Transmitter Localization and Optimizing Initial State in Classical/Quantum Sensor Networks

A Dissertation presented

By

Caitao Zhan

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

May 2024

Stony Brook University

The Graduate School

Caitao Zhan

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Prof. Himanshu Gupta – Advisor
Professor, Department of Computer Science

Prof. Samir Das – Committee Chairperson
Professor and Chair, Department of Computer Science

Prof. C. R. Ramakrishnan
Professor and Graduate Program Director, Department of Computer
Science

Prof. Nengkun Yu
Associate Professor, Department of Computer Science

Prof. Mark Hillery
Professor, Department of Physics and Astronomy, Hunter College of the
City University of New York

This dissertation is accepted by the Graduate School

Celia Marshik
Dean of the Graduate School

Abstract of the Dissertation

Transmitter Localization and Optimizing Initial State in Classical/Quantum Sensor Networks

by

Caitao Zhan

Doctor of Philosophy

IN

Computer Science

Stony Brook University

2024

In shared spectrum systems, it is important to be able to localize simultaneously present multiple intruders (unauthorized transmitters) to effectively protect a shared spectrum from malware-based, jamming, or other multi-device unauthorized-usage attacks. We address the problem of localizing multiple intruders using a distributed set of classical radio-frequency (RF) sensors in the context of a shared spectrum system. In contrast to single transmitter localization, multiple transmitter localization (MTL) has not been thoroughly studied. The key challenge in solving the MTL problem comes from the need to "separate" an aggregated signal received from multiple intruders into separate signals from individual intruders. We solve the problem via a Bayesian-based approach and a deep-learning-based approach.

After addressing multiple transmitter localization with a network of classical RF sensors, we explore using a quantum sensor network for transmitter localization. A quantum sensor network is a network of spatially dispersed sensors that leverage the quantum superposition and quantum entanglement. We pose our transmitter localization problem as a quantum state discrimination (QSD) problem and use the positive operator-valued measurement as a tool for localization in a novel way. Then, we address the additional challenge of the impracticality of general quantum measurement by developing new schemes that replace the QSD's measurement operators with trained parameterized hybrid quantum-classical circuits.

Finally, we investigate problems that are unique in quantum sensors. We look into optimizing the initial state of detector sensors in quantum sensor networks. We consider a network of quantum sensors, where each sensor is a qubit detector that "fires", i.e., its state changes when an event occurs close by. The change in state due to the firing of a detector is given by a unitary operator. The determination of the firing sensor can be posed as a QSD problem which incurs a probability of error depending on the initial state and the measurement operators used. We address the problem of determining the optimal initial state of the quantum sensor network that incurs a minimum probability of error in determining the firing sensor. The optimal initial state is in general an entangled state, and thus there is a demand to generate and distribute the entangled state to a network of sensors. The last part of the thesis is about the efficient generation and distribution of entangled pairs.

Contents

List of Figures

viii

List of Tables

Acknowledgments

I would like to express my sincere gratitude to my advisor, Prof. Himanshu Gupta, for his guidance and support throughout this journey. After chatting with many Ph.D. students in the department about their advisors, I think Prof. Himanshu is one of the professors who provide the most support to and spend the most time on students, especially for tenured senior professors. He has taught me how to conduct research, think scientifically, select good problems to solve, solve problems, write papers, and present the work. When I run into obstacles, Prof. Himanshu can always help me find a way to overcome the obstacle. This thesis came out to be "half classical and half quantum", due to the change of research direction from wireless networks to quantum networks in the middle of my Ph.D. Quantum is very hard, but I always have faith because I believe that Prof. Himanshu's ingenuity and vision will lead me to success. Thank you, Himanshu.

I am also very thankful to my committee members, Professor Samir Das, C.R. Ramakrishnan, Mark Hillery, and Nengkun Yu. Prof. Samir had a big influence on me during the first half of my Ph.D. while I was working on spectrum sensing. My lab seniors told me to include Prof. Samir in the committee no matter what. Thereafter, he became the chair of my RPE, Prelim, and the dissertation committee. Prof. C.R. gave me very insightful suggestions during the second half of my Ph.D. while I was working on quantum networks. His vast knowledge and intelligence helped me approach problems from different angles. I am very fortunate to have the opportunity to collaborate with Prof. Mark, a very respected and famous scientist in the field of quantum optics and quantum physics. My work in quantum sensor networks is rooted in his expertise in quantum state discrimination. It came out that Prof. Nengkun is also an expert in quantum state discrimination. I started asking him questions about quantum state discrimination right after he moved from Australia to Stony Brook. Thus I am also grateful to Prof. Nengkun.

After thanking my advisor and committee members, I give thanks to my Ph.D. peers. First and foremost, I would like to thank Dr. Mohammad Ghaderibaneh, a.k.a. Shahrokh. We joined the lab at the same time in the summer of 2018 and closely collaborated on several projects since then. He is a very reliable teammate and I enjoyed the time we worked together. When I just joined the lab in 2018, I collaborated with my lab senior Dr. Arani Bhattacharya on a paper and he set a very good example for me. Lab seniors Dr. Jian Xu, Dr. Mallesh Dasari, and Dr. Santiago Vargas also gave me tremendous help. Other lab seniors including Dr. Yi, Dr. Qingqing, and Dr. Max are also important. During the COVID year of 2020, Dr. Pranjal Sahu, Mallesh, and I are among the handful of students who still dare to go to the department every day in person, especially from July 2020 to early 2021 when there was no vaccine. During that special time, Pranjal helped me a lot on a project that required training deep neural networks. During the last months of my Ph.D., I collaborated with lab new member Xiaojie Fan extensively on a paper. He is very hard working and I hope I have set a good example for him just like Arani had set a good example for me. Other Ph.D. peers worth mentioning include Tanmay, Prerna, Dr. Anand, Dr. Abeer, Weihai, Zhengyu, Dr. Duin, Pramodh, Ranjani, John, Bo,

Publications

Papers Included in this Thesis

- **Caitao Zhan**, Himanshu Gupta, Arani Bhattacharya, Mohammad Ghaderibaneh, "Efficient Localization of Multiple Intruders in Shared Spectrum System," ACM/IEEE IPSN, Sydney, Australia, 2020.

- **Caitao Zhan**, Mohammad Ghaderibaneh, Himanshu Gupta, "DeepMTL: Deep Learning Based Multiple Transmitter Localization," IEEE WoWMoM, Pisa, Italy, 2021.

- **Caitao Zhan**, Mohammad Ghaderibaneh, Himanshu Gupta, "DeepMTL Pro: Deep Learning Based Multiple Transmitter Localization and Power Estimation," Elsevier PMC, 2022.

- **Caitao Zhan**, Himanshu Gupta, "Quantum Sensor Network Algorithms For Transmitter Localization," IEEE QCE, Bellevue, WA, 2023

- **Caitao Zhan**, Himanshu Gupta, Mark Hillery, "Optimizing Initial State of Detector Sensors in Quantum Sensor Networks," ACM TQC, 2024.

- Mohammad Ghaderibaneh, **Caitao Zhan**, Himanshu Gupta, C.R. Ramakrishnan, "Efficient Quantum Network Communication using Optimized Entanglement Swapping Trees," IEEE TQE, 2022.

Papers Not Included in this Thesis

- Arani Bhattacharya, **Caitao Zhan**, Himanshu Gupta, Samir R. Das, Petar M. Djuric, "Selection of Sensors for Efficient Transmitter Localization," IEEE INFOCOM, 2020

- Himanshu Gupta, Max Curran, **Caitao Zhan**, "Near-Optimal Multihop Scheduling in General Circuit-Switched Networks," ACM CoNEXT, 2020

- Arani Bhattacharya, **Caitao Zhan**, Abhishek Maji, Himanshu Gupta, Samir R. Das, Petar M. Djuric, "Selection of Sensors for Efficient Transmitter Localization," IEEE/ACM TON, 2021.

- Mark Hillery, Himanshu Gupta, **Caitao Zhan**, "Discrete Outcome Quantum Sensor Networks," Physical Review A, 2023

- Mohammad Ghaderibaneh, **Caitao Zhan**, Himanshu Gupta, "DeepAlloc: CNN-Based Approach to Spectrum Allocation in Shared Spectrum Systems," IEEE ACCESS, 2024

- Xiaojie, **Caitao Zhan**, Himanshu Gupta, CR Ramakrishnan, "Optimized Distribution of Entanglement Graph States in Quantum Networks," Under review.

Chapter 1

Introduction

1.1 Background and Motivation

Wireless sensor network (WSN) [135] is a network of spatially dispersed and dedicated sensors that monitor and record the physical conditions of the environment and forward the collected data to a central location [4] via wireless communication. WSN can measure environmental conditions such as temperature, sound, humidity, wind, light, wireless channel state information, radio spectrum, etc. A sensor network becomes a *quantum sensor network* (QSN) when the sensors leverage quantum objects and quantum properties [66] such as quantum coherence and quantum entanglement. Quantum sensors are extremely sensitive to physical quantities such as magnetic field, electric field, quadrature displacement and phase shift in the optic field.

Classical sensors. WSNs have various applications [5, 58, 108]. In this thesis, the application we focus on is spectrum surveillance and monitoring [39] for security and threat detection. The *core problem* involved in this application is *transmitter localization* [25, 306], and in particular, *multiple transmitter localization* (MTL) as the number of transmitter present in an area could be more than one and localizing multiple transmitters are not independent. The reason for being not independent is that a sensor receives an aggregated power from multiple transmitters and separating the power from different multiple sources is impractical. That an aggregated received power is not able to separate is a big challenge for MTL.

Furthermore, in a shared spectrum paradigm, the presence of an evolving set of authorized users (e.g., primary and secondary users) adds to the challenge. The RF spectrum is a natural resource in great demand due to the unabated increase in mobile (and hence, wireless) data consumption [8]. The research community has addressed this capacity crunch via the development of shared spectrum paradigms, wherein the spectrum is made available to unlicensed users (secondaries) as long as they do not interfere with the transmission of licensed incumbents (primaries). The fundamental objective behind such shared spectrum paradigms is to maximize spectrum utilization, the viability of such systems depends on the ability to effectively guard the shared spectrum against unauthorized usage. The current mechanisms however to locate such unauthorized users (intruders) are human-intensive and time-consuming,

Figure 1.1: Classical sensors. Radiofrequency sensors used in this thesis. Details see §2.6

involving the FCC enforcement bureau which detects violations via complaints and manual investigation [130].

Motivated by the above, we seek an effective technique that can accurately localize multiple simultaneous intruders and even in the presence of a dynamically changing set of authorized users. Our solution assumes a network of crowdsourced sensors wherein relatively low-cost spectrum sensors (Fig. 1.1) are available for gathering signal strength in the form of received power. We introduce two different approaches to the MTL problem. The first approach is a hypothesis-driven Bayesian approach, viz. maximum a posterior approach, where wherein each hypothesis is a configuration (i.e. a combination of ⟨location, power⟩ pair of the potential intruders), and the goal is to determine the hypothesis that best explains the sensor observations. The second approach is a deep learning-based approach. First, we encode the sensors' observation data into an image. Then, we frame MTL as a sequence of two steps: image-to-image translation and object detection, each of which is solved using a trained CNN model. The first step of image-to-image translation maps an input image representing sensor readings to an image representing the distribution of transmitter locations, and the second object detection step derives precise locations of transmitters from the image of transmitter distributions. Besides the location, the transmission power is another property of a transmitter that we wish to estimate. We introduce some novel methods to estimate the power of multiple transmitters. We also introduce a novel interpolation method for received signal strength.

Quantum sensors. We continue the research in transmitter localization but with the usage of a new kind of sensor – quantum sensors. Although classical sensors are omnipresent and work well in general, there are big motivations to explore quantum sensors. Quantum sensing is an emerging field that leverages quantum objects and properties at atomic/subatomic scales and has the potential to sense physical parameters at an unprecedented level of precision. Therefore, quantum sensing brings new
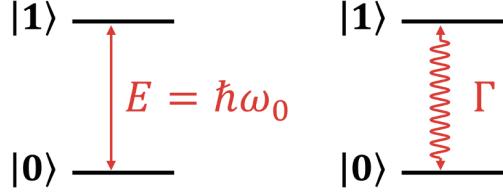
Figure 1.2: Quantum sensor theory. Basic features of a two-state quantum sensor, figure from [66]. $|0\rangle$ is the lower energy state and $|1\rangle$ is the higher energy state. Quantum sensing leverages changes in the transition frequency ω_0 (shifts of the energy level) or the transition rate (transition between energy levels) in response to an external signal V.
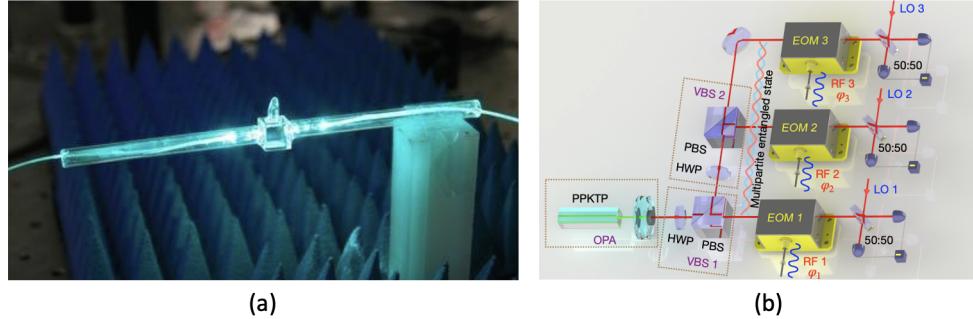
Figure 1.3: Quantum sensors. (a) Fiber-coupled vapor cell for Electric field measurements using Rydberg atoms, figure from [3]; (b) Reconfigurable entangled radio frequency photonic sensor network, figure from [280].

opportunities to new and well-established problems. For example, physicists in the year 2016 used the squeezed state of light to improve the sensitivity of the Laser Interferometer Gravitational-wave Observatory (LIGO) detector [243] and successfully detected gravitational waves. In [280], researchers use some distributed quantum RF-photonic sensors to estimate the amplitude and phase of a radio signal. They showed the performance of sensing a global property of the RF wave is enhanced (beating the standard quantum limit by over 3 dB) by leveraging multipartite entangled state and squeezed light.

Our key idea is to pose the transmitter localization problem as a well-studied *quantum state discrimination* (QSD) [13, 20, 21], which allows us to develop viable transmitter localization schemes using quantum sensors. We design two high-level schemes to localize a transmitter in a given area deployed with a quantum sensor network. The first scheme is based on solving an appropriate quantum state discrimination problem using a global measurement, while the second scheme uses a trained hybrid quantum-classical circuit to process the quantum sensor data. Within the above high-level schemes, we also introduce a two-level localization scheme to improve the performance of the basic one-level schemes. To evaluate our schemes, we model how a quantum sensor's state evolves due to RF signals from a transmitter at a certain distance. Using this model, we evaluate our localization schemes and

demonstrate their effectiveness in our custom-built simulator.

Finally, we investigate two problems that are unique in quantum sensor networks. The first problem is *optimizing the initial state* of detector sensors in quantum sensor networks. We consider a network of quantum sensors, where each sensor is a qubit detector that "fires", i.e., its state changes when an event occurs close by. The change in state due to the firing of a detector is given by a unitary operator. The determination of the firing sensor can be posed as a QSD problem which incurs a probability of error depending on the initial state and the measurement operator used. We address the problem of determining the optimal initial state of the quantum sensor network that incurs a minimum probability of error in determining the firing sensor. The optimal initial state could be an entangled state. Thus, the second problem we look into is the *generation and distribution of entangled states.* Quantum entanglement – correlation between multiple particles – is a phenomenon that has no counterpart in the classical world. It is the physical phenomenon that occurs when a group of particles (electrons, photons, etc) are generated or interact in a way such that the quantum state of each particle of the group cannot be described independently of the state of the others, including when the particles are separated by a large distance. In our context of QSNs, entanglement can serve as a resource to enhance the performance of the QSN. Thus, there is a demand to generate and distribute the entangled state to a network of sensors. This a challenging problem in the field of quantum communication. Physical transmission of quantum states across nodes can incur irreparable communication errors, as the no-cloning theorem proscribes making independent copies of arbitrary qubits. The establish of entanglement over long distances is challenging due to the low probability of success of the underlying physical process (short-distance entanglement and swapping). We propose an efficient heuristic approach that efficiently generates an entanglement pair in a quantum network.

1.2 Thesis Statement

This thesis strives to:

- Develop novel methods to *localize transmitter(s)* efficiently and accurately using classical and quantum sensor networks.

- *Optimize* and *generate* the *initial state* for quantum sensor networks.

1.3 Thesis Organization

Towards the thesis statement, we make the following contributions:

- In Chapter 2, we introduce an efficient hypothesis-based Bayesian approach `MAP*` for multiple transmitter localization (`MTL`) problem (§2.3.1); A closed-form equation for the estimation of transmission power (§2.3.2); A novel received signal strength interpolation method inspired by the power law distribution (§2.3.3); Extend `MAP*` to accommodate the presence of authorized users (§2.4).

- In Capter 3, we introduce a deep learning-based approach `DeepMTL` for the `MTL` problem (§3.3, §3.4); Extend `DeepMTL` via deep learning models to accommodate the presence of authorized users (§3.5); A deep learning-based approach that estimates the transmission power of multiple transmitters (§3.6).

- In Chapter 4, we introduce the concept of quantum sensor networks and the model of a quantum sensor (§4.2); In the context of quantum sensor networks, we pose a transmitter localization problem as a quantum state discrimination problem and introduce a novel quantum localization method `POVM-Loc` and `POVM-Loc Pro` based on positive-operator valued measure (§4.3).

- In Chapter 5, we consider a network of quantum sensors, where each sensor is a qubit detector that "fires", its state changes when an event occurs close by. We address the problem of determining the optimal initial global state of a network of quantum sensors that incur a minimum probability of error in determining the firing sensor (§5.2). We have proposed both theoretical analytical results (§5.4, §5.5) and numerical simulation results(§5.6).

- In Chapter 6, we introduce an efficient heuristic algorithm `Balanced-Tree` for routing an entanglement pair. The algorithm is Dijkstra-based, and the path selection metric is a closed-form expression that models a path as a tree near accurately (§6.4).

Chapter 2

Efficient Localization of Multiple Intruders for Shared Spectrum System

We address the problem of localizing multiple intruders (unauthorized transmitters) using a distributed set of sensors in the context of a shared spectrum system. In contrast to single transmitter localization, multiple transmitter localization (`MTL`) has not been thoroughly studied. In shared spectrum systems, it is important to be able to localize simultaneously present multiple intruders to effectively protect a shared spectrum from malware-based, jamming, or other multi-device unauthorized-usage attacks. The key challenge in solving the MTL problem comes from the need to "separate" an aggregated signal received from multiple intruders into separate signals from individual intruders. Furthermore, in a shared spectrum paradigm, presence of an evolving set of authorized users (e.g., primary and secondary users) adds to the challenge.

In this chapter, we propose an efficient algorithm for the `MTL` problem based on the hypothesis-based Bayesian approach called `MAP`. Direct application of the `MAP` approach to the `MTL` problem incurs prohibitive computational and training cost. In this work, we develop optimized techniques based on `MAP` with significantly improved computational and training costs. In particular, we develop a novel interpolation method, `ILDW`, which helps minimize the training cost. We generalize our techniques via online-learning to the setting wherein there may be a set of dynamically-changing authorized users present in the background. We evaluate our developed techniques on large-scale simulations as well as on small-scale indoor and outdoor testbeds. Our experiments demonstrate that our technique outperforms the prior approaches by significant margins, i.e., error up to 74% less in large-scale simulations and 30% less in real-world testbeds.
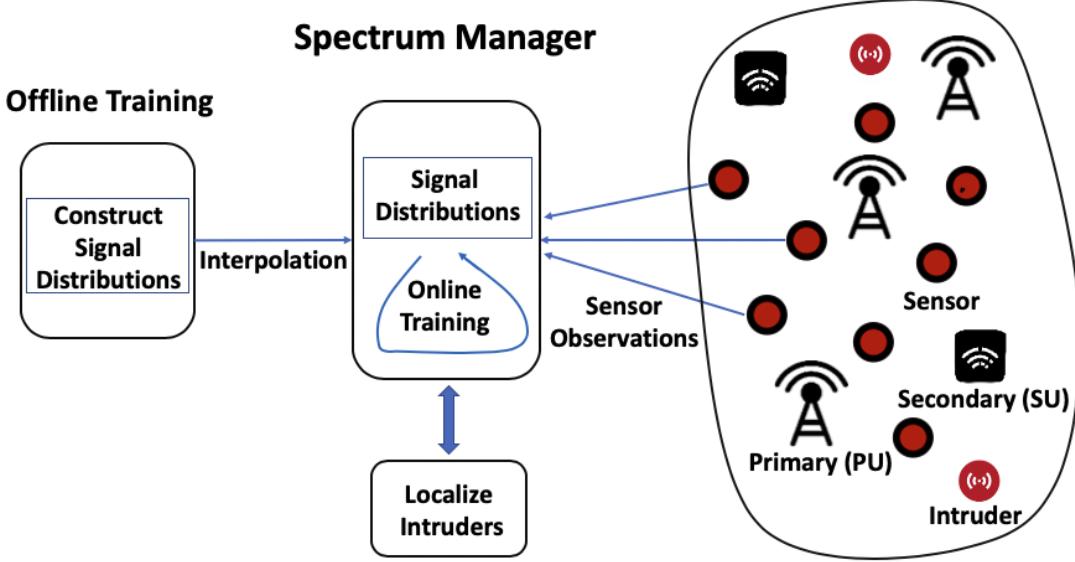
Figure 2.1: Overall approach to localize intruders in a shared spectrum system.

2.1 Introduction

The RF spectrum is a natural resource in great demand due to the unabated increase in mobile (and hence, wireless) data consumption [8]. The research community has addressed this capacity crunch via the development of *shared spectrum paradigms*, wherein the spectrum is made available to unlicensed users (secondaries) as long as they do not interfere with the transmission of licensed incumbents (primaries). E.g., in the recent years, the FCC has made available the CBRS band, i.e., the 3550-3700 MHz band within the 3.5 GHz band, for shared commercial use to allow other users to utilize the otherwise low-usage band which was previously reserved for incumbent users including US Navy radar operators.

The increasing affordability of the software-defined radio (SDR) technologies makes the shared spectrums particularly prone to unauthorized usage or security attacks. With easy access to SDR devices [1, 2], it is easy for selfish users to transmit data on the shared spectrum without any authorization and potentially causing harmful interference to the incumbent users. Such illegal spectrum usage could also happen as a result of infiltration of computer virus or malware on SDR devices. As the fundamental objective behind such shared spectrum paradigms is to maximize spectrum utilization, the viability of such systems depends on the ability to effectively guard the shared spectrum against unauthorized usage. The current mechanisms however to locate such unauthorized users (intruders) are human-intensive and time-consuming, involving FCC enforcement bureau which detects violations via complaints and manual investigation [130]. Motivated by the above, we seek for an effective technique that is able to accurately localize multiple simultaneous intruders and even in the presence of a dynamically changing set of authorized users. In the following, we begin by describing the multiple transmitter localization problem.

Multiple-Transmitter Localization (MTL). The transmitter localization prob-

lem has been well-studied, but most of the focus has been on localizing a *single* intruder at a time. However, it is important to localize *multiple* transmitters *simultaneously* to effectively guard a shared spectrum system. E.g., a malware or virus-based attachment could simultaneously cause many devices to violate spectrum allocation rules; spectrum jamming attacks would typically involve multiple transmitters. More importantly, a technique limited by localization of a single intruder could then be easily circumvented by an offender by using multiple devices. The key challenge in solving the MTL problem comes from the fact that the deployed sensor would receive only a sum of the signals from multiple transmitters, and separating the signals may be impossible. In addition, the other challenge that MTL in the context of shared spectrum system poses is the presence of authorized users—e.g., the incumbent users and the dynamic set of secondary users that have been allocated spectrum by the manager. To the best of our knowledge, no prior localization work has considered the presence of authorized users.

The state-of-the-art technique for the MTL problem is the recent work [130], which essentially decomposes the MTL problem to multiple single-transmitter localization problems based on the sensors with the highest power readings in a neighborhood. However, the technique has a few shortcomings: (i) it implicitly assumes a propagation model, and thus, may not work effectively in areas with complex propagation characteristics, (ii) it is not effective in the case of transmitters being located close-by, a key challenging scenario for MTL problem, and (iii) most importantly, it can't be extended effectively to incorporate background authorized users, a key requirement in the context of shared spectrum systems.

Our Approach. Transmitter localization is generally done based on observations at deployed sensors. In particular, as in prior works [38, 130], we assume a crowd-sourced sensing architecture wherein relatively low-cost spectrum sensors are available for gathering signal strength in the form of received power. Our approach is a hypothesis-driven Bayesian approach, viz. *maximum a posteriori* (`MAP`) approach, wherein each hypothesis is a configuration (i.e. a combination of $\langle location, power \rangle$ pair) of the potential intruders, and the goal is to determine the hypothesis that best explains the sensor observations. This determination is done based on the distributions (gathered during a training phase) of sensor observations for each hypothesis. The `MAP` approach is known to have optimal classification accuracy, but (i) incurs prohibitive computation cost—exponential in the number of potential intruders—when applied to the `MTL` problem, and (ii) requires a significant amount of training cost. The focus of our work is to address these challenges, and design a viable `MAP`-based approach. In particular, using `MAP` as a building block, we develop an optimized approach that runs in polynomial time with minimized training costs. We extend our technique to work in the presence of authorized users by incorporating online (real-time) training.

<u>Motivation for `MAP`.</u> Our motivation for using a `MAP`-based approach is multifold: First, with sufficient training data, `MAP` is known to deliver optimal classification accuracy for the MTL problem [77]. Second, the `MAP` approach doesn't assume any propagation model and thus works for arbitrary signal propagation characteristics. Third, it allows

us to also estimate the intruder's transmit power, which can be very useful in some applications, e.g., where the penalty is proportional to the extent of violation. Last but not the least, it naturally extends to being able to handle the presence of an evolving set of authorized users.

Training Cost and Optimization. The benefits of a `MAP`-based approach come at a cost: the `MAP` framework requires prior training to build probability distributions (PDs) of sensor observations for each hypothesis. However, most of the training occurs offline, one-time, and can be automated e.g. via drones or robots. In our work, we develop strategies to minimize the training cost; in particular, we reduce the number of PDs to be constructed via a *novel interpolation scheme* suited to our unique setting, and evaluate the impact of reduced training on the localization accuracy. We note that the online training to incorporate the presence of authorized users is needed only for the prevailing setting (of authorized transmitters and deployed sensors) and hence incurs minimal cost (see §2.4).

Overall Contributions. The goal of our work is to develop an efficient technique for accurate localization of simultaneously present multiple intruders in a shared spectrum system. The raw data are available at https://github.com/Wings-Lab/IPSN-2020-data. In this context, we make the following four specific contributions.

1. Design an efficient localization algorithm (`MAP*`) for the MTL problem, based on an optimal hypotheses-driven Bayesian approach. The designed approach predicts both locations and transmit powers of the intruders, and does not assume any propagation model and thus, works for arbitrary signal propagation characteristics.

2. Extend the designed algorithm (`MAP**`) to localize effectively in the presence of background authorized users, i.e., primaries with possibly unknown parameters (e.g., location and transmit power) and an evolving set of secondary users.

3. Develop an effective interpolation scheme (`ILDW`) for our unique setting to reduce the one-time training cost of our scheme, without impacting the localization accuracy much.

4. Evaluate our techniques via large-scale simulations as well as over two developed testbeds (indoor and outdoor), and demonstrate the effectiveness of our developed techniques and their superior performance compared to the best-known techniques.

