockchain. [39] proposed a malicious node recognition model to resist malicious behavior of high-reputation nodes in existing WSNs. [40] proposes an abnormal sensor identification using the pairwise similarity of sensing results of helpers. Trust has been investigated in the context of crowd-sensing and collaborative spectrum sensing [1], [41]. Most of these approaches rely on centralized fusion of information, which is both vulnerable and does not scale well. In contrast, we propose anomaly detection in a purely distributed manner using only the SNR and the location of sensors, and the dissemination of information using the blockchain to assign reputation of sensors.

Blockchains for sensor networks: DLTs like blockchain have gained immense interest in various application domains in wireless sensor networks [42]. Blockchains have been employed for dynamic spectrum access [43] and to achieve secure routing among malicious nodes [44]. Blockchains have been used to establish a trust model and for the detection of malicious nodes in [36]. [45] proposed a smart contract based framework to solve the problems of trusted access control and distributed in the IoT. [46] addresses the problem of distributing trust and reputation among trustless nodes, by employing collaboration among miners. [33] proposes spectrum sensing as a service using a smart contract to describe the sensing service parameters and helpers are rewarded only if they perform sensing accurately. In contrast to these approaches we employ peer-based anomaly detection algorithm, a heterogeneous difficulty assignment and *Most-Difficult-Chain* rule to guide credible inferences on detection and localization.

CHAPTER 3

Multi-Modal Autonomous Spectrum Sensing

The distributed nature of policy violations (Targets)[4] in spectrum sharing necessitate the use of mobile autonomous agents (*e.g.*, UAVs, self-driving cars, crowdsourcing) to implement cost-effective and efficient enforcement systems. We define this problem as Multi-agent Planning with Cardinality (MPC), where Cardinality represents multiple, unique agents visiting each infraction location to collectively improve the accuracy of the enforcement tasks. Designed as a practical and deployable system, our solution leverages crowdsourced information to determine the optimum Cardinality and provide a routing schedule for the agents to achieve the desired level of accuracy of detection and localization at minimum possible cost. We show that by estimating spatial orientation of the agents with single antenna, the accuracy is improved by 96% over crowdsourcing only. Using geographical maps as the basis, we solve the scheduling problem with a 3-approximation ratio in polynomial time that exhibits statistically similar performance under variety of urban locale across multiple continents. The longest path traversed by an agent on average is 1.2km per unit diagonal length of a rectangular geographic area, even when there are twice as many infractions as agents. Deploying UAVs to the estimated region of infraction improves localization accuracy by $\approx 70\%$ compared to ground vehicles.

3.1 Multi-Modal Agent Scheduling

The design of an efficient and accurate enforcement system critically depends on, the appropriate amount of agents[5] mobilized to the right location in the shortest possible time. To this end, crowdsourced paradigm [1, 47, 48] has been shown as a viable apparatus. However, it suffers from many inefficiencies like lack of trust and efficient incentive mechanisms and controlability, that may not provide bounded guarantees of accuracy (*e.g.*, detection and location) and cost (*e.g.*,

[4]This includes greedy users who violate spectrum policies, and acts like denial of service and jamming attacks. We primarily focus on violations that leverage the physical layer of the device. e.g., Emulating a higher priority user (or incumbent) using signatures like cyclostationary features embedded in the radio signal. This also includes unintentional infractions such as faulty hardware causing unwanted spectral leakage. These actions alter the signal characteristics, causing harmful interference to rightful users of that band.

[5]We do not impose any restriction on the type of autonomous agents as long as those use the road infrastructure to navigate. These agents can be crowd mobile users, radio nodes mounted on autonomous vehicles or low flying UAVs (for sensing ground based communication and avoid obstacles) or a combination of all.

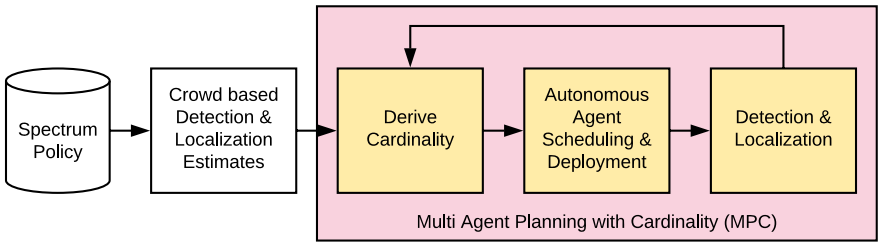


Figure 3.1: Autonomous Spectrum Sensing System: Crowdsourced measurements provide the basis for deploying mobile agents to detect and localize targets.

incentives, capital and operational costs). Instead, we envision a hybrid approach that leverage crowdsourced measurements (akin to eye-witness accounts) to deploy mobile, autonomous agents to the target sites depending on the veracity of these measurements. Our work builds on any crowdsourced paradigm, where the *wisdom of crowd* is simply used to assess the need for additional resources to achieve a desired level of accuracy and cost, thus avoiding unnecessary and restrictive burden on the *crowd* (like undesired mobility, prioritized sensing, low incentive, etc [49, 48]). This system operates in two steps as shown in figure 3.1. The initial assessment of the target[6] from the crowdsourced agents (voluntary mobile users) is broadcased and recorded in `SenseChain` by the validators. This crowdsourced information is used by the agents to derive the *cardinality* necessary to collectively improve the accuracy at a bounded cost. This information is used to deploy mobile agents to perform the additional detection and localization tasks under the constraint of scheduling a fixed number of agents in minimum time.

We define this problem as Multiagent Planning with Cardinality (MPC). Cardinality, refers to the number of unique mobile agents (not including the participants from the crowd) visiting targets, simultaneously or otherwise, to achieve a target accuracy of the enforcement tasks. Accuracy has two primary dimensions: a) Detection of a *bad* signal (often expressed as a confusion matrix [50] and b) Location estimate. Unlike the crowd participants, the mobile autonomous agents can be directed (although at a cost) to a near-optimal orientation or detect signals with desirable operating points to independently maximize along both the dimensions. We adopt the widely used geometric trilateration [1, 51] as the basis to locate a target and calculate the optimum cardinality that minimizes the Geometric Dilution of Precision (GDOP). This is followed by routing a finite number of agents to multiple targets while fulfilling the cardinality determined in the previous step.

[6]Crowdsourced agents may detect infractions with a wide variety of accuracy (false and true positives) due to heterogeneous hardware and their relative proximity to the target. There are many crowdsourced models [1, 48] but our work subsumes any such paradigm without loss of generality.

The solution to this lies at the intersection of finding the shortest path between nodes in a graph and finding a schedule (or order) for the agents to visit a set of targets. However, in MPC, the additional requirement of fulfilling the cardinality for each target, makes the solution orthogonal to the existing literature [52, 26]. It is not necessary to route all the agents (as per the cardinality) to a target at the same time. To ensure a fast convergence of the scheduling algorithm the agents may start from any point and take any path as long as it covers all the targets in the least possible time.

In the final step the accuracy is iteratively improved until the target level is achieved. Trilateration with no GDOP results in a convex polygon that includes the target (Section 3.2). Each agent is initially routed to the centroid of the polygon and then visits each vertex to collect measurements and broadcast these measurements. These sensing reports are validated and added to `SenseChain`. This ensures, the credible aggregation of multiple sensing results (using a reputation based weighted combination) at a very high SNR (due to the proximity of agents to targets), and hence lends to a highly reliable enforcement system. It is to be noted that these tasks may involve deeper signal processing and possibly indoor sensing as well, which is not in the scope of this work. Since, the cost incurred to conduct this localized sensing, is small compared to the overall cost of scheduling it can be safely ignored in the larger context of the cost of enforcement.

Collectively, these three parts constitute a solution to the MPC problem that operate in lockstep with any crowdsourced paradigm to achieve very high accuracy at a bounded cost that is also minimum under the above constraints. The design and analysis of the Autonomous Sensing scheme is further detailed in Chapter 3.

3.2 Background and Known Results

Trilateration under noise: Although our solution is independent of the underlying crowdsourced paradigm, we adopt trilateration based localization [1] to derive the cardinality for the infractions. Trilateration [51] involves estimating the distance (also called *range*) of a receiver from a potential source based on the path loss incurred by a signal using an approximation of the wireless channel. For example, in the Hata-Urban [53] channel model, the distance from a transmitter, d is related to

the path-loss, PL_{out} as,

$$\mathrm{PL}_{out}{=}A{+}B\log(d){+}C \implies d{=}10^{\frac{\mathrm{PL}_{out}-A-C}{B}} \tag{3.1}$$

$$\text{where, } A{=}69.55 + 26.16\log(f_c) - 13.82\log(h_b) - 3.2(\log(11.75h_m))^2 - 4.97$$

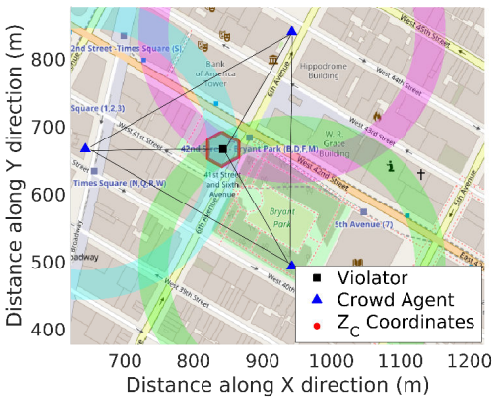
$$B{=}44.9{-}6.55\log(h_b) \text{ and } C{=}0 \quad \text{(Large metropolitan areas)}$$

$$\mathrm{PL}_b \text{ [dBm]}{=}P_t \text{ [dBm]} - \text{SNR [dB]} - P_N \text{ [dBm]}$$

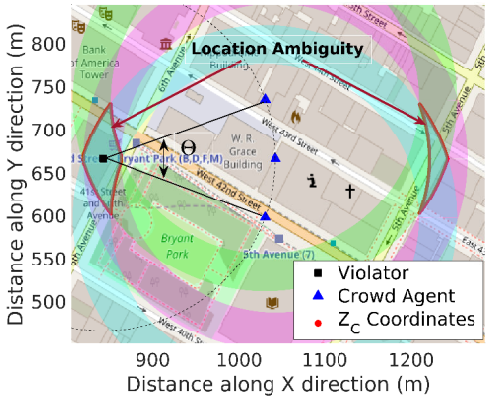
P_t is the transmit power and SNR is the received signal to noise ratio at the agent. P_N denotes the average noise power in absence of any signal is assumed to be -96 dBm.

In (3.1), uncertainty arise from the assumption about P_t, measurement noise in estimating the SNR and approximation of the channel model. These errors are collectively modeled as a random variable $X \sim \mathcal{N}(\mu, \sigma^2)$, with $\mu{=}0$ and $\sigma^2{=}2$ dB. This noise model leads to two limits, [SNR$\pm(X{=}x)$] dB that translates to two range values using (3.1): d_{outer} and d_{inner}, resulting in annular regions (instead of circles) of thickness $d = (d_{outer} - d_{inner})$ for each enforcer. Thus, geometric trilateration using these annular regions provides an *estimate* of the location of the violator. The overlapping area of the annular regions creates a *convex polygon* containing the violator and its area is a measure of accuracy of localization. Figure 3.2a shows an ideal scenario where the location of the violator is estimated by using the measurements reported by three closest (highest SNR) members of the crowd. It has a very low GDOP because the crowd agents are uniformly distributed on all sides of the violator providing an accurate estimate of the target. While, figure 3.2b shows such a scenario where the agents are located within a certain angle of the violation. This produces multiple convex polygons because of GDOP. This is precisely the drawback of any crowdsourced paradigm. It is to be noted that the GDOP can only be eliminated if there is a viable way of positioning the agents, which is not possible in a purely crowdsourced enforcement paradigm. The GDOP is used as the guiding metric to derive the cardinality of a target.

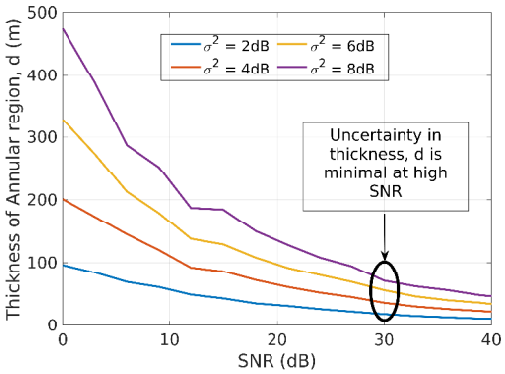
Accuracy and GDOP: Intuitively, it is desirable to choose crowd agents that are operating at high SNR (closer to target). The area of the convex polygon is a function of SNR and the noise model given by (3.1), which defines the thickness of annular regions. Figure 3.2c shows that higher the SNR, lower is the median width of the annular region, d and consequently lower is the uncertainty in the location of the targets Hence, it is desirable to position the agents as close to the target



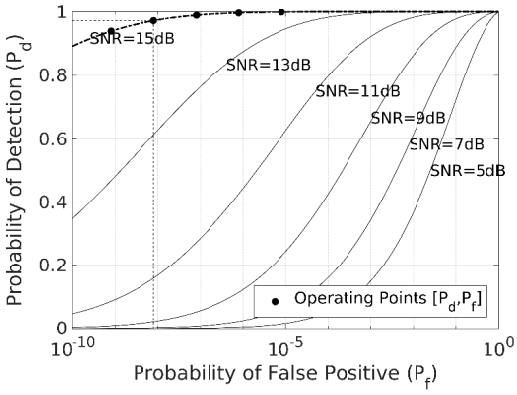
(a) Ideal arrangement of agents leads to low GDOP



(b) High GDOP using crowdsourced measurements.



(c) Thickness of the annulus, d decreases with SNR



(d) Receiver operating Characteristic of a detector

Figure 3.2: In trilateration, the location of a target is given by the intersection of the annular regions. The thickness of the annular regions reduces with SNR based on (3.1). However, GDOP depends on the relative positioning of the agents as well. Also, an ROC curve dictates the performance of any detector and assimilating results from various agents leads to higher accuracy.

as possible. Therefore, one of the objectives is to deploy mobile agents to surround the initial crowdsourced estimate of the convex polygon in order to minimize GDOP. The number of agents required for this is also the cardinality of the target.

ROC of a signal detector: Signal detection and parameter estimation is a rich and well-studied area. The Receiver Operating Characteristic (ROC) curve (figure 3.2d shows the ROC curve for Neyman-Pearson detector [54]) is universally used to define the performance of a classifier or an estimator. In this work, the agents rely on the ROC curve to choose an operating point based on the SNR of the received signal similar to [1, 55]. Directing crowd agents to *always* operate at a desirable operating point can be cost prohibitive but a group of homogeneous autonomous agents

can be mandated to yield a high detection result, especially since the SNR is also very high at the vertices of the polygon as mentioned above. Therefore, our work is independent of any specific detection scheme and simply ensures that the agents are always delivering the highest possible accuracy, *e.g.*, aggregating the operating points chosen by the agents on the 15 dB curve in figure 3.2d will always yield the *best* result for detection.

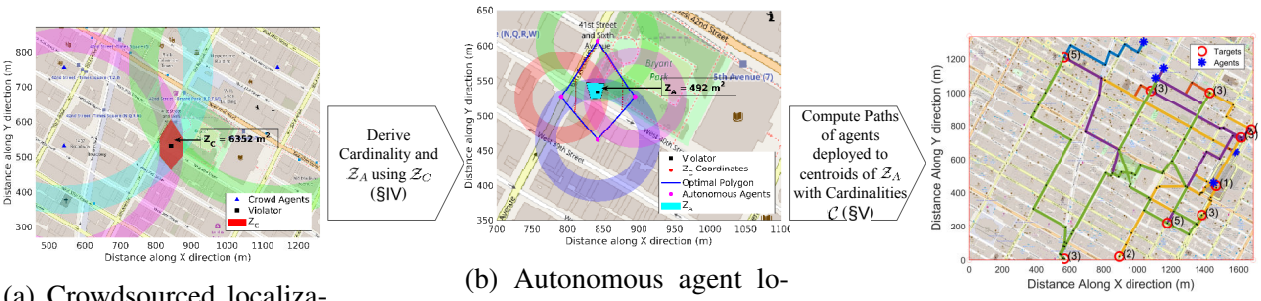
3.3 Multi-Agent Planning with Cardinality

In the context of the MPC problem, let the set of m targets be denoted by $T = \{T_1, ..., T_m\}$ located at coordinates specified by the set $\mathbb{t} = \{t_1, ..., t_m\}$. Let the crowdsourced estimates of the locations of the set of m targets be $\mathbb{t}_C = \{t_{C,1}, ..., t_{C,m}\}$. Let the set of n autonomous agents be denoted by $A = \{A_1, ..., A_n\}$, with coordinates $\mathbb{a} = \{a_1, ..., a_n\}$. Let the set of m convex polygons for targets T, as determined by the crowdsourced and autonomous agent based localization, be denoted by $\mathcal{Z}_C = \{\mathcal{Z}_{C,1}, ..., \mathcal{Z}_{C,m}\}$ and $\mathcal{Z}_A = \{\mathcal{Z}_{A,1}, ..., \mathcal{Z}_{A,m}\}$ respectively.

Algorithm 1: MPC Algorithm

```
1  Function MPC(Map, a, Z_C)
2      γ_th = 10m^2; t_C = getCentroids(Z_C);
3      while True do
4          [C,Z_A] = findCardinality(Map, t_C, Z_C);
5          t = getCentroids(Z_A);
6          P = findAgentSchedule(Map, a, t, C);
           // Take measurements & evaluate actual t
7          if Z_A < γ_th then break; else Z_C = Z_A; t_C = t;
8      end
9      return P;
10 end
```

Algorithm 1 shows the steps in solving the MPC problem. It is initialized with the starting locations of the autonomous agents, $\mathbb{a}$, and $\mathcal{Z}_C$, followed by updating the target location set, $\mathbb{t}_C$ with the geometric centroids of $\mathcal{Z}_C$ in line 2. The accuracy of localization is defined by the area of $\mathcal{Z}_A$, and the target value is chosen to be $10m^2$. Although a higher accuracy can be achieved in theoretical sense, in practice, the accuracy is limited by the feasibility of deploying agents to the vertices of $\mathcal{Z}_A$. In other words, if the vertices of $\mathcal{Z}_A$ fall over (or inside) any structure, then it requires additional resources to further improve the accuracy of localization. Algorithm 1 terminates under such infeasible conditions but provides the maximum accuracy in outdoor setting. This al-



(a) Crowdsourced localization and $\mathcal{Z}_C$. Colored disks portray the annular regions of the crowd agents.

(b) Autonomous agent localization and $\mathcal{Z}_A$. Colored disks portray annular regions of autonomous agents.

(c) Example routing in NYC. Cost metric = $2.374 km/km$.

Figure 3.3: (a), (b) The autonomous agent based localization achieves a 92.25% reduction in the area of the convex polygon containing the violation. (c) Example routes and the cost metric (details in Section 5) for 5 agents and 10 targets in New York City.

gorithm has two key steps: A) Derivation of Cardinality, and B) Scheduling of autonomous agents. Step-A calculates the cardinality ($\mathcal{C}$) and the convex polygons ($\mathcal{Z}_A$) in $findCardinality$ (line 4), using the estimates from the crowdsourced phase, $\mathcal{Z}_C$ and $\mathbb{t}_C$. Figure 3.3a shows an example of a target with crowdsourced detection and localization. Figure 3.3b shows the cardinality for that target, the optimal orientation of the autonomous agents and the improvement in the accuracy of localization over crowdsourced localization by employing Algorithm 2 in Section 3.4. Then the target location set, $\mathbb{t}$ is updated with the geometric centroids of $\mathcal{Z}_A$ in line 5.