2.2   Problem, Related Work, and Methodology

In this section, we describe our model of the shared spectrum systems, formulate the `MTL` problem, and discuss related work. We also describe the building block of our approach, viz., a hypothesis-driven Bayesian localization approach (`MAP`).

Shared Spectrum System. In a shared spectrum paradigm, the spectrum is shared among licensed users (primary users, PUs) and unlicensed users (secondary users,

SUs) in such a way that the transmission from secondaries does not interfere with that of the primaries (or secondaries from a higher-tier, in case of a multi-tier shared spectrum system [273]). In some shared spectrum systems, the location and transmit power of the primary users may be unavailable, as is the case with military or navy radars in the CBRS band [273]. Such sharing of spectrum is generally orchestrated by a centralized entity called *spectrum manager*, such as a spectrum database in TV white space [131] or a central spectrum access system in the CBRS 3.5GHz shared band [106]. The spectrum manager allocates spectrum to requesting secondaries (i.e., permission to transmit up to a certain transmit power at their location) based on their location, spectrum demand, configurations of the primaries, other active secondaries, prevailing channel conditions, etc. SwarmShare [113, 114] is proposed to enable spectrum sharing between the incumbent systems and the coexisting UAV networks in the 6GHz band. Researchers have developed NeXT [112, 115], a software-defined wireless testbed, to support both traditional model-based control and new data-driven control techniques in wireless research.

Authorized and Unauthorized Users. Secondary users that have been explicitly given permission to transmit at their location are termed as *authorized users*; the primary users are also considered as authorized users. Note that the set of authorized users evolve over time, as more and more SUs are allocated spectrum and as some SUs stop using the spectrum after a while. We can assume that each SU is allocated spectrum for a certain duration of time, after which it stops using the spectrum. Other users that transmit without explicit permission (for that given time) are referred to as *unauthorized users* or *intruders*.

Problem Setting and Formal Definition. Consider a geographic area with a shared spectrum. Without loss of generality, we assume a single channel throughout this paper (multiple channels are handled similarly). For localization of unauthorized users, we assume available crowdsourced sensors that can observe the received signal in the channel of interest, and compute (total) received signal strength indicator (RSSI)[1]. These sensors, being crowdsourced, may be at different locations at different times. At any given instant, the shared spectrum area has some licensed primary users and some active secondary users; the PU configurations may not be known as can be the case for military users. The centralized spectrum manager is aware of the set of active SUs at any time, as each SU request is granted for a certain period of time. In addition to the authorized users, there may be a set of intruders present in the area with each intruder in a certain "configuration" (see §2.2.2).

The MTL problem is to determine the set of intruders with their configurations at each instant of time, based on the set of sensor observations at that instant. See Figure 2.1. The basic MTL problem assumes no other transmissions (of authorized users) in the background. The more general MTL problem, where there may be an evolving set of authorized users in the background, is referred to as the MTL-SS problem. We address the MTL problem in §2.3, and then address the more general MTL-SS problem in §2.4.

---

[1]We do not use angle-of-arrival (AoA) measurements [313] as they require additional and complex RF hardware.

10

2.2.1 Related Work

Localization of an intruder in a field using sensor observations has been widely studied, but most of the works have focused on localization of a single intruder [39, 78]. In general, to localize multiple intruders, the main challenge comes from the need to "separate" powers at the sensors [203], i.e., to divide the total received power into power received from individual intruders. Blind source separation is a very challenging problem; only very limited settings allow for known techniques [153, 228] using sophisticated receivers. In our context of hypotheses-driven approach, the challenge of source separation manifests in terms of a large number of hypotheses, a challenge addressed in §2.3. We note that (indoor) localization of a device [14] based on signals received from multiple reference points (e.g, WiFi access points) is a quite different problem (see [300] for a recent survey), as the signals from reference points remain separate, and localization or tracking of multiple devices can be done independently. Recent works on multi-target localization/tracking are different in the way that targets are passive [61, 99, 128], instead of active transmitters in this work.

In absence of blind separation methods, to the best of our knowledge, only a few works have addressed multiple intruder(s) localization, and none of these consider it in the presence of a dynamically changing set of authorized transmitters. In particular, (i) [130] decomposes the multi-transmitter localization problem to multiple single-transmitter localization problems based on the sensors with the highest of readings in a neighbohood, (ii) [193] works by clustering the sensors with readings above a certain threshold and then localizing intruders at the centers of these clusters, (iii) [194] uses an EM-based approach. The techniques of [130, 194] assume a propagation model, while that of [193, 194] require a priori knowledge of the number of intruders present. We have compared our approach with [130, 193] in §2.5, while [194] has a high computational cost and has also been shown to be inferior in performance to [130, 193] even for a small number of intruders. Other related works include (i) [91] that addresses the challenge of handling time-skewed sensor observations in the MTL problem, and (ii) [23] that addresses the sensor selection optimization problem for our proposed hypotheses-based localization approach.

Bayesian methods are a subset of statistics in which probability expresses a degree of belief in an event. Bayes' theorem provides a way to update the probability of a hypothesis as more evidence or information becomes available. It combines prior knowledge with new evidence to form a posterior probablity. Bayesian methods are used to calculate the parameters for movement time distribution models [259] and to predict the distribution and the mean of pointing movement time [317].

2.2.2 MAP: Bayesian Approach for Localization

We localize intruders based on observations from a set of sensors. Each sensor communicates its observation to a centralized entity, the spectrum manager, which runs an appropriate localization algorithm to localize the intruders. In particular, we use a hypotheses-driven Bayesian approach, as described below, where intruders are localized by determining the most likely prevailing hypothesis; this is done based on

joint probability distributions of the sensors' observations (constructed during a priori training). Below, we formalize the above concepts, and the basic localization approach.

Observation; Observation Vector. Throughout this paper, we use the term *observation* at an individual sensor to mean the received power over a time window of certain duration, in the frequency channel of interest (we assume only one channel). In particular, received power is computed from the FFT of the I/Q samples in the time window [39]. We use the term *observation vector* $\mathbf{x}$ to denote a vector of observations from a given set of distributed sensors, with each vector dimension corresponding to a unique sensor.

Hypotheses. Let H_0, H_1, ..., H_m be the set of all hypotheses, where each hypothesis H_j represents a "configuration" of potential intruders. In this chapter, we largely assume an intruder's configuration to be comprised of just its location and transmit power, but the concept of configuration is quite general and could include any attributes (e.g., height, antenna direction, etc.) that affect how its transmitted signal is received at other locations. Moreover, for simplicity, we assume that each intruder transmits at a fixed power (which may be different for different intruders). Thus, in our context, a configuration is simply the set of (location, transmit power) pairs of potential intruders. We assume a bounded number of intruders. We use H_0 to represent the hypothesis with no intruders. See Figure 2.2.

Figure 2.2: Illustration of a hypothesis formed of three transmitters.

If there is only one intruder, then each hypothesis represents the location and transmit power combination of the intruder, and determining the hypothesis is equivalent to localizing the intruder and estimating its power. If we allow multiple intruders at a time, the number of possible hypotheses can be exponential in the number of intruders; we will address this challenge in §2.3.

Inputs. For a given set of sensors deployed over an area, we assume the following available inputs, obtained via a priori training, data gathering and/or analysis:

- Prior probabilities of the hypotheses, i.e. $P(H_i)$, for each hypothesis H_i. Prior probabilities come from known knowledge about the area, intruder's behavior, etc., and can be assumed to be uniform in absence of better knowledge.

- Joint probability distribution (JPD) of sensors' observations for each hypothesis. More formally, for each hypothesis H_j, we assume $P(\mathbf{x}|H_j)$ to be known for each observation $\mathbf{x}$ for the set of deployed sensors. The JPDs can be obtained from prior training, a combination of training and interpolation (§2.3.3), or even by assuming a propagation model to remove the training cost completely.

Maximum a Posteriori (MAP) Localization Algorithm. We use Bayes rule to compute the likelihood probability of each hypothesis, from a given observation vector

x:

$$P(H_i|\mathbf{x}) = \frac{P(\mathbf{x}|H_i)P(H_i)}{\sum_{j=0}^{m} P(\mathbf{x}|H_j)P(H_j)} \qquad (2.1)$$

We select the hypothesis that has the highest probability, for given observations of a set of sensors. That is, the `MAP` Algorithm returns the hypotheses based on the following equation:

$$\arg\max_{i=0}^{m} P(H_i|\mathbf{x}) \qquad (2.2)$$

The above `MAP` algorithm to determine the prevailing hypothesis is known to be *optimal* [77], i.e., it yields the minimum probability of (misclassification) error. The above hypothesis-based approach to localization works for arbitrary signal propagation characteristics, and in particular, obviates the need to assume a propagation model. However, the above `MAP` algorithm does incur a *one-time* training cost to construct the JPDs.

2.3 `MAP`* Optimizing `MAP` for MTL

The `MAP` algorithm of §2.2.2 can be directly applied to localize multiple intruders with optimal localization accuracy. However, `MAP` incurs prohibitive computational costs, especially for a large number of potential intruders. In particular, note that if there are L potential locations, up to T potential intruders, and W possible discrete transmit-power levels, then the hypotheses-driven `MAP` algorithm needs to consider $(LW)^T$ hypotheses—making its runtime complexity exponential in the number of potential intruders, and thus, making it impractical to localize even a moderate number of intruders present simultaneously. In addition, `MAP` also incurs a high training cost. In the following subsections, we develop an optimized algorithm called `MAP`* based on `MAP` but with significantly improved computational and training cost. We start with optimizing the computation cost in §2.3.1. In the following subsection §2.3.2, we derive a closed-form expression to efficiently estimate the intruder's power in the *continuous* domain. Finally, we discuss optimizing the training cost via a novel interpolation scheme `ILDW`.

2.3.1 Optimizing Computation Time

Basic Idea. Note that the `MAP`'s exponential time complexity is due to the exponential number of *combinations* of locations and/or powers of the potential intruders. To motivate our proposed optimized approach, consider a simple example of 2 intruders with fixed power p in a large area. Assume that the "transmission radius" r for power p is much smaller than the area; we define the *transmission radius* as the range till which the received signal is more than a certain noise floor. The key observation is that if the intruders are far away (isolated) from each other (specifically, more than $2r$ distance away), then they could be localized independently. If the intruders are closer, then there is a need to separate aggregated signal at some of the sensors and hence we must apply the standard `MAP` algorithm *within that "subarea"*; however, since

each such subarea is small (a disk of $2r$ radius around each possible location), the computation time is reduced significantly. However, since we do not a priori know the configurations of intruders, we need to consider appropriate possibilities.

In essence, our optimized approach is a divide-and-conquer approach, consisting of a sequence of two procedures each of which is executed iteratively. The first procedure focuses on localizing "isolated" intruders (if any) independently, while the second procedure localizes the remaining intruders—by considering all possible subareas as suggested above. The challenge lies in modifying the MAP algorithm for each iteration of the above procedures—as the hypotheses to consider across iterations of the procedures are not disjoint. We now describe each of the procedures.

Procedure 1. Localize Isolated Intruders. Informally, in this procedure, we localize intruders that are sufficiently separated from other intruders. In other words, we localize intruders x that are surrounded by sensors that receive most of their received power from x. More formally, we localize an intruder x at location l if (i) l's "neighborhood" has at least 3 sensors that receive most of their power from x, and (ii) there are no other intruders in the "vicinity" of l. In essence, we iterate over all locations l, and localize an intruder at l if the above conditions are satisfied with high enough probability, based on the readings of sensors around l. The precise definition of neighborhood above must depend on x's transmission radius which depends on its transmit power; however, as x's transmit power is unknown, we iterate over smaller and smaller neighborhoods.

We now formally describe the procedure. Let R_p denote the transmission radius for a transmit power of p. Let R denote the maximum transmission radius, i.e.,

$$\max_p R_p.$$

In the below description, we use a fractional value f to define a neighborhood and vicinity size. We start f equal to 1, use a disk of radius $f R_p$ as a neighborhood and $R + f R_p$ as the vicinity, and iterate over the procedure for reduced values of f.

(a) Let $f = 1$.

(b) For each location and power pair (l, p), compute $P(\mathcal{H}_{l,p} | \mathbf{x}_{l,p})$ using a form of Equation 2.1 over appropriate JPDs. Here:

- $\mathcal{H}_{l,p}$ represents the hypothesis that an intruder is at location l and using p transmit power. We also implicitly assume that there is no other intruder present within a distance of $R + f R_p$ from l; this ensures that the observations in $\mathbf{x}_{l,p}$ are only due to the intruder at l. See Figure 2.3.

- $\mathbf{x}_{l,p}$ represents the observation vector for all sensors, but the sensors that are within a radius of $f R_p$ around l use an observation of "residual" received powers, as defined below, while the remaining sensors (outside the radius of $f R_p$ around l) use an observation of the "noise floor" (in essence, we are "zeroing" the observations of the far-away sensors). See Figure 2.3.

(c) Denote (l, p) pairs that have $P(\mathcal{H}_{l,p} | \mathbf{x}_{l,p})$ higher than a certain threshold as *peaks*. If a location l is a peak and there are no other peaks within a distance of $R + f R_p$, then **localize an intruder at l with transmit power p.**
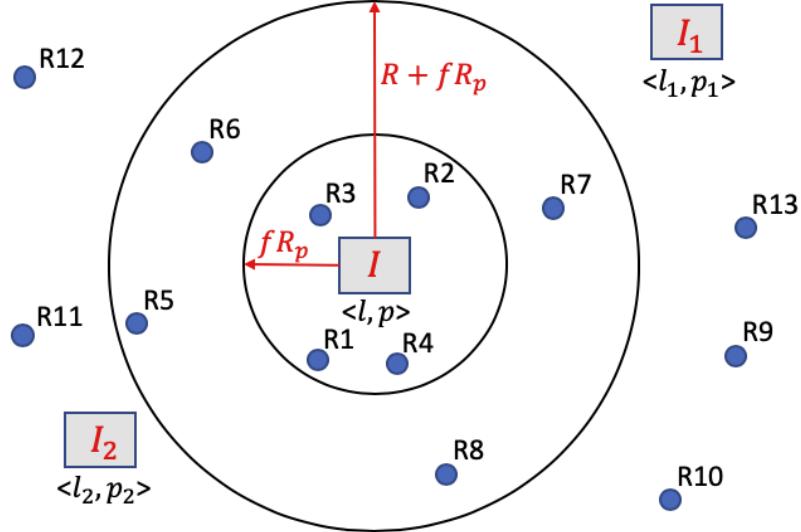
Figure 2.3: Illustration of Hypothesis $\mathcal{H}_{l,p}$ in Step (b) of Procedure 1. Here, the intruder I at location l is transmitting at power p, with no other intruder within a distance of $R + fR_p$ from I. The observation vector $\mathbf{x}_{l,p}$ consists of residual received powers from $R1$ to $R4$, and "noise floor" from the remaining sensors.

(d) For each sensor s, define its *residual received power* (RRP) as the total received power reduced by the sum of mean powers received from already localized intruders; the desired mean values are available from the given JPDs.

(e) Reduce f and go back to step #2 above, unless no new intruders were localized in (c) above. In our experiments, we used $f = 1, 1/2, 1/4$ and $1/8$.

The above procedure is partly inspired by the recent localization work [193]. However, instead of discarding sensors based on their individual power and clustering the rest as in [193], we "discard" sensors based on their neighborhood readings (i.e., likelihood $P(\mathbf{x}|H_i)$ values) and then "cluster" the remaining sensors. Also, we "cluster" iteratively, for smaller and smaller neighborhoods.

Procedure 2. Localize Intruders Situated Close-By. Once we have localized separated intruders as above, we now localize the remaining intruders, if any, by applying the general MAP algorithm independently over "subareas" that still have some sensors with high-enough RRP (residual received power), but no intruder localized in the "vicinity." Formally, the procedure is as follows. Let T be the maximum number of intruders allowed within a disk of radius R, the maximum transmission radius.

(a) Let s be the sensor with the highest RRP; if s's RRP is below a certain threshold (tantamount to noise), then quit.

(b) For $t = 2$ to T: Use MAP (from §2.2.2) to try to localize t transmitters within a disk of radius R around s, using observations of sensors within a radius of $2R$ from s. We use a certain threshold for a posterior probability, in a similar way as for Procedure 1.

(c) Update RRP of each sensor, and go to step (a) above.

Time Complexity. The worst-case time complexity of the first procedure is $O(LWG_R \log(G_R))$, where L and W are the number of potential locations (total grid cells) and transmit power levels respectively, and G_R is the maximum number of grid cells within a transmission range of an intruder. Here, the first term $O(LWG_R)$ is the time to compute the likelihood values in each iteration, since the number of sensors involved in each computation is at most G_R. Note that the number of iterations is bounded by $\log(G_R)$, as f is reduced by a constant multiplicative factor. The worst-case time complexity of the second procedure is $O(G_R(G_R)^T)$ where T is the maximum number of intruders allowed/possible in a transmission region (i.e., a circle of radius at most R). Thus, the overall time complexity of the above localization algorithm is $O(L.W.G_R.\log(G_R) + G_R.(G_R)^T)$. Generally, we would expect T to be a small constant, as more than 3 intruders in a R-radius region with a R transmission range would interfere with each other. If we also consider G_R as a small constant, the overall time complexity can be considered to be $O(L.W)$. In the following subsection, we further reduce the time complexity by removing the factor of W.

2.3.2 Intruder Power Estimation in the Continuous Domain

In this subsection, we derive a *closed-form* expression to estimate an intruder's power in the continuous domain, for the special case of single intruder and Gaussian probability distributions [98]. The derived result essentially removes the assumption of discrete power levels, and reduces the number of hypotheses to consider by a factor of W. We use this result within Procedure 1 of the previous subsection to further optimize its time complexity and performance.

Estimating Intruder Power, Given a Location. Consider the special case of a single intruder in an area. In this case, each hypothesis can be represented as $\mathcal{H}_{l,p}$, for each location l and power p of the potential intruder. Let us focus on a particular location l^* and the corresponding hypotheses $\mathcal{H}_{l^*,p}$. For a given observation vector $\mathbf{x}$, we wish to estimate the power P that corresponds to the hypothesis with maximum likelihood among the hypotheses $\mathcal{H}_{l^*,p}$.

$$P = \arg max_p P(\mathcal{H}_{l^*,p}|\mathbf{x})$$

The value P can be computed by computing $P(\mathcal{H}_{l^*,p}|\mathbf{x})$ for each p, but our goal is to derive a closed-form expression for P from the given JPDs; such an expression yields power estimate in continuous domain without computing $P(\mathcal{H}_{l^*,p}|\mathbf{x})$ for each possible discrete p.

For each sensor (location) j, let $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*})$ represent the probability distribution (PD) of j's observations $\mathbf{x}_j$ when the intruder is at l^* transmitting with power p^*, the power used at training. For a fixed l^* and p^*, the set of PDs $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*})$ are equivalent to the JPDs defined in §2.2 under the assumption of conditional independence[2]. Let us assume that the above PDs are Gaussian distributions [98], and thus, can be

---

[2]PD $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p})$ can be computed $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*})$ for any p, as the path-loss can be assumed to be independent of the transmit power, and JPD $\mathcal{P}(\mathbf{x}|\mathcal{H}_{l^*,p})$ can be computed as product of PDs $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p})$ due to the conditional independence assumption.

represented as $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*}) = N(\mu_j, \sigma_j^2)$ for a given l^* and p^*. In the above setting, the power value P that maximizes $P(\mathcal{H}_{l^*,p}|\mathbf{x})$ can actually be derived as a closed-form expression; we state the result formally in the below lemma.

Lemma 1. *Consider the special case of a single intruder in an area. For a specific location l^* and power p^* (the only power used during training), let $\mathcal{P}(\mathbf{x}_j|\mathcal{H}_{l^*,p^*})$ represent the PDs of the sensor observations at location j. Now, given the above PDs for various j and an observation vector $\mathbf{x}$, the power value $P = \arg max_p P(\mathcal{H}_{l^*,p}|\mathbf{x})$ is given by:*

$$p^* + \frac{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}(x_j - \mu_j)}{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}},$$

where $\gamma = \prod_{j=1}^{S} \sigma_j^2$ and S equals the number of sensors in the neighborhood of l^*. ∎

The proof is in Appendix 6.6. Here, we give its intuition based on a special case. Consider the special case wherein each σ_j is 1 for all j. In this special case, the Lemma's equation reduces to $P = p^* + \frac{\sum_{j=1}^{S}(x_j - \mu_j)}{|S|}$, which implies that if each observation x_j is c more than its mean μ_j then P is also c more than p^*. We note that the above result does *not* extend to the case of multiple intruders. In short, the proof is a process of solving maximum likelihood estimation and multiple intruders introduce transcendental functions, thus cannot derive a closed-form solution.

Use of Lemma 1 in `MAP`***.** For localization of multiple intruders, Lemma 1 can only be used in Procedure 1 of §2.3.1, due to its assumption of a single intruder. In particular, we can Procedure 1 of §2.3.1 as follows.

- We replace R_p by R, the maximum transmission radius.

- For each location l, using Lemma 1, we first compute the power $p(l)$ such that the hypothesis $\mathcal{H}_{l,p(l)}$ has the most likelihood (among the hypotheses at l) using the observations from sensors within a radius of R.

- Then, in the rest of the procedure, we only consider the (location, power) pairs of the type $(l, p(l))$ for any l.

The rest of the Procedure 1 remains unchanged. The above change has two benefits. First, the powers predicted in Procedure 1 are now continuous rather than discrete. Second, the above removes the factor of W from the time complexity of `MAP`* and reduces it to $O(LG_R \log(G_R) + G_R(G_R)^T)$ which becomes $O(L)$ if we consider G_R and T to be relatively small constants.

2.3.3 `ILDW`: Optimizing Training Cost

As in supervised machine learning algorithms, our Bayesian approach also needs training data. We use the term *training* to denote the process of collecting data and building up the JPDs for the hypotheses. Note that this training phase is done

only one-time,[3] and hence, a certain cost is acceptable. The training cost incurred during such data gathering depends greatly on the exact mechanism used for such purposes, e.g., drones with appropriate routes can be used to gather such data [205]. In general, the cost of training would depend on the number of JPDs that need to be constructed, with the cost reduced with a reduction in the number of JPDs needed. In this subsection, we design effective *interpolation* schemes that are useful in reducing the number of JPDs gathered which in turn will reduce the overall training cost. Note that reduction in JPDs constructed from raw data is bound to negatively impact the accuracy—we will evaluate this trade-off in our evaluations and show that impact on accuracy is minimal even with a significant reduction in training cost.

Probability Distributions. First, we note that making the following reasonable assumptions and observations can greatly reduce the number of JPDs/PDs to be constructed.

- If we assume conditional independence of sensor observations, then JPDs can be computed from independently constructed probability distributions (PDs) of received powers at *individual sensors.*

- Since received power at a sensor location x due to multiple transmitters is merely a sum of received powers [130, 213] due to individual transmitters, we can compute PD at x for a particular hypothesis involving a set S of intruders from PDs due to each individual intruder in S.

- Lastly, we need to only construct a PD for one transmit power for each transmitter and sensor location pair, since path loss is independent of transmit power.

Based on the above observations, if there are L discrete locations in an area for sensors or intruders, then a `MAP`-based approach requires L^2 PDs. Below, we propose to minimize the number of PDs to be constructed via data gathering/training, by estimating the remaining unconstructed PDs via interpolation.

Minimizing Training Cost with `ILDW`. Consider a particular location l^* of a potential intruder. Our eventual goal is to compute the PD for each of the L possible sensor locations for this location l^* of a potential intruder; a PD may be computed either by constructing it directly from gathered sensor observations or by estimation via interpolation from the constructed PDs. In particular, for effective interpolation, we construct PDs at coarser-grid sensor locations and estimate via interpolation the PDs at the remaining finer-grid locations. See Figure 2.4. The exact coarseness at which the PDs are constructed is determined by the accuracy of the interpolation scheme for a given area and/or the impact on localization accuracy due to estimated PDs. Below, we describe the interpolation scheme that we use for our purposes.

---

[3]JPDs depend on the channel state and hence, must be updated periodically to account for any changes in the environment (e.g., terrain, buildings, etc.); however, such environment changes are infrequent. Also, note that the online-training of §2.4 is done repeatedly, but only for specific sensors and authorized users, and thus incurs minimal cost. See [302] for spectrum sensing in both spatial and temporal domains.
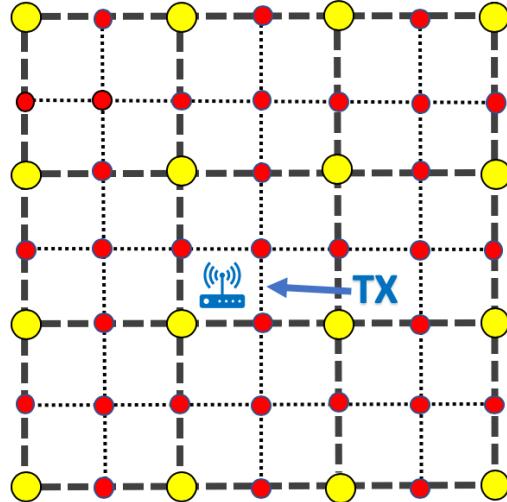
Figure 2.4: Training for PDs at coarse-grained locations (yellow bigger dots), while estimating PDs using interpolation at the remaining fine-grained locations (red smaller dots).
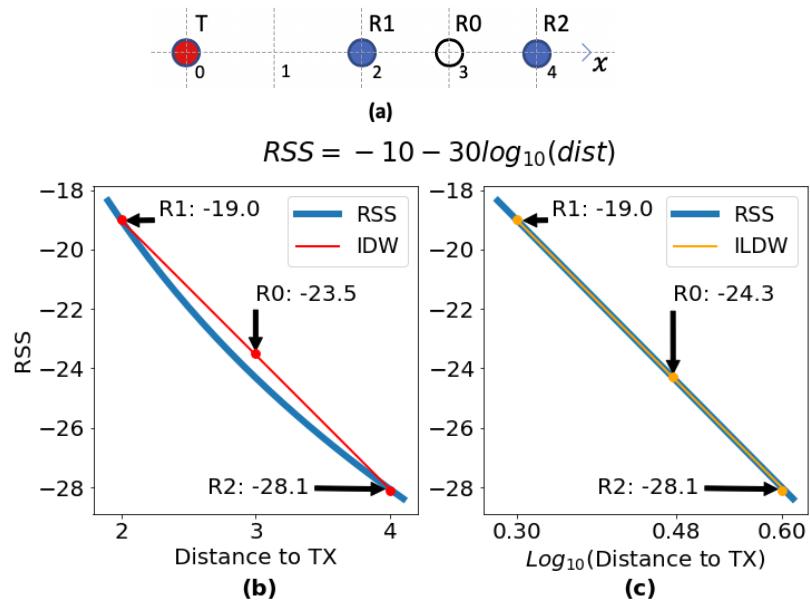
Figure 2.5: Illustration of `ILDW` vs. IDW. (a) Transmitter (T), points with known (R1 and R2) and unknown (R0) received signal strength (RSS) values. (b) Log-normal RSS function (= -10 - 30log$_{10}$(distance)) plotted for varying distance from the transmitter T, along with IDW-estimated RSS value at a point between R1 and R2. (c) Log-normal RSS function and ILDW-estimated RSS value at a point between R1 and R2, plotted on a logarithmic distance scale.