Step-B uses the locations, $\mathbb{t}$ and cardinality $\mathcal{C}$ from Step-A to determine the paths, $\mathcal{P}$ for each agent by calling the subroutine $findAgentSchedule$ in line 6, outlined in Section 4.2. Figure 5.1a shows an example schedule in a major city in the US with a small set of agents and targets. Two properties are evident from the schedule: 1) Paths for different agents overlap but the same agent never visits a target more than once and 2) The agents can start and finish at any target as long as it minimizes the length of the longest traversing agent. These two properties collectively lead to Algorithm 3 in Section 3.5.1 that iteratively prunes the paths as the cardinality for the targets are fulfilled, terminating with the quickest possible schedule for all agents.

Then the agents are deployed to each target and measurements are taken to validate the calculated $\mathcal{Z}_A$ and $[P_d, P_f]$ (true and false positives). If $\mathcal{Z}_A$ is greater than the threshold γ_{th}, then steps A and B are repeated by setting $\mathcal{Z}_C = \mathcal{Z}_A$ and $\mathbb{t}_C = \mathbb{t}$, until $\mathcal{Z}_A$ is less than the threshold γ_{th}. In other words at each round of enforcement we use the estimated convex polygon, $\mathcal{Z}_A$ and locations $\mathbb{t}$ as the inputs for the next round of enforcement. This procedure ensures that each violation is localized with target accuracy threshold with no ambiguity. The output of Algorithm 1

are the paths of all the agents ($\mathcal{P}$).

In practice, once an agent visits a target, it performs a single round of patrolling by visiting the vertices of the optimal polygon circumscribing $\mathcal{Z}_C$ as shown in figure 3.3b. At each vertex the agent collects measurements (SNR) and estimates the annular region based on the noise model mentioned in Section 3.2. Since, each target is visited by a number of agents equal to its cardinality the average of all the measurements minimizes the error in $\mathcal{Z}_A$ and $[P_d, P_f]$. This aggregation is independent of the MPC algorithm and can be designed to achieve other objectives like trust and fault tolerance. The cost of a single round of patrolling by the agents is negligible, since the area of $\mathcal{Z}_A$ are very small compared to the cost of scheduling the agents to the the target locations. Hence, this cost is ignored from the overall cost of scheduling.

3.4 Step-A: Determination of Cardinality

Algorithm 2: Algorithm to determine Cardinality

Function *findCardinality(Map, $\mathbb{t}_C$, $\mathcal{Z}_C$)*
  for *j=1:size($\mathcal{Z}_C$)* **do**
    numEdges = $\mathcal{Z}_C$.Edges;
    `// Find optimal circumscribing polygon`
    for *i=3:numEdges* **do**
      MinPoly[i] = *findMinPoly*($\mathcal{Z}_C$,i);
      $\bar{\mathcal{Z}}_A[i]$ = *findConvexPoly*(*MinPoly*[i], $\mathbb{t}_C$[j]);
      $\overline{Cost}_{Loc}$[i]=$\frac{\bar{\mathcal{Z}}_A[i]}{\mathcal{Z}_C} + \lambda i$;
    end
    [$Cost_{Loc}$[j], $\mathcal{C}$[j]] = *min*($\overline{Cost}_{Loc}$);
    $\mathcal{Z}_A[j] = \bar{\mathcal{Z}}_A[\mathcal{C}[j]]$; `// Convex Polygons`
  end
  return $\mathcal{C}$, $\mathcal{Z}_A$;
end

Definition 1 (Cost of Localization) *The Cost of Localization for target $T_j \in T$, given $\mathcal{Z}_{C,j}$, i agents deployed to target T_j, and the convex polygon $\mathcal{Z}^i_{A,j}$ is defined as,*

$$Cost\ of\ Localization = \frac{\mathcal{Z}^i_{A,j}}{\mathcal{Z}_{C,j}} + \lambda i \tag{3.2}$$

where $\mathcal{Z}^i_{A,j}$ denotes the convex polygon with i agents on the vertices of the smallest polygon circumscribing $\mathcal{Z}_{C,j}$ and $\mathcal{Z}^i_{A,j}/\mathcal{Z}_{C,j}$ denotes the improvement in the accuracy of localization ($\mathcal{Z}^i_{A,j} \ll$

$\mathcal{Z}_{C,j}$) over crowdsourcing after deploying i agents. The trade-off parameter, λ is a non-negative value that trades off the localization accuracy with the cost of deploying more agents.

Definition 2 (Cardinality) *The Cardinality of a target $T_j \in T$, denoted by C_j is the total number of unique agents A_i, that are required to visit T_j. For each T_j $\exists$ $C_j \geq 1$.* The set of cardinality for the m targets is denoted by $\mathcal{C} = \{C_1, ..., C_m\}$.

Thus, the desired cardinality, C_j for each target, $T_j \in T$, is the number of unique agents (i) for which the Cost of Localization for target T_j is minimum.

$$C_j = \arg\min_i (Cost\ of\ Localization) \tag{3.3}$$

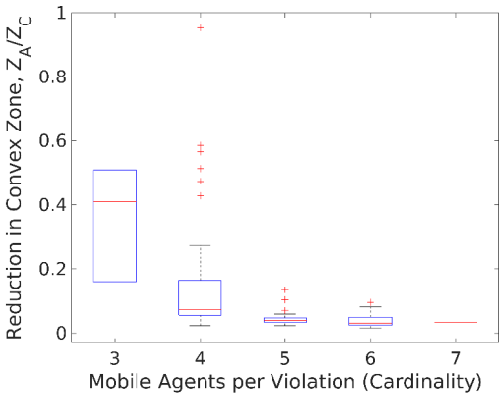
3.4.1 Algorithm to determine Cardinality

The key idea here is to find an optimal polygon for each target T_j that circumscribes the convex polygon, $\mathcal{Z}_{C,j}$. By deploying the autonomous agents to the vertices of this optimal polygon we can ensure that the target is localized with low GDOP and high accuracy while choosing optimum operating points on the ROC for signal detection.

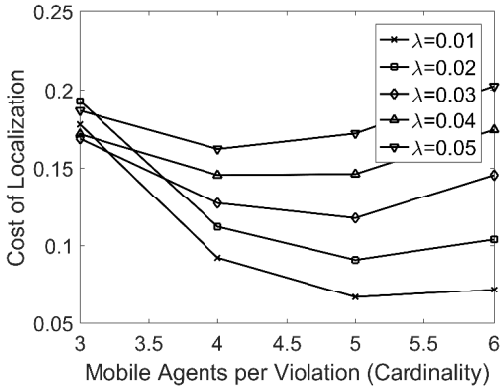
Initially the number of edges of $\mathcal{Z}_C$ is extracted in line 3. For each target T_j, the optimal polygon that circumscribes $\mathcal{Z}_{C,j}$ is determined. To do this we scan through the number of agents (i) starting from 3 agents to a maximum number that is equivalent to the number of edges of $\mathcal{Z}_{C,j}$. For each i we find the smallest polygon, $MinPoly$ with i number of sides that circumscribes $\mathcal{Z}_{C,j}$ (line 5). Line 6 calculates the convex regions $\bar{\mathcal{Z}}_A[i]$, when i agents are deployed to the vertices of $MinPoly$. This step involves, computing the annular regions and trilaterating as described in Section 3.2 assuming that the target is at $\mathbb{t}_C$[j] and the agents are at the vertices of $MinPoly$. The overlapping area of these annular regions is the convex polygon $\bar{\mathcal{Z}}_A[i]$. Next, we determine the cost of localization for each i according to Definition 1. The optimal circumscribing polygon is the polygon that gives the minimum cost of localization. Thus, line 9 gives the Cost of Localization for target T_j and the cardinality of T_j is equal to the number of sides of the optimal polygon. The above steps are repeated, to determine the cardinality, $\mathcal{C}_j$ and $\mathcal{Z}_{A,j}$ for all targets, $T_j \in T$.

3.4.2 Impact on Localization

Figure 3.4 shows the fidelity of localization of the autonomous agents oriented as described in Algorithm 2. Figure 3.4a shows the reduction in the area of the convex polygons encompassing



(a) Reduction in the convex polygon ($\mathcal{Z}_A/\mathcal{Z}_C$) with Cardinality



(b) Cost of Localization vs Cardinality.

Figure 3.4: Accuracy and cost of localization using Algorithm 2. For $\lambda = 0.01$, the optimal cardinality is 5 and the median reduction in the area of the convex polygon is 96%.

the targets over crowdsourced localization. It shows that the higher the number of agents deployed to the target, the smaller is $\mathcal{Z}_A$ and the higher is the accuracy of localization. Figure 3.4b shows the dependence of the optimal cardinality on the trade-off parameter, λ. At larger values of λ the cost of deploying more agents is higher. Hence, at larger values of λ, a lower cardinality provides a lower cost of localization.

3.4.3 Impact on Detection

The set homogeneous autonomous agents can be directed to operate at a desirable operating point on the ROC. Also, since Algorithm 2 positions the agents on the vertices of the optimal polygon (figure 3.3b) and the SNR is very high at these vertices they guarantee a near optimum detection result (F-score[56] ≈ 1) regardless of the chosen operating point, as shown in figure 3.2d. Such a guarantee cannot be made for crowd agents as mandating the crowd to operate at a fixed operating point may be cost prohibitive. Thus, it is guaranteed that autonomous agents provide better detection accuracy than crowd agents.

3.5 Step-B: Schedule Autonomous Agents

After ascertaining the improvement in localization based on the optimal cardinality, unique agents are routed to each target. The starting point of the scheduling phase is the construction of an undirected, weighted graph $G = (V, E)$ from the road network of the geographical area being enforced for spectrum policies, where the roads are mapped as edges E and the intersections as

vertices V. A Path in $G = (V, E)$ is defined as a subgraph $P = (V_s, E_s)$, if V_s is a set of k vertices of its base graph G and $E_s = \{(x_1, x_2), (x_2, x_3), ..., (x_{k-1}, x_k)\} \subseteq E$ is the set of $k-1$ edges that connect those vertices. The length of a path depends on the number of its edges and their weights, $w(v_i, v_j) = w(v_j, v_i)$. In this paper, the weight associated with each edge is the geographical distance between the corresponding vertices.

The *Cost of Scheduling* n agents to visit m targets is the time it takes for all the agents to cover m targets while satisfying the cardinality for each target. This time is determined by the agent that takes the longest time to traverse its path. Since, all agents are assumed to travel at the same speed, the time taken by an agent is determined by the sum of the edge weights of E_s. Finding the costliest path, $P = (Vs, Es)$ is the central goal of this work. For practical purposes, $m \geq n$.

Definition 3 (Cost of Scheduling) *Given the path P_i of an agent a_i of length l_i, the Cost of Scheduling is the length of the path of the longest travelling agent.*

$$Cost\ of\ Scheduling = max_{\forall i} c(P_i) = max_{\forall i} l_i$$

Where, the cost $c(.)$ of a path of k vertices (targets) is the sum of its edge weights,

$$c(P) = \sum_{i=1}^{k} w(x_i, x_{i+1})$$

Definition 4 (Uniqueness) *Uniqueness is the necessary condition that requires distinct agents A_i to visit each target T_j in order to fulfill its cardinality C_j).*

For example, if $C_j = 2$, *Uniqueness* guarantees that even if agent A_i traverses multiple times through target T_j, it still needs another agent, other than A_i to visit T_j to fulfill the cardinality of 2. In practice, unique agents provide additional layers of information that can be assimilated for higher accuracy [1].

Definition 5 (Schedule) *Given a set of m targets $\{T_1, T_2, ..., T_m\}$ at $t_1, t_2, ..., t_m$, where $t_j \in V$, a set of n agents $\{A_1, A_2, ..., A_n\}$ at $a_1, a_2, ..., a_n$, where $a_i \in V$, $m \geq n$, and $max_{\forall j}(C_j) < n$, the Schedule is to find the paths $P_i, \forall A_i \in A$ to visit m targets in the shortest possible time with C_j unique agents visiting T_j, $\forall T_j \in T$.*

The solution is the set of paths, $\mathcal{P} = \{P_1, ..., P_n\}$ such that the the cost of scheduling (according to Definition 3) is minimum, while ensuring the number of unique agents visiting each

target T_j is exactly equal to $C_j, \forall T_j \in T$ (Definition 2 and 4). The targets can be visited at any time during their traversal without any constraint of waiting time or synchronization, until the cardinality is satisfied for each target.

3.5.1 Algorithm for the Schedule

Algorithm 3: Path Pruning Algorithm

Function *findAgentSchedule(Map,* $\mathbb{a}$*,* $\mathbb{t}$*,* $\mathcal{C}$*)*
  targets_assigned=[$\mathbb{t}$;...;$\mathbb{t}$];$\mathcal{X}$=[n,...,n]; `// Visits Count`
  $\bar{G}$=*findMissionGraphs*(*Map*,$\mathbb{a}$, $\mathbb{t}$);
  `// Order for all agents to cover all targets`
  [$\mathcal{P}$,costs] = TSP($\bar{G}$, $\mathbb{a}$, $\mathbb{t}$);
  `// Compute shortest path to find Schedule`
  while $\mathcal{X} \neq \mathcal{C}$ **do**
    i=0; k=getMax(costs);
    `// Prune redundant edges`
    while *True* **do**
      l=$\mathcal{P}$[k][end-i];
      if $\mathcal{X}$*[l]* $>$ $\mathcal{C}$*[l]* **then**
        targets_assigned[k][end-i]=[]; break;
      end
      i=i+1;
    end
    `// Reevaluate TSP for costliest agent`
    $\bar{G}$[k]=graph(Dijkstra([$\mathbb{t}$,$\mathbb{a}(i)$],*G*[k]));
    [$\mathcal{P}$[k], costs[k]] = TSP($\bar{G}$[k], $\mathbb{a}$[k], targets_assigned[k]);
    $\mathcal{X}$[l]=$\mathcal{X}$[l]-1;
  end
  return $\mathcal{P}$, max(costs);
end
Function *findMissionGraphs(Map,* $\mathbb{a}$*,* $\mathbb{t}$*)*
  City_Graph=graph(Map);`// Extract connectivity`
  $\bar{G}$=[];`// Extract Mission Graph for each agent`
  for $i = 0$ ***to*** *size(*a*)* **do**
    DistanceMatrix=Dijkstra([$\mathbb{t}$,$\mathbb{a}(i)$],City_Graph)
    $\bar{G}$[i]=graph([$\mathbb{t}$,$\mathbb{a}(i)$],DistanceMatrix)
  end
  return $\bar{G}$;
end

The algorithm is initialized with the locations of the agents, $\mathbb{a}$ and the targets, $\mathbb{t}$ with corresponding Cardinality $\mathcal{C}$, projected on to a graph $G = (V, E)$ (where V is the vertex set and E is the edge set), extracted from open source map engines like OpenStreetMap [57]. For one round of

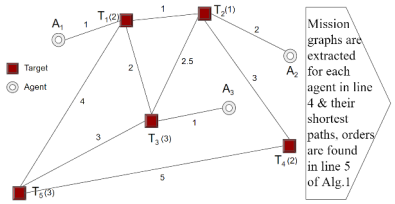
enforcement activity, the locations of the targets are assumed to be constant while the agents follow a schedule to visit the targets. Line 2 in Algorithm 3 initializes these steps. It is assumed that the Dispatch, where the algorithm is executed has prior information about the initial conditions.

The goal of finding the shortest path between the points in a graph is accomplished by creating a *Mission-Graph* for every agent in $\mathbb{a}$ as shown in line 3 and the corresponding subroutine in lines 20 – 28. The Mission-Graph is defined as a complete graph $\bar{G}_i = \bar{V}_i, \bar{E}_i$, where, $\bar{V}_i = T \cup A_i$. The edge weight, $\bar{w}(p, q)$ is the length of the shortest route between nodes $p, q \in \bar{V}$, computed using Dijkstra's shortest path algorithm in lines 24 and 25. In other words, the Mission-Graph provides the best geographical route for each agent A_i to reach every target T_j and also the shortest route between any two targets in $\bar{V}$. Given, the shortest paths in the *Mission-Graph*, Line 4 calculates the schedule (order) for each agent to cover all the targets in $\bar{V}$ in the shortest time, which is also the sum of edge-weights $\bar{w}(p, q)$ in the path P_i. This is equivalent to solving the TSP for each agent and dropping the last edge of the TSP tour to obtain the path P_i. Considering the best achievable performance to solve the TSP for each agent, we utilize the approach in [58].

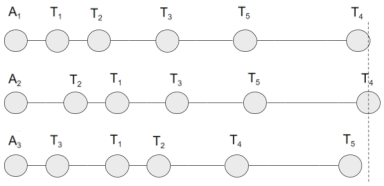
Pruning for least costly path: Given the objectives in Definitions 2 & 5, the algorithm iteratively prunes the path of the costliest agent obtained from line 4 (shortest tour on the *Mission-Graph*) to find a schedule with the minimum cost of scheduling. Central to the pruning step is the adherence to the cardinality C_j for each target. This is outlined in Lines 5–17. The pruning begins by selecting the costliest agent indexed by k (the agent that traverses the longest path) and choosing the farthest target (indexed by l) that the agent k visits, as indicated in lines 6 and 8 respectively. Line 9 checks for the condition if the cardinality of this target, $\mathcal{C}[l]$ has been fulfilled by other agents visiting it prior to the agent k. The variable $\mathcal{X}$ keeps track of the number of visits for each target, which is initialized in line 2 with the maximum number of visits possible for each agent, n ($max_{\forall j}(Cj) < n$ in Definition 5). It is decremented by one in line 16 every time a target is removed and the path is pruned to minimize the cost of scheduling. The intuition behind this approach is that by removing this redundant node (cardinality already fulfilled) from $\mathcal{P}[k]$, it produces a local minima for the overall time taken among all agents. The condition in line 9 is checked for each target in $\mathcal{P}[k]$ and after all the redundant paths are removed, the shortest route among the remaining targets in $\mathcal{P}[k]$ is computed again using Dijkstra's algorithm, followed by finding the shortest tour by solving the TSP [58] in lines 14 and 15 respectively and the visits count variable $\mathcal{X}$ is decremented. The reason for recomputing Dijkstra's algorithm and the shortest tour in lines 14 and 15 is to ensure that once a node is removed from the current TSP tour, the weight of the new edge between the

nodes immediately prior and after the one removed *may not* be equal to the sum of the two edges prior to the removal. In other words, if $a \rightarrow b \rightarrow c$ is a TSP tour and edge b is removed in line 10 then $w(a,c) \neq [w(a,b) + w(b,c)]$. Although the inequality strictly depend on the graph G (city map), it cannot be ascertained a priori and hence line 14 and 15 ensures that the final schedule is always has the minimum cost. This process (lines 5-17) is repeated until the cardinality is fulfilled for all targets in $\mathbb{t}$ (line 5) and the last calculated shortest tour given by line 15 is the final schedule for the agents, returned as $\mathcal{P}$ along with the final cost of scheduling in line 18.

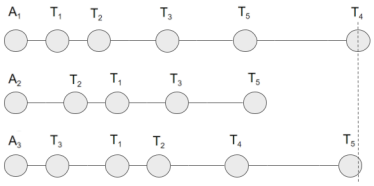
Example Illustration of Algorithm 3: Figure 3.5 shows an example of the iterative evolution of Algorithm 3. Figure 3.5a shows a simple city graph, G with edges representing the roads along with 5 targets and 3 agents located at the intersection of these roads. The cardinality of the targets $T_1 - T_5$ is $\{2, 1, 3, 2, 3\}$ respectively. After computing the shortest tour in the *Mission-Graph* for all agents in lines 3 and 4, the costs (length of the lines) and the resultant paths (order) for each agent are indicated in Figure 3.5b. In the first iteration, agent A_2 travels the longest to cover all the targets and is identified as the costliest agent. The farthest target in A_2's path is T_4. As T_4 has a cardinality of 2, requiring only 2 of the 3 agents to visit, it is considered to have a redundant visit in A_2's path. In other words, T_4 can be visited in shorter time by the other two agents and fulfill the cardinality of 2. Hence, removing this redundant and costly target T_4 from agent A_2's path (as per lines 9 – 11) reduces the cost of A_2's path while fulfilling the cardinality for all the targets. The new path for A_2 and its cost is determined from the new Mission-Graph (without T_4) as per line 14 and 15. Figure 3.5c shows the paths and the costs of the agents at the beginning of Iteration 2. In the second iteration, A_1 is determined to be the costliest agent and T_4 as the farthest target. However, as T_4 has a cardinality of 2 and has two agents visiting it (including A_2). So, this is not considered as a redundant visit. So, the algorithm continues to look for the farthest target in A_1's path that has redundant visits, until it detects T_2 as the farthest redundant visit, which is removed from A_1's path. Note, T_5 and T_3 in A_1's path, both require all three agents to visit as they have cardinality of 3, hence those two nodes cannot be removed from A_1's path. The updated path and its corresponding cost is shown in figure 3.5d. Similarly, in the third iteration, A_3 is the costliest agent and T_2 is the farthest redundant target in its path (removing other nodes will not meet the cardinality for those). While the removal of T_2 does not improve the cost, as T_4 can only be reached via T_2, it should be noted that removal of any targets and the corresponding edges does not increase the cost (due to the triangular rule governing Euclidean graphs [58]). In Iteration 4, shown in figure 3.5e, A_3 is still the costliest agent, and T_1 is the farthest redundant target because its cardinality



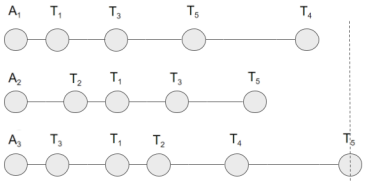
(a) City map with 3 agents, 5 targets with different cardinality and edge weights



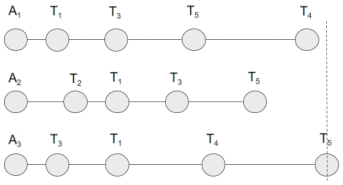
(b) Iter 1: Initial Path Estimate: A_2-costliest agent, T_4-farthest redundant target



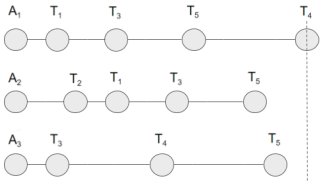
(c) Iter 2: Remove T_4 from A_2's path. A_1-costliest agent, T_2-farthest redundant target



(d) Iter 3: Remove T_2 from A_1's path. A_3-costliest agent, T_2-farthest redundant target



(e) Iter 4: Remove T_2 from A_3's path. A_3-costliest agent, T_1-farthest redundant target



(f) Iter 5: Remove T_1 from A_3's path, A_1- costliest agent with all cardinality fulfilled

Figure 3.5: Example illustration of Algorithm 3 with 3 agents and 5 targets (T_1 - T_5 with cardinality $\{2, 1, 3, 2, 3\}$ respectively). In each step the costliest agent (longest travelling agent) is identified and the redundant target (cardinality already fulfilled) is removed. After 5 iterations, Agent A_1 is the costliest. This cost is normalized with the diameter of the graph used a cost metric for evaluation in Section 5. The algorithm also provides the *best* schedule for the remaining agents.

can be met in shorter time by the other two agents. Consequently, T_1 is removed from A_3's path and after recomputing the new schedule and the path cost the final schedule is obtained as shown in in figure 3.5f. The cost of scheduling for this graph is the total cost of the A_1's path because that is the minimum time required to visit all the targets while fulfilling the cardinality.

3.5.2 Analysis of Algorithm 3

We show that the Schedule is NP-hard, hence there is no optimal solution in polynomial time.

Claim 1. *The Schedule is NP-hard.*

Proof. Consider a subproblem of the Schedule, with 1 agent having to visit all the targets, with all the targets having a cardinality of 1. This is equivalent to solving the TSP for that agent. Since, the TSP is NP-hard and it is a special case of the Schedule, it is inferred to be at least NP hard. □

3.5.3 Complexity of Algorithm 3

In absence of an optimal algorithm, Algorithm 3 yields a solution for the Schedule in polynomial time.

Lemma 1. *Algorithm 3 has complexity of $O(nm^4)$, where n is the number of agents and m is the number of targets in G.*

Proof. Since the cardinality of the targets is fixed, the number of iterations of Algorithm 3 is bounded by a fixed number. The algorithm is initiated with all agents visiting all targets ($\mathcal{X}$ in line 3) and executes until each target is visited by a number of agents equal to its cardinality. Each iteration removes one redundant visit from the costliest agent (as shown in Figure 3.5). So, for each target T_j, the algorithm executes $(n - C_j)$ times and therefore, the total iterations in Algorithm 3 for all the targets is,

$$(n - C_1) + (n - C_2) + ... + (n - C_m) = m.n - \sum_{i=1}^{m} C_i \tag{3.4}$$

Recall that, $n < m$ and $max_{\forall i \in m}(C_i) \leq n$. Hence,

$$\sum_{i=1}^{m} C_i \leq m.n \implies m.n - \sum_{i=1}^{m} C_i \geq 0 \tag{3.5}$$

The 3/2-approximation for TSP [58] used in Algorithm 3 has a complexity of $O(m^3)$. Since, the TSP is computed once in every iteration, the complexity of Algorithm 3 is,

$$=O(m.n - \sum_{i=1}^{m} \mathcal{C}_i).O(m^3) \qquad = O((m.n - \sum_{i=1}^{m} \mathcal{C}_i).m^3)$$

$$=O(n.m^4 - m^3.\sum_{i=1}^{m} \mathcal{C}_i) = \qquad O(n.m^4) \text{ From (3.5)}$$

Hence, Algorithm 3 has a complexity of $O(nm^4)$. □

Note: The 2-approximate solution of TSP based on the Minimum Spanning Tree (MST) of the corresponding graph [59], has a complexity of $O(m.log(m))$. Using such an implementation in Algorithm 3, the complexity can be improved to $O(n.m^2log(m))$. Using the MST has the added advantage of solving the Schedule for non-metric graphs, such as in the presence of traffic the costs

Table 3.1: Notations used in Section 3.5.4

Notation	Interpretation
i	Agent i, $i \in \{1, 2, ..., n\}$
t_k	Target j, $j \in \{1, 2, ..., m\}$
P_i	Path of Agent i returned by Algorithm 3
P_i^*	Path of Agent i returned by OPT
l_i	Cost of Agent i obtained by Algorithm 3
l_i^*	Cost of Agent i obtained by OPT
$l_i(t_k)$	Increase in cost of agent i by adding target t_k in Algorithm 3
$l_i^*(t_k)$	Increase in cost of agent i by adding target t_k in OPT
T_x^i	Set of targets visited by both P_i and P_i^*
T_y^i	Set of targets visited by P_i, but not by P_i^*
T_z^i	Set of targets visited by P_i^*, but not by P_i

of edges are no longer just a function of distance. This problem is out of scope of this work and will be investigated in future.

3.5.4 Approximation Ratio for Algorithm 3

In absence of a polynomial time, optimal solution for the Schedule, an approximation ratio is a bound, which guarantees that any solution from Algorithm 3 is always within a constant factor of the solution from an optimal algorithm. In other words, using the notations in Table 5.1, if agent p is the costliest in Algorithm 3 and agent q is the costliest in the optimal algorithm, then $l_p \leq 3.l_q^*$ is provably correct. Let, OPT be the optimal algorithm for the Schedule problem that returns the paths $P_i^*, \forall i \in A$, with minimum cost among all the possible paths that fulfills the cardinality of the targets. It is to be noted, that we do not make assertions on the design of OPT except to acknowledge that a Minimum Spanning Tree (MST) can be constructed from the targets in any optimal path $P_i^*, \forall i \in A$, similar to deriving a solution of the TSP problem (used in Algorithm 3) from a corresponding MST. Also, $P_i, \forall i \in A$, in Algorithm 3 (computed in lines 6–21) can be obtained using the round-trip MST of the Mission-Graphs ($\bar{G}_i$) instead of the 3/2-approximate TSP approach [58]. Under such implementation, we observe that if $T_y^i = 0$, i.e. targets in $P_i \subseteq$ targets in P_i^*, then by the construction of MST [60] we observe Property 1.

Property 1. *If $T_y^i = 0$, then l_i is no worse than twice the optimal cost l_i^*. i.e, $l_i \leq 2.l_i^*$.*

Furthermore, the following properties can be observed based on the design of Algorithm 3

and the definition of OPT.

Property 2. *Since, Algorithm 3 and OPT both return the costliest paths among all the agents (say l_p and l_q^*), the paths travelled by any other agent, must not be costlier than l_p or l_q^*. Thus, for any agent $i \in A$ we have, $l_i \leq l_p$ for Algorithm 3 and $l_i^* \leq l_q^*$ for OPT.*

Property 3. *In Algorithm 3 and OPT, all targets must be visited by the same number of agents (Definition 2 in §4.2).*

Property 4. *If a target t_k is removed from an agent i's path, it must have been the costliest path at some prior iteration of the algorithm (line 8–15). So, if agent p is the costliest agent at the end of the algorithm, the increase in agent i for visiting t_k must be such that $l_i + l_i(t_k) \geq l_p$.*

Property 5. *From Table 5.1, we can express the costs l_i and l_i^* of agent i as,*

$$l_i = l_i(T_x^i) + l_i(T_y^i)$$
$$l_i^* = l_i^*(T_x^i) + l_i^*(T_z^i)$$

Theorem 1. *Algorithm 3 is 3-approximation for the Scheduling Problem.*

Proof Overview: Let the costliest paths returned by Algorithm 3 and OPT be l_p and l_q^* respectively. Our goal is to find a relationship between these two quantities, by first establishing an inequality between the costs of the same agent in Algorithm 3 and OPT, and then using the inequality and the properties to relate the costs of different agents in the two algorithms. This result is used to relate the costs of agents which have non-overlapping targets in the paths obtained from Algorithm 3 and OPT. We consider two cases: 1) The targets in $P_p \subseteq$ the targets in P_p^* and 2) The targets in $P_p \nsubseteq$ the targets in P_p^*.

The complete proof of the approximation ratio is detailed in Appendix A.

3.6 3D Localization and Detection

In general, the targets can be located in a 3-dimensional space. For infractions that occur at ground level or low elevations, Unmanned Ground Vehicles (UGVs) are sufficient for desired accuracy of localization and detection. However, for targets in higher elevations (e.g., drones) or in high-rise buildings, Unmanned Aerial Vehicles (UAVs) are a better choice. UAV systems have

broader perception [61], better tracking, and a higher degree of flexibility in terms of mobility compared to UGVs, but may be constrained in terms of battery life, and radio resources. On the other hand, UGVs are more capable of patrolling larger areas, carry higher payload and radio equipment and can act as a mother-ship for the UAVs. Coordinated as a team, UGVs and UAVs can deliver exponentially more effective operations than as separate, non-integrated systems. The accuracy of localization of targets in 3D space can be improved by deploying UAVs as it has low 3D-GDOP and the volume of the estimated convex polyhedron containing the target is reduced[7]. Moreover, since UAVs are able to navigate closer to the infractions, the SNR of the received signal is very high, leading to high accuracy of detection.

3.6.1 Outdoor-to-Indoor path loss

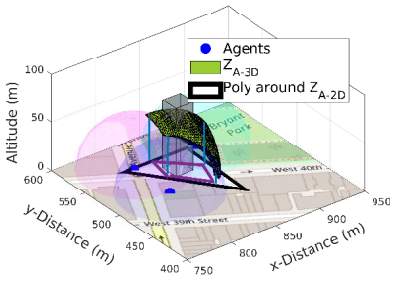
The trilateration based localization for targets located in 3D (typically within buildings) is dictated by the Outdoor-to-Indoor (O2I) building penetration loss is modelled [64],

$$\begin{aligned} \mathrm{PL} &= \mathrm{PL}_{out} + \mathrm{PL}_{\mathrm{in}} + \mathrm{PL}_{\mathrm{tw}} + \mathcal{N}\left(0, \sigma_P^2\right) \\ \mathrm{PL}_{out} &= A + B\log(d) + C \quad \text{and} \quad \mathrm{PL}_{\mathrm{in}} = 0.5 d_{\mathrm{2D-in}} \\ \mathrm{PL}_{tw} &= \mathrm{PL}_{npi} - 10\log_{10}\sum_{i=1}^{N}\left(p_i \times 10^{-0.1 L_{material\_i}}\right) \end{aligned} \tag{3.6}$$

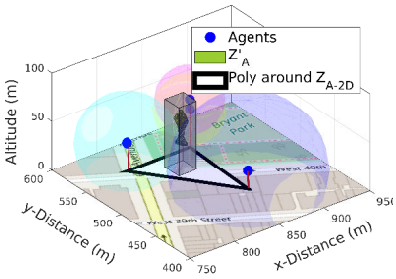
where PL [dBm]$=P_t$ [dBm] $-$ SNR [dB] $- P_N$ [dBm]. PL_{out} is the outdoor path loss as in (3.1). The distance to the target, d is a function of PL_{out}. PL_{in} is the inside loss dependent on the depth into the building, and d_{2D-in} is a single, link-specific, uniformly distributed variable between 0 and 25 m for Urban environments [64]. PL_{tw} is the building penetration loss through the external wall with standard deviation σ_P. PL_{npi} is an additional loss added to the external wall loss to account for non-perpendicular incidence. $L_{\text{material}\_i} = a_{\text{material}\_i} + f_c \cdot b_{\text{material}\_i}$ is the penetration loss of material i and p_i is proportion of i^{th} material, where $\sum_{i=1}^{N} p_i = 1$ and N is the number of materials. The value of L_{material_i} for a building is determined by the construction material (from OpenStreetMap[57]) and its penetration loss [64]. From (3.6), the distance d from each Agent to the Target is,

$$d = 10^{\left(\frac{\mathrm{PL} - \mathrm{PL}_{in} - \mathrm{PL}_{tw} - A - C}{B}\right)} \tag{3.7}$$

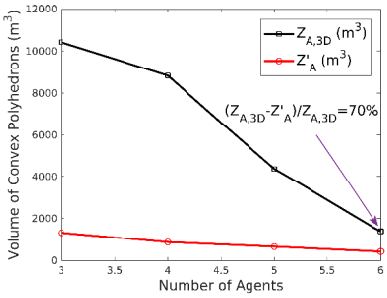
[7]Regulations by the FAA [62], impose restrictions on path planning of UAVs above 400 ft and above groups of people. These regulations may be wavered [63], and are subject to constant revision.



(a) UGV localization and $\mathcal{Z}_{A,3D}$. The 2D projection of $\mathcal{Z}_{A,3D}$ is $\mathcal{Z}_{A,2D}$. Polygon around $\mathcal{Z}_{A,2D}$ is shown



(b) UAVs are deployed to the vertices ofpolygon around $\mathcal{Z}_{A,2D}$ and at height h_{UAV}.



(c) Comparison of localization accuracy for UGVs and UAVs

Figure 3.6: (a) Localization of a target at a high elevation from UGV agents. The 2D projection of $\mathcal{Z}_{A,3D}$ is $\mathcal{Z}_{A,2D}$. Polygon around $\mathcal{Z}_{A,2D}$ is shown, (b) The UAV agents are scheduled to the vertices of a polygon around $\mathcal{Z}_{A,2D}$ and at an elevation h_{UAV}, (c) The volume of the estimated polyhedron is much less for UAVs, showing $\approx 70\%$ improvement.

This estimated range range along with the noise model (in $\mathcal{S}$3.2) produces the 3D annular region as shown in figure 3.6a. In presence of multiple Agents, the intersection of these 3D regions result in a convex polyhedron, that contain the target.

3.6.2 3D Trilateration

In the context of the 3D localization problem, let the UGV estimates of the locations of the set of m targets be $\mathbb{t}_{A,3D}$. Let the set of m convex polyhedrons for targets T, as determined by the UGV and UAV based localization, be $\mathcal{Z}_{A,3D}$ and $\mathcal{Z}'_A$ respectively. $\mathcal{Z}_{A,3D}$ and $\mathbb{t}_{A,3D}$ is determined by Algorithm 2 for targets distributed in 3D. $\mathbb{t}_{A,3D}$ are the centroids of $\mathcal{Z}_{A,3D}$. Algorithm 4 computes the 3D localization. $\mathcal{Z}_{A,2D}$ and $\mathbb{t}_{A,2D}$ are the 2D projection of $\mathcal{Z}_{A,3D}$ and $\mathbb{t}_{A,3D}$ on to the city map, as shown in line 2 figure 3.6a.

Algorithm 4: Algorithm to perform 3D Localization

1 **Function** *do3DLocalization(Map,* $\mathcal{Z}_{A,3D}$, $\mathbb{t}_{A,3D}$, $\mathbb{C}$*)*
2 $\quad \mathbb{t}_{A,2D} = \mathbb{t}_{A,3D}[:,1{:}2]$; $\mathcal{Z}_{A,2D} = \mathcal{Z}_{A,3D}[:,1{:}2]$; $\sigma_h = 20m$
3 $\quad [x_{UAV}, y_{UAV}] =$ *findMinPoly*$(\mathcal{Z}_{A,2D}, \mathcal{C})$;
4 $\quad h_{UAV} \sim \mathcal{N}(\mathbb{t}_{A,3D}[:,3], \sigma_h)$;
5 $\quad \mathcal{Z}'_A =$ *findConvexPoly*$([x_{UAV}, y_{UAV}, h_{UAV}], \mathbb{t}_{A,3D})$;
6 $\quad$ **return** $\mathcal{Z}'_A$;
7 **end**

We deploy the same number of UAVs to each target as UGVs (equal to the cardinality, $\mathcal{C}$[j] of each target, T_j). The x and y coordinates of the UAVs are the vertices of the minimum polygon

(with a number of sides equal to the cardinality of each target) circumscribing each convex polyhedron in the set, $\mathcal{Z}_{A,2D}$ as in line 3. The heights of each UAV is sampled from the normal distribution, $\mathcal{N}(\mathbb{t}_{A,3D}[:,3],\sigma_h)$. Line 5 calculates the 3D convex regions $\mathcal{Z}'_A$, when i UAVs are deployed to $[x_{UAV}, y_{UAV}, h_{UAV}]$. This step involves, computing the 3D annular regions and trilaterating as described in $\mathcal{S}$3.6.1 assuming that the target is at $\mathbb{t}_{A,3D}$ and the agents are at $[x_{UAV}, y_{UAV}, h_{UAV}]$. The overlapping volume of these 3D annular regions is the convex polyhedron $\mathcal{Z}'_A$. Circumscribing the estimates $\mathbb{t}_{A,3D}$ at different elevations improves the accuracy of localization (reduces the volume of $\mathcal{Z}'_A$) and the likelihood of the target being located in $\mathcal{Z}'_A$. Each agent has a complete topological view of the geographical area of interest (city). The exact location of each building on the city map is extracted from [57] and the dimension of each building is obtained from city databases such as [65]. For targets approximately located in buildings (as estimated from Z'_A), the 3D convex polyhedron containing the target can be further reduced by considering its overlap with the building dimensions, thereby increasing the accuracy of localization. Figure 3.6c shows the performance of UGV and UAV localization for 1000 targets scattered at various elevations (less than 100m) and achieve $\approx$70% improvement in localization accuracy.