`ILDW` Interpolation Scheme. Consider a fixed transmitter location l^*, and let us assume locations $R_1, R_t, \cdots, R_n$ for which we know the path loss from l^*. Now, consider a new point R_0 for which we wish to estimate the path-loss from l^*. This is a traditional interpolation problem and well-known schemes such as inverse distance weighting (IDW), Ordinary Kriging (OK), k-NN, etc. have been evaluated even in the special context of signal strength or received power [38]. However, our specific context has a unique element. We *know* the location l^* of the transmitter from which the path-loss is being estimated—as we are in the training phase wherein we are gathering observations with the transmitter at l^*. In light of the above unique element of our setting, and the observation of wireless signal characteristics, we use a custom interpolation technique which is a nontrivial modification of the IDW scheme, called *inverse log-distance weighting* (`ILDW`). The traditional IDW interpolation scheme estimates the path loss at R_0 by taking a weighted average of the path losses at $R_1, R_t, \cdots, R_n$, with the weight being the inverse of the distance from R_0.

In our proposed `ILDW` scheme, we still estimate the path loss at R_0 as a weighted average of values at R_i's, but assign weights differently. In particular, we assign the weight for the point R_i as the inverse of the "distance" between R_0 and R_i in the domain where each point is represented merely by its logarithmic distance from l^*, the known transmitter's location—i.e., each point R_i is mapped to a point $\log d(R_i, l^*)$ on a line. This mapping is motivated by the expectation that the actual path loss would be somewhat similar to the log-distance path loss. Thus, the weight for the point R_i is assigned to be

$$w_i = \frac{1}{|\log d(R_i, l^*) - \log d(R_0, l^*)|},$$

where $d()$ is the Euclidean distance function and the path loss at R_0 is estimated as:

$$\mathbf{u_0} = \frac{\sum_{i=1}^{n} w_i \mathbf{u_i}}{\sum_{i=1}^{n} w_i},$$

where $\mathbf{u_i}$ denotes the path loss at point R_i from l^*. In the above equation for weights, if the denominator is zero, then we assign w_i to be equal to the maximum of the weights among the given points (and if all denominators are 0, each weight is assigned to be 1). For an illustration of the above scheme, see Figure 2.5. In the IDW scheme, R_1 and R_2 will get equal weights, but under the `ILDW` scheme they will get weights of 5.57 and 8.00 respectively. More importantly, it can be easily shown that, for log-distance path loss, `ILDW` estimates the path loss for R_0 accurately from two unknown points R_1 and R_2, if $d(R_1, l^*) < d(R_0, l^*) < d(R_2, l^*)$.

The above discussion has been on using `ILDW` for estimating path-loss values. In general, it can be easily used to estimate PDs from the PDs at neighboring points—essentially, we can use `ILDW` to estimate both the mean and standard deviation of a Gaussian PD from other means and standard deviations respectively.
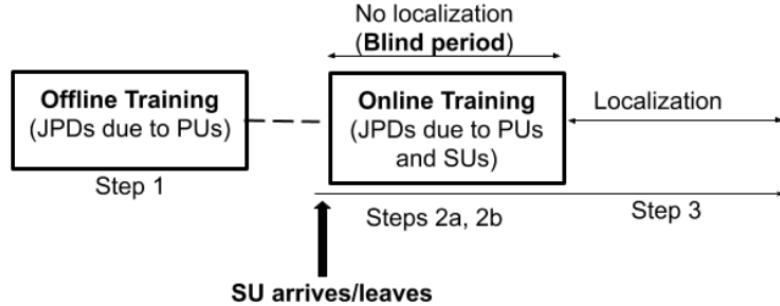
Figure 2.6: `MAP**`'s overall approach

2.4 `MAP**` Localizing in Presence of Authorized Users

We have implicitly assumed till now that the only transmitters present in the area are the intruders which need to be localized. In this section, we adapt our `MAP*` approach described in the previous section to the setting wherein there may be authorized transmitters in the background and the localization technique must take their presence into account. In particular, in a shared spectrum paradigm, there are primary users and an evolving set of active secondary users transmitting in the background. The key challenge comes from the fact that the set of authorized users is not static and changes over time as allocation requests are granted and/or active secondary users become inactive over time.

One simple way to handle background users is to just localize every transmitter, and then remove the authorized users. However, any localization approach (including ours) is susceptible to performance degradation with an increase in the number of transmitters to be localized, especially if some of them are situated close together. Thus, this simple approach of localizing every transmitter is unlikely to be effective, as shown in our evaluations, especially when the number of primaries and active secondaries can be large. Thus, here, we develop an approach based on learning PDs in real-time in response to changes in the set of secondary users.

`MAP**`: Localizing with Authorized Users. Our problem is to localize intruders in a shared spectrum system with fixed primaries and changing set of secondaries. Our `MAP**` approach uses a combination of a priori (offline) and online training to construct JPDs for appropriate hypotheses based on gathered observations, and then use these JPDs to localize intruders in real-time using the `MAP*` approach described in the previous section. We start with defining a few useful notations.

We use $\mathcal{R}$ to denote the set of (fixed) primaries, and $\mathcal{K}$ to denote the set of secondaries at a given instant, and $\mathcal{I}_j$ to denote the jth configuration of intruders (we can assume the zeroth configuration to represent no intruders). We use $\tau = \mathcal{R} \cup \mathcal{K} \cup \mathcal{I}_j$ to denote the set to all transmitters (authorized and unauthorized) at a given instant. Finally, we use $\mathcal{P}(\mathbf{x}|(\tau = X))$ to denote the joint probability distribution (JPD) of observation vectors from the deployed sensors when the prevailing hypothesis is that the set τ of transmitters is X. `MAP**` is the sequence of the following steps.

1. (Offline Step.) Construct JPDs $\mathcal{P}(\mathbf{x}|\mathcal{R})$ and $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{I}_j \cup \mathcal{R}))$ for all j. Since

these JPDs are independent of the secondaries, they do not change and can be done once a priori.

2. (Online Steps.) Whenever $\mathcal{K}$ (set of secondaries) changes:
   (a) Construct JPD $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{R} \cup \mathcal{K}))$.
   (b) Compute $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{R} \cup \mathcal{I}_j \cup \mathcal{K}))$ for all j, from above constructed JPDs, viz., $\mathcal{P}(\mathbf{x}|\mathcal{R})$, $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{I}_j \cup \mathcal{R}))$, and $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{R} \cup \mathcal{K}))$. See the below observation.

3. (Real-time Localization.) Periodically, each sensor sends its observation to a centralized entity (spectrum manager) which uses MAP* to localize any intruders present. Here, localization essentially means determining the most likely prevailing hypothesis among the hypotheses $\tau = (\mathcal{R} \cup \mathcal{I}_j \cup \mathcal{K})$, based on the JPDs $\mathcal{P}(\mathbf{x}|\tau = (\mathcal{R} \cup \mathcal{I}_j \cup \mathcal{K}))$ constructed in earlier steps.

Note that steps 1 and 2a are essentially learning the authorized users' signal charecteristics and view them as the "background signals". If there are no authorized users, then the background signals are "quiet". Else, then the background signals have some "sound". We now state the observation that forms the basis of JPD computation in step 2b; note that the noise due to sensor's hardware gets duplicated when "adding" two JPDs, but can be easily removed.

Observation 1. *The JPD $\mathcal{P}(\mathbf{x}|(\tau = A \cup B))$ and be computed from JPDs $\mathcal{P}(\mathbf{x}|(\tau = A))$ and $\mathcal{P}(\mathbf{x}|(\tau = B))$. Similarly, JPD $\mathcal{P}(\mathbf{x}|(\tau = A))$ can be computed from the JPDs $\mathcal{P}(\mathbf{x}|(\tau = A \cup B))$ and $\mathcal{P}(\mathbf{x}|(\tau = B))$.*

Blind Period due to Step 2. Note that the steps 2a and 2b construct or compute the JPDs needed for localization, and thus, during their execution, the localization cannot be done. Thus, it is important that the duration of this "blind period" in minimal. Fortunately, step 2b being a simple mathematic computation takes only in the order of milliseconds under efficient implementation, while 2a merely entails gathering a sufficient number of observations to construct the desired JPD which could take anywhere from milliseconds to a few seconds, as an observation takes only a fraction of a millisecond [39].

Mobility of Users and Sensors. We note that MAP* works seamlessly for mobile intruders and sensors, due to the constructed PDs. However, MAP** has the following limitation: the sensors must remain static in between two consecutive online-training periods (i.e., step 2 of above). If a sensor X moves, then either X's observation must be ignored, or that X needs to online-train itself in its new location (and there should be no intruders during this individual online-training phase). Note that active SUs are expected to remain static anyway, as they are allocated spectrum for a specific location.

2.5   Large-Scale Simulation Results

To evaluate our techniques in a large scale area (a few km square), we conducted simulations over a geographic area using path-loss values from the Longley-Rice propagation model generated by open sourse software SPLAT! [179]. We describe the simulation setting below and discuss the results.

2.5.1 Settings

Generating Probability Distributions. To evaluate our techniques over a large area with 100s of sensor nodes, we need to run simulations with an assumed propagation model. We use the well-known Longley-Rice [42] Irregular Terrain With Obstruction Model (ITWOM), which is a complex model of wireless propagation based on many parameters including locations, terrain data, obstructions, soil condition, etc. We consider an area of 4km × 4km in the NY state and use the 800 MHz band for SPLAT! We discretize the area using 40 vertical and 40 horizontal grid lines—yielding 1600 cells each of size 100m × 100m. To generate a probability distribution (PD) at a sensor location x due to a transmitter at location l transmitting at power p^*, we compute the received power at x using transmit power minus path-loss from SPLAT!, and use it as the mean of the probability distribution. For the complete PD, we assume Gaussian distributions and use a standard deviation between 1 and 3, with higher values for pairs (x, l) with smaller distance. As mentioned before, the PD due to multiple simultaneous transmitters can be computed as just a "sum" of the Gaussian distributions due to individual transmitters [130, 213].

Algorithms Compared. For the MTL problem, we compare our MAP* algorithm with SPLOT [130] and CLUS [193] (see §2.2.1). As mentioned before, [194] has been shown to be inferior in performance to both SPLOT and CLUS in their respective works, and thus, not evaluated here. CLUS uses k-means [204] for clustering and needs to be provided with the number of clusters. To do a somewhat fair comparison, we provide CLUS with a *range* of the number of intruders and use the elbow-point method to pick the best number of clusters/intruders. In particular, the range of intruders passed to CLUS is 1 to $2x$, where x is the actual number of intruders present.

Table 2.1: Simulation Evaluation Parameters.

Param.	Value	Description
Q_1'	0.6	Threshold for Procedure 1's hypothesis posterior
Q_2'	0.1	Threshold for Procedure 2's hypothesis posterior
R	1000	Transmission radius when power is p^*, (m)
p^*	30	Transmit power during training, (dBm)
δ_p	2	Range of intruders' power is $[p^* - \delta_p,\ p^* + \delta_p]$

For SPLOT, we use the same set of parameter values as in [130] except that we use the confined area radius to be 800m for our large area setting ([130] only considered small 15m × 15m areas; 800m is roughly the maximum transmission radius in our large-scale setting and other values yielded worse results). Table 2.1 gives the main parameters of MAP* used in our evaluations. Recall that the transmission radius is the distance between the TX and RX for which the RX's RSS is at the noise floor (we use -80dBm).
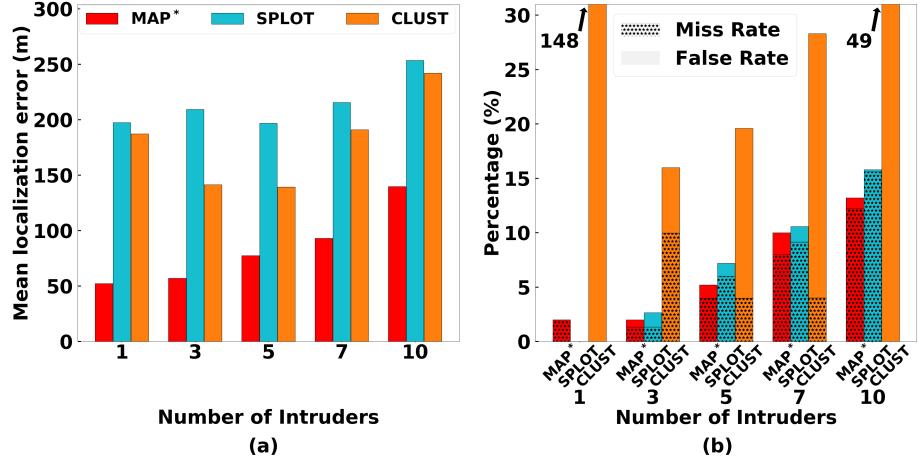
Figure 2.7: Localization performance of various algorithms in a large scale area, for varying number of intruders

2.5.2 Five Evaluation Metrics.

We use the following metrics to evaluate the localization methods.

1. Localization error (L_{err}).

2. Miss rate (M_r).

3. False alarm rate (F_r).

4. Power error (P_{err}).

The above metrics are best explained using a simple example. Given a multi-intruder localization solution, we first compute the L_{err} as the minimum-cost matching in the bi-partite graph over the ground-truth and the solution's locations, where the cost of each edge in the graph is the Euclidean distance. We use a simple greedy algorithm to compute the min-cost matching. The unmatched nodes are regarded as false alarms or misses. E.g., if there are 4 intruders in reality, but the algorithm predits 6 intruders then it is said to incur 0 misses and 2 false alarms and if it predicts 3 intruders then it incurs 1 miss and 0 false alarms. The M_r and F_r metrics are on a per-intruder basis, so in the above two examples: M_r is 0 and 1/4 and F_r is 2/4 and 0. In the plots, we stack miss rate and false alarm rate together to show the overall difference between the true number of intruders and predicted number of intruders. P_{err} is the average difference between the predicted power and the actual power of the matched pair in the above bi-partite graph.

Finally for interpolation schemes, we use the metric (5) interpolation error (I_{err}) defined as the estimated path-loss minus the ground-truth path-loss value.

2.5.3 Results

In this subsection, we evaluate the performance of our techniques for varying parameter values, viz., number of intruders and sensors in the field, and training cost. Here,

the training cost is defined relative (specifically, as a percentage of) to the full training scenario wherein we construct each of the 1600×1600 PDs (one for each pair of transmitter and sensor locations) directly from observations. E.g., $x\%$ training cost indicates that we construct $1600 \times (16x)$ PDs directly, and interpolate the remaining $1600 \times (1600 - 16x)$ PDs; our proposed interpolation scheme only interpolates for sensor locations. In general, when we vary a specific parameter, the other parameters are set to their default values which are: 9% for training cost, 5 for number of intruders, and 240 for number of sensors. For each experiment, the said number of sensors and intruders are deployed randomly in the field, with the intruders deployed in the continuous location domain while the sensors deployed only at the centers of the grid cells. Each data point in the plots is an average of 50 experiments.
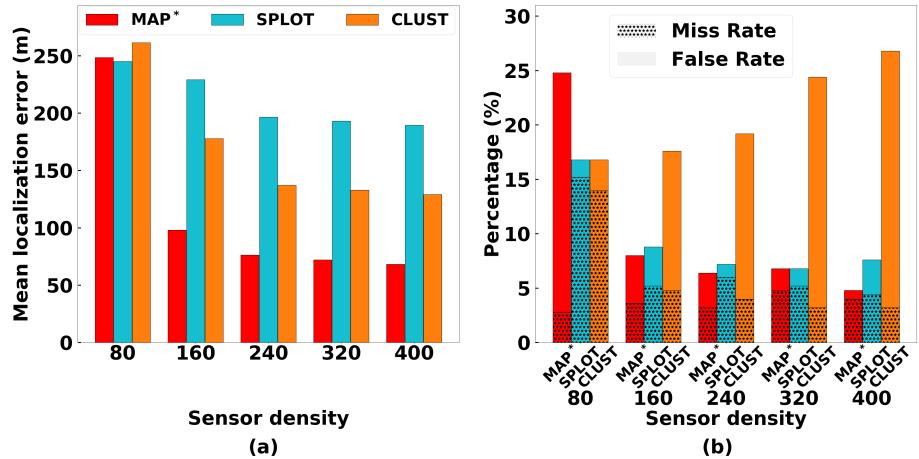
Figure 2.8: Localization performance of various algorithms in a large scale area, for varying sensor density

Varying Number of Intruders. First, we compare the localization accuracy of various algorithms for varying number of intruders. See Figure 2.7. We vary the number of intruders from 1 to 10. We observe that the localization error of MAP* is the minimum across the three algorithms. The localization error is $45\% - 74\%$ less than SPLOT. In terms of the M_r and F_r, MAP* also performs others which confirms the overall performance of MAP* to be the best among the algorithms compared. In terms of absolute performance, note that the localization error of 50-150m indicates an error of 1-2 grid cells, and thus is minimal in the context of the large area of 4km by 4km with 1600 cells and a sensor population of 240. Investigating further, we observe that misses in MAP* are mostly due to the interpolated PDs (note that only 9% of the PDs are constructed from the actual sensor observations and the remaining 91% are interpolated), while SPLOT's misses are mainly from the case of two or more intruders being close to each other. This demonstrates the superior ability of MAP* to localize intruders that are close-by via the designed sequence of Procedures 1 and 2.

Intruder Power Estimation, and Computation Time. Table 2.2 shows the mean absolute error (MAE) and mean error (ME) of the intruder's predicted power by MAP*. Note that CLUS and SPLOT do not predict the intruder's power, and hence, are not

Table 2.2: MAP* Power Error (dB)

# Intru.	MAE	ME
1	0.56	-0.07
3	1.02	0.89
5	1.31	0.97
7	1.52	1.16
10	1.47	1.04

Table 2.3: Running time (s)

# Intru.	MAP*	SPLOT	CLUS
1	0.55	0.56	0.03
3	1.07	1.02	0.11
5	5.74	1.35	0.23
7	8.14	1.63	0.30
10	16.50	1.89	0.41

shown. We observe that MAP* is able to predict intruder's power quite accurately. The errors increase with the increase in number of intruders. Also, the mean error begins at near zero and then turns positive. Table 2.3 shows the running time of various algorithms over an Intel i7-8700 3.2 GHz processor. We see that CLUS is the fastest, and the running times of MAP* and SPLOT are comparable for a small number of intruders, but for a larger number of intruders, MAP* takes longer time than SPLOT mainly because of more number of iterations of the computationally intensive Procedure 2.
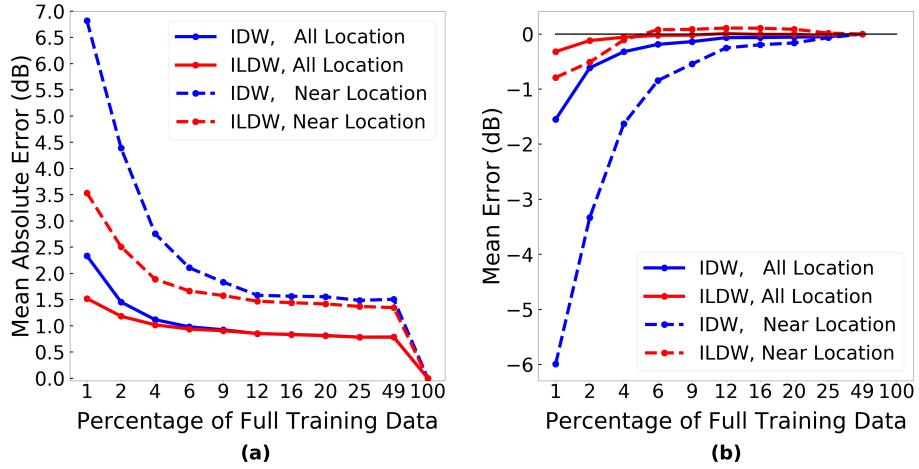
Figure 2.9: Estimation errors for interpolation schemes for varying training data

Varying Sensor Density. We now vary the total number of sensors in the field and observe the impact on the performance of various algorithms. See Figure 2.8, where the number of sensors is varied from 80 to 400. We see that all algorithms perform

better with an increasing number of sensors as expected, with `MAP*` performance improving significantly (in both L_{err} as well as $f_r + m_r$) as the number of sensors is increased from 80 to 160. More importantly, except for a very low number of sensors (i.e., 80), `MAP*` handily outperforms the other two algorithms.

Varying Training Cost. Finally, we now investigate how the training cost (i.e., the number of PDs constructed from raw observations) affects the performance of our `MAP*` algorithm. Note that the other algorithms do not depend on the training data, hence not shown. We first evaluate the interpolation error of our `ILDW` scheme for varying training cost (number of known PDs) by comparing with the traditional `IDW` scheme on which it is based. See Figure 2.9, which plots the mean absolute error (MAE) as well as mean error (ME). As the interpolation error is substantially higher for points that are closer to the transmitter, we plot MAE and ME as averaged over all interpolated points as well as over just the points close (less than 800m away) to the transmitter. Note that the PDs at sensor locations closer to the transmitter would have a stronger bearing on the localization accuracy, and thus, the MAE and ME values for points closer to the transmitter are of more significance. We observe here that as expected both MAE and (absolute value of) ME decrease with an increase in the training cost for both `IDW` and `ILDW`, but MAE and ME of `ILDW` are significantly lower than that of `IDW`, especially for low percentages of training cost and when the points are close to the transmitter.

We now plot the performance of `MAP*` for varying training data; see Figure 2.10. As expected, the performance metrics show general improvement with an increase in the amount of training. More importantly, we note that with 5-10% of training, `MAP*` achieves performance comparable to that with 100% training, suggesting that our interpolation scheme is largely effective as long as 5-10% of PDs are constructed from raw observations.
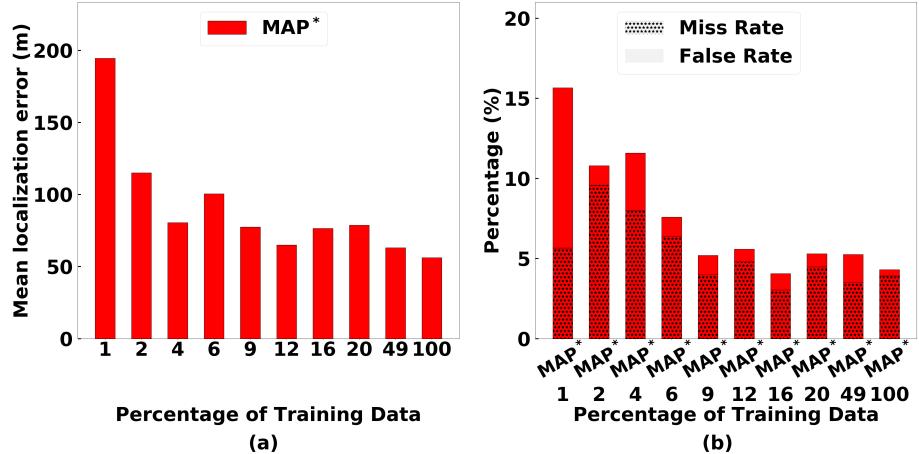
Figure 2.10: Localization performance of `MAP*` in a large scale area, for varying training data

In Presence of Authorized Users (`MAP**`). We now evaluate the performance of our `MAP**` approach which is tailored to work in the presence of authorized users.

To evaluate MAP**, we place 5 authorized users in the area—with 2 primary and 3 secondary users. The primary users are placed at fixed locations, while the secondaries are put at random locations. We assign each authorized user a random power in the range of 30 to 32dBm, while, as before, a random power between 28 and 32dBm to the intruders. To ensure that these 5 authorized users do not "interfere" with each other, we ensure that the distance between any two of these authorized users is at least 1000m. We compare MAP** with the simpler approach called MAP*+ that uses MAP* to localize all transmitters (authorized as well as intruders) and then remove the predicted transmitters that are closest to the authorized users. See Figure 2.11, which shows that MAP** easily outperforms MAP*+ for varying number of intruders.
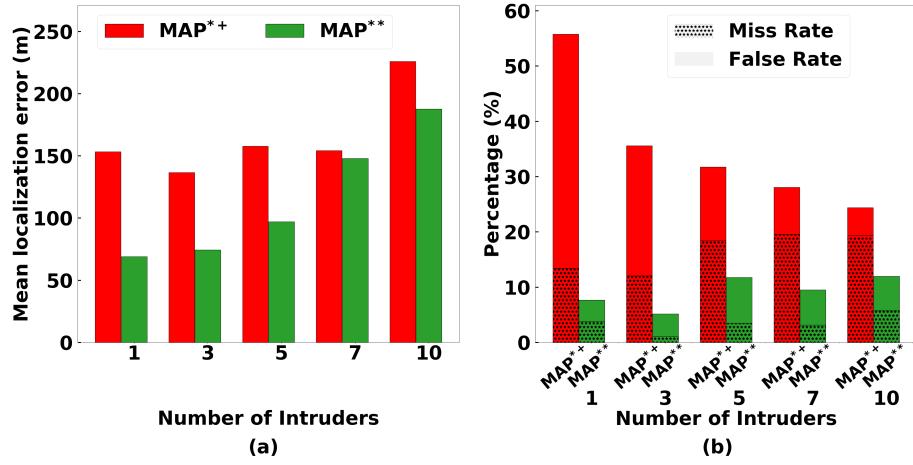
Figure 2.11: Localization performance of MAP*+ and MAP** in large-scale simulations with authorized users present, for varying number of intruders

2.6 Testbed Implementation

In this section, we implement our techniques over commodity devices and evaluate them over two small-scale testbeds—one indoor and one outdoor. Outdoor environment is a realistic setting for our target application of shared spectrum systems, while the indoor environment provides more challenging signal attenuation characteristics due to walls and other obstacles.

Sensor and Transmitters Used. Our low-cost (sub $100, see [64] for a measurement study of low-cost spectrum sensors) sensing device is composed of a single-board computer Odroid-C2 with an RTL-SDR dongle that connects to a dipole antenna. We deploy 18 of these sensing devices in our indoor and outdoor testbeds and configure them for low gain. For transmitters/intruders, we use USRP B210 and HackRF devices powered by laptops; we place these on a cart for mobility. These transmitter devices are uncalibrated, and there is no way to assign a specific transmit power. However, they have a configurable parameter called *gain* which is almost perfectly correlated to power when the gain is in a specific range, i.e., when the transmitter's gain is increased by 1, the receiver's signal strength increases by 1dB. We thus use
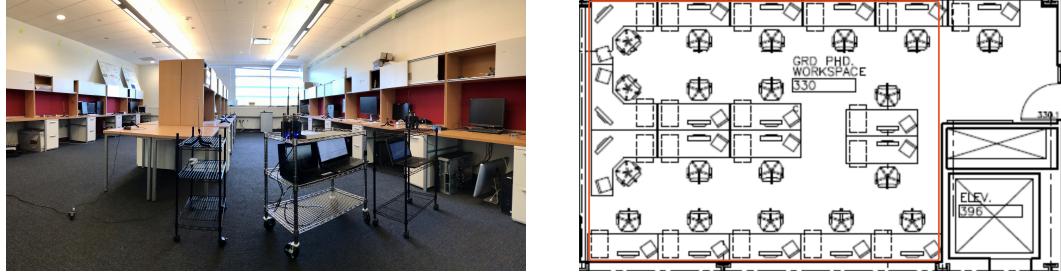
(a) Indoor lab environment

(b) Floor plan

Figure 2.12: Indoor testbed. (a) Our lab used for the indoor testbed, (b) The lab's floor plan.
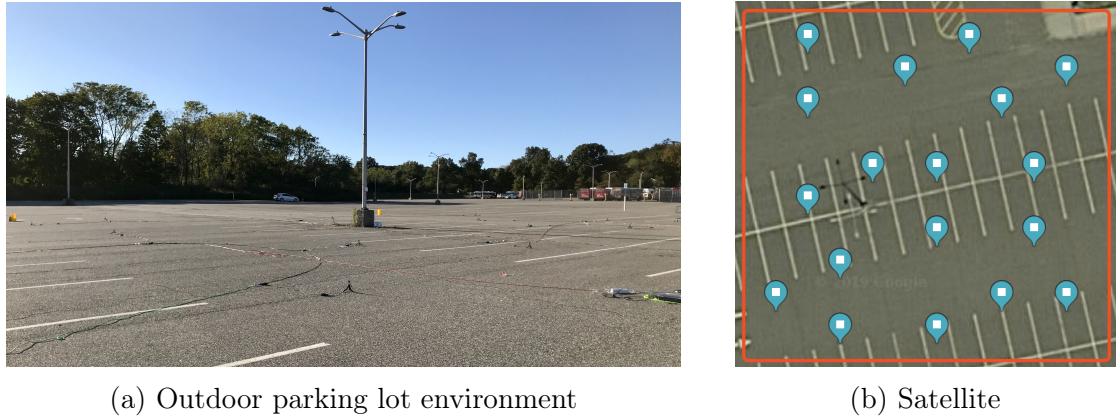
(a) Outdoor parking lot environment

(b) Satellite

Figure 2.13: Outdoor testbed. (a) Parking lot picture, (b) Satellite image of the parking lot; the red box is the area of the experiment, and the stars are the locations of sensing devices during evaluation.

the gain parameter to adjust transmit power in the USRP devices. For indoor experiments, the location is manually derived, while for outdoor experiments, we use GPS dongles connected to the laptops. For collecting sensor observations, we implemented a Python repository in Linux that measures spectrum in real time at 915MHz ISM band and 2.4Msps sample rate. The repository collects I/Q samples fetched from the RTL-SDR dongle and computes the RSS value, then records the RSS along with timestamp and location. These three pieces of information are sent to a server that runs the localization algorithms.