CHAPTER 4

`SenseChain`: Blockchain based Distributed Fusion System

Autonomous enforcement of spectrum policies requires the distributed fusion of sensing results from a set of trust-less spatially scattered sensors to detect and localize spectrum violations with the highest possible accuracy. In `SenseChain`, we leverage the distributed consensus mechanism employed in Blockchain networks to record sensing reports and accrue the reputation of sensors, leading to a highly reliable and accurate enforcement system. Specifically, we define and analyze a detection mechanism to identify falsifying sensors using a distributed anomaly detection system and use the Blockchain to record the individual's behavior. The reputation is then based on the combination of the difficulty level of the consensus method and the degree of falsehood in the reported sensor values. The recorded sensing information and the reputation of the sensors are used to make credible inferences without relying on centralized, explicitly trusted entities.

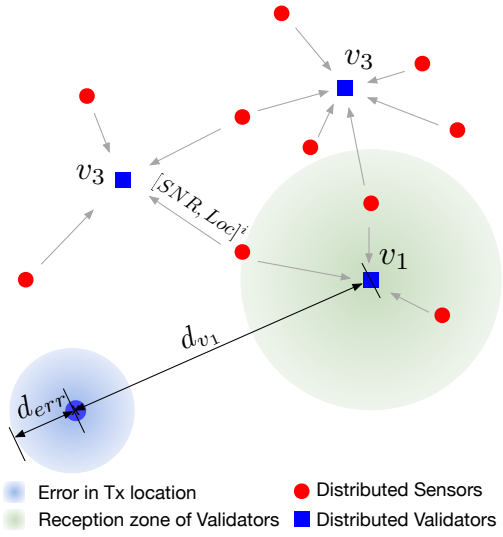
4.1 Overview of `SenseChain`

Recent interest in applying Distributed Ledger Technology (DLT) like Blockchain, beyond cryptocurrencies [66, 42] has led to creative applications that maintain the integrity of transactions while assuring provenance of information being transacted on. Smart Contracts [67] are often seen as a viable way of deploying these transactional systems on a Blockchain. While such applications benefit from continued proliferation of DLT platforms (e.g. Ethereum, Hyperledger, Hashgraph, etc), these are fundamentally restricted to the features of such platforms that limit the innovation and scope for new applications. Interestingly, the core features of DLTs like immutability and distributed validation of transactions can be found in many applications that rely on data aggregation and dissemination among untrustworthy entities. Central to this, is the definition of *Transaction* and *Consensus*. While transactions are unique entries in an electronic ledger that encapsulate exchange of valuables (money, goods, data, etc.) between multiple parties in a cryptographically secure manner, consensus is responsible for all the constituents in a network to agree on one common version of the ledger without involving globally trusted intermediaries (e.g., trusted web servers or human arbiters like lawyers etc.). We envision a connected world of Things where instead of any one universal Blockchain for a gamut of applications (Ethereum, Hyperledger, etc.), there

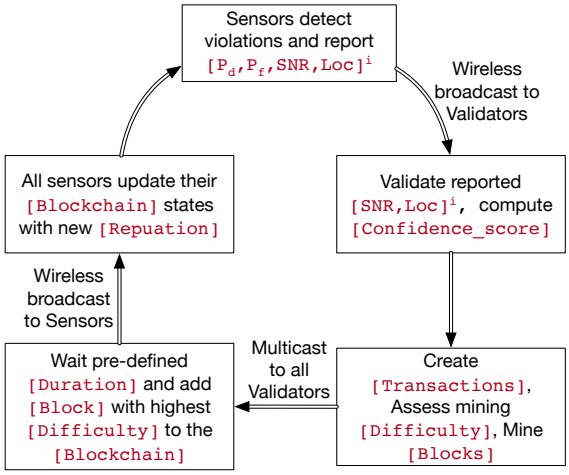
are numerous independent islands of DLTs, employing proprietary definitions of transactions and consensus algorithms that are tailored for specific applications. In practice there are no technical barriers to implement such an idea, unless it needs to scale with a global footprint. We believe that many applications need not be scaled beyond a certain geographical area or a small interconnected network like a Smart-City, Smart-Home, Micro-grid and Mobile Adhoc Network (MANET) of sensors.

`SenseChain` is such an example, where distributed sensors detecting violation of spectrum access policies (malicious or otherwise) can take advantage of Blockchain technology to record and disseminate sensing information among the distributed sensors, while benefiting from the core properties of DLT. In the post sensing phase, the sensing results may be incorrectly reported by agents with a malicious intent to disrupt the information fusion. Therefore, `SenseChain` also includes an anomaly detection system that separates the *good* actions from the *bad* and then assign a reputation metric to each sensor based on individual actions. This reputation can then be utilized in fusing the results of the sensors via a weighted function. The challenge in such reputation based systems is an implicit reliance on a separate trusted infrastructure to detect falsifying sensors and disseminate reputation metric. `SenseChain` eliminates such restrictions by assigning the task of validation to the sensors itself and requiring to compute a Proof-of-Work to include their validation in the Blockchain. Thus, the Blockchain serves as a historical ledger of sensing reports and anomalous behaviour of sensors since the genesis of the enforcement system, that is immutable and is available to all the nodes in the network.

Figure 4.1a shows a distributed spectrum enforcement scenario, where certain agents assume the role of Sensor (red circles) while others assume the role of Validator (blue squares). Each sensor broadcasts their sensing results for peer validation, which may include false reports from some of the sensors. Depending on the reception zone (based on transmit power of the sensors) each Validator is tasked with identifying false reports for different number of sensors as highlighted by the green area in figure 4.1a. The first step is to estimate the approximate location of the transmitter based on the reported sensor values. There are many such examples in the literature using a variety of methods like multilateration [68], centroid [69], clustering, etc [70]. All of these methods estimate the location of the transmitter with some degree of uncertainty that can be modeled as an error term, d_{err} in figure 4.1a, around the true location of the transmitter (which is unknown to the validators). Guided by the estimated position of the transmitter and the characteristics of d_{err}, Validators use the Log-distance path loss model to validate the reported SNR of the sensors using



(a) Distributed anomaly detection of spectrum Sensors



(b) Reputation assignment and weighted fusion using Blockchain

Figure 4.1: `SenseChain`: Distributed agents can interchange roles as Sensor or Validator. Each Validator assesses false sensor reports, assigns confidence scores and creates transactions for each peer sensor. Transaction blocks are mined with Proof-of-Work difficulty proportional to number of sensors in a block.

its own sensed SNR and distance to transmitter as a reference. This results in an annulus validation zone for each sensor within the broadcast zone. Consequently, if a sensor is outside the annulus, it is concluded as a *bad* action with high certainty, while a score is calculated proportional to the thickness of the annulus if the sensor lies inside the annulus. In the context of this work, we define this validation step as *anomaly detection*.

The anomaly detection phase is followed by recording the confidence score in the Blockchain for provenance and calculation of historical reputation. Figure 4.1b shows the protocol that also includes the Blockchain based accumulation of this reputation metric. The confidence score for each sensor along with the sensing values constitute a transaction and a block is mined by the Validator for all the sensors it validates. In `SenseChain` the role of Blockchain is two fold: 1) Provide an immutable record of sensing reports and anomalous behavior by the sensors and 2) By requiring the *Difficulty* for mining to be proportional to the number of sensors being validated, it acts as a measure of credibility of the validators as well. The mined blocks are multicast within the Validators to converge on the block that is mined with the highest difficulty as it is deemed as the most credible validation for the set of sensors included in that block. We define this as the *Most-Difficult-Chain* consensus. Once each Validator gets an updated Blockchain state, it broadcasts that to its validated sensors to update their local states as well. In that way, in the following round

if any sensor chooses to assume the role of a validator it will always have historical confidence scores that is used to calculate the most current reputation of the sensor. The design and analysis of `SenseChain` is detailed in Chapter 4.

4.2 Models and Preliminaries

System Model: Let $\mathcal{S}=\{s_1, \ldots, s_N\}$ be the set of N sensors in the area. Let $\mathcal{V}=\{v_1, \ldots, v_M\}$ be the set of M validators in the area. The target that is being sensed is denoted by $\mathcal{T}$. The target may be a primary user, a rogue source or a transmitter to be localized. The sensors and validators are mobile entities with omnidirectional antennas and have a limited broadcast range. The validators have their own overlay network to communicate among themselves. Validators and sensors are mobile crowd devices. Devices with more processing capabilities assume the role of a validator. As such, typically there are much more crowd sensors than validators (i.e., $M \ll N$).

Sensing Model: Each sensor $s_i \in S$ senses the target $\mathcal{T}$ and reports the SNR and location. The sensors may report additional parameters like the probability of detection (P_{d_i}) and probability of false alarms (P_{f_i}). The validators rely on the SNR and location for anomaly detection and to assign reputations to sensors. The reputations of sensors can be used by the validators to fuse the information of sensors for more reliable inference. The weighted aggregation of P_d and P_f as in [1], using the normalized reputations as weights would increase the credibility of detection. e.g., $P_d^{fused} = \sum_i w_i P_{d_i}$, where w_i is the reputation of sensor s_i normalized by the aggregate reputation of all the sensors. Similarly, the weighted localization as in [71], using the normalized reputations leads to more credible and accurate localization.

Blockchain Model: Each validator, v_j receives the sensing reports (referred to as 'transactions') from all the sensors within a finite range (referred to as a 'Validator Range') within a fixed duration (referred to as the 'sensing phase'). The validator detects whether the report is valid or an anomaly, evaluates and appends a confidence score to each transaction from all sensors within its range. These transactions are aggregated by the validator to create a candidate block with a certain difficulty (Proof of Work [72]). Each validator appends its candidate block to the blockchain which is then broadcasted to all the validators. The validators arrive at a consensus on the most difficult blockchain. The blockchain is used to extract the reputation of each sensor. This work does not rely on a specific type of financial incentive mechanism, any approach that rewards the validators for their effort (Difficulty) [73] and the sensors according to their reputation is valid.

Threat Model: *1) Malicious Sensors (Sensing threat):* The sensors may falsify either their reported SNR, their reported location or both. *2) Malicious Validators (Validation threat):* The validators may forge information in the block they create by falsifying the confidence scores. Since, the validators may assume the role of a sensor, whose reputation depends on the reputation-weighted fusion of reports from other sensors, the validators are intrinsically discouraged from falsifying.

Protocol: There are three key phases of activity in `SenseChain` that each entity in the system adheres to: 1) Sensing Phase: the sensors sense the target and report their findings to the validators in the vicinity, 2) Validation Phase: the validators assess the truthfulness of the reports and creates a block by aggregating the reports, and 3) Blockchain Phase: the validators broadcast mined blocks to arrive at consensuses on the Blockchain before calculating the reputation of sensors.

In the sensing phase, each sensor $s_i \in \mathcal{S}$ senses the target $\mathcal{T}$, and creates a report with its perceived signal-to-noise ratio (SNR), SNR^i and an estimate of its location, Loc^i and broadcasts the report, $[SNR, Loc]^i$ to all the validators. Additionally, the sensors may report other sensing parameters depending on the application of interest. However, for applications in wireless communications, the SNR and the location are identified as fundamental sensing parameters that all sensors must report. A sketch of the protocol steps is shown in figure 4.1b. It is to be noted that there is not explicit synchrony in the network and all nodes will converge at the same Blockchain state eventually by employing the *most-difficult-chain* rule described in subsequent sections.

4.3 Anomaly Detection

The goal of anomaly detection is to gauge the truthfulness of a sensor in a distributed manner, using only the fundamental sensing information that is reported by the sensor. In the validation phase, each validator $v_j \in \mathcal{V}$ receives the reports from all the sensors located within the 'Validator Range' (limited to a distance R around the validator). Algorithm 5 describes the anomaly detection performed by each validator.

First, the validator $v_j \in \mathcal{V}$ estimates the region most likely to contain the target $\mathcal{T}$ using the reports from the sensors as in line 4. This is done using a multilateration based localization [74] using a reputation-weighted fusion [71] of the sensing reports. The centroid of the estimated region is taken as the location estimate of the target, denoted by $Loc_{\mathcal{T}}$. The estimated location of the target may vary from the true location due to two major reasons: 1) potential falsification in the sensing

Algorithm 5: Anomaly Detection Algorithm

1 **Function** *AnomalyDetection(Map,* $[SNR, Loc]$*)*
2 $R = 100$; $d_0 = Diameter(Map)$;
3 n_{v_j} = count($[SNR, Loc]$);
4 $[Loc_{\mathcal{T}}, d_{err}]$=DistributedTargetLocalization($[SNR, Loc]$);
5 d_{v_j} = $EuclideanDistance(Loc_{\mathcal{T}}, Loc_{v_j})$;
6 **for** *i=1:*n_{v_j} **do**
7 $[d_{s_i}^{min}, d_{s_i}^{max}]$ = Estimate Annulus as in Section 4.3.1;
8 $\hat{d}_s$ = $EuclideanDistance(Loc_{\mathcal{T}}, Loc^i)$;
9 **if** $(\hat{d}_s \geq d_s^{min}$ & $\hat{d}_s \leq d_s^{max})$ & $(d_s^{max} - d_s^{min} < R)$ **then**
10 $S_{s_i} = 1 - \frac{(d_{s_i}^{max} - d_{s_i}^{min})}{d_0}$
11 **else**
12 $S_{s_i} = 0$;
13 **end**
14 **end**
15 **return** S_{s_i} for all $s_i \in \mathcal{S}$
16 **end**

reports from malicious sensors, and 2) inaccuracies in the path loss model, receiver heterogeneity and other noise sources. The effect of falsified sensing reports on the localization of the target can be mitigated by clustering the location estimates similar to [75]. To account for any errors in the estimated location, we model the error in the location estimate and the true location using the random variable d_{err}. This represents the circular region with a radius equal to d_{err} around the estimated location, that is likely to include the true location of the target. This is shown in figure 4.1a. The distance from the target to the validator v_j, denoted by d_{v_j} is estimated as the euclidean distance between the estimated location of the target, $Loc_{\mathcal{T}}$ and the location of the validator, Loc_{v_j} (line 5). Due to the uncertainty in the location of the target, the true distance from the target to the validator will be a value in the interval $[(d_{v_j} - d_{err}), (d_{v_j} + d_{err})]$.

Lines 6-14 detail the steps involved in the detection of anomalies for each sensor $s_i \in \mathcal{S}$. The core of the anomaly detection algorithm involves two steps: 1. based on the SNR reported by sensor s_i, the validator determines an *annular region* which should contain the sensor, and 2. if the sensor's reported location lies outside the estimated annulus, it is considered an anomaly.

4.3.1 Estimation of the annulus validation zone

Using the reported SNR, SNR^i the validator evaluates the received power, P_{r,s_i} experienced by the signal transmitted from the target at each sensor s_i. The average noise floor (NF) for a short range communication, similar to 802.11a/g, is approximately $-96dBm$ [76]. Thus, (P_{r,s_i}) is

given by,

$$P_{r,s_i}(dBm) = SNR^i(dB) + NF(-96dBm) \tag{4.1}$$

The validator estimates the distance to the target using the Log-distance path loss model [77],

$$PL_{s_i} = PL_{v_j} + 10\gamma \log_{10} \frac{d_{s_i}}{d_{v_j}} + \chi \tag{4.2}$$

where PL_{s_i} and PL_{v_j} is the path loss experienced by the transmission from the target at sensor s_i and validator v_j respectively, and d_{s_i} is the true distance to the sensor from the target. γ is the path loss exponent and χ is a zero-mean gaussian random variable to account for the shadowing effect. Since the path loss is equal to the difference between the transmitted and received powers, and the transmitted power (of the target) remains constant during the sensing phase, we can rewrite (4.2) in terms of the received power as,

$$-P_{r,s_i} = -P_{r,v_j} + 10\gamma \log_{10} \frac{d_{s_i}}{d_{v_j}} + \chi \tag{4.3}$$

where P_{r,v_j} is the received power from the signal transmitted from the target at the validator v_j. Note, that since the estimate of d_{v_j} is erroneous, the estimate of d_{s_i} from (4.3) will lie within a range, [$d_{s_i}^{min}$, $d_{s_i}^{max}$]. The distance from the target to each sensor s_i, denoted by $\hat{d_{s_i}}$ is estimated as the euclidean distance between the estimated location of the target, $Loc_{\mathcal{T}}$ and the reported location of the sensor, Loc^i (line 8).

The annular region estimated for sensor s_i by validator v_j is centered at the location of the target, $Loc_{\mathcal{T}}$ and defined by the inner and outer radii, $d_{s_i}^{min}$ and $d_{s_i}^{max}$ respectively. $d_{s_i}^{min}$ and $d_{s_i}^{max}$ is calculated from (4.3) as,

$$\begin{aligned} d_{s_i}^{min} &= (d_{v_j} - d_{err}) \times 10^{\left(\frac{P_{r,v_j} - P_{r,s_i} - \chi}{10\gamma}\right)} \\ d_{s_i}^{max} &= (d_{v_j} + d_{err}) \times 10^{\left(\frac{P_{r,v_j} - P_{r,s_i} - \chi}{10\gamma}\right)} \end{aligned} \tag{4.4}$$

The thickness of the annulus is given by ($d_{s_i}^{max} - d_{s_i}^{min}$). By considering the minimum and maximum values of d_{v_j} in (4.4), we account for the impact on the annulus, by the error in the estimated and true location of the target. i.e., when the target is located anywhere within the circle as shown in

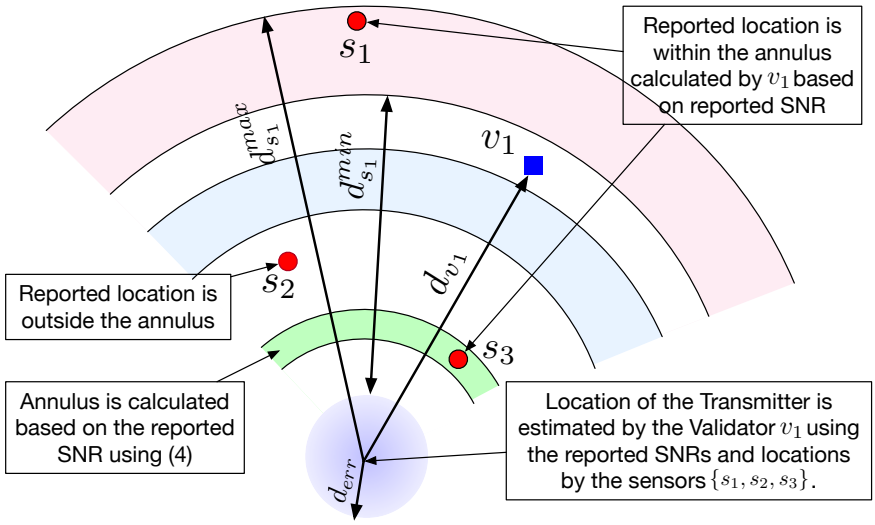


Figure 4.2: Anomaly detection: When reported sensor location is outside the annulus it is detected as an anomaly else it is associated with a confidence score to represent its truthfulness.

figure 4.1a, the distance from the target to the sensor is bounded in the interval $[d^{min}_{s_i}, d^{max}_{s_i}]$. This is because $d^{min}_{s_i}, d^{max}_{s_i}$ represent the distance to the sensor from the closest and furthest possible location of the target respectively. The annular regions estimated by a validator v_1 for several sensors is shown in figure 4.2.

4.3.2 Anomalies and confidence score

Any falsification in the reported $[SNR, Loc]^i$ can be identified in two steps:

- If the validator received a report from a sensor s_i, whose reported location, Loc^i is outside the range of the validator, it is flagged as an anomaly. i.e., if $(d_{s_i} - d_{v_j}) > R$, then sensor s_i must have falsified.
- If the reported location of the sensor does not exist in the estimated annulus computed by the validator based on the reported SNR^i, then it is also flagged as an anomaly. i.e., if $\hat{d_{s_i}} < d^{min}_{s_i}$ or $\hat{d_{s_i}} > d^{max}_{s_i}$, then sensor s_i must have falsified. In figure 4.2, s_2 is detected as a falsifying sensor.

Note that, it is only important to detect falsifications in the sensor report, it is not required to identify the type of falsification (i.e., whether a sensor falsified in its reported SNR or location or both).

If $d^{min}_{s_i} \leq \hat{d_{s_i}} \leq d^{max}_{s_i}$, the sensor report may have been truthful with a confidence level. In figure 4.2 reports from s_1 and s_3 are not detected as anomalies. Since annulus estimated for s_3

is smaller, and the reported location of s_3 is within the annulus, s_3 must have reported the truth or falsified by an insignificant amount. Thus, s_3 is more likely to be truthful than s_1. The larger the thickness of the annulus, the higher uncertainty in the truthfulness of the sensor. Thus, the thickness of the annulus serves as a measure of the confidence on the truthfulness of the sensor. The confidence score of a sensor s_i, denoted by S_{s_i} is defined as, the normalized thickness of the annulus,

$$S_{s_i} = \begin{cases} 1 - \frac{(d_{s_i}^{max} - d_{s_i}^{min})}{d_0}, \text{if } (d_{s_i}^{min} \leq \hat{d_{s_i}} \leq d_{s_i}^{max}) \quad \& \\ \qquad\qquad\qquad\qquad\qquad\qquad (d_{s_i}^{max} - d_{s_i}^{min} < R) \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{, otherwise} \end{cases} \tag{4.5}$$

where, d_0 is a reference distance (the diameter of the region of interest) used for normalization. When a sensor falsifies information and it is flagged as an anomaly it would be assigned a confidence score of 0. When most of the sensors in the validator range are truthful, the thickness of the estimated annulus would be very small (since d_{err} is small), and the confidence in the truthfulness of the sensors increases (i.e., S_{s_i} is close to 1). If most of the sensors falsify, the thickness of the annulus is large (since d_{err} is large) and the uncertainty in the truthfulness increases. i.e., a falsifying sensor may not be detected as an anomaly. However, in this case, the score assigned to the falsifying sensor would be smaller than 1.

4.4 Blockchain based reputation

Algorithm 6 describes the blockchain related functionality of each validator. For each sensing report, the validators prepare a transaction, by including a transaction id, a sensor id, the sensing report $[SNR, Loc]^i$ and the confidence score for that report, as shown in line 4 and figure 4.3. The transactions for all the sensors are aggregated (line 6) and are ready to be inserted in to a block. The process of creating a block is called *mining* and is outlined in lines 7 and 8. Figure 4.4 depicts the structure of the Blockchain and its key features. Each block is composed of a block header and a block body which contains the list of transactions from the sensors. The block header includes the hash of the block, the hash of the previous block, the difficulty target for that block, a nonce and a timestamp. The merkle root field represents the hash value of the current block. Merkle tree hashing is commonly used in distributed systems and P2P networks for efficient data verification [78]. The nonce field is used for the proof-of-work algorithm, and it is the trial counter value that

Algorithm 6: Blockchain based Reputation Algorithm

Function *Reputation*($S_{s_i} \forall s_i \in \mathcal{S}$, $[SNR, Loc]$, N, $Blockchain$)
  n_{v_j} = count($[SNR, Loc]$);
  for *i=1:*n_{v_j} **do**
    $transaction = < transID, sensorID, [SNR, Loc]^i, S_{s_i} >$;
  end
  $transactions$ = Aggragate all transactions;
  Evaluate Difficulty of v_j, D_{v_j} from (4.6);
  $Block$ = CreateBlockwithDifficulty($transactions$,D_{v_j});
  $CandidateBlockchain$ = Add $Block$ to $Blockchain$;
  Broadcast $CandidateBlockchain$ to all validators;
  `// Consensus on Most-Difficult-Chain`
  Wait for Block-wait-time ($\tau_{\mathcal{B}}$) = 7s;
  Receive all Candidate Blockchains;
  $Blockchain$ = Select Most Difficult Blockchain;
  for *i=1:*N **do**
    Evaluate Reputation of s_i, $\mathcal{R}_{s_i}$ from (A.1);
  end
  return $\mathcal{R}$, $Blockchain$;
end

produced the hash with leading zeros. The difficulty target specifies the number of leading zero bits that the hash should contain to be considered valid. The implementation details are outlined in Section 5.

4.4.1 Difficulty of mining

When a new validator joins a network of validators, it gets a copy of the current blockchain. In addition to anomaly detection, validators also perform the functions of a blockchain miner [73]. Each validator generates a block $\mathcal{B}$ by aggregating the transactions, iterating over a nonce value and calculating the hash of a block with the nonce value included [72]. For the block $\mathcal{B}$ to be considered *valid*, a value of a hash function has to be less than a target T, i.e., $\mathrm{hash}(\mathcal{B}) < T$, where hash is a cryptographic hash function. The process of creating a valid block, typically requires a large amount of effort, which serves as a Proof-of-Work for the validators. The difficulty is a measure of how hard it is to find a hash below a given target T [79]. Unlike in many blockchain implementations we use a heterogeneous difficulty assignment mechanism, where each validator is assigned a different difficulty target. We define the difficulty of each validator $v_j \in \mathcal{V}$ as,

$$D_{v_j} = \left\lceil D_{max} \times \frac{n_{v_j}}{N} \right\rceil \qquad \forall v_j \in \mathcal{V} \tag{4.6}$$

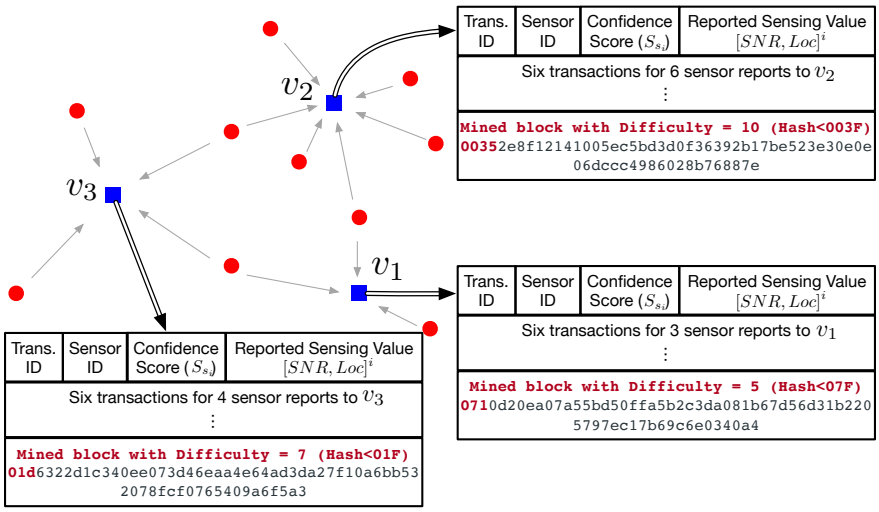


Figure 4.3: The validators create a valid block by aggregating all transactions from sensors and calculating a hash less than the target based on the difficulty.

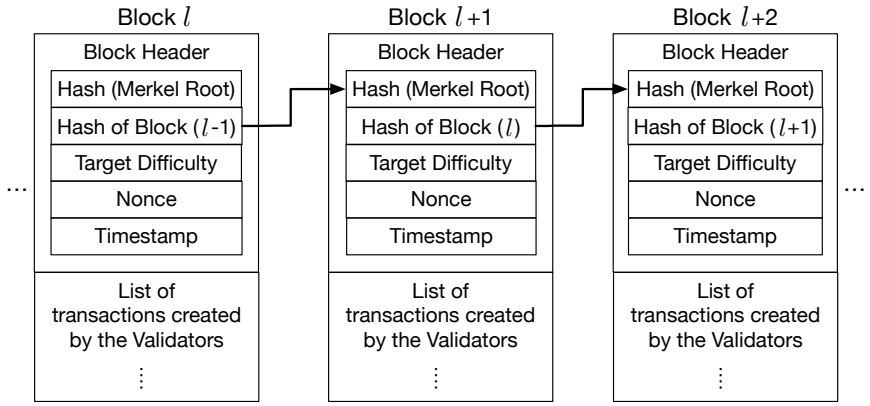


Figure 4.4: Blockchain structure showing chained blocks with header fields and body. The target difficulty changes from block to block depending on the most difficult block that was successfully mined.

where n_{v_j} is the number of sensors within the Validator Range R for validator v_j, and D_{max} is the maximum difficulty (designer's choice). By requiring the *Difficulty* for mining to be proportional to the number of sensors being validated, it acts as a *measure of credibility of the validators* as well. The difficulty assumes an integer value in the interval $[1, D_{max}]$, and represents the number of leading zero bits in the target T. An example of valid blocks (with valid hashes) created by validators with different difficulty targets is shown in figure 4.3.

Blockchain Structure: The average time for a validator to create a valid block is directly proportional to the difficulty of the target and inversely proportional to the average hash rate [80]. All the validators are assumed to have the same mining power (hash rate). Thus, a validator assigned with a higher difficulty target would on average, take a longer time to mine a block compared to a

validator with a lower difficulty target. Validators are rewarded according to their effort (the difficulty). Each validator contends with all the other validators in the entire area in creating a valid block.

Lines 7-10 describe the validators' functions in creating a blockchain with a valid block. Each validator first determines it's mandated difficulty level using (4.6). Then the validator creates a valid block with the assigned difficulty level as discussed above, by incorporating the pool of transactions. Once a valid block is created it is added to the current blockchain and broadcasted. To provide sufficient time for all validators (with different difficulty levels) to generate a valid block, the validators wait for a fixed amount of time (referred to as the 'Block-wait time' denoted by $\tau_{\mathcal{B}}$) to receive all the broadcasted candidate blockchains. The block time is set by design to account for the block mining time, the propagation time of blocks to reach all validators, and for all validators to reach a consensus. Since, the propagation time is much less than the block mining time, the value of $\tau_{\mathcal{B}}$ is determined empirically, as the average time required to mine a block with a difficulty of D_{max}. All the validators in the network receive the candidate blockchains. The validators wait till the end of the 'Block-wait time', $\tau_{\mathcal{B}}$, and select the candidate blockchain with the most difficulty as the valid blockchain as discussed in the following section.

Choice of Maximum Difficulty (D_{max}): A higher maximum difficulty (D_{max}), sets a lower target value for the calculated hash [72] and determines the value of $\tau_{\mathcal{B}}$. For a lower D_{max} (a high target value), validators can generate a valid block faster, with low computation cost, so the delay in disseminating the information (reputation) is less. However, blockchain forks may occur more frequently and the security (immutability) of the blockchain itself is lowered. This is because the amount of computation required by validators to generate a valid block is also less. Thus, there is a trade-off between the computational power (or the delay in settlements) and the level of security.

4.4.2 Most-Difficult-Chain consensus

The validators select the candidate Blockchain with the highest difficulty, which is termed as the *Most-Difficult-Chain rule* (unlike the Longest-chain rule in Bitcoin [73]). In the event that multiple validators succeed in creating a block, the blockchain may fork. Even if a blockchain fork occurs, the blockchain would converge, because each validator selects the chain with the highest aggregate difficulty and generate a new block following the most difficult blockchain. Even though blocks are mined and may arrive at the validators at different times, the chain can be synchronized

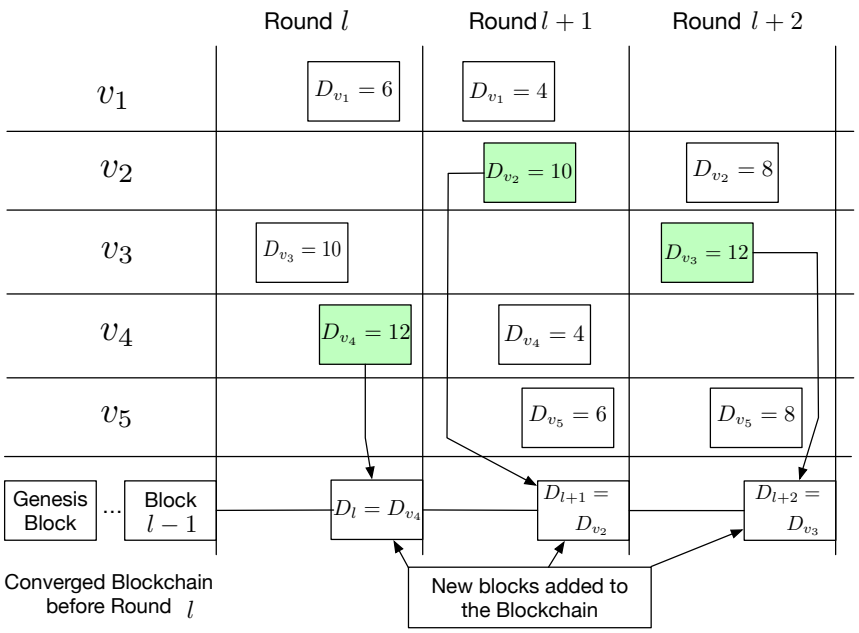


Figure 4.5: Consensus using the *Most-Difficult-Chain* rule: At the end of each round, the most difficult block that is successfully mined by the validators is added to the blockchain.

by exchanging status messages between the validators similar to [81].

The difficulty defined in (4.6) is proportional to the number of sensors within the range of a validator. Thus the most difficult block corresponds to the validator with the most number of sensors, which makes the most credible inference by harnessing the power of the crowd. Since the validator creates one transaction for each sensor within it's range, the difficulty of a block is proportional to the number of transactions in the block. Hence, the most difficult chain is the blockchain with the most number of transactions (scores and sensing data). The reputation of sensors' are assigned based on their transactions. Thus, the *most difficult blockchain would contribute to the most credible assignment of reputation* for sensors and would contribute to reliable inference using sensing data.

Figure 4.5 shows an example of the consensus on the most-difficult blockchain among 5 validators. Before round l, a majority of the validators arrive at consensus on a blockchain. Each round involves a sensing, validation and a blockchain exchange phase. In round l, a valid block is created by validators v_1, v_3 and v_4. Note that validators v_2 and v_5 were unable to create a valid block within time $\tau_{\mathcal{B}}$. Each validator adds its own block to the blockchain and multicasts it to all other validators. Due to the various delays in mining and propagation time, the validators may have different local views of the blockchain state. To guarantee the consensus properties and thus convergence to one canonical blockchain state, the `SenseChain` protocol relies on the

assumption that the majority of the consensus validators follow the most-difficult chain. Thereby, in round l the blockchain with the highest aggregate difficulty, i.e., the blockchain from v_4 is agreed upon as the canonical blockchain state by the majority. Similarly, in rounds $l+1$ and $l+2$ the blockchains from v_2 and v_3 respectively, represent the canonical blockchain states.

4.5 Historical reputation and provenance

The reputation of a sensor s_i is calculated from the information stored in the blockchain. Let $l = 1, \ldots, L$ represent the L blocks in the blockchain. For a blockchain of length L, the reputation is defined by the non-linear sigmoid function [82], whose exponent is the historical weighted average of the difficulty of the block, D_l and the scaled confidence scores of the sensor recorded in that block, $S_{s_i,l}$ and is defined as,

$$\mathcal{R}_{s_i} = \frac{1}{1+e^{-exp_{s_i}}} \tag{4.7}$$
$$\text{where, } exp_{s_i} = \frac{\beta \sum_{l=1}^{L} a_{i,l} D_l (2S_{s_i,l}-1)}{L.D_{max} \sum_{l=1}^{L} a_{i,l}}$$

where $a_{i,l}$ represents the association of sensor s_i with block l. $a_{i,l}$=1 if block l contains information about s_i, and $a_{i,l}$=0 otherwise. $\beta = 8$ is a sensitivity factor that asymptotically drives the sigmoid reputation function $\mathcal{R}_{s_i}$ to a value of 1 (or 0), when the exponent is largely positive (or negative) respectively. The term $(2S_{s_i,l}-1)$, scales the confidence score so that the resulting value is in the interval $[-1, 1]$. Note that the reputation of each sensor is calculated by the validator, by using the most-difficult-chain, scanning through all the blocks, extracting the confidence scores and difficulties of each block. Even though a malicious validator may forge information in its current block, since the reputation calculation relies on all records from the genesis block and due to the immutability of the blockchain, the impact on the calculated reputation of a sensor will be very small as the blockchain grows. The reputation $\mathcal{R}_{s_i}$ assumes a value in between 0 and 1. When a sensor continuously acts truthfully (or maliciously), the confidence scores recorded for that sensor in each block would be close to 1 (or 0), asymptotically driving the reputation R_{s_i} to a value of 1 (or 0) respectively. The nonlinear nature of the reputation function ensures that a sensor which engages vastly in either truthful or false behaviour would have a reputation of 1 or 0 respectively. Over time, this allows the validators to identify sensors that are always truthful or always malicious. Reputation of sensors that frequently alter their behaviour is subject to more

pronounced variation. Thus, the reputation of a sensor serves as a measure of the credibility of it's reports. The validators use the reputation of sensors assimilated over time to perform reputation weighted fusion of the sensing data in order to accurately detect and localize the target.

4.6 Reputation weighted Fusion

Agents sense the spectrum infractions and report fundamental sensing parameters such as SNR and location and possibly auxiliary parameters like the probability of detection (P_{d_i}) and probability of false alarms (P_{f_i}). The reputations of sensors is used by the validators to fuse the information of the distributed trust-less sensors for more reliable inference. This is achieved by performing a reputation-weighted aggregation of sensing results. For example, the weighted aggregation of P_d and P_f as in [1], using the normalized reputations as weights increases the credibility of detection result. e.g., $P_d^{fused} = \sum_i w_i P_{d_i}$, where w_i is the reputation of sensor s_i normalized by the aggregate reputation of all the sensors defined as $w_i = \mathcal{R}_{s_i} / \sum_i \mathcal{R}_{s_i}$. Similarly, the weighted localization as in [71], using the normalized reputations leads to more credible and accurate localization. Thus, the reputation-based weighted fusion of sensing information from trust-less agents enables credible and accurate distributed enforcement of spectrum policies.