Testbeds. The **indoor** testbed is built in a lab of our Computer Science building. Figure 2.12 depicts the lab with its floor plan. The red box in the floor plan is the area where experiments are conducted. The area is 9.6×7.2 m^2 (or 2177 square feet) large, with four rows of desks. The middle two rows are separated by a wooden board. The area is imagined to be divided into 48 grid cells each of size 1.2m $\times$ 1.2m, with the help of ceiling tiles each of which is 0.6m $\times$ 0.6 m. The **outdoor** testbed is over an open space parking lot. See Figure 2.13. The area is 32m $\times$ 32m. We divide the area into 100 grid cells with each cell representing an area of 3.2m $\times$ 3.2m. The GPS device returns the location in (latitude, longitude) and the program converts it

into coordinates. We use an outdoor WiFi router and long power cords for network and electrical connection respectively. During the evaluation, the 18 sensing devices are placed on the ground and are randomly spread out.

Training. In both the testbeds, for training (i.e., constructing non-interpolated PDs), we first pick 18 random grid cells and place sensors in their approximate centers. Then, we manually move the transmitter around in a cart through each of the grid cells. For the USRP transmitter, we use a gain value of 45 in the indoor environment and 58 in the outdoor testbed. We use a higher gain for outdoors to allow the transmitter to have a larger transmission range in a larger area. With each grid cell, the transmitter transmits from 3 to 4 different points within each grid cell, and for each such location of the transmitter, the sensors (at the 18 picked locations) gather tens of signal strength readings. From these readings, we construct a Gaussian probability distribution from each grid cell location of the transmitter. More specifically, for a particular grid cell location of the transmitter, we average over the readings from multiple TX positions within that particular grid cell—this process of averaging different positions of the TX inside a grid cell makes the Gaussian distributions more robust to multipath fading and shadowing. The overall training process takes an hour for indoors, and about two and a half hours for outdoors.
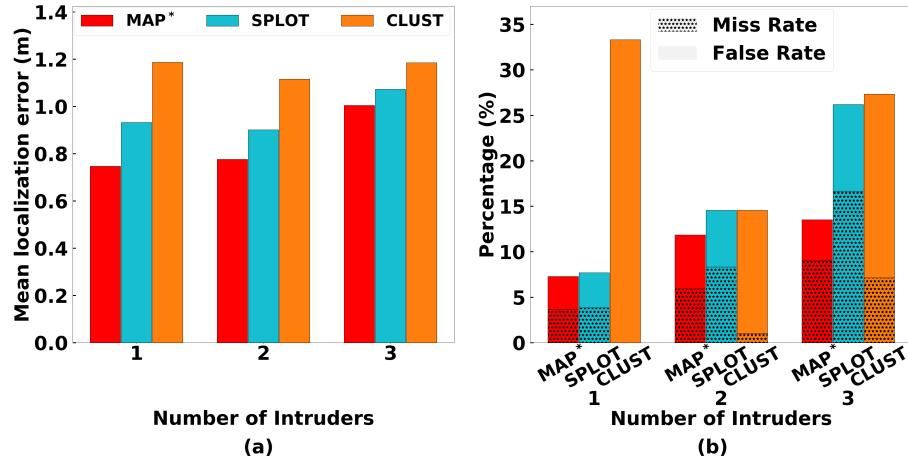
Figure 2.14: Localization performance of various algorithms in an indoor testbed

Evaluation. For evaluation, in both testbeds, we place the 18 sensors at centers of grid cells that are randomly chosen and are different from the cells chosen for the training above. The chosen locations for the outdoor testbed are shown in Fig. 2.13(b). We choose the intruder's gain/power to be in the range of $[p^* - 1, p^* + 1]$, where p^* is the gain/power used during the training phase as mentioned above. Roughly half of our experiments involve close-by (in the same or adjacent grid cells) intruders. Localization is done on a laptop which listens to HTTP requests containing the sensors' observations.
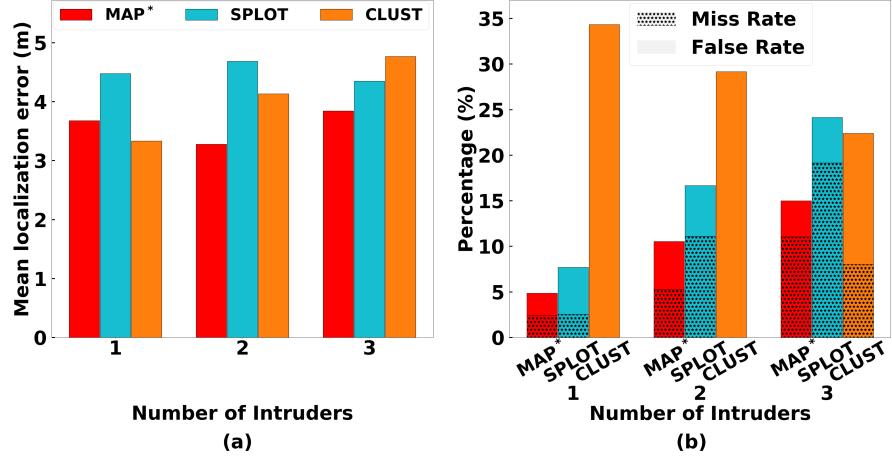
Figure 2.15: Localization performance of various algorithms in an outdoor testbed

2.6.1 Results

Localization Metrics. Figure 2.14-2.15 show the localization results for the indoor and outdoor testbeds respectively. Overall, the results indicate that `MAP*` performs the best across all metrics, with the overall performance gap between `MAP*` and `SPLOT` increasing with the increase in number of intruders. When the number of intruders is 3, the performance of `SPLOT` is significantly worse than `MAP*` due to a significantly higher (84% for indoors and 53% for outdoors) sum of miss and false-alarm rates and 43% higher localization error. The `CLUS` algorithm generally performs the worst, but its performance doesn't have a strong correlation with the increase in the number of intruders; recall that `CLUS` is given the range of number of intruders as an extra piece of information compared to the other algorithms. In terms of absolute performance, we see that the localization error of `MAP*` is roughly around 1 or less grid cell, and the sum of miss-rate and false-alarm is between 5-15%.

Table 2.4: Interpolation Mean Absolute Error (MAE) and Mean Error (ME) in dB for `IDW` and `ILDW`

Environment	IDW (MAE)	ILDW (MAE)	IDW (ME)	ILDW (ME)
Indoor	2.6	1.7	1.7	0.25
Outdoor	6.2	2.7	5.8	0.48

Interpolation Error. Table 2.4 shows the interpolation mean absolute error (MEA) as well as mean error (ME) of `IDW` and `ILDW` when the transmitter and receiver are close by (i.e., within a distance of 3 grid cells). When the transmitter and receiver are far away, the difference between `IDW` and `ILDW` is small and thus not shown. We see that when compared with `IDW`, our `ILDW` interpolation scheme decreased the mean absolute error by 35 percent in the indoor environment and 56 percent in the outdoor environment. In terms of mean error, `ILDW` reduced the error compared to `IDW` by

as large as 86 percent and 92 percent respectively. This is because `IDW` mostly tends to estimate the value to be larger than the ground truth, while `ILDW`'s estimates are more even across the ground truth.

Table 2.5: Power Prediction Mean Absolute Error (MAE) and Mean Error (ME) in dB for indoor and outdoor testbed

# Intruder	Indoor (MAE)	Outdoor (MAE)	Indoor (ME)	Outdoor (ME)
1	0.34	0.50	-0.02	0.02
2	0.57	0.63	0.10	0.54
3	0.77	0.90	0.49	0.76

Intruder Power. Table 2.5 shows the errors in the predicted powers of the intruders in `MAP*`. We see that the outdoors have a slightly higher power prediction error, likely because of a larger number of grid cells. We also note that with the increase in the number of intruders, the error in predicted power increases.

2.7   Conclusion

In this chapter, we have developed an efficient Bayesian approach with a noval interpolation scheme to localize multiple transmitters in presence of authorized users, and demonstrate its superior power over large-scale simulations and smaller scale indoor and outdoor testbeds. In our future work, we wish to extend our techniques to allow a continuous location domain and design methods to further minimize training cost. In addition, we will consider alternate signal measurements such as angle-of-arrival (AoA).

Chapter 3

DeepMTL: Deep Learning Based Multiple Transmitter Localization and Power Estimation

In this chapter, we address the problem of Multiple Transmitter Localization (`MTL`). `MTL` is to determine the locations of potential multiple transmitters in a field, based on readings from a distributed set of sensors. In contrast to the widely studied single transmitter localization problem, the `MTL` problem has only been studied recently in a few works. `MTL` is of great significance in many applications wherein intruders may be present. E.g., in shared spectrum systems, detection of unauthorized transmitters and estimating their power are imperative to efficient utilization of the shared spectrum.

In this chapter, we present `DeepMTL`, a novel deep learning approach to address the `MTL` problem. In particular, we frame `MTL` as a sequence of two steps, each of which is a computer vision problem: image-to-image translation and object detection. The first step of image-to-image translation essentially maps an input image representing sensor readings to an image representing the distribution of transmitter locations, and the second object detection step derives precise locations of transmitters from the image of transmitter distributions. For the first step, we design our learning model `sen2peak`, while for the second step, we customize a state-of-the-art object detection model `YOLOv3-cust`. Using `DeepMTL` as a building block, we also develop techniques to estimate the transmit power of the localized transmitters. We demonstrate the effectiveness of our approach via extensive large-scale simulations and show that our approach outperforms the previous approaches significantly (by 50% or more) in performance metrics including localization error, miss rate, and false alarm rate. Our method also incurs a very small latency. We evaluate our techniques over a small-scale area with real testbed data and the testbed results align with the simulation results.

3.1 Introduction

The RF spectrum is a limited natural resource in great demand due to the unabated increase in mobile (and hence, wireless) data consumption [8, 192]. In 2020, the U.S. FCC moves to free up 100 MHz of previously military-occupied mid-band spectrum in the 3.45-3.55 GHz band for paving the way for 5G development. Also, the research and industry communities have been addressing this capacity crunch via the development of *shared spectrum*. Spectrum sharing is the simultaneous usage of a specific frequency band in a specific geographical area and time by a number of independent entities where harmful electromagnetic interference is mitigated through agreement (i.e., policy, protocol) [83]. Spectrum sharing techniques are also normally used in 5G networks to enhance spectrum efficiency [138]. However, protection of spectrum from unauthorized users is important in maximizing spectrum utilization.

The increasing affordability of the software-defined radio (SDR) technologies makes the shared spectrum particularly prone to unauthorized usage or security attacks. With easy access to SDR devices (e.g. HackRF, USRP), it is easy for selfish users to transmit data on a shared spectrum without any authorization and potentially cause harmful interference to the incumbent users. Such illegal spectrum usage could also happen as a result of the infiltration of computer viruses or malware on SDR devices. [138] depicts three cases of spectrum attack. As the fundamental objective behind such shared spectrum paradigms is to maximize spectrum utilization, the viability of such systems depends on the ability to effectively guard the shared spectrum against unauthorized usage. The current mechanisms however to locate such unauthorized users (intruders) are human-intensive and time-consuming, involving the FCC enforcement bureau which detects violations via complaints and manual investigation [130]. Motivated by the above, we seek an effective technique that is able to accurately localize multiple simultaneous intruders (transmitters). Below, we describe the multiple transmitter localization problem.

Multiple Transmitter Localization (`MTL`). The transmitter localization problem has been well studied, but most of the focus has been on localizing a *single* transmitter at a time. However, it is important to localize multiple transmitters simultaneously to effectively guard a shared spectrum system. E.g., a malware or virus-based attachment could simultaneously cause many devices to violate spectrum allocation rules; spectrum jamming attacks would typically involve multiple transmitters. More importantly, a technique limited by the localization of a single intruder could then be easily circumvented by an offender by using multiple devices. The key challenge in solving the multiple transmitter localization (`MTL`) problem comes from the fact that the deployed sensor would receive only a *sum* of the signals from multiple transmitters, and separating the signals may be impossible.

<u>Prior Works.</u> The `MTL` problem has been recently addressed in a few prior works, among which `SPLOT` [130], `MAP` [307], and `DeepTxFinder` [331] are the most prominent. `SPLOT` essentially decomposes the `MTL` problem to multiple single-transmitter localization problems based on the sensors with the highest power readings in a neighborhood. However, their technique implicitly assumes a propagation model, and thus,
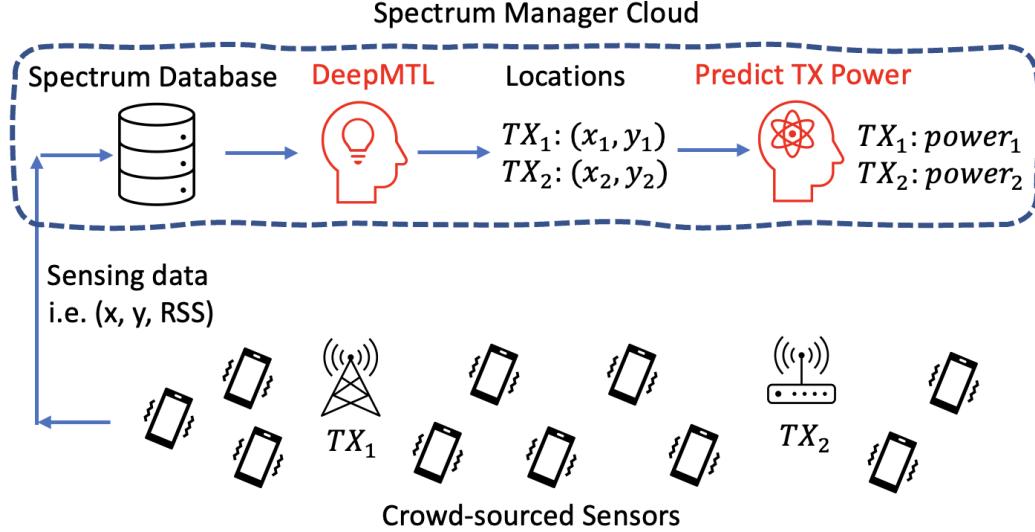
Figure 3.1: Multiple transmitter localization using a distributed set of sensors. Sensing data is uploaded to a spectrum manager server in the cloud. `DeepMTL` is a deep learning approach to multiple transmitter localization which helps protect spectrum against unauthorized usage. After that, the prediction of transmission powers happens using `DeepMTL` as a building block.

may not work effectively in areas with complex propagation characteristics, and it is not effective in the case of transmitters being located close by (a key challenging scenario for `MTL` problem). Our recent work `MAP` solves the `MTL` problem using a hypothesis-driven Bayesian approach; in particular, it uses prior training in the form of distributions of sensor readings for various transmitter locations, and uses the training data to determine the most likely configuration (i.e., transmitters' locations and powers) for a given vector of sensor readings. However, to circumvent the high computational cost of a pure Bayesian approach, `MAP` uses a divide and conquer heuristic which results in somewhat high number of misses and false alarms while still incurring high latency. `DeepTxFinder` uses a CNN-based learning model approach; however, they use a separate CNN model for a specific number of transmitters and thus may incur high model complexity and training costs while also limiting the number of transmitters that can be localized. In our evaluations, we compare our work with each of the above approaches.

`DeepMTL`: **Our Two-Step Approach.** As in prior works [38, 130], we assume a crowdsourced sensing architecture (See Fig. 3.1) wherein relatively low-cost spectrum sensors are available for gathering signal strength in the form of received power. We use a convolutional neural network (CNN) based approach to solve the `MTL` problem. In particular, we frame `MTL` as a sequence of two steps: image-to-image translation and object detection, each of which is solved using a trained CNN model. The first step of image-to-image translation maps an input image representing sensor readings to an image representing the distribution of transmitter locations, and the second object detection step derives precise locations of transmitters from the image of transmitter

distributions. We name our `MTL` approach as `DeepMTL`.

Motivation. Our overall approach and its various aspects are motivated by the following considerations. **First**, we use a learning-based strategy to preclude assuming a propagation model [130] or conducting surveys of sensors reading distributions [307]. The assumption of the propagation model suffers from the fact that even sophisticated propagation models yield unsatisfactory accuracy and thus lead to degraded performance. Among all learning-based strategies, deep learning can implicitly capture the environment characteristics (e.g., objects, walls, landscape) in the neural network layers' weights learned through the training of the data [11]. Even though a learning-based approach incurs a one-time high training cost, it generally incurs minimal latency during inference, which is an important consideration for our `MTL` problem. The intruder detection should incur minimal latency to be effective. **Second**, the geographical nature of the `MTL` problem suggests that convolutional neural networks (CNNs) are well-suited for efficient learning of the desired function. In particular, the features of the `MTL` problem can be represented in an image (2D matrix) corresponding to their geographic locations, which can be fed as an input to an appropriate CNN model that can leverage the spatial correlation among the input features to facilitate efficient learning. **Lastly,** we use a two-step architecture to facilitate efficient training by essentially providing an additional intermediate image. In particular, we are able to map each step to well-studied standard computer vision problems, allowing us to build upon known techniques.

Overall Contributions. The goal of our work is to develop an efficient technique for accurate localization of simultaneously present multiple transmitters/intruders. We also extend our technique to address various extensions such as power estimation and the presence of authorized users. Overall, we make the following contributions.

1. For the `MTL` problem, we develop a novel two-step CNN-based approach called `DeepMTL` approach. For the first step of image-to-image translation, we develop a CNN model that translates an image representing the sensor readings into an intermediate image that encodes distributions of transmitter locations (Section 3.3). For the second step of mapping transmitter distributions to precision locations via object detection, we customize the well-known object detection method YOLOv3 (Section 3.4).

2. For localization of transmitters in the presence of authorized users, we augment the `DeepMTL` model by adding a pre-processing step based on a CNN-model that first reduces the sensor readings by the power received from the authorized users (Section 3.5).

3. To estimate the transmit power of the intruders, we augment our `DeepMTL` model with a power-estimation CNN-model which iteratively estimates the power of transmitters in sub-areas (Section 3.6).

4. We evaluate our techniques via large-scale simulations as well as small-scale testbed data and demonstrate their effectiveness and superior performance compared to the prior works (Section 3.7).

A preliminary version of this paper appeared at IEEE WoWMoM 2021 [304].

3.2   Background, MTL Problem and Our Approach

In this section, we describe the background of the shared spectrum systems, formulate the MTL problem, then describe our methodology.

Shared Spectrum System. In a shared spectrum paradigm, the spectrum is shared among licensed users (primary users, PUs) and unlicensed users (secondary users, SUs) in such a way that the transmission from secondaries does not interfere with that of the primaries (or secondaries from a higher-tier, in case of a multi-tier shared spectrum system). In some shared spectrum systems, the location and transmit power of the primary users may be unavailable, as is the case with military or navy radars in the CBRS band. Such sharing of spectrum is generally orchestrated by a centralized entity called *spectrum manager*, such as a spectrum database in TV white space [131] or a central spectrum access system in the CBRS 3.5GHz shared band [106]. The spectrum manager allocates spectrum to requesting secondaries (i.e., permission to transmit up to a certain transmit power at their location) appropriately so as to avoid interference with primaries. Users who transmit without explicit permission are referred to as unauthorized users or *intruders*; the MTL problem is to essentially localize such intruders.

MTL Problem. Consider a geographic area with a shared spectrum. Without loss of generality, we assume a single wireless frequency[1] throughout this paper[2]. For localization of intruders, we assume available crowdsourced sensors that can observe received signals in the wireless frequency of interest, and compute (total) received signal strength (RSS). RSS can be measured using low-cost sensors and has been shown to achieve good accuracy for single-transmitter localization [14]. In the related work Section 3.8, we will discuss signal metrics other than RSS, such as AoA, ToA, etc. At any instant, there may be a set of intruders present in the area with each intruder at a certain location transmitting with a certain power which may be different for different intruders.

The MTL problem is to determine the set of intruders with their locations at each instant of time, based on the set of sensor observations at that instant. For the main MTL problem, we assume that there are no primary or authorized users, and thus, assume that the sensor readings represent aggregate received power from the transmitters we wish to localize. However, in Section 3.5, we investigate the more general MTL problem where the background primary and/or secondary users may also be present.

---

[1]To avoid confusion with image channels, we use *wireless frequency* instead of the perhaps more appropriate *wireless channel* term.

[2]Multiple wireless frequencies can be handled independently. Note that if we assume the wireless propagation characteristics to be similar for different frequencies, then we do not need to train different models for each of them. Our localization techniques would still work for scenarios wherein the intruders may change their transmit frequencies dynamically.

Sensor Reading Input Image

Translated Image of Peaks (TX)

Peak Detection

Image-to-image Translation

Step 1

Peak–Finding/ Object Detection

Step 2

5 TX at (15, 97), (47, 77), (49, 79), (91, 53), (92, 33)

Intermediate Image

Predicting 5 Bounding Boxes Use Box Center as TX Location
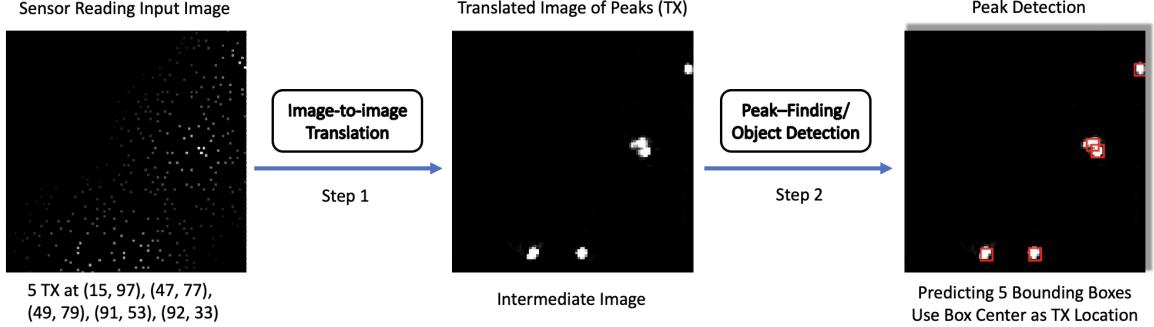
Figure 3.2: The overall two-step CNN architecture of the `DeepMTL` model. The first step is the `sen2peak`, whose higher idea is to translate the input image of sensor readings to the image of peaks where each peak implies a transmitter. The `sen2peak` architecture is illustrated in Fig. 3.4. The second step is `YOLOv3-cust`, a customized version of YOLOv3, to perform object/peak detection in the output image of the first step. This step returns the precise location coordinates of TX. The `YOLOv3-cust` architecture is illustrated in Fig. 3.5. A zoom-in of the peak detection result of the second step is in Fig. 3.6.

Our Approach. In our context, each sensor communicates its observation to a centralized spectrum manager which then runs localization algorithms to localize any potential (multiple) transmitters. We design and implement a novel two-step localization algorithm named `DeepMTL`, as illustrated in Fig. 3.2, based on CNN models. The first step (Section 3.3) is a four-layer image-to-image translation CNN model that is trained to translate an input image representing sensor readings to an image of transmitters' locations distributions. Each distribution of a transmitter can be visualized as a mountain with a peak, so we name this model `sen2peak`. The second step (Section 3.4), called `YOLOv3-cust`, is a customized object-detection method built upon YOLOv3[215] which localizes the objects/peaks in the translated image. The high-level motivation behind our two-step design is to frame the overall `MTL` problem in terms of well-studied learning problem(s). The two steps facilitate efficient learning of the models by supplying an intermediate image with the training samples.

3.3 `DeepMTL` Step 1: Sensor Readings to TX Location Distributions

In this section, we present the first step of our overall approach to the `MTL` problem, i.e., the image-to-image translation step which translates/transforms the sensor reading to distributions of TX locations. Here, we first create a grayscale image to represent the input sensor readings; this image encodes both the sensors' RSS readings and the sensors' physical location. We then train and use a convolutional neural network (CNN) model to transform this input image to an output image that represents the distribution of TX locations. Pixels in the output image that have higher values will have a higher chance of having a TX being present at that location.
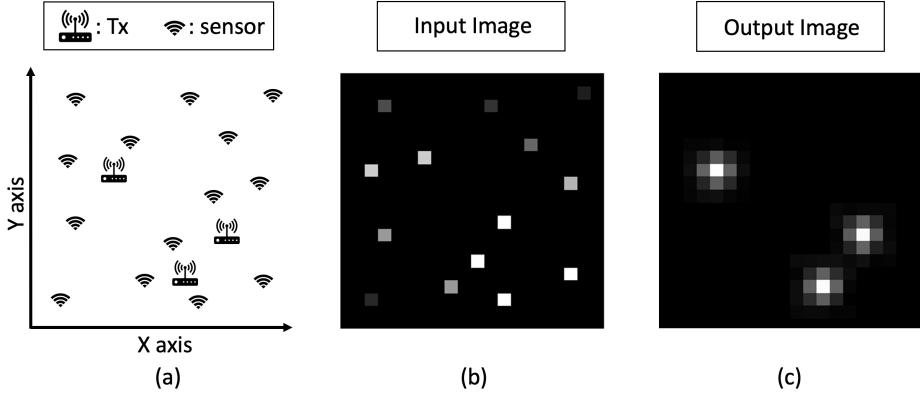
Figure 3.3: Illustration of `DeepMTL` first step's input and output images. (a) Area with distributed sensors and transmitters to be localized. (b) Input image representing the sensor readings (RSS) and locations. (c) Output Image, where we put a 2D Gaussian distribution with its "peak" at the transmitter's location.

Input/Output Image Sizes and Tiling Approach for Large Areas. We need to represent data by images of certain sizes. Typically, an image should be a size of a few hundred pixels by a few hundred pixels, since a thousand pixels by thousand pixels images will consume too much GPU memory. In this chapter, we pick 100×100 as the size for both our input and output images in the first image-to-image translation step. Given an area that we want to monitor and a 100×100 size image, we will know how large an area a pixel will represent and we call it a pixel subarea. A large pixel subarea could certainly lead to high localization errors, due to very coarse granularity. We can address this by using a "tiling" technique, wherein we divide the given area into tiles, then represent each tile by 100×100 size image and use our localization techniques in the tile. We can do some post-processing to handle cross-tiling issues (e.g., [331] uses overlapping tiles and employs a voting scheme inside the overlapping tile area).

3.3.1 Input Image Representing Sensors' Readings

We localize transmitters based on observations from a set of sensors, i.e. solve the `MTL` problem assuming only intruders. The input of the localization method is sensor observations. Here, an *observation* at a sensor is the received power (RSS, in decibels) over a time window of a certain duration, in the frequency of interest (we assume only one wireless frequency). RSS is computed using FFT over the I/Q samples collected in a time window. More specifically, in our evaluations, we use a Python API [120] that computes the power spectral density from a sequence of signal data (I/Q samples), and then, we choose the RSS at the frequency of interest. Different than [130, 307], we represent the sensor information, i.e., their locations and observations, in a 2D input image. We use a 2D grayscale image and let us denote it $\mathbf{X}$. The pixel $\mathbf{X_{i,j}}$ denotes the observation of the sensor at the grid cell whose index is (i, j). For example, $\mathbf{X}_{10,20} = -50$ denotes there is a sensor at coordinate $(10, 20)$ with an RSS reading of

39

-50 dB. If there is no sensor at location (i, j), we assign the noise floor $\mathcal{N}$ (i.e. -80 dB) value to $\mathbf{X_{i,j}}$. Note that the above pixel values (representing the sensor observations) are not the standard image pixel values that lie in the $[0, 255]$ range. Also, since the path loss computed by propagation models during simulations could be real numbers, the sensor observation values could be real numbers. So we use a 2D matrix with real numbers instead of an image object.

Before passing this sensor reading image as input to our CNN model, we do a normalization step; we first subtract the $\mathcal{N}$ from each value and then divide it by $-\mathcal{N}/2$. Let $\mathbf{X}'$ denote the 2D matrix after the normalization of $\mathbf{X}$. The value $\mathbf{X}'_{\mathbf{i,j}}$ will be zero at locations without sensors, and $\mathbf{X}'_{\mathbf{i,j}}$ will be a positive real number (in most cases, less than two) for locations with sensors. E.g., if $\mathbf{X}_{10,20} = -50$, then the $\mathbf{X}'_{10,20}$ equals to $(-50 - (-80))/40) = 0.75$. Fig. 3.3 (b) shows how a matrix is used to represent the input information that contains both the RSS and the spatial location of the distributed sensors in an area that exists 14 sensors in Fig. 3.3(a).

3.3.2 Output Image Representing TX locations' Distributions

We now focus on designing the output image to represent the distribution of TX locations; the output image is essentially the "label" assigned to each input image that guides the training of the CNN model. Fig. 3.3(c) illustrates the output image of the image-to-image translation step in Fig. 3.3(a) that contains three transmitters.

A straightforward representation that represents the TXs with locations is to just use an array of (x, y) elements where each (x, y) element is the location of a transmitter, as in [331]. However, this simple representation is less conducive to efficient model learning, as the representation moves away from spatial representation (by representing locations as positions in the image) to direct representation of locations by coordinate values. E.g., in [331]'s CNN-based approach to MTL problem, the authors assume a maximum number N of transmitters and train as many as $N + 2$ different CNN models and thus, limiting the overall solution to the pre-defined maximum number of transmitters. Instead, in our approach, we facilitate the learning of the overall model, by solving the MTL problem in two steps, and in this step of translating sensors' reading to transmitter locations' distributions, we represent the output also as an image. This approach allows us to use a spatial learning model (e.g. CNN) for the second step too, and preclude use of regression or fully-connected layers in the first step.

Inspired by recent work on wireless localization problem [11] that represents the input and output as images, we represent our output of the first step as an image as well. The output image is a grayscale image implemented as a 2D matrix with real numbers. In the output image, we use 25 (5×5) pixel values to represent the presence of a transmitter. It is desirable to use an odd side length square (e.g., 3×3, 5×5, 7×7) for symmetry. For a 100×100 size input we use, while 3×3 gives too little information for a transmitter and 7×7 generates too many overlaps for close by transmitters, 5×5 is the sweet spot. Other pixels far away from any transmitter are zero-valued. Among multiple potential ways to represent a transmitter presence by a number of pixels, we found that using a 2D Gaussian distribution around the pixel

Input: Sensor Reading Image

C[1,8,5,1,2], G[4,8], ReLU   C[8,32,5,1,2], G[16,32], ReLU   C[32,8,5,1,2], G[4,8], ReLU   C[8,1,5,1,2]

Output: Translated Image of Peaks (TX)

1 x 100 x 100

8 x 100 x 100

32 x 100 x 100

8 x 100 x 100

1 x 100 x 100

5 TX at (15, 97), (47, 77), (49, 79), (91, 53), (92, 33)

C – Conv2d
G – Group Norm

Step 1: Image-to-image translation
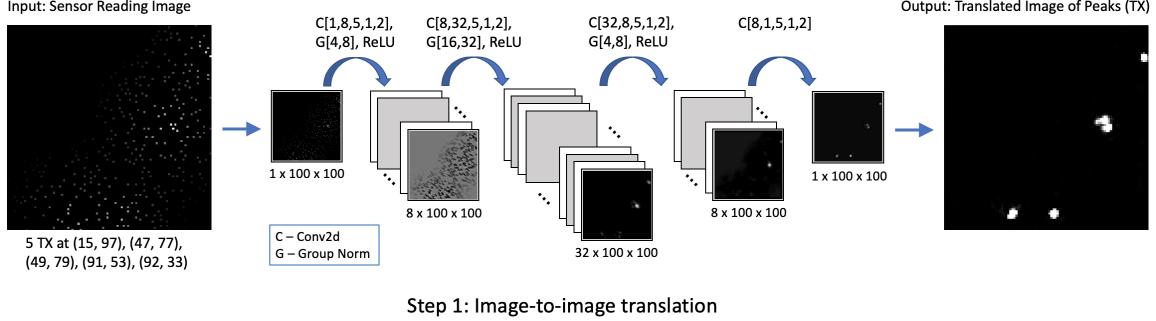
Figure 3.4: Architecture of the first step CNN, a four layer image-to-image translation model (`sen2peak`). The figure displays how the data volume flows through the various convolutional layers. C stands for Conv2d, and for each Conv2d layer, the five values shown are [number of input channels, number of output channels, kernel size, stride, padding]. G stands for group normalization, and, for each group normalization, the two values shown are [number of groups, number of channels]. See §3.3 for details.

of TX location, as shown in Fig. 3.3(c), yields the best model performance. Thus, a geographic area with multiple transmitters present is represented by a grayscale image with multiple Gaussian distributions, with each Gaussian distribution's peak inside the pixel corresponding to transmitter's location. Based on preliminary performance tests, we pick the amplitude of the 2D Gaussian peak to 10, the standard deviation to 0.9, and located the center of the distribution at the location of each transmitter. Note that the location of the TX is in the continuous domain and usually not at the center of the grid cell.

3.3.3 Image-to-Image Translation: `sen2peak` CNN Model

At a higher level, we use a deep and spatial neural network, in particular a CNN, to learn the approximation function that maps the input image (of sensor readings) to the output image (of Gaussian distributions for TX locations). We refer to this as the *image-to-image translation* model. Our approach is inspired by the recent work [11] that frames a different wireless localization problem as an image-to-image translation problem. We incorporate the idea into our multiple transmitter localization problem and utilize recent advances in the computer vision area. Encoder-decoder based CNN models like U-Net [220] with down-sampling and up-sampling convolutional layers have been successful in effectively learning image-to-image translation functions. However, in our setting, we observe that the usage of down-sampling layers (such as max-pooling) degrades the performance of the model, especially in the case when transmitters may be close to each other wherein the model is unable to distinguish the nearby transmitters and generate a single large distribution in the output image. To circumvent this, we avoid using any down-sampling layers in our model and redesign the image-to-image translation model as described below.

`sen2peak` **CNN Model**. We refer to our image-to-image translation CNN model as `sen2peak`, as it translates sensors' readings to "peaks" with Gaussian distributions

corresponding to transmitter locations. It has four [3] convolutional layers, as shown in Fig. 3.2(a). We use an input size of 100×100. The number of convolutional filters varies for different layers, with up to 32 in one of the layers. We tried doubling the filter numbers at each layer, but it did not lead to significant improvement (it does yield a lower error, but the output image does not improve significantly to impact the second step of our architecture). We use a kernel size of 5×5, a stride of 1, and a padding of 2. This ensures that the dimensions do not decrease and all the pixels are treated uniformly, including the ones at the edge of the image. With the above four convolutional layers, the receptive field [166] of each neuron in the output layer is 17×17. Normalization layers can improve the learning process. We chose group normalization [279] and put it after the first three convolutional layers. We compared group and batch normalization [121] methods in our context and observed better performance with the group normalization. For the activation layers, we select the rectified linear unit (ReLU) and put it after the group normalization layers.

The Loss Function. Our inputs (X) and output (Y) are images. We use the L2 loss function which computes the mean squared error aggregated over individual pixels. More formally, our loss function is defined as:

$$\frac{1}{N} \sum_{i}^{N} ||\texttt{sen2peak}(X_i) - Y_i||^2 \tag{3.1}$$

where N is the number of samples used in computing the loss, $|| \cdot ||^2$ is the L2 loss function, X_i and Y_i are the i_{th} sample's input and output images respectively, and $\texttt{sen2peak}(X_i)$ is the predicted output image corresponding to the input X_i. During training, we use Adam [133] as the optimizer that minimizes the loss function. We set the learning rate to 0.001 and the number of epochs to 20 and the model converges well.

3.4 $\texttt{DeepMTL}$ Step 2: TX Locations' Distributions to Precise Locations

In this section, we present the second step of our overall localization approach. We refer to this step as the *peak detection* step, as the goal is to detect the peaks within the Gaussian distributions in the input image (which is also the output image of the first step). The first step outputs an image that has multiple distributions (presumably, Gaussian), whose peaks need to be interpreted as precise locations of the transmitters/intruders. As, our end goal is to determine the precise locations of the present transmitters, we develop techniques to detect peaks within the output image of the first step. We propose two different strategies for the peak-detection task. The first strategy is a straightforward peak detection algorithm based on finding local maximal

---

[3]We observe that a four-layer lightweight and symmetric $\texttt{sen2peak}$ model produces good results and adding more layers gives marginal improvement.
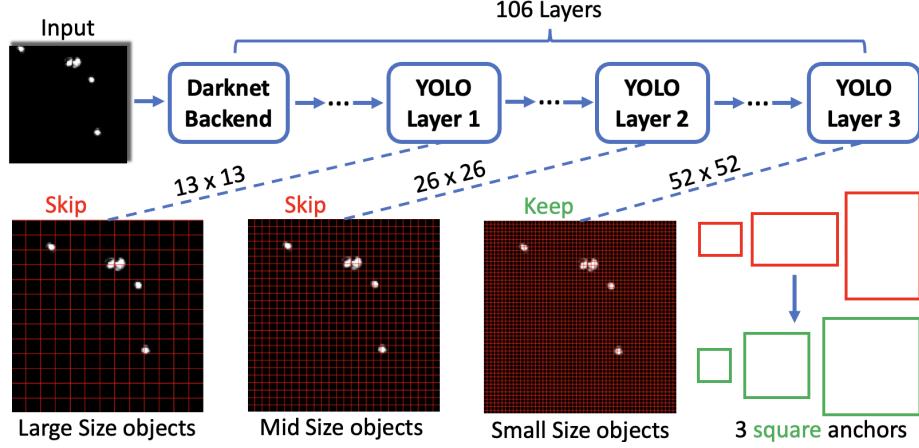
Figure 3.5: Our `YOLOv3-cust` in the second step of the `DeepMTL`. The two major customizations are: (i) Use only the third YOLO layer that detects small-size objects (the output of `YOLOv3-cust` is the bounding box predicted by the third YOLO layer and we use the center of the bounding box as the transmitter location), and (ii) change the rectangle anchors to square anchors.

values, while the second strategy is based on framing the problem as an object detection task; for the second strategy, we utilize a widely used state-of-the-art computer vision model called YOLOv3 [215].

Simple Peak Detection Method. The simple and straightforward peak detection method is to designate pixels with locally maximal values as peaks, subject to certain thresholds. More formally, we use a threshold x for a peak value and also use a parameter r to define a r-radius neighborhood of a pixel. Then, any pixel whose value is more than x and is the maximum among all pixels with a r-radius neighborhood, is designated as a peak (transmitter location). We use $x = 2$ and $r = 3$, in our evaluations. Note that each pixel represents a subarea; thus, a pixel designated as pixel only implies the transmitter location at the *center* of the corresponding subarea. To localize the transmission more precisely with the pixel's subarea, we use a scheme that localizes the transmitter within the subarea by computing a weighted average of the peak pixel's coordinate and the peak's neighbor pixels' coordinates. The weight of a pixel is the predicted pixel value itself from the first step `sen2peak`. We refer to the above simple approach for the second step of `DeepMTL` as `simplePeak`.

3.4.1 Object-Detection Based Precise Localization: `YOLOv3-cust`

The simple hand-crafted method described in the previous subsection performs reasonably well in most cases in our simulations. However, its key drawback is that it needs appropriate threshold values that may vary from case to case; such thresholds can be difficult to determine, especially since the input images (with distributions) are not expected to be perfect as they are themselves the output of a learning model. Inaccurate threshold values can lead to false alarms and misses. Also, the previous method is not sufficiently accurate at the sub-pixel level, where each pixel may rep-
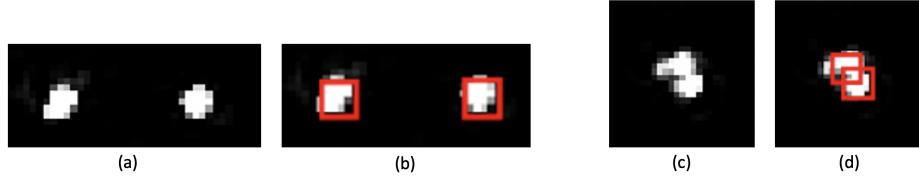
Figure 3.6: (a) is the zoom-in of two peaks at the bottom of the Fig. 3.2 example. (c) is the zoom-in of the two close-by peaks in the middle right of the Fig. 3.2 example. (b) and (d) shows the bounding boxes that `YOLOv3-cust` outputs for (a) and (c) respectively.

resent a large area such as $10m \times 10m$ or even $100m \times 100m$. Thus, we propose a CNN-based learning method that overcomes the above shortcomings. CNN has been widely used for object detection in different areas [6, 160].

We frame this problem as an object detection task where the objective is to detect and localize known objects in a given image. We observe that our second-step peak detection problem is essentially an object detection problem where the "object" to detect is a "peak". Thus, we turn the `MTL` problem of localizing multiple transmitters into detecting peaks in the images output by `sen2peak` model. For object/peak detection, we design `YOLOv3-cust`, our customized version of YOLOv3 [215]. Fig. 3.6 is a zoom-in of localizing two close-by transmitters (peaks) in Fig. 3.2(b).

Peak Detection Using `YOLOv3-cust`. Object detectors are usually comprised of two parts: (i) a backbone which is usually pre-trained on ImageNet, and (ii) a front part (head), which is used to predict bounding boxes of objects, probability of an object present, and the object class. For the front part, object detectors are usually classified into two categories, i.e., one-stage detectors such as the YOLO [214] series, and two-stage detectors such as the R-CNN [97] series. We choose the one-stage YOLO series because of its computational efficiency, high popularity, and available ways to customize it for our specific context. We refer to the customized version as `YOLOv3-cust`, see Fig. 3.5. Implementing a 106-layer deep neural network with a complex design from scratch is out of the scope of our work. Thus, we use a publicly available source repository [158] and made customization on top of it. We refer to the architecture that uses `sen2peak` and `YOLOv3-cust` in sequence as `DeepMTL`, our key product. In addition, we use `sen2peak` in combination with the uncustomized original YOLOv3, and refer to it as `DeepMTL-yolo` (still change the class number to one).

Customization of YOLOv3. Overall, we incorporated four customizations to YOLOv3, of which two are significant and the other two are relatively minor. See Table 3.1. YOLOv3 is designed to be a general object detector that can detect objects of various sizes, shapes, and classes within input images of various sizes. However, in our context, the input images are of a fixed size, with only a single class of objects which are relatively small and semi-circular. Based on the above observations, we make changes to the original YOLOv3 that both decrease the model complexity and improve its performance.

Customization Details. The first and second changes presented in Table 3.1 are

Table 3.1: Differences between the original YOLOv3 and our `YOLOv3-cust`.

YOLOv3	YOLOv3-cust
Has three YOLO layers at 13×13, 26×26, and 52×52 for detection	Only use the last 52×52 YOLO layer for detection (skip the first two YOLO layers)
Has 3 different rectangle anchors for each YOLO layer	Has 3 square anchors
Every 10 batches, randomly choose a new input image dimension size	Do not randomly choose new input dimension size
Has 80 different categories of object class	Only has one category for the peak class

major changes and we elaborate on them in the following paragraphs. Making predictions at three different scales is one of the highlights of YOLOv3 and an improvement compared to the previous version YOLOv2 which was prone to missing at detecting small objects. As shown in Fig. 3.5, the coarse-grain 13×13 YOLO layer-1 is designed for detecting large size objects, the 26×26 YOLO layer-2 is designed for detecting middle-sized objects, and the fine-grained 52×52 YOLO layer-3 is designed for detecting small-sized objects. Since the peaks in our translated images are always small objects, we only use the last 52×52 YOLO detection layer (and skip the first two YOLO layers). As shown in Fig. 3.5, by "skipping" the two YOLO layers means that we do not use them in computing the overall loss function and their outputs are not used in predicting the bounding boxes. In our `YOLOv3-cust`, the only YOLO layer predicts 8112 bounding boxes, since it has a dimension of 52×52 and each cell results in the prediction of 3 bounding boxes; this is in contrast to the original YOLOv3, which predicts 10647 bounding boxes ($3 \times (13 \times 13 + 26 \times 26 + 52 \times 52) = 10647$).

The anchor box is one of the most important hyperparameters of YOLOv3 that can be tuned to improve its performance on a given dataset. The original YOLO's anchor boxes are 10×13, 16×30, and 33×23 (for the input image of size 416×416 pixels), which are essentially bounding boxes of a rectangular shape. These original YOLOv3 anchors were designed for the Microsoft COCO [157] data set and were chosen since they best describe the dimensions of the real-world objects in the MS COCO data set. In our context, since the peaks are generally squares—we use the anchor boxes to be 15×15, 25×25, and 35×35.

Input Image for `YOLOv3-cust`. The first step `sen2peak`'s output image is 100×100, while the second step `YOLOv3-cust`'s input is required[4] to be a three-channel (RGB) image with each channel being a size of 416×416. To feed the output of `sen2peak` to `YOLOv3-cust`, we do the following: (i) First, we duplicate the `sen2peak`'s output

---

[4]YOLOv3 was developed before our work and the YOLOv3 authors set the input size of the CNN model to $3 \times 416 \times 416$. Although we are customizing their YOLOv3 model, we cannot change the input size because changing it will change the convolutional layer structure, which will preclude us from using the pre-trained weights in the YOLOv3 backbone.
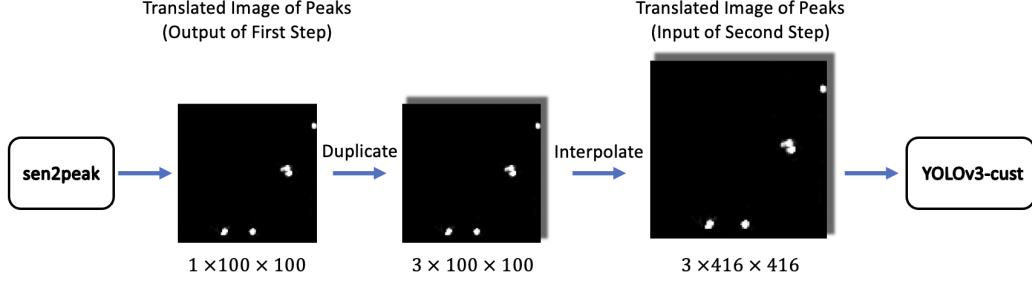
Figure 3.7: The data processing of `sen2peak`'s output to get `YOLOv3-cust`'s input of correct size.

image to create two more copies and thus create a three-channel image of 100×100 size channels; (ii) Next, we resize the 100×100 channels to 416×416 channels using the PyTorch's default "nearest neighbor" interpolation. See Fig. 3.7.

Output of `YOLOv3-cust`. YOLO treats object detection as a regression problem. The regression target (or "label") for an object is a five-value tuple $(x, y, length, width, class)$. In our case, there is only one $class$. x and y are real number location coordinates of the center of the bounding box, which we use as the location of the transmitter. $Width$ and $height$ determine the size and shape of the object—which we consistently set to be 5 each to signify a 5×5 square. Note that the center of the bounding box is in the continuous domain. Thus, we are able to get the sub-pixel level location of the transmitters.

3.5 Localization in the Presence of Authorized Users

Till now, we have assumed that the only transmitters present in the area are the intruders which need to be localized. In this section, we solve the more general `MTL` problem, where there may be a set of authorized users in the background. This is referred to as the multiple transmitter localization - shared spectrum (`MTL-SS`) problem [307].

In particular, in a shared spectrum paradigm, there are primary users and an evolving set of active secondary users transmitting in the background. Different than the intruders whose locations are unknown, the authorized users' locations are known and we wish to utilize this known information to better localize the unknown intruders. The key challenges come from the fact that the set of authorized users is not static and changes over time as allocation requests are granted and/or active secondary users become inactive over time. A straightforward way to handle background authorized users is to localize every transmitter, and then remove the authorized users. However, any localization approach is susceptible to performance degradation with the increase in the number of transmitters to be localized. Thus, the straightforward approach of localizing every transmitter is likely to be error-prone. Therefore, we attempt to develop a new approach that uses `DeepMTL` as a building block that uses the information of the location of the authorized uses in a way other than removing them after localizing all. The new approach tries to subtract the received signal strength

46
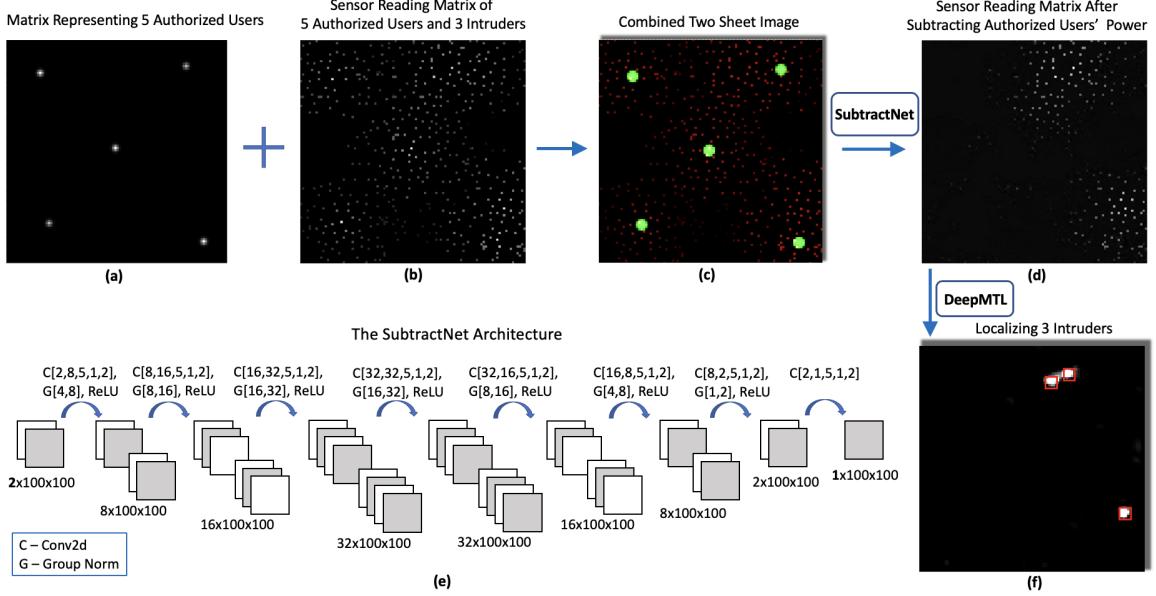
Figure 3.8: Overall architecture of the second approach to localize 3 intruders in the presence of 5 authorized users. The input of the `SubtractNet` is (c), which is stacking authorized user matrix (a) and the sensor reading matrix (b). (d) is the output of `SubtractNet`, where the transmission power of the authorized users is subtracted from the area. The details of the `SubtractNet` model is in (e). (f) is the localization output after feeding (d) into `DeepMTL`.

at the sensors by a value received from the authorized users. This subtraction is done by a novel CNN model; we refer to it as `SubtractNet`. Then we feed the image with subtracted powers to the `DeepMTL` and get the locations of the intruders. See Fig. 3.8(c)–(d)–(f). We describe `SubtractNet` in the following paragraphs.

`SubtractNet` Input Image. The sensor reading has two sources, one is the intruders and the other is the authorized users. We aim to subtract the power of the authorized users and retain the power from the intruders. So the input of the `SubtractNet` will contain two kinds of information: the authorized users' information (Fig. 3.8(a)), including both the location and the transmitter power, and the sensor reading matrix (Fig. 3.8(b)) that encode the power from all transmitters. To incorporate the two kinds of information, we first encode the authorized user information into a matrix that has the same dimension as the sensor reading matrix. Then stack the two matrices together. The combined stacked image is nothing but a two-channel image, which can be interpreted as Red and Green channels. The sensor reading matrix is the Red channel and the authorized user matrix is the Green channel. There is no Blue channel. To represent the authorized transmitter in the Green channel, we use a Gaussian peak similar to what we did in the `sen2peak` for representing transmitters (Section 3.3). The difference is that in `sen2peak`, all the peaks have a uniform height, whereas in `SubtractNet`, the height of the peak is the power of the authorized transmitter. So the higher the power of the authorized transmitter, the higher the peak in the Green channel. Another difference is that the authorized

47

transmitters are approximated at discrete locations instead of the continuous locations as in `sen2peak`.

<u>`SubtractNet` Output Image</u>. The `SubtractNet`'s output image is just a one-channel image and represents the sensor readings due to the intruders only.

<u>`SubtractNet` CNN Architecture</u>. We refer to the model that subtracts the power from the authorized users as the `SubtractNet`. It has a similar design philosophy with `sen2peak`. `SubtractNet` is also an image-to-image translation neural network. Compared to `sen2peak`, it doubled the number of layers, mainly because `SubtractNet` needs a bigger receptive field than `sen2peak`. A bigger receptive field can let the CNN model update sensors that are further away from the authorized user. For the loss function, we use the L2 loss function, similar to the loss function used in Equation 3.1, merely replacing the `sen2peak` with `SubtractNet` in Equation 3.1. The training details are also the same as in `sen2peak`.

3.6   Estimating the Transmit Power of Transmitters

In this section, we extend our techniques to estimate the transmit power of the intruders; we refer to the overall problem as Multiple Transmitter Power Estimation (`MTPE`). Estimation of the transmit power of transmitters can be very useful in shared spectrum systems. In particular, estimated transmit powers of the primary users (if unknown, as in the case of military users or legacy systems) can be used to set a "protective" region around them—inside which secondary users can be disallowed [247]. Estimating the transmit power of secondary users can also be useful. E.g., if the violation in a shared spectrum system is based on a certain minimum threshold, then it is important to estimate the transmit power to determine a violation. Also, the estimated transmit power of secondary users can be used to "circumvent" their intrusion—i.e., for the primary users to appropriately increase their transmit power to overcome the harmful interference from the secondary users. In general, estimating the transmission power is beneficial to various operations such as node localization, event classification, jammer detection [301].