CHAPTER 5

Evaluation and Results

5.1 Performance Evaluation of Autonomous Sensing System

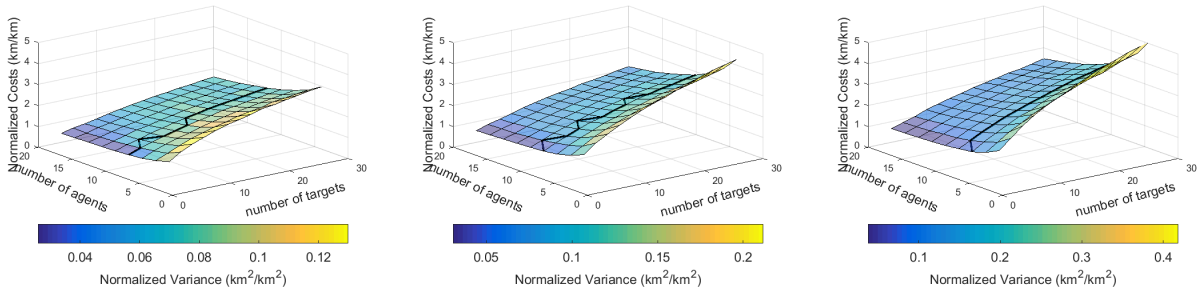
For practicality, the algorithm was evaluated in three prominent cities: New York City (NYC), Paris and London, primarily to understand the performance on different graphs, with the roads mapped as edges, and the intersections to vertices. To compare the outcome of Algorithm 3 among different cities, the cost metric (Definition 3) is normalized by the diameter of the graph and its unit is represented as km/km. For a rectangular area, the diameter is simply the diagonal. The choice of these three cities were attributed to the unique features of their road networks and intersections. The New York City map exhibited a dominantly grid-based uniform structure, while the Paris Network included a central star connected hub with roads protruding out from it and the London Network was comprised of dominant common edges in the form of bridges connecting sets of nodes. Figure 5.1 shows the cost metric and paths for one instance in each of the cities where 5 agents are routed using Algorithm 3 among 10 targets with different cardinality. For this particular instance, Paris exhibits the highest cost followed by London with NYC exhibiting the least cost. This behaviour is attributed to the features of the road network in these cities. In Paris, the agents have to travel via the central hub to cover the targets. A similar feature exists in London with the bridges providing the links between targets on either side of the river. Both of these features contribute to higher travel time. In comparison, NYC has highly connected, grid-like road

(a) Example routing in London, cost metric = $1.2317km/km$

(b) Example routing in NYC, cost metric = $1.5393km/km$

(c) Example routing in Paris, cost metric = $1.9896km/km$

Figure 5.1: Example routes and the normalized cost metric (km/km) for Algorithm 3 for 5 agents and 10 targets for (a) London, (b) NYC and (c) Paris. It shows how the road network influence the cost metric



(a) Normalized cost metric in London (b) Normalized cost metric in NYC (c) Normalized cost metric in Paris

Figure 5.2: Normalized cost metric for **Average Cardinality = 3** for (a) London (b) NYC and (c) Paris.The dark line highlights the points beyond which the cost variation is below 10%. The variance is indicated using the color scale.

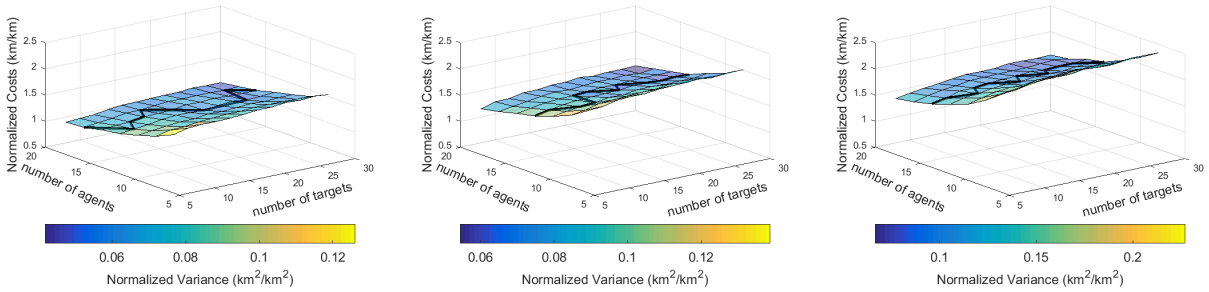
systems with plenty of connectivity between the targets, resulting in a relatively lower cost metric compared to the other two cities. To further investigate the Scheduling performance, we perform a parameter space analysis on larger geographical area and more agents and targets.

5.1.1 Parameter Space Analysis

For each graph (city) the number of agents (n), the number of targets (m), the location of agents (a), the location of targets (t) and the cardinality of targets (C_j) were varied and the effect on the paths (P_i) and the costs (l_i) of the agents were recorded. The location of agents and targets were chosen randomly among the available nodes in the graph. The agents are varied from 4 to 20 and number of targets from 4 to 30 and executed 500 unique arrangements of agents and targets. The cardinality was varied from 1 to n ($1 \leq C_j \leq n$). However, the distribution of C_j is controlled in two ways: (1) Constant average cardinality, and 2) Constant number of total visits across all the agents (Constant total cardinality).

5.1.1.1 Constant Average Cardinality

The cardinality was distributed among the targets such that an average cardinality of 3 is maintained across all agents. This ensures that even with increasing number of agents, the average number of visits required for the targets is 3, such that for a fixed number of violations the cost reduces as we deploy more agents. Figure 5.2 shows the mean and variance of the normalized cost metric for the three cities. The fixed average cardinality justifies the drop in cost observed when more agents are deployed. The cost increases with increasing violations, since the total number of visits required also increases linearly as the average cardinality is fixed. However, it is observed



(a) Normalized cost spread in London

(b) Normalized cost spread in NYC

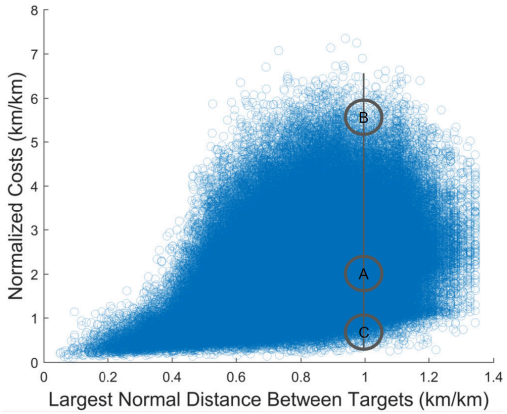
(c) Normalized cost spread in Paris

Figure 5.3: Normalized cost metric for **Total Visits = 40**. The dark line highlights the points beyond which the cost variation is below 10%. The variance is indicated using the color scale.

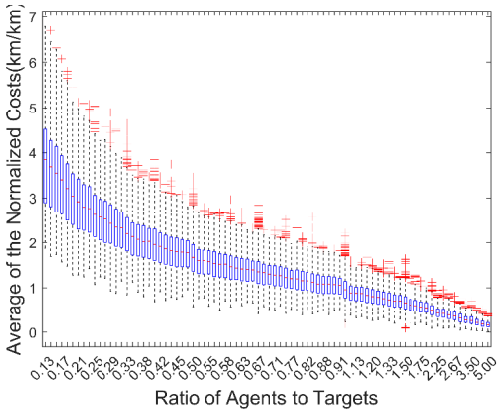
that the rate of reduction in the cost metric drops with increasing number of agents, denoted by the dark line, which shows the 10% reduction in the cost metric for different number of targets. This line indicates a boundary for cost-effective enforcement as adding more agents does not lead to substantial improvement in the cost metric. Further, the variance of the cost metric (the color axis) increases with the number of targets and drops with the number of agents. This is influenced by the fact that, as there are more infractions, the potential of having diverse target distributions (such as clusters or well separated targets) increases resulting in a larger variation. Figure 5.2 also reveals that the algorithm performs statistically similar with respect to the mean and variance of the cost metric among the cities regardless of the attributes of the city maps. However, closer inspection indicates that Paris portrays a slightly larger cost followed by New York and London. This behaviour is attributed to the features of the road network in these cities. In Paris, the agents have to travel via the central hub to cover the targets. This feature contributes to a higher travel time. In comparison, NYC has highly connected, grid-like road systems with plenty of connectivity between the targets, resulting in a relatively lower cost metric compared to Paris. It is interesting to observe that in London and New York the 10% line is located between 10-12 agents and for Paris between 8-10, suggesting that in London and New York the costs can be further improved by increasing the number of agents compared to Paris.

5.1.1.2 Constant Total Number of Visits

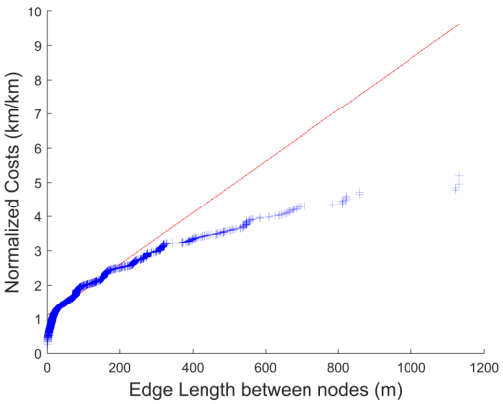
The total number of visits was fixed at 40 to ensure that even with increasing number of targets the total number of visits required by all the targets combined is limited to 40, such that the average cardinality reduces as the number of targets increases, limiting the growth in the cost. Due



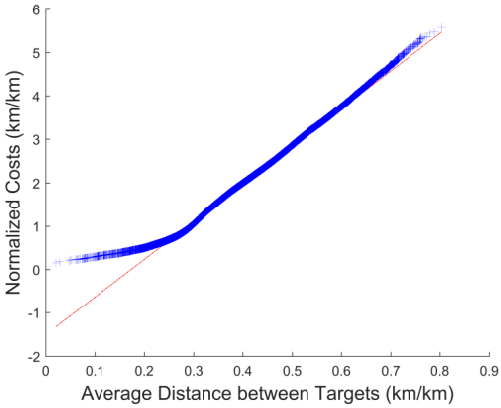
(a) Variation of Cost with maximum separation of targets.



(b) Variation of Cost with ratio of agents to targets



(c) Cost vs edge length



(d) Cost vs average inter-target distance

Figure 5.4: Comparison of the distribution of Normalized Cost Metric for NYC with that of (a) Edge lengths and (b) Average Distance between Targets.

to this constraint, it is observed that the change in cost with the number of targets was minimal compared to the previous case. Practical implications of limiting the total number of visits include situations where the cost must be maintained at a specific value even with increasing violations. This is a metric that is controlled by the Dispatch to conserve enforcement resources. The variation of the mean normalized cost metric in figure 5.3 portrays a similar pattern with minuscule increase from London to Paris. However, it is observed that the variance decreases with increasing number of targets unlike in the previous case. This is because the lesser the targets the higher is the average cardinality introducing more variation in the paths of agents for lesser number of violations.

5.1.2 Overall System Performance

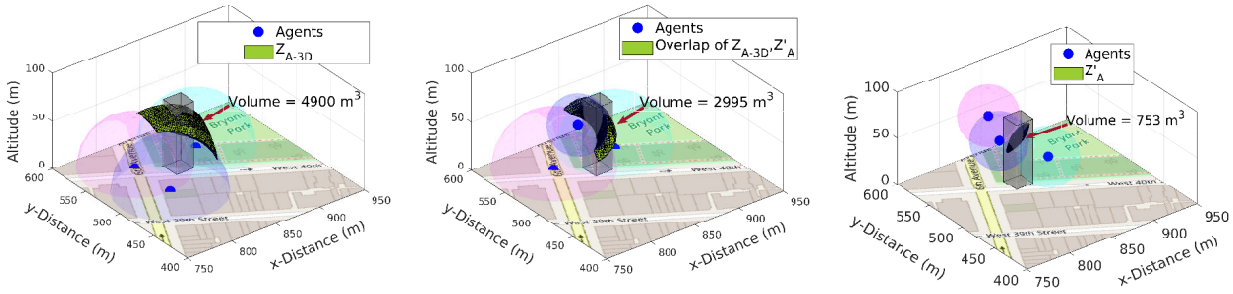
The overall performance metrics for Algorithm 3 is shown in Figure 5.4 for an average cardinality of 3. Figure 5.4a indicates increasing cost metric with increasing separation of targets,

confirming its influence on the cost. The large variation of cost observed at the same target separation, is due to the dependence of the cost metric on the distribution of the targets. Among the regions highlighted on Figure 5.4a at a target separation of 1, region A exhibits expected behaviour, where the cost metric is comparable to the separation of the targets. Region B has an unusually high cost metric due to widely distributed and hostile violations (high cardinality) with few available agents. On the contrary, region C portrays a much lower cost than the separation, which occurs when there are more agents than targets that are clustered within a small area.

The variation of the average cost metric with the ratio of agents to targets shown in figure 5.4b, confirms that deploying more agents for the same number of infractions, decreases the cost. The plot shows the scalability of the algorithm to larger systems of agents and targets. i.e if the hostility in an environment increases, increasing the number of agents by the same factor will ensure similar cost performance. The figure also shows the load balancing capability of the algorithm, which is naturally guaranteed, since it prunes the path of the costliest agents. This balances the number of targets visited by each agent, ensuring that no agent is overworked. The behaviour of the cost metric with a feature of the city graph (length of the edges in the city graph) and a feature of the mission graph (average inter-target distance) are shown as QQ plots in figures 8c and 8d. The high correlation of the cost metric with the average inter-target distance in figure 8d is due to the fact that, at larger inter-target distances, the cost metric is influenced more by the average target separation and less by other factors such as clustering of targets and features of the road network. In fact, while the city graphs differ among the cities (due to different road topologies), the features of the mission graphs are similar, explaining the similar trends of the cost metric among the cities observed in figures 6 and 7.

5.1.3 3D Localization and Detection

Figure 5.5 shows an example of an infraction event occurring at an elevation (at an altitude of 30m in a high-rise building) and the improvement in the accuracy of localization over UGVs, when UAVs are deployed. Figure 5.5a shows the 3D convex polyhedron, $\mathcal{Z}_{A,3D}$ containing the target as estimated by three UGVs positioned at the vertices of $\mathcal{Z}_C$ as described in Section 3.4. Eventhough, the UGVs can estimate the target location with reasonable accuracy (as the centroid of $\mathcal{Z}_{A,3D}$), we can further improve the accuracy of localization by deploying UAVs. The volume of the 3D convex polyhedron estimated by deploying a single UAV in conjunction with two UGVs, shows a 40% improvement in the localization accuracy over a purely UGV based localization (as



(a) Localization with 3 UGVs ($\mathcal{Z}_{A,3D}$).

(b) Localization with 2 UGVs and 1 UAV.

(c) Localization with 3 UAVs.

Figure 5.5: (a) Localization from UGV agents only, (b) Replacing one UGV with one UAV achieves 40% reduction in the volume of the convex polyhedron, compared to (a), (c) Localization with UAVs alone achieves 85% reduction compared to (a) in the volume of the convex polyhedron. The colored rings portray the inner spheres of the autonomous agents.

in figure 5.5b). Localization from a set of UAVs, results in more accurate localization of the target as depicted in figure 5.5c, showing a 85% improvement over the localization from UGVs. This demonstrates the superiority of UAVs in 3D localization.

5.1.4 Conclusion

In this section, we analyzed and evaluated the architected solution for the MPC problem that uses algorithms to derive the near-optimum cardinality of the targets and to compute schedule for all the agents to fulfill the cardinality in the shortest possible time. Through rigorous simulations and analysis, we draw four firm conclusions: 1) The autonomous agents are able to detect and localize targets with higher accuracy than a purely crowdsourced regime. 2) The scheduling algorithm is polynomial, has a provable bound of 3-approximation ratio, and provides the shortest paths for the agents while conforming to the cardinality requirement, 3) The scheduling algorithm exhibits strong generality across different geographical regions, by producing statistically similar results for varying degree of violations, 4) UAVs achieve significant improvement in localization accuracy over UGVs. While we await practical system implementation, the encouraging results from this work lay the foundation towards adopting a real-time, autonomous enforcement system for spectrum policies.

5.2 Performance Evaluation of `SenseChain`

`SenseChain` is evaluated on an integrated mobile sensing and Blockchain simulator built on Matlab. The simulation parameters are shown in table 5.1. We analyze the various facets of

`SenseChain` using practical simulations.

Table 5.1: Simulation Parameters

Parameters	Value/Model
Area	300m $\times$ 300m
Node Distribution	Uniform Distribution
Mobility Model	Random Waypoint
Propagation Model	Log-distance propagation model [77]
Path-loss exponent (γ)	3 (urban area)
Carrier Frequency (f)	600 MHz
Number of Validators	5
Number of Sensors	20
Antenna Type	Omnidirectional
Broadcast Range	100
Maximum Difficulty (D_{max})	16
Block-wait Time ($\tau_{\mathcal{B}}$)	7 s
Target location error (d_{err})	Uniformly distributed in [20,30] m

5.2.1 Evaluation Framework

Sensing Environment: We consider a random network topology with several Sensors, Validators and a single mobile target ($\mathcal{T}$). Random Waypoint mobility is chosen for movement of the various entities in the area of interest and each node is equipped with omnidirectional antenna. The target continuously transmits a signal at a fixed transmit power (40 mW). During the sensing phase, all the sensors receive the signal and compute their SNR from the received power (using (4.1) and NF = -96 dBm [76]). The sensors broadcast their reports ($[SNR, Loc]$) to the validators. Malicious behaviour of a sensor is emulated as a random variation (referred to as the *degree of falsification*) about the true SNR and true location of that sensor. Sensors exhibit malicious behaviour with varying probabilities and varying degrees of falsification. The validators receive reports only from sensors within the broadcast range (100 m). The diameter of the network, d_0 is 424 meters. The validators assign a confidence score to the sensing reports as in Algorithm 5.

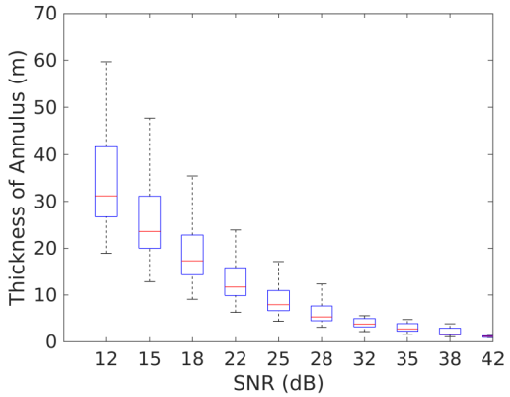
Blockchain Simulator: The blockchain environment in this work is different from typical implementations in two ways: 1) Heterogeneous difficulty assignment: The difficulty varies for each

validator and in each mined block in the blockchain. 2) Consensus: The validators arrive at consensus on the *Most-Difficult-Chain* to avoid forking. The simulation works as follows. For each sensing report, the validator creates a transaction by inserting the sensor id, the sensing report ($[SNR, Loc]$) and the confidence score. The sensor id is an integer index in the interval $[1, N]$, to represent the N sensors (e.g., sensor id of s_i is i). The transaction id is created by hashing transaction data through SHA-256 [83] twice, similar to typical blockchain implementations. The timestamps refers to the time at which the block was created, encoded as a Unix Epoch timestamp. The hash function used to generate the block id is SHA-256. The genesis block is created without any transactions by including a timestamp and creating a block with a hash corresponding to difficulty D_{max}.

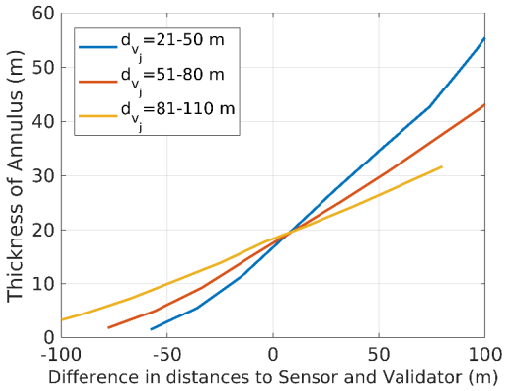
At each round, each validator generates a block by aggregating the transactions, iterating over a nonce value (a random integer) and calculating the hash of the block. Each block is mined with a different difficulty (see Section 4.4.2). The block is considered *valid*, when its hash is less than a target T that depends on the difficulty. This is verified by checking whether the hash of the block has at least as many leading zero bits as T, for that validator. Once a valid block is created it is added to the blockchain and multicast to the Validators. Note that the size of the block (the number of transactions) is not fixed as it depends on the number of sensors being validated. All the validators receive the candidate blockchains within $\tau_{\mathcal{B}}$ time (7 seconds). Each validator calculates the total difficulty of each candidate blockchain, and selects the one with the highest aggregate difficulty. Thus, the validators arrive at consensus on the *Most-Difficult-Chain*. A total of 1000 blocks were mined for each analysis presented below. For each sensor $s_i \in \mathcal{S}$, the validators scans through each block in the blockchain to extract the entries corresponding to its sensor id i. The validators then compute the reputation of each sensor using (A.1).