There are several works that estimates the transmission power of a single transmitter, often jointly with its location [132, 247, 301]. Our previous work [307] can estimate the power of multiple transmitters. The similarity among all four of these methods is that they estimate the power and location jointly. In this chapter, we propose a new method that leverages the capabilities of `DeepMTL` by using it as a building block. We first localize the transmitters by `DeepMTL`. Then given the localized locations, estimate the transmitters' transmission power by a newly designed CNN model `PredPower`. Although `PredPower` is designed to only estimate the power of a single transmitter, we use it together with a machine learning-based error correction method that can mitigate the errors while applying `PredPower` to the multiple transmitter power estimation scenario.

In this section, we develop a technique to predict the transmission powers of the intruders. Here, for simplicity, we assume no background authorized users, though, the techniques in this section also work in the presence of authorized users. We lever-

age our accurate and robust localization solver that tolerates varying transmission power for different transmitters (the varying transmission power needs to be in a range). We propose an efficient approach and its overall methodology at a high level is as follows. And then in the next subsection, we describe our `PredPower` model.

1. We use `DeepMTL` to localize the multiple transmitters in a field.

2. We develop a CNN model `PredPower` to predict the power of a single isolated (far away from other intruders) intruder.

3. For other (non-isolated) intruders, we still use `PredPower` to predict their powers but employ a post-processing "correction" technique to account for nearby intruders.

3.6.1 `PredPower`: Predicting Power of a Single Isolated TX

<u>`PredPower` Input Image</u>. Let us consider an "isolated" transmitter T. To predict T's power, we start with creating a smaller-size image by cropping the original sensor readings image with the area of a certain size around T. In our evaluations in Section 3.7, the transmitters have a transmit radius[5] of around 20 pixels, which is equivalent to 200 meters.[6] For this setting, we used a cropped area of 21×21 around the isolated transmitter T to predict its power, with T at the center of this area; also, in this setting, we define a transmitter to be *isolated* if there is no other transmitter within a 20-pixel distance.[7] Note that the above cropping process requires the location of the transmitter to be known, and hence, we undertake the above power-estimation process after the localization of the transmitters using the `DeepMTL` model. We crop images from the same dataset where `DeepMTL` is trained on.

<u>`PredPower` Output Power</u>. The output of the `PredPower` is a single pixel whose value is the predicted power of the transmitter located at the center of the cropped image. Before coming into this single pixel output design, we tried using the height or radius of the peak from the output of `sen2peak` to indicate the power. But we figured out that the height or radius of the peak is hard to accurately predict and therefore is not an accurate indicator of the power. So we reduced the output complexity and designed the output as a simple single pixel whose value directly represents the power of the transmitter. By simplifying both the input side and output side, we can design and implement a novel CNN model that can accurately predict the power of a single transmitter, as described in the following paragraph.

---

[5]I.e., sensors beyond a distance of 20 pixels away from a transmitter x receive only negligible power from x.

[6]Transmission ranges of a standard 2.4 GHz and 5 GHz WiFi at default transmission powers (100 mW) are roughly 45m and 15m respectively. In our simulations (Section 3.7), we use the 600 MHz frequency band. As the lower the signal frequency, the higher the transmission range, a transmission range of around 200m is reasonable.

[7]Ideally, transmitters with a transmit radius of 20 pixels should entail defining isolated transmitters as ones that have no other transmitters within a 40-pixel distance, and then use a 41×41 area around the isolated transmitter. However, in our evaluations, our chosen values yielded a more efficient technique with sufficient accuracy.
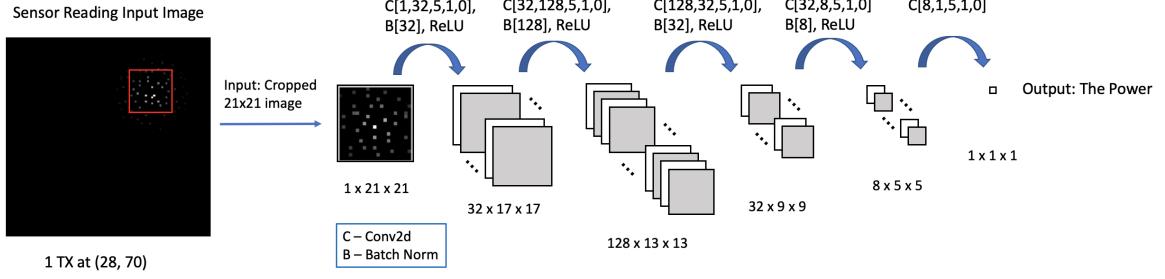
Figure 3.9: Architecture of the `PredPower`, a five-layer CNN model that takes in a cropped image from the original input image and outputs the predicted power of one transmitter. The figure displays how the data volume flows through the various convolutional layers. C stands for Conv2d, a 2D convolutional layer, and for each Conv2d layer, the five values shown are [number of input channel, number of output channel, kernel size, stride, padding]. B stands for batch normalization 2d, and for each batch normalization, the value shown is [number-of-features].

<u>PredPower CNN Architecture.</u> We refer to our CNN model that estimates the power of a single transmitter as `PredPower`. See Fig. 3.9. It has a similar design to `sen2peak` as well, where it has no max-pooling layers and no fully connected layers. We do not use the fully connected layers and design a fully convolutional network since the usage of fully connected layers will destroy the spatial relationships. `PredPower` has five CNN layers and each CNN layer has a kernel size 5×5, striding 1 and padding 0. With this setting, a pixel in the output layer has a receptive field of 21×21, which is exactly the size of the input cropped image. Also note that the pixel is exactly at the location where the transmitter is assumed to be located (recall that the transmitter is at the center of the cropped image). We tried both batch normalization and group normalization and found that batch normalization is better than group normalization, which is the opposite to the `sen2peak` scenario. ReLU is used as the activation function.

Loss Function. The output of the last convolutional layer is technically a 3D cube, although $1 \times 1 \times 1$. So we flatten it in the end to get one scalar value. We use a L2 loss function, which is formally defined as:

$$\frac{1}{N} \sum_{i}^{N} (\texttt{PredPower}(X_i^c) - y_i)^2, \tag{3.2}$$

where N is the number of training samples, X_i^c is the cropped input image for the ith sample and y_i is the ground truth power for the ith sample. $\texttt{PredPower}(X_i^c)$ is the predicted power. We use Adam as the optimizer and set the learning rate to 0.001 and the number of epochs to 20, which is sufficient for the model convergence.

3.6.2 Estimating Powers of Multiple Transmitters

Our end goal is to estimate the power of multiple transmitters at the same time. When the multiple transmitters are far away and isolated from each other, the problem

reduces to single transmitter power estimation, which `PredPower` handles well. The hard part is to estimate the transmit powers of multiple transmitters that are close by. In this case, a sensor will receive an aggregated power from multiple transmitters. We assume that blind source power separation is not viable.

Overall High-Level Approach. For each localized intruder by using `DeepMTL` (whether isolated or not), we crop the 21×21 size area around it and feed it to `PredPower`, and estimate its power. If it is actually isolated, then the predicted power is final. If it is not isolated, then we apply a post-processing correction phase to account for the overestimation of the powers, as described below.

Correction Method for Close by Transmitters. Let us first consider the case where there are two close-by transmitters T_0 and T_1. We use `PredPower` to estimate the power of two transmitters and get p_0' and p_1' respectively. Let us say the ground truths are p_0 and p_1 respectively. The estimated power will most likely be higher than the ground true power, i.e., $p_0' > p_0$ and $p_1' > p_1$. Because `PredPower` can only "see" one transmitter, it will view two transmitters in the areas as a combined single one. Let us focus on T_0 and assume $\delta_0 = p_0' - p_0$. The intuition is that δ_0 has some underlying patterns that we are able to recognize. We model δ_0 as a function of some features related to T_0 and T_1. We model δ_0 as follows,

$$\delta_0 = \theta_0 \cdot p_0' + \theta_{(1,1)} \cdot d_{01} + \theta_{(1,2)} \cdot p_1' + \theta_{(1,3)} \cdot \frac{p_1'}{d_{01}} \qquad (3.3)$$

where d_{01} is the distance between T_0 and T_1, and the four θs are the coefficients for the four terms respectively. The first term is related to T_0 itself, and the other three terms are related to T_1. We observe that the smaller the d_{01}, the larger the value of δ_0. And the bigger the p_1', the larger the value of δ_0. So d_{01} has a negative correlation with δ_0 while p_1' has a positive correlation. $\frac{p_1'}{d_{01}}$ is a combination of two terms to increase the number of features. We also tried a few other features, but we decided to use only these three features for a close-by transmitter as a balance of model accuracy and model complexity.

Equation 3.3 is for the case of one close-by transmitter, we then extend the equation to handle multiple close-by transmitters in the following Equation 3.4,

$$\delta_0 = \theta_0 \cdot p_0' + \sum_{i=1}^{m} (\theta_{(i,1)} \cdot d_{0i} + \theta_{(i,2)} \cdot p_i' + \theta_{(i,3)} \cdot \frac{p_i'}{d_{0i}}) \qquad (3.4)$$

where m is the number of close by transmitters for T_0, the transmitter of interest, d_{0i} is the distance between T_0 and close by T_i, and p_i' is the uncorrected power predicted by `PredPower`. For the ith close by transmitter, we introduce three terms $d_{0i}, p_i', \frac{p_i'}{d_{0i}}$, and assign three coefficients $\theta_{(i,1)}, \theta_{(i,2)}, \theta_{(i,3)}$ to the three terms respectively. So for m close by transmitters, there are $1 + 3m$ number of terms in the Equation 3.4.

After modeling δ_0, in Equation 3.5, we "correct" p_0' by subtracting δ_0 from p_0' to get a more accurate estimation of the power of transmitter T_0.

$$p_0^{correct} = p_0' - \delta_0 \qquad (3.5)$$

Estimating the parameter θ. Equation 3.4 is essentially a linear model and we can train it by using either linear, ridge, or LASSO regression models [204]. We perform experiments using ridge regression (alpha=0.01). We set a distance threshold for a neighbor transmitter to be classified as a close by transmitter. Note that the transmitters will have a different number of close by transmitters. So, let us denote M as the maximum number of close-by transmitters we see in the dataset. When training the linear model in Equation 3.4, we train a model that assumes a maximum M number of close-by transmitters, i.e., the linear model has $1 + 3M$ terms. The $3M$ terms are organized in a group of three (i.e., three features) and the groups are sorted by distance in ascending order. Then, for a transmitter with a smaller than M number of close-by transmitters, let us say m, only the first $1 + 3m$ terms will have a meaningful value. And for the rest $3(M - m)$ terms, we set the value to zero, i.e., impute the missing value with zero.

3.7 Evaluation

To evaluate the performance of our proposed techniques, we conduct large-scale simulations over two settings based on two different propagation models. In particular, we consider the log-distance-based propagation model and the Longley–Rice model obtained from SPLAT! [179]. We evaluate various algorithms, using multiple performance metrics as described below.

Performance Metrics. We use the following metrics 1, 2, and 3 to evaluate the localization methods and use the 4th metric to evaluate the power estimation methods.

1. Localization Error (L_{err})

2. Miss rate (M_r)

3. False Alarm rate (F_r)

4. Power Error (P_{err})

Given a multi-transmitter localization solution, we first compute the L_{err} as the minimum-cost matching in the bi-partite graph over the ground truth and the solution's locations, where the cost of each edge in the graph is the Euclidean distance between the matched ground truth node location and the solution's node location. We use a simple greedy algorithm to compute the min-cost matching. The unmatched nodes are regarded as false alarms or misses. We also put an upper threshold on the cost (L_{err}) of an eligible match. E.g., if there are four intruders in reality, but the algorithm predicts six intruders then it is said to incur zero misses and two false alarms, so the M_r is zero and the F_r is one-third. If the algorithm predicts three intruders then it incurs one miss and zero false alarms, so the M_r is one-fourth and the F_r is zero. In the plots, we stack the miss rate and false alarm rate to reflect the overall performance.

Algorithms Compared. We implement[8] and compare six algorithms in two stages. In stage one, we compare three versions of our techniques, viz., `DeepMTL`, `DeepMTL-yolo`, and `DeepMTL-peak`. Recall that `DeepMTL`, `DeepMTL-yolo`, and `DeepMTL-peak` use `sen2peak` in the first step, and `YOLOv3-cust`, original YOLOv3, and `simplePeak` respectively in the second step. In the first stage of our evaluations, we will show that `DeepMTL` outperforms `DeepMTL-yolo` and `DeepMTL-peak` in almost all performance metrics. Thus, in the second stage, we only compare `DeepMTL` with schemes from three prior works, viz., `SPLOT` [130], `DeepTxFinder` [331], and `MAP` [307] and show that `DeepMTL` outperforms the prior works.

Training and Testing Dataset. We consider an area of $1km \times 1km$, and use grid cells (pixels) of $10m \times 10m$, so the grid is 100×100. The transmitters may be deployed anywhere within a cell (i.e., their location is in the continuous domain), while the sensors are deployed at the centers of the grid cells (i.e. their location is in the discrete domain). For each instance (training or test sample), the said number of sensors and transmitters are deployed in the field randomly. For each of the two settings (propagation models described below), we create a 100,000 sample training dataset to train our models and create another 20,000 sample testing dataset to evaluate the trained model.

We will evaluate the performance of various techniques for a varying number of transmitters/intruders and sensor density. When we vary a specific parameter, the other parameter is set to its *default* value; the number of transmitters varies from 1 to 10 and the default value is 5; the sensor density varies from 1% to 10% and the default value is 6% (600 sensors in a 100×100 grid). The two default numbers 5 and 6% are chosen because they are in the middle of their ranges. When not mentioned, the default values are used. The transmitter power varies from 0 to 5 dBm and is randomly picked. To minimize overfitting, the training dataset and testing dataset have sensors placed at completely different locations.

We train the `DeepMTL` model using the 100,000 sample dataset. To train the `DeepTxFinder` [331], we partition the 100,000 sample training dataset into ten datasets based on the number of transmitters in the samples which varies from 1 to 10. These ten datasets are used to train the ten "localization" CNN models in `DeepTxFinder`, and the full dataset of 100,000 samples is used to train the `DeepTxFinder` model that determines the number of transmitters. For the `MAP` scheme [307], we assume the availability of all required probability distributions. We note that using a simple cost model (number of samples need to be gathered), the overall training cost for `MAP` is an order of magnitude higher than `DeepMTL` and `DeepTxFinder`. Lastly, `SPLOT` [130] does not require any training.

Two Propagation Models and Settings. The sensor readings (i.e. the dataset) are simulated based on a propagation model [292]. To demonstrate the generality of our techniques, we consider two propagation models as described below.

Log-Distance Propagation Model and Setting. Log-Distance propagation model is a generic model that extends Friis Free space model which is used to predict the path loss for a wide range of environments. As per this model, the path loss (in dB)

---

[8]Source code at: https://github.com/caitaozhan/deeplearning-localization.

between two points x and y at a distance d is given by: $PL_d = 10\alpha \log d + \mathcal{X}$, where α (we use 3.5) is the path-loss exponent and $\mathcal{X}$ represents the shadowing effect that can be represented by a zero-mean Gaussian distribution with a certain (we use 1) standard deviation. Power received (in dBm) at point y due to a transmitter at point x with a transmit power of P_x is thus: $P_x - PL_d$. Power received at point y due to multiple sources is assumed to be just an aggregate of the powers (in linear) received from each of the sources.

SPLAT! Model and Setting. This is a complex model of wireless propagation based on many parameters including locations, terrain data, obstructions, soil conditions, etc. We use SPLAT! [179] to generate path-loss values. SPLAT! is an open-source software implementing the Longley-Rice [42] Irregular Terrain With Obstruction Model (ITWOM) model. We consider a random area in Long Island, New York of $1km \times 1km$ large and use the 600 MHz band to generate path losses.
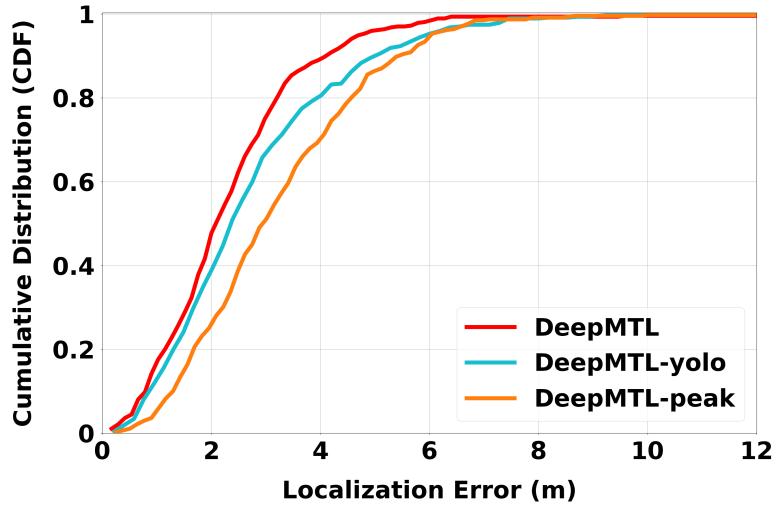
Figure 3.10: Cumulative probability of localization error of `DeepMTL`, `DeepMTL-yolo` and `DeepMTL-peak`, for the special case of single transmitter localization with 6% sensor density.

3.7.1 `DeepMTL` vs. `DeepMTL-yolo` vs. `DeepMTL-peak`

In this subsection, we compare the three variants of our technique, viz., `DeepMTL`, `DeepMTL-yolo`, and `DeepMTL-peak`. For simplicity, we only show plots for the log-distance propagation model setting in this subsection (we observed similar performance trends for the Longley-Rice propagation model too).

Performance Results. In Fig. 3.10, we plot the cumulative density function (CDF) of the localization error, for the simple case of a single transmitter. We observe that `DeepMTL` outperforms the other variants, as it yields a higher cumulative probability for a lower range of errors. In addition, we evaluate the three variants for varying number of transmitters (Fig. 3.11) and sensor density (Fig. 3.12), and evaluate the
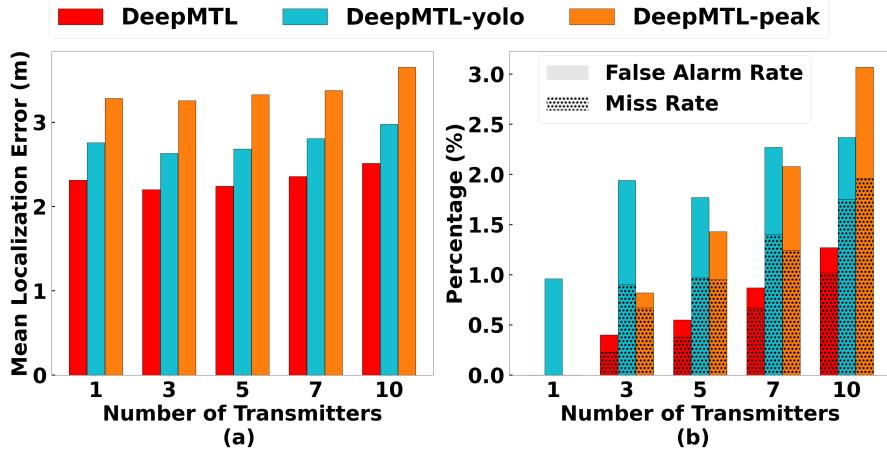
Figure 3.11: (a) Localization error and (b) miss and false alarm rates, of `DeepMTL`, `DeepMTL-yolo` and `DeepMTL-peak` variants for varying number of transmitters in log-distance dataset (propagation) model.

localization error as well as the false alarm and miss rates. We observe that `DeepMTL` consistently outperforms the other two variants across all plots and performance metrics. As expected, the performance of all algorithms degrades with an increase in the number of transmitters (in terms of false alarms and miss rates) or with a decrease in sensor density. In general, the localization error of `DeepMTL` is around 15-30% lower than the other variants. Impressively, the total cardinality error (i.e., false alarms plus miss rates) is fewer than 1% for the `DeepMTL` technique, when the sensor density is 6% or above.

When the sensor density is as low as 1%, the performance of all methods significantly decreases. Because when the sensor density is 1% or lower, the input image will be very sparse and contain only a few pixels. `DeepMTL`'s first part `sen2peak` has a receptive field of 17×17. This area will contain an average of less than three sensors when the sensor density is 1% ($17 \times 17 \times 0.01 = 2.89$). This number is considered too low and note that 2.89 sensors are not enough for the trilateration localization method, which needs three sensors. Our CNN models need to function well with enough pixels that contain useful information. So we suggest the sensor density to be at least 2% to achieve reasonable results.

Table 3.2: Compare Localization Running Time (s) for 1 to 10 Number of Intruders

Intru.	DeepMTL-peak	DeepMTL-yolo	DeepMTL	MAP	SPLOT	DeepTxFinder
1	0.0013	0.0180	0.0180	8.78	1.53	0.0015
3	0.0014	0.0183	0.0186	15.1	1.79	0.0016
5	0.0016	0.0192	0.0189	19.3	2.06	0.0017
7	0.0018	0.0196	0.0194	24.1	2.32	0.0019
10	0.0023	0.0205	0.0206	28.5	2.72	0.0022

Running Time Comparison. For the running time comparison of the variants, see
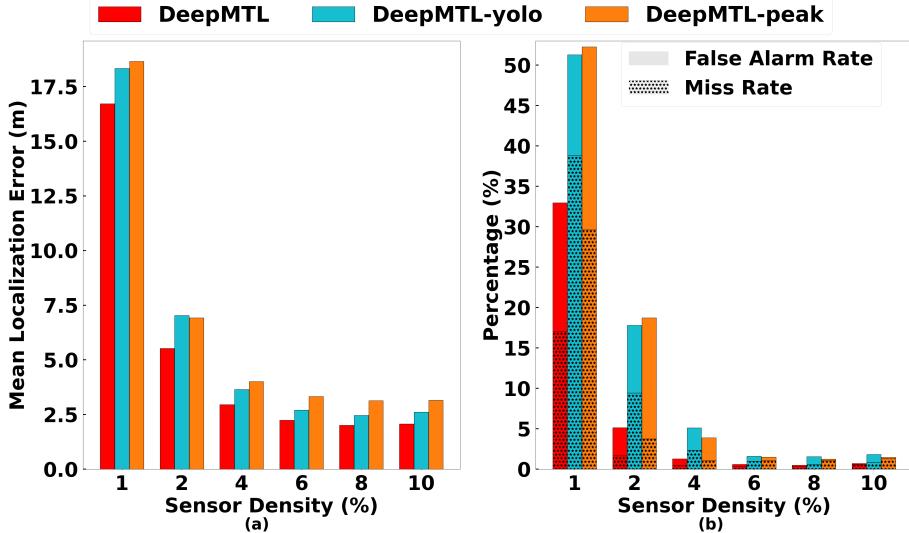
Figure 3.12: (a) Localization error and (b) miss and false alarm rates, of `DeepMTL`, `DeepMTL-yolo` and `DeepMTL-peak` variants for varying sensor density in log-distance dataset (propagation) model.

Table 3.2. Our hardware is an Intel i7-8700 CPU and an Nvidia RTX 2070 GPU. We observe that, as expected, `DeepMTL` and `DeepMTL-yolo` which use a sophisticated object-detection method do incur higher latency (around 20 milliseconds) than `DeepMTL-peak` (around two milliseconds). As our key performance criteria is accuracy and the run time of `DeepMTL` is still quite low, we choose `DeepMTL` for comparison with the prior works in §3.7.2.

Localizing Transmitters Close By. Localizing two or more transmitters close by is a hard part of the `MTL` problem. Fig. 3.6(c) and (d) gives an example of when an advanced object detection algorithm will work while a simple local maximal peak detection might not. Fig. 3.6(c) and (d) shows `DeepMTL` can successfully localize two transmitters as close as three pixels apart. When a pixel represents a $10m \times 10m$ area, then it is 30 meters apart. If a pixel represents a smaller area, such as $1m \times 1m$, it has the potential to localize two transmitters as close as three meters apart.

Two YOLO Thresholds. YOLO has two important thresholds to tune that can affect the miss rate and false alarm rate. One is the confidence threshold (`conf`) and the other is the non-maximum suppression threshold (`nms`). An object will be recognized as a peak only if its confidence level is larger than `conf`. If two recognized peaks' bounding boxes have a large overlap, and their intersection of union is higher than `nms`, then the two peaks will be considered as one peak. The peak with a higher confidence level keeps while the other peak with a lower confidence level discards. A higher `conf` will bring a lower false alarm rate but a higher miss rate, and a higher `nms` will bring a lower miss rate but a higher false alarm rate. We pick `conf` = 0.8 and `nms` = 0.5 for `DeepMTL` as we observe these values bring a good balance between false alarm rate and miss rate. In particular, a high `conf` of 0.8 precludes "fake peaks" at locations with no transmitters. Also, a low `nms` weakens `DeepMTL`'s ability
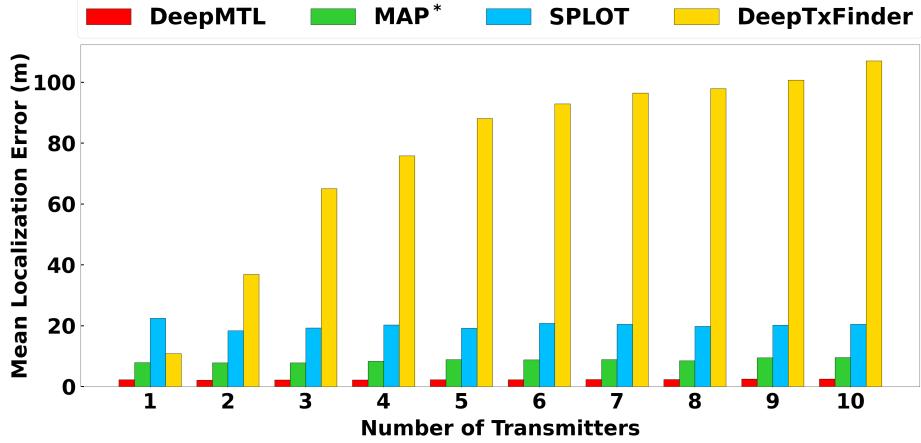
Figure 3.13: Localization error of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying number of transmitters in the log-distance dataset.
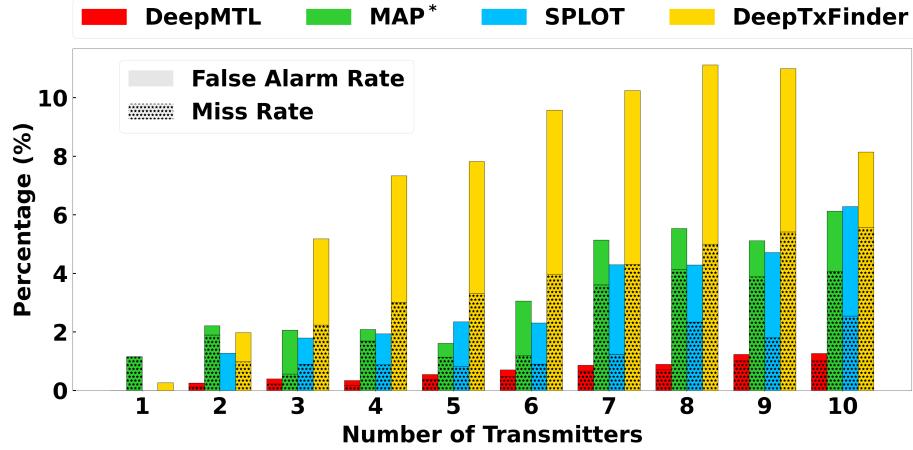
Figure 3.14: Miss and false alarm rates of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying number of transmitters in the log-distance dataset.

to localize two close by transmitters, while a high *nms* yields a high false alarm rate (by incorrectly interpreting a single transmitter as multiple close by transmitters); thus, we chose `nms` of 0.5.