5.2.2 Performance of anomaly detection

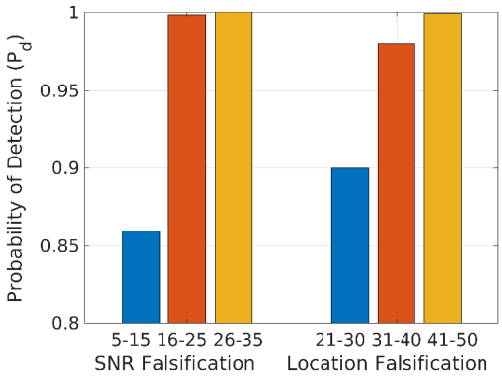
A falsifying sensor is detected as an anomaly, when its reported location exists outside the annulus (see Section 4.3), else a confidence score is associated with the sensor report, to reflect the confidence in the truthfulness of the sensor. The smaller the thickness of the annulus the more confident the validator will be on the truthfulness of the sensor. The thickness of the annulus is defined in Section 4.3.1 and is a measure of the confidence score. Figure 5.6a shows the dependence of the thickness of the annulus on the SNR reported by the sensors. The thickness of the annulus estimated by the validator decreases with increasing values of the SNR reported by the



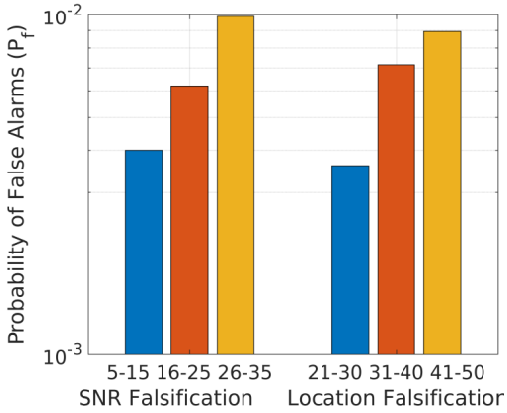
(a) Variation of annulus width with reported SNR



(b) Annulus width with Sensor and Validator distances



(c) P_d with varying falsification in SNR (dB) and Location (m)



(d) P_f with varying falsification in SNR (dB) and Location (m)

Figure 5.6: Performance of Anomaly Detection. (a) and (b) show the dependence of the thickness of the annulus on the SNR of the sensor and the distances from the target to the validator and the sensor. (c) and (d) show that when the degree of falsification is high, a validator is more likely to detect an anomaly, however the false alarms are also high.

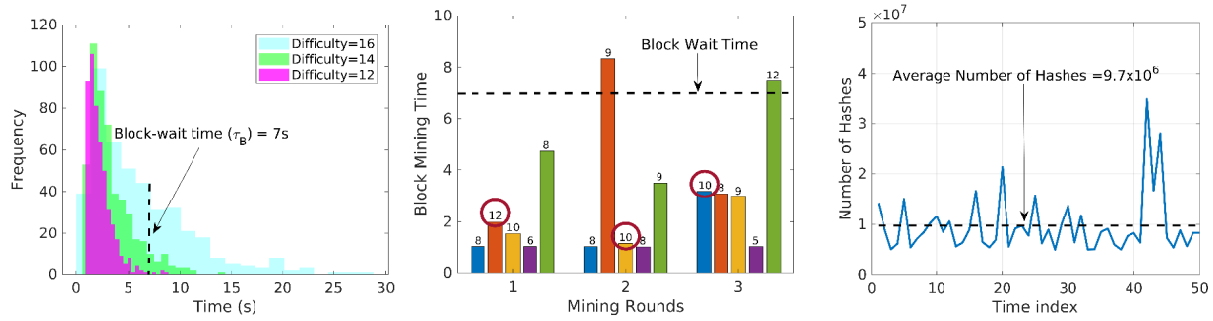
sensor. When the sensor reports a high SNR, the thickness of the annulus estimated by the validator is small. A sensor that falsifies by reporting a high SNR value is more likely to be detected as an anomaly since the thickness of the annulus is very small (and the reported location is likely to exist outside the annulus). For a sensor that falsely reports a low SNR value, the thickness of the estimated annulus would be large. Hence, it is possible for the reported location to exist within the annulus. But the confidence score for such sensors would be very low due to the large thickness of the annulus.

Figure 5.6b shows the dependence of the thickness of the annulus on the difference of the distances from the target to the validators and the reported location of the sensor Loc^i. i.e., $d_{s_i} - d_{v_j}$. Consider two validators v_1 and v_2, with $d_{v_1} < d_{v_2}$. When the sensor is located closer to $\mathcal{T}$ than

either validator, the thickness of the annulus estimated by v_1 would be less than v_2. That is v_1 would be able to, more accurately assess the truthfulness of the sensor. When the sensor is located further away from $\mathcal{T}$ than either validator, the thickness of the annulus estimated by v_2 would be less than v_1. In this case v_2 would be able to, more accurately assess the truthfulness of the sensor.

The performance of the anomaly detector with varying degree of falsification is shown in figures 5.6c and 5.6d. Figure 5.6c shows the probability of detection (P_d) and figure 5.6d shows the probability of false alarms (P_f) in detecting anomalies in the sensing reports. Recall that the annulus for a sensor is estimated using the reported SNR. If the sensor reports its true location and SNR, it will exist within the annulus. First, consider the effect on P_d and P_f with the degree of falsification in the reported SNR (i.e., the difference in the reported SNR and the true SNR) but reported location is true. When the degree of falsification in SNR increases, the reported location of the sensor is more likely to be outside the annulus. Hence, P_d increases with the degree of falsification. Even for low degrees of falsification in SNR (5-15 dB), P_d is relatively large (≈ 0.86). However, when a sensor falsifies to a higher degree, the possibility of flagging truthful sensors as anomalies increases. i.e, P_f also increases. Since the falsifying sensor is included in the weighted, distributed localization of $\mathcal{T}$ (as explained in Section 4.3), a higher degree of falsification leads to a higher possibility of error in the location of $\mathcal{T}$. Consequently, this leads to errors in the estimated annulus. Thus, a truthful sensor may exist outside the estimated annulus and may be detected as an anomaly. However, P_f is very low (less than 10^{-2}) even for higher degrees of falsification in SNR. When the falsification in the SNR is less, the error in the location estimate of $\mathcal{T}$ is less and false alarms are less.

Consider the effect on P_d and P_f by degree of falsification in the reported location while the reported SNR is true. When the degree of falsification in location increases, the more likely is the reported location of the sensor to be outside the annulus, and it is more likely to be detected as an anomaly. Hence, P_d increases with degree of falsification. Since the degree of falsification affects the distributed localization of $\mathcal{T}$, the possibility of flagging truthful sensors as anomalies also increases. i.e, P_f also increases. Overall, we see that Algorithm 5, achieves high probability of detection (≥ 0.86) even for low degrees of falsification and a low probability of false alarms (≤ 0.01) even for high degrees of falsification.



(a) Block mining times of validators with varying difficulty targets

(b) Block mining time per validator and winning block in each round

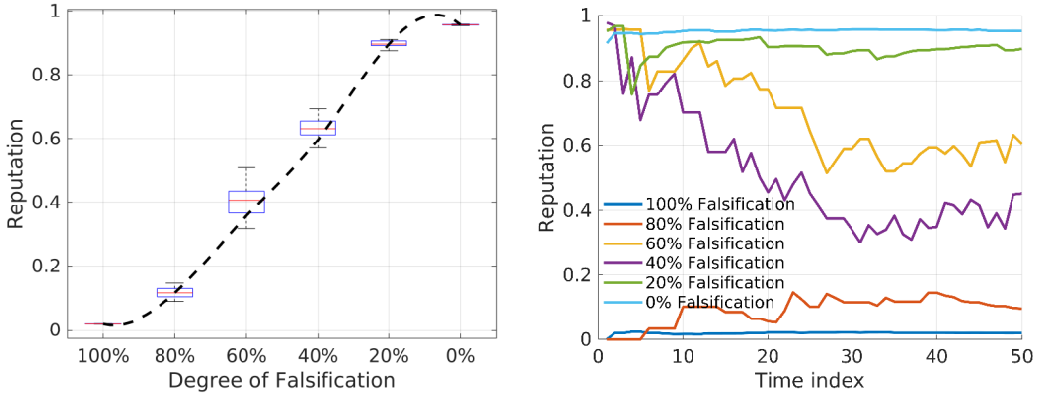
(c) The number of hashes generated by the winning validator over time

Figure 5.7: Impact of the difficulty of mining on the block mining time and the winning block. In (b) each color bar represents the blocks mined by each validator in order from v_1 to v_5. The numbers on the top of the color bar indicates the difficulty with which the block was mined. The winning block in each mining round are annotated by the red circles.

5.2.3 Performance of Blockchain based reputation

Blockchain performance: The performance of mining is shown in figure 5.7. Figure 5.7a shows the variation in the block mining time with varying difficulty of validators. The dotted line shows $\tau_{\mathcal{B}}$, i.e., the block-wait time which is equal to the average block time to mine a block with maximum difficulty ($D_{max} = 16$). When the difficulty level is high, the average time required to mine a block is more, since more amount of hashes are required on average, to find a hash value less than the target. The validators with a less difficulty target have a higher probability of mining a block within $\tau_{\mathcal{B}}$. The probability that a block is mined by a validator v_j within $\tau_{\mathcal{B}}$, when $D_{v_j} = 12$, $D_{v_j} = 14$ and $D_{v_j} = 16$, is 92%, 78% and 50% respectively. Even though validators with a lower difficulty have a higher probability of mining a block within $\tau_{\mathcal{B}}$, only the most-difficult block mined within $\tau_{\mathcal{B}}$ is added to the chain. This gives all the validators a chance to contribute to the blockchain and get rewarded.

The average time required to mine a block is proportional to the difficulty level and inversely proportional to the mining power of the validators [80]. Figure 5.7b shows the amount of time required by each validator to generate a valid block in each mining round. The block added to the blockchain at each mining round is determined by the most difficult block mined within the wait time of $\tau_{\mathcal{B}} = 7s$. The validators contend with all other validators in the area. As shown in the figure, in the first mining round v_2 is assigned the highest difficulty ($D_{v_2} = 12$) and generates a valid block within $\tau_{\mathcal{B}}$. Thus, the block from v_2 leads to the most difficult blockchain (as detailed in Section 4.4.2), and is agreed upon as the canonical blockchain. In the second mining round, a



(a) Reputation with degree of falsification by the Sensors

(b) Reputation of falsifying Sensors over time

Figure 5.8: Reputation assignment with varying degree of malicious activity over time

block is mined with the highest difficulty by v_3 ($D_{v_3} = 10$) and is added to the blockchain. In the third mining round, even though v_5 has a higher difficulty ($D_{v_5} = 12$) it is unable to mine a block within $\tau_{\mathcal{B}}$. The block mined within $\tau_{\mathcal{B}}$ with the highest difficulty is from v_1 and it is added to the blockchain. Note that in the second and third mining rounds, v_2 and v_5 respectively are unable to create a valid block within $\tau_{\mathcal{B}}$. Figure 5.7c shows the number of hashes generated by the winning validators (whose blocks are added to the blockchain) in each mining round. This serves as a measure of the amount of computation performed, time spent and power consumed by the validators in each round. Even though there are occasional spikes in the computational power (high number of hashes), most of the time the computational power is consistent. On average about 9.7 million hashes are calculated in each mining round.

Reputation Assignment: Figure 5.8 shows the reputation assigned to nodes, with varying degrees of malicious activity. Figure 5.8a shows the nonlinearity of the reputation function. Sensors that predominantly exhibit good behaviour ($falsification$<10%) by truthfully reporting sensing information asymptotically accumulate a reputation of 1. Sensors that continuously falsifies sensing information ($falsification$>90%) will accumulate a reputation close to 0. The reputation of sensors that arbitrarily alter their (truthful or malicious) behaviour by falsifying reports with a certain probability, are more susceptible to change based on their relatively dominant behaviour. The reputation of sensors which exhibit $falsification$ between 20% to 80% fall within the linear range of the sigmoid function. The reputation of these sensors are more likely to change depending on their dominant (truthful or malicious) behaviour.

The variation in the reputation of sensors over time is shown in figure 5.8b. Sensors cre-

ate malicious reports with a probability equal to their percentage of falsification. Over time, the reputation values of sensors that exhibit consistently, either truthful or malicious behaviour settle much quicker, compared to sensors that exhibit alternating behaviour. It is clear that after about 30 blocks the reputation of nodes settle to within 10% of their steady state reputation. It is important to note that even in the presence of a malicious validator the impact on the reputation is minimal. This is because, the reputation of any node is assimilated from the entire blockchain. Even if a malicious validator creates valid blocks with forged information, the impact on the reputation by these forged blocks, decreases significantly with the number of validators and the length of the blockchain.

5.2.4 Conclusion

In this section, we evaluated the proposed anomaly detection and reputation assignment scheme called `SenseChain`, which disseminates the reputation information via a blockchain. Through rigorous simulation and analysis we draw the following conclusions on `SenseChain`: 1) anomalies in sensing reports can be detected with high accuracy in a fully distributed, peer-based manner, 2) the *Most-Difficult-Chain* rule enables distributed consensus on the most-credible chain among spatially scattered agents, 3) the non-linear function to aggregate historical confidence scores and corresponding Difficulty, enables the reputation assignment based on a agents' degree of truthful (or malicious) behaviour. Thus, the distributed anomaly detection by validators and the use of the *Most-Difficult-Chain* to capture and disseminate the behaviour of agents, provides a fast and tamper-proof means to arrive at distributed consensus on the reputation of agents, among trustless entities. The reputation of agents is used in the fusion sensing information and achieve credible distributed detection and localization of targets.

5.3 Discussions

Ongoing Experimental testbed: The theoretical and simulated results in this work is currently being validated by over the air experiments. The Targets are emulated with software defined radios (e.g., Ettus USRP B210 [84]) transmitting in the 3.6 GHz band scattered geographically (outdoor and indoor). The cardinality of the violations is first estimated by a crowdsourced paradigm [1, 3]. The autonomous agents are a combination of UAVs and UGVs with software defined radios mounted on them. Each agent is provided with a set of rules (signal detectors) to check for spec-

trum infractions. The validators compute the schedule for each agent (Algorithm 3) based on the crowdsourced information and communicates it to all the Agents. The agents use GNSS (Global Navigation Satellite System) based automated navigation [85] to navigate to each target. At each target, the Agents perform SNR measurements, and broadcasts the quadruplet, $[SNR, loc, P_d, P_f]$ to other agents and moves on to the next Target. A fraction of agents serve as validators and assess the truthfulness of the sensing report and the reputation of the agent that sent the report. At the end of a single round of enforcement (when the cardinality of all the targets have been met), the validators compute the convex zone containing the target and the overall time of scheduling (determined by the agent travelling the longest path traversed). These measures are used to test this autonomous spectrum enforcement framework.

CHAPTER 6
Conclusion

In this work, we architected and analyzed a fully Autonomous Enforcement System that leverages heterogeneous mobile, autonomous agents to detect and localize dynamic spectrum infractions and a distributed reputation system to achieve reliable inferences among distributed trustless agents. This contribution is a necessary precursor to advance signal processing for enforcing spectrum etiquette using mobile, autonomous agents. Through rigorous simulations and analysis, we draw two firm conclusions:

1) The multi-modal autonomous agents can be scheduled in polynomial time to detect and localize dynamic and distributed spectrum violations with higher accuracy than purely static or crowdsourced regimes. We have shown the efficacy of using heterogeneous, autonomous agents in the enforcement of spectrum policies.

2) The anomalies in sensing reports can be detected with high accuracy in a fully distributed distributed manner. The distributed reputation system captures the sensors' degree of truthful (or malicious) by leveraging the distributed consensus mechanisms in Blockchain networks. Thus, the distributed anomaly detection by validators and the use of the *Most-Difficult-Chain* to capture and disseminate the behaviour of sensors, provides a fast and tamper-proof means to arrive at distributed consensus on the reputation of sensors, among trustless entities. The reputation of sensors can be used to fuse sensing information and achieve a highly reliable enforcement system (credible distributed detection and localization of targets), among distributed trustless agents.

While we await practical system implementation, the encouraging results from this work lay the foundation towards adopting a real-time, autonomous enforcement system for spectrum policies.

APPENDIX A

Approximation Ratio of Scheduling Algorithm 3

Proof. **CASE 1:** If the targets in $P_p \subseteq$ the targets in P_p^* (i.e., $T_y^p = 0$), then it follows from from Properties 1 and 2 that $l_p \leq 2.l_p^* \leq 2.l_q^*$. Hence, the approximation ratio is 2.

CASE 2: In the the case where the targets in $P_p \nsubseteq$ the targets in P_p^* (i.e., $T_y^p > 0$), from Property 3, there must exist a target t_k and an agent i such that t_k is in P_i^* but not in P_i (otherwise, in Algorithm 3, some targets would be visited by more agents than in OPT). Hence from Property 4 we have,

$$\begin{aligned} l_p &\leq l_i + l_i(t_k) \\ &\leq l_i(T_x^i) + l_i(T_y^i) + l_i(t_k) && \text{From Property 5} \\ &\leq 2\Delta l_i^* + l_i(T_y^i) && \text{From Properties 5 \& 1} \\ &\leq 2\Delta l_q^* + l_i(T_y^i) && \text{From Property 2} \end{aligned} \tag{A.1}$$

CASE 2a: If $l_i(T_y^i) \leq l_q^*$, then we have, $l_p \leq 2\Delta l_q^* + l_i(T_y^i) \leq 3\Delta l_q^*$, giving an approximation ratio of 3.

CASE 2b: If $l_i(T_y^i) > l_q^*$, consider a set $A_c \subseteq A$ with c, for which there is a target in P_i^* but not in P_i (i.e., $T_z^i > 0$). Then P_i must be costlier than $P_i^*, \forall i \in A_c$ (since $l_i(T_y^i) > l_q^*, \forall i \in A_c$). For the set of all agents $j \notin A_c$ since $T_z^j = 0$, the targets in P_j must at least include the same set of targets as in P_j^*. Then, from Property 3, for the set of all agents $i \in A_c$, the collection of all P_i^* must at least include all the targets visited by the collection of P_i. Hence we have,

$$\sum_{i \in A_c} T_y^i \subseteq \sum_{i \in A_c} T_z^i \tag{A.2}$$

Then from Property 1 and 2 we have,

$$\sum_{i \in A_c} l_i \leq 2. \sum_{i \in A_c} l_i^* \leq 2.c.l_q^* \tag{A.3}$$

Now from (A.2) and Property 5 and 1,

$$c.l_q^* \leq \sum_{i \in A_c} l_i(T_y^i) \leq 2. \sum_{i \in A_c} l_i^*(T_z^i) \leq 2.c.l_q^* \tag{A.4}$$

From (A.3), since $\sum_{i \in A_c} l_i \leq 2. \sum_{i \in A_c} l_i^*$, there must exist some agents $i \in A_d$ such that $A_d \subseteq A_c$, for which $l_i \leq 2.l_i^*$. Let any agent $i \in A_d$ contain b number of targets in T_z^i. Then by considering b iterations of Property 4 to remove all T_z^i from i, we have,

$$\begin{aligned}
b.l_p &\leq b.l_i + l_i(T_z^i) \leq 2b.l_i^* + 2l_i^*(T_z^i) && \text{From Property 1} \\
&\leq 2b.l_i^*(T_x^i) + 2(b+1).l_i^*(T_z^i) && \text{From Property 5} \\
&\leq 2(b+1).(l_i^*(T_x^i) + l_i^*(T_z^i)) \\
&\leq 2(b+1).l_i^* \leq 2(b+1).l_q^* && \text{From Property 5,2} \\
\implies l_p &\leq (2 + 2/b).l_q^* && \text{(A.5)}
\end{aligned}$$

Let $|T_z^i|$ and $|T_y^i|$ denote the number of targets in T_z^i and T_y^i respectively. For Property 3 to be satisfied, we have, $\sum_{i \in A} |T_z^i| = \sum_{i \in A} |T_y^i|$. Since for all agents $j \notin A_c$, $|T_z^j| \leq |T_y^j|$, we have $\sum_{i \in A_c} |T_z^i| \geq \sum_{i \in A_c} |T_y^i|, \forall i \in A_c$. Since, for all such agents, T_y^i contains at least one target, and predominantly T_z^i contains more targets than T_y^i, then at least some agents should have more than a single target in T_z^i, i.e., there will exist some agents i such that $b > 1$. Thus, considering any such agent i, from (A.5) we have, $l_p \leq 3.l_q^*$. Hence, Algorithm 3 is 3-approximation for the U-MPC problem.

In cases where $\forall i \in A_c$, T_z^i consists of only a single target, i.e., $b = 1$, from (A.2), all T_y^i cannot contain more than a single target. This leads to the fact that, $\sum_{i \in A_c} T_y^i = \sum_{i \in A_c} T_z^i, \forall i \in A_c$, which is only possible if $\forall j \notin A_c$, all P_j and P_j^* contain the same set of targets (i.e., $T_y^j = 0$). Now, if $p \notin A_c$, then it is similar to CASE 1 (since $T_y^p = 0$) and we have, $l_p \leq 2.l_q^*$. Now, consider the case where $p \in A_c$. For any agent $g \in A_c$, from Property 3, it follows that, the single target in T_y^g (say t_y) should also be in T_z^i for some other agent $i \in A_c$. This means that $t_y = T_y^g = T_z^i$, and hence, we have $l_g(t_y) = l_g(T_y^i) \geq l_q^*$. Since $l_g(t_y)$ is the cost calculated based on the round-trip MST, and since agents g and i and the target t_y belong to the same graph of the city, we have, $l_g(t_y) = l_i(t_y)$, provided that t_y is not the first target visited by agents g or i. Similarly, the single target in T_y^i must also be in T_z^h (say t_z) for some agent $h \in A_c$. This means that $t_z = T_y^i = T_z^h$,

and from Property 1 and 5 we have, $l_h(t_z) = l_h(T_z^h) \leq 2l_h^*(T_z^h) \leq 2l_h^* \leq 2l_q^*$ and $l_i(t_z) = l_h(t_z)$. Therefore, for any agent $i \in A_c$, we get,

$$\begin{aligned}
l_q^* \geq l_i^* &= l_i^*(T_x^i) + l_i^*(T_z^i) && \text{From Property 1 \& 5} \\
&\geq l_i(T_x^i)/2 + l_i(T_z^i)/2 && \text{From Property 1} \\
&\geq l_i(T_x^i)/2 + l_i(t_y)/2 \\
&\geq l_i(T_x^i)/2 + l_g(t_y)/2
\end{aligned}$$

Since $l_g(t_y) \geq l_q^*$, therefore we get, $l_i(T_x^i) \leq l_q^*$. Considering agents i and h, we get,

$$\begin{aligned}
l_i &= l_i(T_x^i) + l_i(T_y^i) = l_i(T_x^i) + l_i(t_z) \text{ From Property 5} \\
&= l_i(T_x^i) + l_h(t_z) \leq l_q^* + 2.l_q^* \\
&\leq 3.l_q^*
\end{aligned}$$

Thus, if $p \in A_c$, we get, $l_p \leq 3.l_q^*$. Hence, we conclude that the approximation ratio for Algorithm 3 is no worse than 3. □

BIBLIOGRAPHY

[1] Dutta, A., Chiang, M.: See something, say something, crowdsourced enforcement of spectrum policies. IEEE Transactions on Wireless Communications **15**(1) (Jan 2016) 67–80

[2] Nika, A., Li, Z., Zhu, Y., Zhu, Y., Zhao, B.Y., Zhou, X., Zheng, H.: Empirical validation of commodity spectrum monitoring. In: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM. SenSys '16, New York, NY, USA, ACM (2016) 96–108

[3] Achtzehn, A., Riihijärvi, J., Castillo, I.A.B., Petrova, M., Mähönen, P.: Crowdrem: Harnessing the power of the mobile crowd for flexible wireless network monitoring. In: HotMobile. (2015)

[4] Maqsood Ahamed Abdul Careem, Dutta, A., Wang, W.: Spectrum enforcement and localization using autonomous agents with cardinality. IEEE Transactions on Cognitive Communications and Networking **5**(3) (Sep. 2019) 702–715. © 2019 IEEE. Reprinted, with permission, from Maqsood Ahamed Abdul Careem and Aveek Dutta and Weifu Wang, "Spectrum Enforcement and Localization Using Autonomous Agents With Cardinality", IEEE TCCN 2019, 2019

[5] Maqsood Ahamed Abdul Careem, Dutta, A., Wang, W.: Multi-agent planning with cardinality: Towards autonomous enforcement of spectrum policies. 2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN) (2018) 1–10. © 2019 IEEE. Reprinted, with permission, from Maqsood Ahamed Abdul Careem and Aveek Dutta and Weifu Wang, "Multi–Agent Planning with Cardinality: Towards Autonomous Enforcement of Spectrum Policies", IEEE DYSPAN 2018, 2018

[6] Maqsood Ahamed Abdul Careem, Dutta, A.: Sensechain: Blockchain based reputation system for distributed spectrum enforcement. 2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN) (2019) 1–10. © 2019 IEEE. Reprinted, with permission, from Maqsood Ahamed Abdul Careem and Aveek Dutta, "SenseChain: Blockchain based Reputation System for Distributed Spectrum Enforcement", IEEE DYSPAN 2019, 2019

[7] Federal Communications Commision: "Report and Order and second further notice of Rule making,"

[8] Cave, M., Webb, W. In: Spectrum management: using the airwaves for maximum social and economic benefit. Cambridge University Press (2015)

[9] Yang, D., Xue, G., Fang, X., Tang, J.: Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In: Proceedings of the 18th Annual International Conference on Mobile Computing and Networking. Mobicom '12, New York, NY, USA, ACM (2012) 173–184

[10] : La sheriffs launch crowdsourced crowd control: Leedir, a surveillance app that uses your photos and videos (Apr 2014)

[11] : If you see something, say something (Jun 2019)

[12] Weiss, M.B.H., Lehr, W., Altamimi, M., Cui, L.: Enforcement in dynamic spectrum access systems. (2012)

[13] Altamimi, M., Weiss, M.B.H., McHenry, M.: Enforcement and spectrum sharing: Case studies of federal-commercial sharing. (2013)

[14] Park, J., Reed, J.H., Beex, A.A., Clancy, T.C., Kumar, V., Bahrak, B.: Security and enforcement in spectrum sharing. Proceedings of the IEEE **102**(3) (March 2014) 270–281

[15] Atia, G., Sahai, A., Saligrama, V.: Spectrum enforcement and liability assignment in cognitive radio systems. In: 2008 3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks. (Oct 2008) 1–12

[16] Harik, E.H.C., Guérin, F., Guinand, F., Brethé, J., Pelvillain, H.: Uav-ugv cooperation for objects transportation in an industrial area. In: 2015 IEEE International Conference on Industrial Technology (ICIT). (March 2015) 547–552

[17] Tokekar, P., Hook, J.V., Mulla, D., Isler, V.: Sensor planning for a symbiotic uav and ugv system for precision agriculture. IEEE Transactions on Robotics **32**(6) (Dec 2016) 1498–1511

[18] Harik, E.H.C., Guinand, F., Pelvillain, H., Guérin, F., Brethé, J.: A decentralized interactive architecture for aerial and ground mobile robots cooperation. In: 2015 International Conference on Control, Automation and Robotics. (May 2015) 37–43

[19] Tanner, H.G.: Switched uav-ugv cooperation scheme for target detection. In: Proceedings 2007 IEEE International Conference on Robotics and Automation. (April 2007) 3457–3462

[20] Phan, C., Liu, H.H.T.: A cooperative uav/ugv platform for wildfire detection and fighting. In: 2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing. (Oct 2008) 494–498

[21] Pajak, D.: Algorithms for Deterministic Parallel Graph Exploration. PhD thesis, Universite Sciences et Technologies - Bordeaux I (September 2014)

[22] Fraigniaud, P., Gasieniec, L., Kowalski, D.R., Pelc, A.: Collective tree exploration. Networks **48**(3) (2006) 166–177

[23] Tamas Kalmar-Nagy, G.G., Bak, B.D.: The multiagent planning problem. Complexity **2017** (2017) 12

[24] Sharon, G., Stern, R., Goldenberg, M., Felner, A.: The increasing cost tree search for optimal multi-agent pathfinding. Artificial Intelligence **195** (2013) 470 – 495

[25] Sundar, K., Rathinam, S.: An exact algorithm for a heterogeneous, multiple depot, multiple traveling salesman problem. In: 2015 International Conference on Unmanned Aircraft Systems (ICUAS). (June 2015) 366–371

[26] Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.L.: A review of dynamic vehicle routing problems. European Journal of Operational Research **225**(1) (2013) 1 – 11

[27] Sundar, K., Venkatachalam, S., Rathinam, S.: Formulations and algorithms for the multiple depot, fuel-constrained, multiple vehicle routing problem. CoRR **abs/1508.05968** (2015)

[28] Yadlapalli, S.K., Rathinam, S., Darbha, S.: An approximation algorithm for a 2-depot, heterogeneous vehicle routing problem. In: 2009 American Control Conference. (June 2009) 1730–1735

[29] Czyzowicz, J., Gasieniec, L., Kosowski, A., Kranakis, E. In: Boundary Patrolling by Mobile Agents with Distinct Maximal Speeds. Springer Berlin Heidelberg, Berlin, Heidelberg (2011) 701–712

[30] Wang, W., Zou, W., Zhou, Z., Zhang, H., Ye, Y.: Decision fusion of cooperative spectrum sensing for cognitive radio under bandwidth constraints. In: 2008 Third International Conference on Convergence and Hybrid Information Technology. Volume 1. (Nov 2008) 733–736

[31] Wang, W., Zou, W., Zhou, Z., Zhang, H., Ye, Y.: Decision fusion of cooperative spectrum sensing for cognitive radio under bandwidth constraints. In: 2008 Third International Conference on Convergence and Hybrid Information Technology. Volume 1. (Nov 2008) 733–736

[32] Liggins, M.E., Chee-Yee Chong, Kadar, I., Alford, M.G., Vannicola, V., Thomopoulos, S.: Distributed fusion architectures and algorithms for target tracking. Proceedings of the IEEE **85**(1) (Jan 1997) 95–107

[33] Bayhan, S., Zubow, A., Wolisz, A.: Spass: Spectrum sensing as a service via smart contracts. In: 2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN). (Oct 2018) 1–10

[34] Wei, H., Sun, H.: Using bayesian game model for intrusion detection in wireless ad hoc networks. IJCNS **3** (01 2010) 602–607

[35] Jr, J., Ulvila, J.: Evaluation of intrusion detectors: A decision theory approach. (01 2001) 50–

[36] She, W., Liu, Q., Tian, Z., Chen, J., Wang, B., Liu, W.: Blockchain trust model for malicious node detection in wireless sensor networks. IEEE Access **7** (2019) 38947–38956

[37] Zawaideh, F., Salamah, M., Al-Bahadili, H.: A fair trust-based malicious node detection and isolation scheme for wsns. In: 2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes Systems (IT-DREPS). (Dec 2017) 1–6

[38] Zhang, W., Zhu, S., Tang, J., Xiong, N.: A novel trust management scheme based on dempster—shafer evidence theory for malicious nodes detection in wireless sensor networks. J. Supercomput. **74**(4) (April 2018) 1779–1801

[39] Yin, G., Yang, G., Wu, Y., Yu, X., Zuo, D.: A novel reputation model for malicious node detection in wireless sensor network. In: 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing. (Oct 2008) 1–4

[40] Li, H., Han, Z.: Catch me if you can: An abnormality detection approach for collaborative spectrum sensing in cognitive radio networks. IEEE Transactions on Wireless Communications **9**(11) (November 2010) 3554–3565

[41] Jana, S., Zeng, K., Cheng, W., Mohapatra, P.: Trusted collaborative spectrum sensing for mobile cognitive radio networks. IEEE Transactions on Information Forensics and Security **8**(9) (Sep. 2013) 1497–1507

[42] Khan, M.A., Salah, K.: Iot security: Review, blockchain solutions, and open challenges. Future Generation Computer Systems **82** (2018) 395 – 411

[43] Kotobi, K., Bilen, S.G.: Secure blockchains for dynamic spectrum access: A decentralized database in moving cognitive radio networks enhances security and user access. IEEE Vehicular Technology Magazine **13**(1) (March 2018) 32–39

[44] Saad, M., Yuksel, M.: Routechain : Towards blockchain-based secure and efficient bgp routing. (2019)

[45] Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., Wan, J.: Smart contract-based access control for the internet of things. IEEE Internet of Things Journal **6**(2) (April 2019) 1594–1605

[46] Goka, S., Shigeno, H.: Distributed management system for trust and reward in mobile ad hoc networks. In: 2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC). (Jan 2018) 1–6

[47] Li, Z., Nika, A., Zhang, X., Zhu, Y., Yao, Y., Zhao, B.Y., Zheng, H.: Identifying value in crowdsourced wireless signal measurements. In: WWW. (2017)

[48] Nika, A., Zhang, Z., Zhou, X., Zhao, B.Y., Zheng, H.: Towards commoditized real-time spectrum monitoring. In: Proceedings of the 1st ACM Workshop on Hot Topics in Wireless. HotWireless '14, New York, NY, USA, ACM (2014) 25–30

[49] Yang, D., Xue, G., Fang, X., Tang, J.: Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In: Proceedings of the 18th Annual International Conference on Mobile Computing and Networking. Mobicom '12, New York, NY, USA, ACM (2012) 173–184

[50] Fawcett, T.: An introduction to roc analysis. Pattern Recognition Letters **27**(8) (2006) 861 – 874 ROC Analysis in Pattern Recognition.

[51] Parkinson, Bradford W., Spilker, James J., Jr., Axelrad, Penina, Enge, P.: Global Positioning System, Volume 1 - Theory and Applications. American Institute of Aeronautics and Astronautics (1996)

[52] Gavish, B., Srikanth, K.: An optimal solution method for large-scale multiple traveling salesmen problems. Operations Research **34**(5) (1986) 698–717

[53] Xiong, J., Jamieson, K.: Arraytrack: A fine-grained indoor location system. In: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation. nsdi'13, Berkeley, CA, USA, USENIX Association (2013) 71–84

[54] Levy, B.C.: Principles of Signal Detection and Parameter Estimation. 1st edn. Springer Publishing Company, Incorporated (2010)

[55] Kumar, V., Park, J.M., Bian, K.: Blind transmitter authentication for spectrum security and enforcement. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. CCS '14, New York, NY, USA, ACM (2014) 787–798

[56] Powers, D.M.W.: Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. Journal of Machine Learning Technologies **2**(1) (2011) 37–63

[57] OpenStreetMap: OpenStreetMap - http://www.openstreetmap.org

[58] Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University (1976)

[59] Prim, R.C.: Shortest connection networks and some generalizations. The Bell System Technical Journal **36**(6) (Nov 1957) 1389–1401

[60] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. In: Introduction to Algorithms, Third Edition. 3rd edn. The MIT Press (2009) 1111–1115

[61] MacArthur, D.K., Crane, C.D.: Unmanned ground vehicle state estimation using an unmanned air vehicle. In: 2007 International Symposium on Computational Intelligence in Robotics and Automation. (June 2007) 473–478

[62] Federal Aviation Administration: Recreational fliers & modeler community-based organizations

[63] FAA: Fact sheet, small unmanned aircraft regulations (part 107)

[64] 3GPP, ETSI: "5G; Study on channel model for frequencies from 0.5 to 100 GHz (3GPP TR 38.901 version 14.1.1 Release 14),"

[65] NYC: Pluto and mappluto

[66] Wood, D.: Ethereum: A secure decentralised generalised transaction ledger. (2014)

[67] Buterin, V.: A next generation smart contract & decentralized application platform. (2015)

[68] Zekavat, R., Buehrer, R.M.: Handbook of Position Location: Theory, Practice and Advances. 1st edn. Wiley-IEEE Press (2011)

[69] Wang, J., Urriza, P., Han, Y., Cabric, D.: Weighted centroid localization algorithm: Theoretical analysis and distributed implementation. IEEE Transactions on Wireless Communications **10**(10) (October 2011) 3403–3413

[70] Yang, J., Chen, Y., Lawrence, V.B., Swaminathan, V.: Robust wireless localization to attacks on access points. In: 2009 IEEE Sarnoff Symposium. (March 2009) 1–5

[71] Altoaimy, L., Mahgoub, I., Rathod, M.: Weighted localization in vehicular ad hoc networks using vehicle-to-vehicle communication. In: 2014 Global Information Infrastructure and Networking Symposium (GIIS). (Sep. 2014) 1–5

[72] Meshkov, D., Chepurnoy, A., Jansen, M.: Revisiting difficulty control for blockchain systems. IACR Cryptology ePrint Archive **2017** (2017) 731

[73] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf

[74] Careem, M.A.A., Dutta, A., Wang, W.: Multi-agent planning with cardinality: Towards autonomous enforcement of spectrum policies. (10 2018) 1–10

[75] Chawla, S., Gionis, A.: k-means-: A unified approach to clustering and outlier detection. In: SDM. (2013)

[76] : IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007) (March 2012) 1–2793

[77] Rappaport, T.: Wireless Communications: Principles and Practice. 2nd edn. Prentice Hall PTR, Upper Saddle River, NJ, USA (2001)

[78] Khan, M.A., Salah, K.: IoT security: Review, blockchain solutions, and open challenges. Future Generation Computer Systems **82** (2018) 395–411

[79] Eyal, I.: The miner's dilemma. In: 2015 IEEE Symposium on Security and Privacy. (May 2015) 89–103

[80] O'Dwyer, K.J., Malone, D.: Bitcoin mining and its energy footprint. In: 25th IET Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014). (June 2014) 280–285

[81] Ethereum: Ethereum wire protocol (eth) (cited July 2019)

[82] Elfwing, S., Uchibe, E., Doya, K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Networks **107** (Nov 2018) 3–11

[83] of Commerce, U.D., of Standards, N.I., Technology: Secure Hash Standard - SHS: Federal Information Processing Standards Publication 180-4. CreateSpace Independent Publishing Platform, USA (2012)

[84] Ettus Research: "USRP B200 and B210 Product Overview,"

[85] ArduPilot Dev Team: "Planning a Mission with Waypoints and Events,"