3.7.2 `DeepMTL` vs. Prior Works

In this subsection, we compare `DeepMTL` with `SPLOT`, `MAP`, `DeepTxFinder` in both log-distance (Fig. 3.13, 3.14, 3.15) and SPLAT (Fig. 3.16, 3.17, 3.18) propagation models and thus, datasets. We observe similar performance trends for both datasets, i.e., `DeepMTL` significantly outperforms the other approaches by a large margin (in many cases, by more than 50% in localization errors, false alarms, and miss rates). For all techniques, as expected, the performance is generally worse in the SPLAT dataset compared to the log-distance dataset.
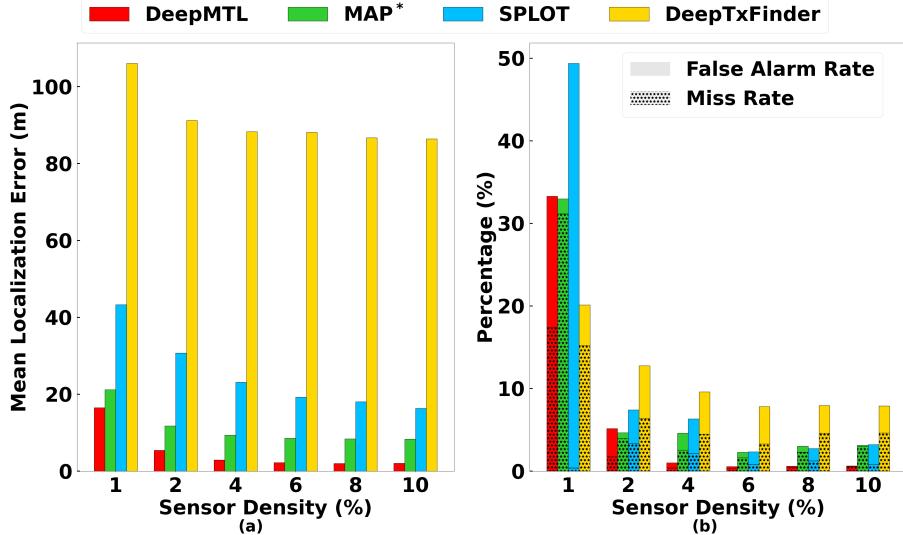
Figure 3.15: (a) Localization error, and (b) miss and false alarm rates, of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying sensor densities in the log-distance dataset.

Varying Number of Transmitters. Fig. 3.13 and Fig. 3.16 show the localization error with varying number of transmitters, in the two datasets. We see that `DeepMTL` has a mean localization error of only 2 to 2.5 meters (roughly, one-fourth of the side length of a pixel/grid cell) in the log-distance dataset and about 5 to 6 meters in the SPLAT dataset. In comparison, the localization errors of `MAP`, `SPLOT`, `DeepTxFinder` are two to three times, eight to nine times, and few tens of times respectively more than that of `DeepMTL`. Fig. 3.14 and Fig. 3.17 show the miss and false alarm rates with varying number of transmitters in the two datasets. We observe that `DeepMTL`'s summation of miss and false alarm rate is only 1% even at ten transmitters in the log-distance dataset, and about 4% for the case of SPLAT! dataset. In comparison, the summation of miss and false alarm rates for other schemes is at least 6% and 10% respectively for the two datasets, when there are ten transmitters.

Varying Sensor Density. Fig. 3.15 and Fig. 3.18 plot the performance of various algorithms for varying sensor density in the two datasets. For very low sensor density of 1%, all algorithms perform badly (in comparison with higher sensor densities), but `DeepMTL` still performs the best except that `MAP` performs best at 1% in terms of false alarm rate and miss rate. For higher sensor densities, we observe a similar performance trend as above—i.e., `DeepMTL` easily outperforms the other schemes by a large margin. For the SPLAT! dataset at the 6% sensor density, the summation of false alarm rate and miss rate is 2%, which is higher than the 1% summation for the log-distance dataset.

Running Times. The run time of `DeepMTL` (in tens of milliseconds) is orders of magnitude faster than `MAP` and `SPLOT` (both in seconds). See Table 3.2. The `DeepMTL` run time is an order of magnitude slower than `DeepTxFinder` (in a few milliseconds), due to the deep `YOLOv3-cust` taking up over 90% of the run time.

Summary and Analysis. In summary, our approach significantly outperforms the
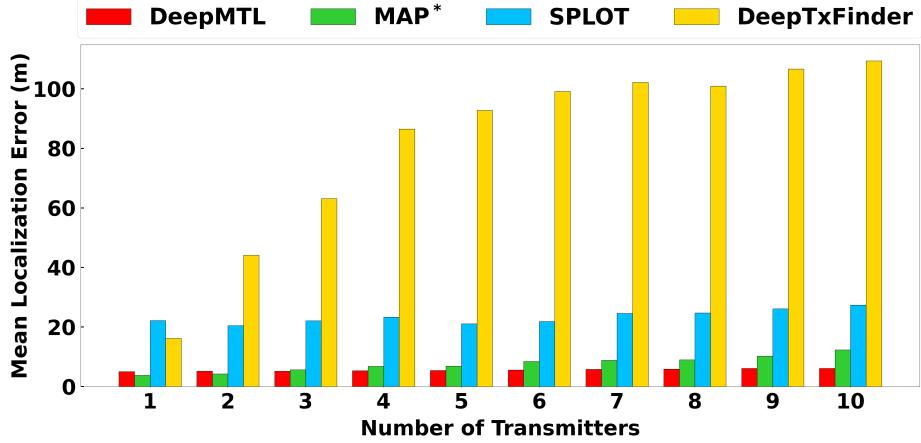
Figure 3.16: Localization error of `DeepMTL`, `MAP`, `DeepTxFinder` and `SPLOT` for varying number of transmitters in the SPLAT! Dataset.
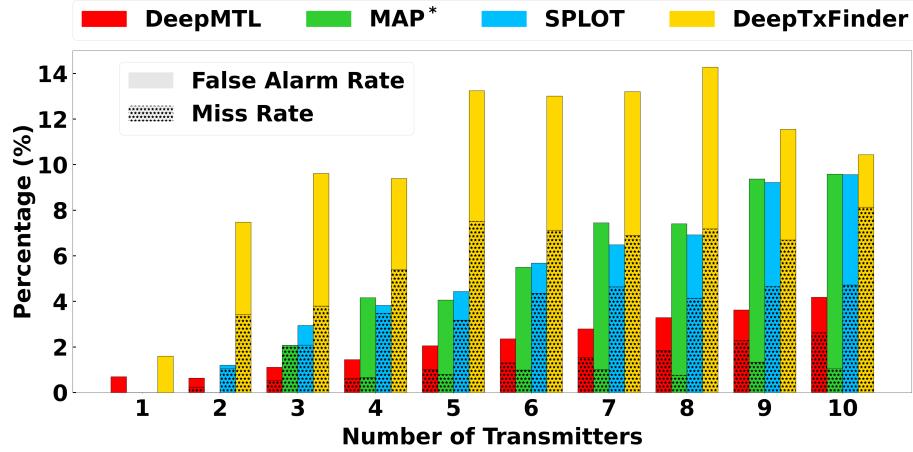
Figure 3.17: Miss and false alarm rates of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying number of transmitters in the SPLAT! Dataset.

other approaches in all the accuracy performance metrics, as well as in terms of latency. In particular, our approach also significantly outperforms the other CNN-based approach `DeepTxFinder`. The main reason for `DeepTxFinder`'s inferior performance is its inability to accurately predict the number of TXs—which forms a fundamental component of their technique. In contrast, `DeepMTL` can circumvent explicit pre-prediction of number of transmitters by using a well-developed object-detection technique which works well for multiple objects especially in our context of simple objects.

3.7.3 Transfer Learning

We demonstrate transfer learning (generalizability) by showing that the second step in `DeepMTL` does not need to be retrained for different radio frequency propagation
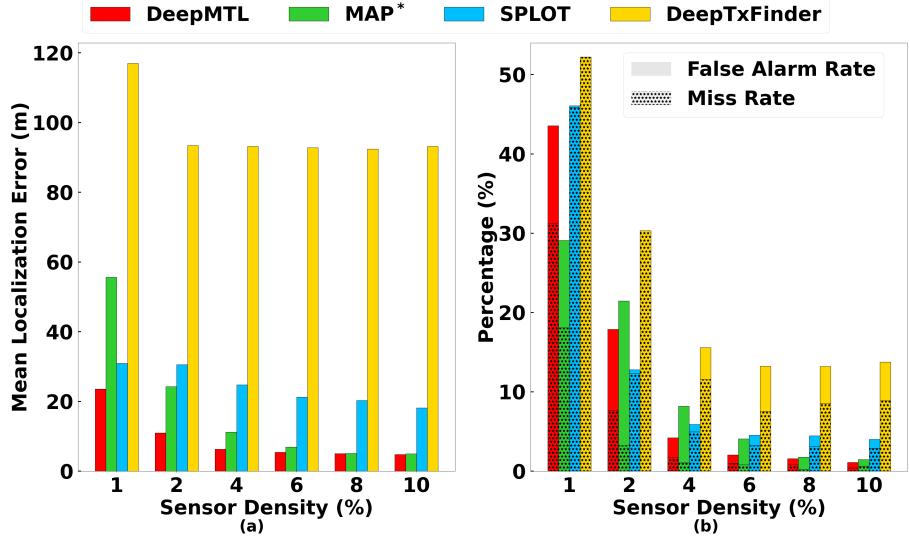
Figure 3.18: (a) Localization error, and (b) miss and false alarm rates, of `DeepMTL`, `MAP`, `SPLOT`, and `DeepTxFinder` for varying sensor densities in the SPLAT! Dataset.
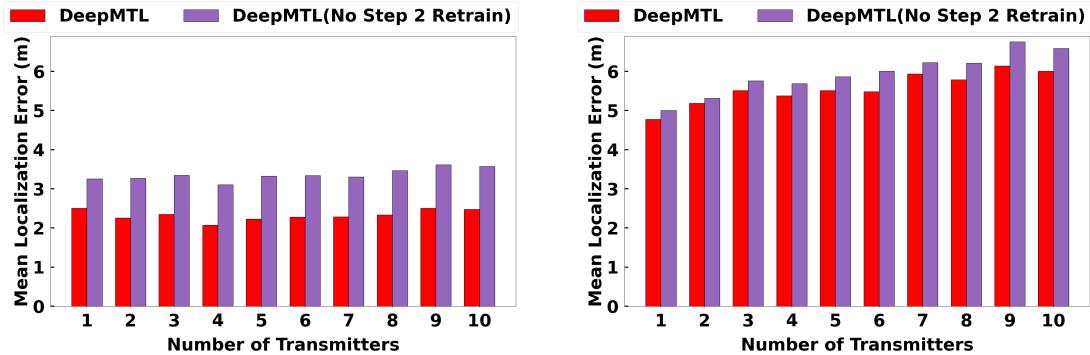
(a) First step trained in log-distance data, while the second step trained in SPLAT! data. Tested on the log-distance data.

(b) First step trained in SPLAT! data, while second the step trained in log-distance data. Tested on the SPLAT! data.

Figure 3.19: Localization error for varying number of transmitters when the first and second step of `DeepMTL` are trained on different training dataset.

models and terrains. In the previous experiments, the two steps of `DeepMTL` are both trained in the same setting, either log-distance or SPLAT!. We do the following two combinations to show that the second step does not need to retrain:

1. The first step is trained in the log-distance setting and the second step is trained in the SPLAT! setting. Tested on the log-distance data.

2. The first step is trained in the SPLAT! setting and the second step is trained in the log-distance setting. Tested on the SPLAT! data.

In both combinations, the second step `YOLOv3-cust` is trained on a different dataset compared to the first step `sen2peak`. Fig. 3.19a shows that the localization error
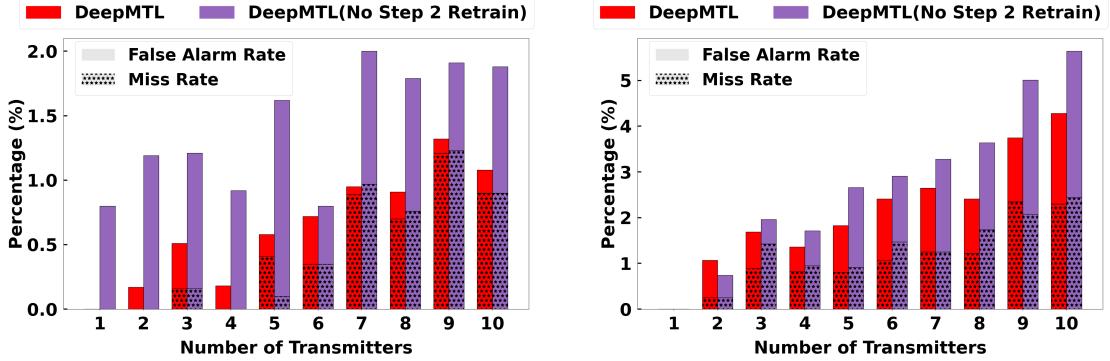
(a) First step trained in log-distance data, while second step trained in SPLAT! data. Tested on the log-distance data.

(b) First step trained in SPLAT! data, while second step trained in log-distance! data. Tested on the SPLAT! data.

Figure 3.20: The miss rate and false alarm rate for varying number of transmitters when the first and second step of `DeepMTL` are trained on different training dataset.

increases one-third in the first combination compared to the case where both the first and second steps are trained on log-distance dataset. Fig. 3.19b shows that the localization error increases only five percent in the second combination compared to the case where both the first and second steps are trained on SPLAT! dataset. The miss rate and false alarm rate for both combinations also increase minimally, i.e. the summation of miss rate and false alarm rate only increases around 1% in absolute value. See Fig. 3.20. This implies that the second step of `DeepMTL`, `YOLOv3-cust`, is general and does not need to retrain for different radio frequency propagation models and terrains. This is because the first step `sen2peak` is translating sensor readings images from different geographical areas to the same Gaussian peaks. The first step `sen2peak` still needs to be retrained for different situations to translate the sensor readings to the peaks.

3.7.4 Localize Intruders in the Presence of Authorized Users

The previous experiment setting is based on the assumption that all transmitters we are localizing are intruders. Different than the previous setting, here, we put five authorized users and they are spread out in the field, so those five will not interfere with each other. This is the more general version of the `MTL` problem, where there are some authorized users in the background. Fig. 3.21 shows the localization error of two approaches localizing intruders in the presence of five authorized users with a varying number of intruders. It is observed that the first approach, localize then remove authorized users, has a ten to twenty percent smaller localization error compared to the second approach, subtract authorized user power then localize. This is due to the inaccuracy of power subtraction from the `SubtractNet`. Fig. 3.22 shows the miss and false alarm of two approaches localizing intruders in the presence of five authorized users with a varying number of intruders. It is observed that the second approach, subtract authorized TX power then localize, is having a high false alarm when the
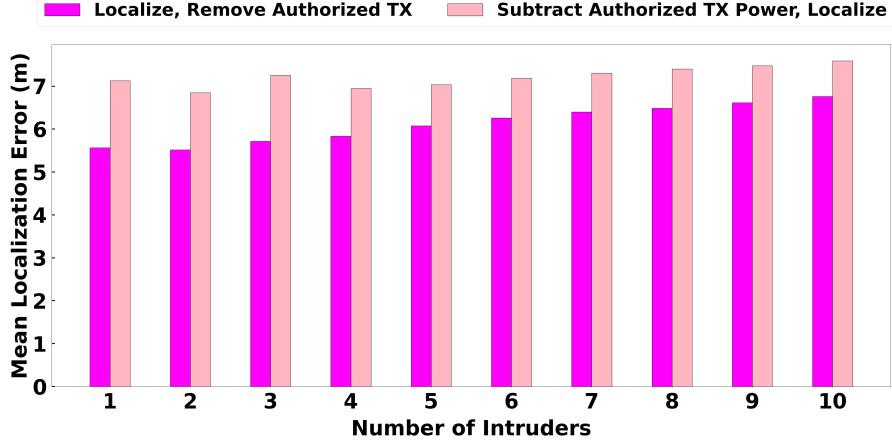
Figure 3.21: The localization error of two approaches in the presence of five authorized users with varying number of intruders.

number of intruders is three or less. So for `SubtractNet`, subtracting the power of five background authorized users from six transmitters (five out of six transmitters are authorized users, one intruder) is relatively more difficult than subtracting the power of five authorized users from nine users (five out of nine transmitters are authorized users, four intruders). Also statistically, getting one false alarm when there are one intruder and five authorized users is 100% false alarm rate, while getting one false alarm when there are two intruders and five authorized users is only 50% false alarm rate (the denominator is the number of intruders). Thus, the false alarm rate for one and two number of intruders looks to differ a lot, but in reality, the false alarm cases do not differ a lot). When the number of intruders is three or four, the two approaches are comparable. But when the number of intruders is larger than four, the second approach is having a lower miss and false alarm rate. In summary, the two approaches both have their strengths. The main advantage for the second approach is that the sum of miss rate and false alarm rate is lower when the number of intruders is large.

3.7.5   Power Estimation Evaluation

In this subsection, we evaluate the transmitter power estimation performance. In all experiments, the power range is 5 dB. The power error is presented in absolute value. A power error of 0.5 dB implies a relative power error of 10%. First, we compare the single transmitter power estimation between `MAP` and `PredPower`, and then compare the multiple transmitter power estimation between `MAP`, `PredPower` with error correction, and `PredPower` with error correction.

Figure 3.23(a) shows the performance of single transmitter power estimation in the log-distance propagation model scenario with varying sensor density. In this case, `MAP` has a 10 to 20 percent smaller power estimation error. Figure 3.23(b) shows the performance of single transmitter power estimation in the SPLAT! model with varying sensor density. In this case, `PredPower` is significantly lower in power error.
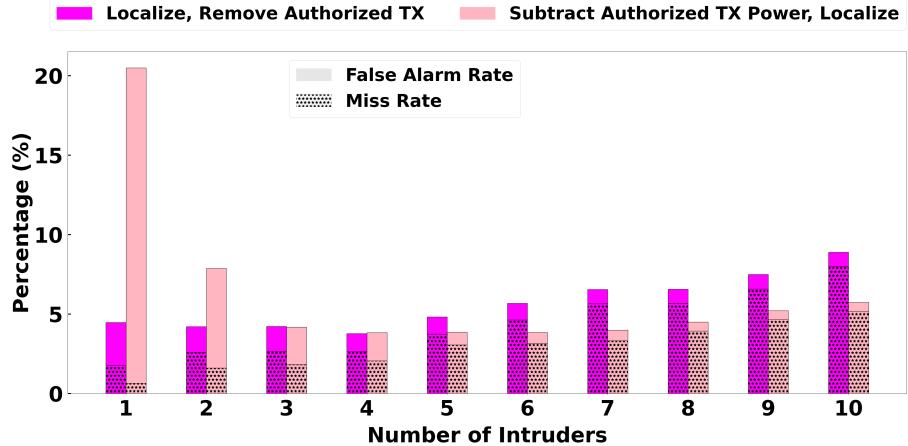
Figure 3.22: The miss and false alarm of two localization approaches in the presence of 5 authorized users with varying number of intruders.

So in average, `PredPower` outperforms `MAP` in single transmitter power estimation. We can also conclude that for `PredPower`, a higher sensor density will decrease the power estimation error. While a 2% of sensor density will lead to a higher error, a sensor density of 6% is enough to give relatively good results.

For multiple transmitter power estimation, we compare three methods in two propagation models and show that `PredPower` with error correction has the best performance among the three methods. `PredPower` without error correction is expected to perform the worst and it suggests that the post-processing error correction stage for `PredPower` is important and works well. Figure 3.24 shows the power estimation error of three methods with a varying number of transmitters while the sensor density is 6%. In this figure, `MAP` is the best only when the number of transmitters is one (which is consistent with Fig 3.23(a)). Also the number of transmitters is one is the only case when `PredPower` with correction and without correction has the same performance. This is also expected because there is no need to error correction when there is only one transmitter in the area. In all other cases, we see that `PredPower` with error correction is the best, `PredPower` without error correction is the worst, and `MAP` is in the middle. In Figure 3.25, which shows experiment results running in the SPLAT! propagation model, we see a similar pattern compared to Figure 3.24. The difference is that `PredPower` with error correction is always the best and the power error is larger than the log-distance model scenario. For example in Figure 3.24, the power estimation error for `PredPower` with error correction goes up to 0.6 dB, where as in Figure 3.25, the error goes up to 1 dB.

3.7.6 Evaluation over Testbed Data

In this subsection, we show that our `DeepMTL` performs well in real-world collected data. For this, we repurpose our testbed data from [307] as described below. We start with describing our testbed data from [307].
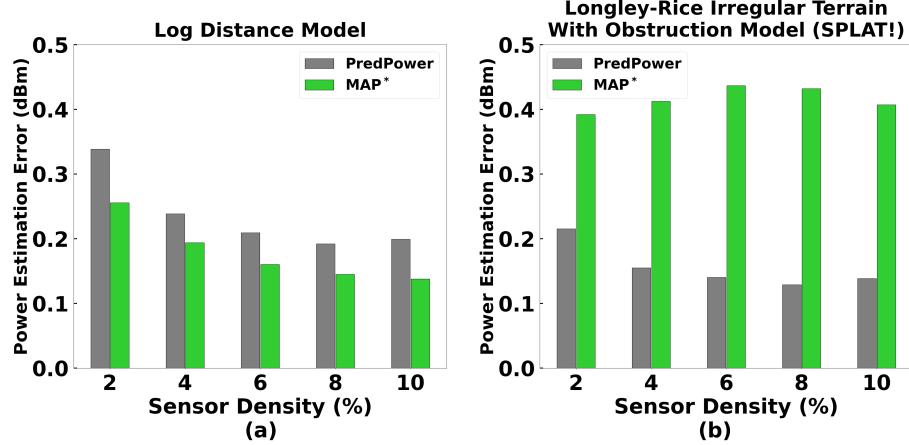
Figure 3.23: The single transmitter power estimation error of `PredPower` and `MAP` in two propagation models, (a) Log-distance model and (b) Longley–Rice Irregular Terrain with Obstruction Model (SPLAT!), for varying sensor densities.

Testbed Data. In [307], we conducted a testbed in an outdoor parking area of $32m \times 32m$ large.[9] Each grid cell has a size of $3.2m \times 3.2m$, with the grid size being 10×10. We place a total of 18 sensors on the ground. The sensors consist of Odroid-C2 (a single-board computer) connected to an RTL-SDR dongle and the RTL-SDR connects to dipole antennas. The transmitters are USRP or HackRF connecting to a laptop. We collect raw Inphase-Quadrature (I/Q) samples from the RTL-SDR at the 915 MHz ISM band. We perform FFT on the I/Q samples with a bin size of 256 samples to get the signal power values, and then utilize the mean and standard deviation of the power at frequency 915 MHz reported from each of the sensors.

Transforming the Data from 10×10 **to** 100×100 **Grid.** Note that `DeepMTL`'s input requires a 100×100 input, while the above data is over a 10×10 grid. Also, the sensor density in the above data is 18%, while we desire a sensor density of around 4-6% to have a fair comparison with our simulation based evaluations in previous subsections. To achieve these objectives, we transform the above 10×10 data to a 100×100 grid data in two steps as follows.

1. Increase the data granularity from 10×10 to 20×20, by dividing each cell into 2×2 cells; we randomly pick one of these four smaller cells to represent the original cell (i.e., to place the sensor if it existed in the original cell). See the red-bordered boxes in Fig. 3.26(a)-(b). We refer to the full 20×20 grid as a *tile*.

2. Now, we duplicate the 20×20 tile 25 times using a 5×5 pattern to generate a 100×100 grid. See Fig. 3.26(b)-(c).

The above steps effectively increase the area from the original $32m \times 32m$ to $160m \times 160m$. Note that the first step above only splits each original cell into four smaller

---

[9]Dataset publicly available at: https://github.com/Wings-Lab/IPSN-2020-data
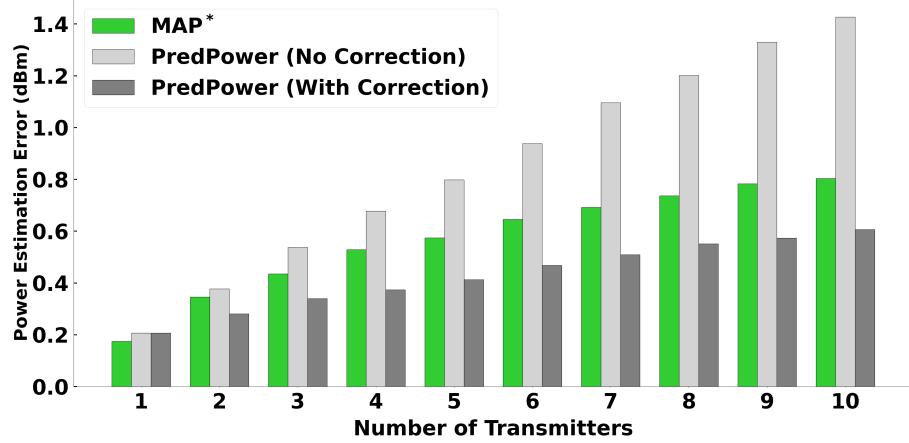
Figure 3.24: The transmitter power estimation error of `MAP`, `PredPower` with and without correction in Log-distance model for varying number of intruders

cells without increasing the whole area size. The 100×100 grid will have a sensor density of 4.5% and each grid cell represents an area of $1.6m \times 1.6m$.

We note that the second duplication step can introduce inaccurate sensor readings at the tile's "edges", due to any transmitters from adjoining tiles. To circumvent this issue, we place *transmitters* only within the internal 10×10 cells of each 20×20 tile (i.e., avoid placing a transmitter on the five-cell edge of each tile). This yields a total of 2500 potential positions to place a transmitter in the final 100×100 grid. With the above setting, we generate training and testing datasets consisting of 25,000 and 12,500 samples respectively.

Testbed Results. The performance of `DeepMTL` on this real world based data is shown in Fig. 3.27. Compared to `DeepTxFinder`, `DeepMTL` is significantly better in localization error and false alarm rate and miss rate in almost all cases, which aligns to the results in the previous subsections based on data generated from either log-distance model or SPLAT!. The localization error of `DeepMTL` in Fig. 3.27(a) is around 1.3 meters. The error increases mildly with the increase in the number of transmitters. The localization error in the testbed data is smaller compared to both log-distance data results (Fig. 3.13) and SPLAT! data results (Fig. 3.16). This is because a grid cell here is representing a smaller area. In the log-distance data, the localization error is roughly one-fourth the side length of the grid cell. In the SPLAT! data result, the localization error is roughly half the side length of its grid length. In the testbed data, the localization is roughly eighty percent the side length of a grid cell. So the localization error in the testbed data is the highest relative to the length of a grid cell it represents. The sum of false alarm rate and miss rate is 3% when the number of transmitters is five and is 5% when the number of transmitters is ten. The results are a little bit worse than the results in the SPLAT! data (Fig. 3.17), where the sum is 2% for five transmitters and 4% for ten transmitters.
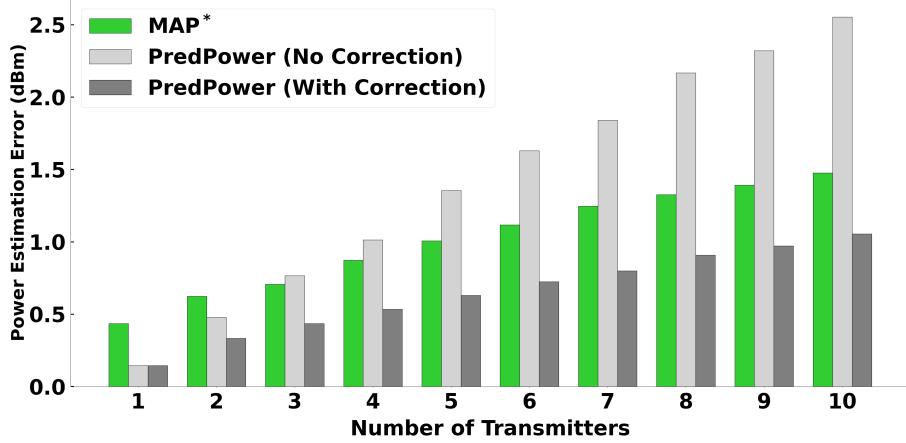
Figure 3.25: The transmitter power estimation error of `MAP`, `PredPower` with and without correction in Longley–Rice Irregular Terrain with Obstruction Model (SPLAT!) for varying number of intruders.

3.8 Related Work

Spectrum sensing is usually being realized by some distributed crowdsourced low-cost sensors. Electrosense [211] and its follow-up work Skysense [216] are typical works of spectrum sensing. In this crowdsourced sensing paradigm [38], sensors collect I/Q samples (in-phase and quadrature components of raw signals) and compute PSD (power spectral density), which is RSS. Crowdsourced low-cost sensors do not have the capability to collect AoA (angle of arrival) data because it requires more expensive antenna arrays. They also do not have the capability to collect ToA (time of arrival) data because it requires the transmission of a wide-band known sequence [65], which is impossible in the case of localizing (blind) intruders. Spectrum sensing platforms serve as the foundation of spectrum applications, and transmitter localization is one of the main applications. Other applications include signal classification [212], spectrum anomaly detection [153], sensor selection [24, 25], spectral occupancy estimation [226], etc.

Transmitter localization. Localization of an intruder in a field using sensor observations has been widely studied, but most of the works have focused on localization of a single intruder [39, 78]. In general, to localize multiple intruders, the main challenge comes from the need to "separate" powers at the sensors [203], i.e., to divide the total received power into power received from individual intruders. Blind source separation is a very challenging problem; only very limited settings allow for known techniques using sophisticated receivers [153, 228]. We note that (indoor) localization of a device [14] based on signals received from multiple reference points (e.g, WiFi access points) is a quite different problem (see [300] for a recent survey), as the signals from reference points remain separate, and localization or tracking of multiple devices can be done independently. Recent works on multi-target localization/tracking such as [128] are different in the way that targets are passive, instead of active transmit-
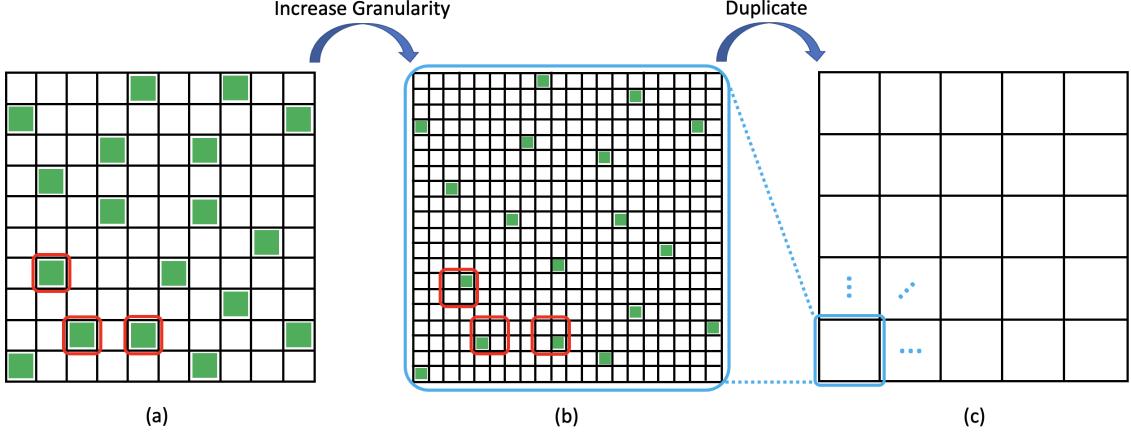
Figure 3.26: (a). The original 10×10 testbed grid with 18 sensors (green cells) representing a $32m \times 32m$ area. (b). The 20×20 grid (a tile) obtained by replacing each original cell by 2×2 smaller cells; a sensor, if present in the original cell, is placed in a random cell within the 2×2 grid (see the green cells). (c). The final 100×100 grid obtained by duplicating the 20×20 tile 25 times using a 5×5 pattern. The final geographic area is $160m \times 160m$.

ters in the `MTL` problem. Among other related works, [91] addresses the challenge of handling time-skewed sensors observations in the MTL problem.

Wireless localization techniques mainly fall into two categories: geometry mapping and fingerprinting-based. Geometry mapping mainly has two subcategories: ranging-based such as trilateration (via RSS/RSSI, ToA [285], TDoA) and direction-based such as triangulation (via AoA). Fingerprinting-based methods can use all signal physical measurements including but not limited to amplitude, RSS/RSSI, ToA, TDoA, AoA, and CSI. Whenever deep learning is used for localization, it can be considered as a fingerprinting-based method, since it requires a training phase to survey the area of interest and a testing phase to search for (predict) the most likely location. In [293], a diffusion model is used to generate the LoRa CSI fingerprints in orchards, and a complex-valued fully-connected block classifier is used for localization.

Deep learning for wireless localization. Quite a few recent works have harnessed the power of deep learning in the general topic of localization. E.g., DeepFi in [263] designs a restricted Boltzmann machine that localizes a single target using WiFi CSI amplitude data. DLoc in [11] uses WiFi CSI data as well. Its novelty is to transform CSI data into an image and then uses an image-to-image translation method to localize a single target. MonoDCell in [217] designs an LSTM that localizes a single target in an indoor environment using cellular RSS data. [65] designs a three-layer neural network that locations a single transmitter. `DeepTxFinder` in [331] uses CNN to address the same `MTL` problem using RSS data in this chapter.

Transmitter Power Estimation. There are several works that estimate the transmission power of a single transmitter. [301] studies the "blind" estimation of transmission power based on received-power measurements at multiple cooperative sensor nodes using maximum likelihood estimation. Blind means there is no prior knowledge
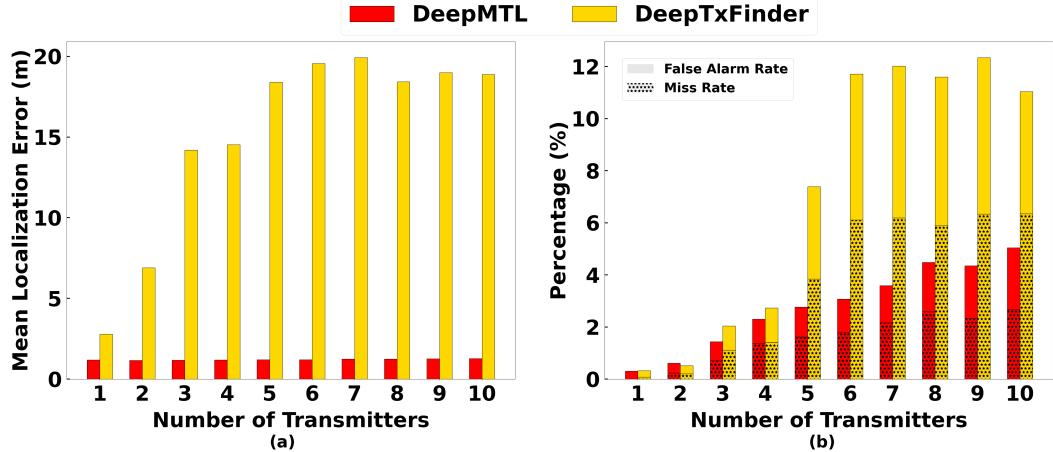
Figure 3.27: The localization error (a), false alarm rate and miss rate (b) of `DeepMTL` and `DeepTxFinder` in a real world collected data for varying number of intruders.

of the location of the transmitter or transmit power. [247] proposes an iterative technique that jointly estimates the location and power of a single primary transmitter. In [132], the primary transmitter location and power are jointly estimated by a constrained optimization method. [307] uses the maximum likelihood estimation method to estimate the power of an isolated single transmitter and adopts an online learning method to estimate the power of multiple close-by transmitters simultaneously.

Machine learning empowered applications. Machine learning includes supervised, unsupervised, semi-supervised, self-supervised, and reinforcement learning [185, 187]. [186] has studied reinforcement learning in the setting of episodic Markov decision processes. [188] studies the off-policy evaluation problem in reinforcement learning with linear function approximation. Machine learning techniques have empowered many applications. Studies have shown that image topology information can be leveraged to improve the performance of generative adversarial networks [251], and breast cancer treatment response biomarker [250]. Novel meta-learning algorithms and diffusion models are developed for generalizable magnetic resonance imaging reconstruction [26, 29, 30]. RELAX-MORE [28] leverages self-supervised learning techniques to achieve out-performing state-of-the-art on less qMRI training data. See [27, 31] for reviews of optimization-based deep learning models for MRI Reconstruction. XGBoost is great at capturing the upward trend using the portfolio features constructed by PolyModel theory [319] and deep learning-based techniques have been shown to further enhance portfolio construction [318]. Deep reinforcement learning can be used to minimize battery energy storage cost [118] and maintain the fidelity of equivalent model for active distribution networks [117]. [175] proposes machine learning methods that help transportation engineers and policymakers conduct accurate traffic performance evaluations [167, 176, 177]. Novel machine learning techniques have been proposed to deal with noisy labeled data [245] and genetic data [244]. Social media bots can be detected effectively via behavioral patterns [276] and metric learning [275]. AI-assisted audio command recognition enables collaborative human-robot

drone inspection of bridges [151]. In [53, 54], algorithms are developed to make machine learning models more transparent, accountable, and explainable [55, 56]. Network function virtualization [255, 258, 294] has great potential in low-performance edge devices [256] as more applications and ISP functionalities are expected to be offloaded to edge clouds [105, 257]. Machine learning can help the simulation and analyses of multi-platelet recruitment simulation [327]. In [229, 316], semi-supervised learning algorithms are developed to segment platelet images. In [260], a machine learning-guided imaging approach is used to segment platelet geometries and quantify adhesion dynamics parameters. DeepVS [284] combines 1D CNN and attention models to exploit local features and temporal correlations to improve RF-based vital signs sensing. An LSTM encoder-decoder model is proposed to generate Chinese poetry [169]. Machine learning also has applications in e-commerce [277], recommendation system [145, 146, 147] financial time series data [162], and spectrum allocation [94]. [172] provide a novel multimodal transformer to fuse clinical notes and structured electronic health record data for better prediction of in-hospital mortality. [173] studies the attention abnormality in Trojaned BERTs, and [174] proposes a Trojan attention loss for backdoor training which enhances the Trojan behavior. [152] conducts an examination of malware detection using machine learning techniques. [150] introduces innovative selection methods within the active learning framework. Semi-supervised contrastive learning [296] and neural rendering [297] can greatly improve medical image segmentation. [298] calibrates multi-model representations and [290] uses sampling techniques for linear contextural bandits. Novel decision transformer [48] based methods and explainable machine learning techniques [46, 47] have been applied to the network intrusion detection problem. Reinforcement learning [44] has been applied in various network utility optimization problems [43, 45] . FIAT [281] uses machine learning methods to achieve frictionless authentication for IoT traffic. Graph neural networks have been shown to enhance quantum approximate optimization algorithms as a warm-start technique [155]. DeepContrast [237] shows that a deep learning approach can potentially replace the need for Gd based contrast agent in brain MRI for approximating cerebral blood volume [159]. [329] leverages graph structure information to better classify paper taxonomy. A new node encoder, Graph Variational Diffusion Network [328], is proposed to improve the robustness of node representation. StyleGAN is upgraded by replacing its mapping network with 8 attention-based transformers to improve face editing [148] and visualization [314]. A novel time-domain self-restoration coding scheme [312] is proposed to enable efficient screen-camera communication. OptML [15] uses neural network to enable fast and efficient cross band channel prediction. [119] uses natural language processing techniques [235] to understand the well-being change (anxiety, stress, loneliness) across age and gender during the COVID-19. To leverage idle supercomputer [310] nodes, [178] introduces malleTrain to fit gaps in schedules for training deep neural networks without model information. Various machine learning methods have shown applications in 3D scene reconstruction [326], heart rate prediction [196], and stock prediction [195].

Mobile health sensing and edge computing. RF sensing enables some important

mobile health applications such as heart and respiratory rate monitoring [282, 287]. RF-based solutions support practical and longitudinal respiration monitoring owing to their non-invasive nature [116, 286]. [283] proposed a robust RF-based respiration monitoring. Low-quality RF sensing data will negatively affect the sensing task, thus reliable signal quality detection is crucial [288]. Apart from RF sensing, acoustic sensing can also enable important applications such as face authentication [321, 322]. Apart from RF sensors, other sensors such as inertial sensors, photoplethysmography, and actigraphy also play a big role in mobile health and wearable IoT, such as finger motion tracking [163], end-stage kidney fluid intake prediction in [242] and predicting salivary cortisol levels in pancreatic cancer patients [73]. In [74], a semi-supervised graph instance transformer is proposed for mental health inference. Graph neural networks are useful in various IoT sensing environments [75] and mobile health sensing [72]. In traditional wireless sensor networks, the sensing data is uploaded (via wireless) to a centralized server [104, 209], and the server processes the sensing data. In an emerging computing paradigm called edge computing, however, sensing data is mainly processed locally on resource-constrained sensors, e.g., on-device machine learning [122, 142, 311]. Thus, various works target to scale up task execution on resource-constrained systems [168], such as SmartOn [170], Antler [171] and intermittently-powered systems [123, 189]. Works tackle the computation overhead and battery limitation of on-device edge computing via interactive learning framework [324], offloading scheme [199] and reinforcement-learning-based scheduling technique [325]. Studies have shown that deep learning can reconstruct ambiance information [52] for mental health purposes. Large language models could help build a practical benchmark for cloud configuration generation [291]. Other works study the non-volatile hybrid memory in mobile memory systems [268, 269, 270]. Studies have shown the existence of an optimal amount of connectivity [180] in a blockchain network [181].

3.9 Conclusion

In this chapter, we have designed and developed a novel deep-learning-based scheme (`DeepMTL`) for the multiple transmitter localization (`MTL`) problem. We extended this problem to localizing the intruders in the presence of authorized users and developed a novel technique to solve it. We also developed a novel technique that can solve the multiple transmitter power estimation (`MTPE`) problem. Solving the general `MTL` and `MTPE` are both achieved by utilizing our robust `DeepMTL` as a building block. We evaluated all our methods extensively through data simulated from two propagation models as well as small-scale data collected from a real-world testbed. Our developed technique outperforms prior approaches by a significant margin in all performance metrics.

Chapter 4

Quantum Sensor Network Algorithms for Transmitter Localization

A quantum sensor (QS) is able to measure various physical phenomena with extreme sensitivity. QSs have been used in several applications such as atomic interferometers, but few applications of a quantum sensor network (QSN) have been proposed or developed. We look at a natural application of QSN—localization of an event (in particular, of a wireless signal transmitter). In this paper, we develop effective quantum-based techniques for the localization of a transmitter using a QSN.

Our approaches pose the localization problem as a well-studied quantum state discrimination (QSD) problem and address the challenges in its application to the localization problem. In particular, a quantum state discrimination solution can suffer from a high probability of error, especially when the number of states (i.e., the number of potential transmitter locations in our case) can be high. We address this challenge by developing a two-level localization approach, which localizes the transmitter at a coarser granularity in the first level, and then, at a finer granularity in the second level. We address the additional challenge of the impracticality of general measurements by developing new schemes that replace the QSD's measurement operator with a trained parameterized hybrid quantum-classical circuit. Our evaluation results using a custom-built simulator show that our best scheme is able to achieve meter-level (1-5m) localization accuracy; in the case of discrete locations, it achieves near-perfect (99-100%) classification accuracy.

4.1 Introduction

Quantum sensors, being strongly sensitive to external disturbances, are able to measure various physical phenomena with extreme sensitivity. These quantum sensors interact with the environment and have the environment phenomenon or parameters encoded in their state [66]. In [71, 264], multiplexer technologies are proposed to facilitate the readout of large (nearly 2000) arrays of superconducting transition-edge sensor bolometers (i.e., quantum detectors), largely accelerating the development of astronomy. A group of distributed quantum sensors, if prepared in an appropriate
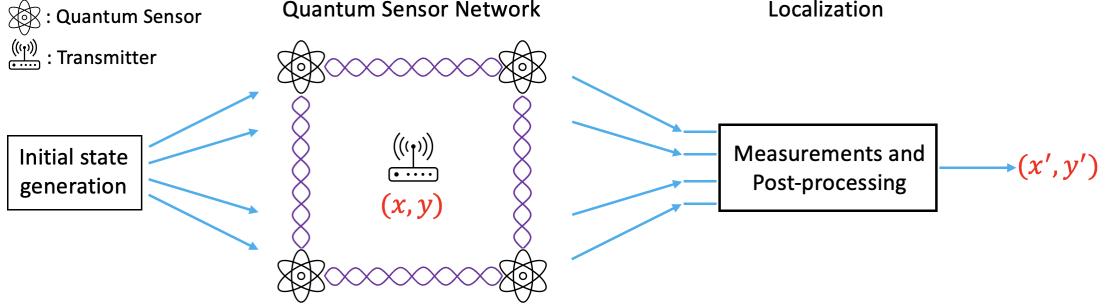
Figure 4.1: Overall architecture of using a QSN to localize a transmitter.

entangled state, can further enhance the estimation of a single continuous parameter, improving the standard deviation of measurement by a factor of $1/\sqrt{m}$ for m sensors (Heisenberg limit) [96].

Recently, many protocols have been developed for the estimation of a single parameter or multiple independent parameters [96, 208] using one or multiple (possibly, entangled) sensors. But, the use of a distributed set of quantum sensors working collaboratively to estimate more complex physical/environmental phenomena, as in many classical sensor network applications [5, 58, 108], has not been explored much. In this paper, we explore a potential quantum sensor network application— localization of events. In particular, we develop effective techniques for radio frequency (RF) transmitter localization and thus demonstrate the promise of QSNs in the accurate localization of events. Our motivation for choosing RF transmitter localization as the event localization application is driven by the significance of transmitter localization in wireless/mobile applications and recent advances in quantum sensor technologies for RF signal detection (see §4.2).

Transmitter Localization using QSNs. Our approach to transmitter localization using QSNs essentially involves posing the localization problem as a quantum state discrimination (QSD) problem [20] which is to identify the specific state a given quantum state is in (from a given set of states in which the system can be) by performing quantum measurements on the given quantum system. The overall architecture is illustrated in Fig. 4.1. First, a probe state is generated and distributed to the QSN. Then, once the quantum sensors have been impacted (i.e., the overall quantum state changed) due to the transmission from the transmitter's signal, an appropriate quantum measurement is made on the quantum state of the network. The outcome of the measurement determines the quantum state, and thus, the location of the transmitter. However, the above process can be erroneous, as solving the QSD problem even optimally can incur a certain probability of (classification/discrimination) error. This paper's goal is to develop an approach with a minimal localization error. In that context, our developed schemes in this paper are based on two ideas that extend the above basic QSD-based approach:

1. We use a two-level approach that localizes the transmitter in two stages: first, at a coarse level using a set of sensors over the entire area, and then, at a fine level within the "block" determined by the first level.

2. In addition, we circumvent the challenge of implementing a general measurement operation, by instead using a trained parameterized hybrid quantum-classical circuit that essentially implements the measurement operation and predicts the transmitter location from quantum sensor data.

Our evaluation results show that our best scheme (which combines the above two ideas) is able to achieve meter-level (1-5m) localization accuracy; in case of discrete locations, it achieves near-perfect (99-100%) classification accuracy.

Contributions. In the above context, we make the following contributions.

1. We model the transmitter localization problem as a well-studied quantum state discrimination (QSD) problem, which allows us to develop viable transmitter localization schemes using quantum sensors.

2. We design two high-level schemes to localize a transmitter in a given area deployed with a quantum sensor network. The first scheme is based on solving an appropriate quantum state discrimination problem using a global measurement, while the second scheme uses a trained hybrid quantum-classical circuit to process the quantum sensor data. Within the above high-level schemes, we also introduce a two-level localization scheme to improve the performance of the basic one-level schemes.

3. To evaluate our schemes, we model how a quantum sensor's state evolves due to RF signals from a transmitter at a certain distance. Using this model, we evaluate our localization schemes and demonstrate their effectiveness in our custom-built simulator.

Paper Organization. The paper is organized as follows. In §4.2, we present our quantum sensor model, formally define the transmitter localization problem and discuss related work. In the following two sections, we describe our two classes of algorithm: quantum-state-discrimination (QSD) based scheme, and parameterized-quantum-circuit (PQC) based scheme. We discuss our evaluation results in §4.5, and give concluding remarks in §4.6.

4.2 Sensor Model, Problem, Related Work

In this section, we start with motivating our choice of RF transmitter localization as an application for QSN, and then model the impact of an RF received signal on the quantum state of a quantum sensor. We then formulate the quantum localization problem and discuss related work.

Motivation for Transmitter Localization. Accurate detection and localization of a wireless transmitter (typically, using a radio-frequency (RF) wireless signal) is important in a variety of wireless and/or mobile applications, e.g., as an intruder detection in shared spectrum systems [39], localization of devices/users in indoor settings (e.g., supermarkets, museums, virtual/augmented reality applications [103]),

etc. In general, transmitter localization is a key technology for location-based services and an improvement in transmitter localization will be very beneficial to a variety of applications. In particular, in shared spectrum systems [39], there is a need to guard the shared spectrum against unauthorized usage which entails detecting and localizing unauthorized transmitters that may use and/or jam the spectrum illegally. Classical techniques for transmitter localization involve triangulation [289] or fingerprinting techniques [14] (see [57] for a survey).

Advances in quantum technologies have led to the creation of efficient quantum sensors for radio-frequency (RF) signal detection that are much more sensitive than the classical antenna-based RF sensors and are expected to cover the entire RF spectrum [184]. E.g., in [280], researchers use some distributed entangled RF-photonic quantum sensors to estimate the amplitude and phase of a radio signal, and the estimation variance beats the standard quantum limit by over 3 dB. Thus, QSNs may have a great potential in accurate localization of wireless transmitters, which is a problem of great significance in many applications.

Quantum Sensor Model. Impact on a quantum sensor due to a physical phenomenon is typically modeled by an appropriate unitary operator that results in a quantum phase change [66]. Below, we model the change in the quantum phase of a sensor's state due to an RF-received signal. Since the RF received signal (and thus the change in quantum phase) depends on the sensor's distance from the transmitter, we can use the phase change that occurs during the sensing period to localize an RF transmitter.

<u>Sensor's Hamiltonian.</u> A quantum sensor's Hamiltonian $\hat{H}(t)$ is a sum of two[1] components [66]:

$$\hat{H}(t) = \hat{H}_0 + \hat{H}_V(t)$$

where $\hat{H}_0$ is the internal Hamiltonian of the system and $\hat{H}_V(t)$ is the change in the Hamiltonian due to an external signal $V(t)$. The internal Hamiltonian $\hat{H}_0$ remains fixed and is equal to $E_0 |0\rangle \langle 0| + E_1 |1\rangle \langle 1|$, where E_0 and E_1 are energies corresponding to the $|0\rangle$ and $|1\rangle$ states respectively. The signal Hamiltonian $\hat{H}_V(t)$ is given by:[2]

$$\hat{H}_V(t) = -\frac{1}{2}\gamma V_\parallel(t)\hat{\sigma}_z$$

where σ_z is the Pauli-Z matrix, $V_\parallel(t)$ is the parallel component of the signal $V(t)$, and γ is the coupling of the qubit to the parallel component. In essence, the above induces a change in the spin in the z axis direction resulting in a qubit *phase shift*. Above, $V_\parallel(t)$ at the sensor is given by:

$$V_\parallel(t) = E \sin(2\pi f t + \theta)$$

---

[1] The third component of control Hamiltonian is chosen to tune the sensor in a controlled way [66]; we assume $\hat{H}_{control} = 0$ in our analysis [79].

[2] Here, we ignore the transverse component of $\hat{H}_V(t)$ [79], since, in most sensor applications, the energy difference $\Delta E = E_1 - E_0$ is much higher than the energy changes introduced by the signal $V(t)$ [66].

where E is the signal's (electric field) maximum amplitude, f is the signal frequency and θ is the signal's phase.

Evolution Unitary Operator. Assume at time $t = 0$, the quantum state is $|\phi_0\rangle$. Then at time $t = t'$, the state $|\phi_{t'}\rangle$ is,

$$|\psi_{t'}\rangle = \hat{U}(0, t') |\psi_0\rangle$$

where the time evolution unitary operator $\hat{U}(0, t')$ due to the signal is given by:

$$
\begin{aligned}
\hat{U}(0, t') &= e^{\frac{i}{\hbar} \int_0^{t'} \hat{H}_V(t) dt} \\
&= e^{\frac{i}{\hbar} \int_0^{t'} (-\frac{1}{2}\gamma V_\|(t)\hat{\sigma}_z) dt}
\end{aligned}
$$

where $\hbar = 6.626 \times 10^{-34} J \cdot s$ is the plank constant. The unit of coupling γ is $J/(V \cdot m^{-1})$, and the unit of $V_\|(t)$ is $V \cdot m^{-1}$.

Phase Shift over a Sensing Time Window. Let us represent $\hat{U}(0, t')$ as [161, 315]

$$\hat{U}(0, t') = e^{-\frac{i}{2}\phi\hat{\sigma}_z} \tag{4.1}$$

where the phase shift $\phi = \int_0^{t'} \frac{\gamma}{\hbar} V_\|(t) dt$, accumulated during the sensing time $[0, t']$ due to the signal $V(t)$ is estimated as follows. Note that $V_\|(t)$ is a sinusoidal function— and hence, the phase shift in one full cycle ($t' = 1/f$) is zero. To address this, we invert the qubit whenever the sinusoidal function turns from positive to negative using a π pulse [66]. Thus, the accumulated phase in one cycle $\phi = \int_0^{1/f} \frac{\gamma}{\hbar} V_\|(t) dt = \frac{2}{\pi\hbar}\gamma E \frac{1}{f}$. Since the sensing time t' is expected to be much larger than $1/f$, the phase shift accumulated during the sensing time $[0, t']$ can be estimated by:

$$\phi = \frac{2}{\pi\hbar}\gamma E t' \tag{4.2}$$

Thus, for a fixed sensing time duration, the phase shift in the sensor's quantum state accumulated due to the signal is proportional to E, the signal's maximum amplitude, which is a function of the distance from the transmitter (see §4.5).

Impact on Multiple Quantum Sensors. Consider a set of m quantum sensors distributed over an area, with a global m-qubit quantum state of $|\psi_0\rangle$. Consider a transmitter at a certain specific location in the area. Let $\hat{U}_i$ be the impact on the ith sensor due to the transmitter over a sensing time window. Then, the overall impact on the *global* quantum state is represented by a *tensor product* of m individual unitary operators, i.e., $\bigotimes_{i=1}^{m} \hat{U}_i$, and the evolved global state state is $\bigotimes_{i=1}^{m} \hat{U}_i |\psi_0\rangle$.

Problem Definition. Consider a network of quantum sensors distributed in a geographic area and a potential transmitter/intruder in the area. Let the initial state of the system of quantum sensors network be $|\psi_0\rangle$. As described above, due to the transmission from the intruder, the quantum state evolves to $|\psi_{t'}\rangle = \hat{U} |\psi_0\rangle$ over a period of time t'. The transmitter localization problem is to determine the location of the transmitter based on the evolved quantum state $|\psi_{t'}\rangle$.

Related Work. Radio transmitter localization using a set of sensors/receivers has been widely studied [57, 91, 299]. Localization methods can be roughly classified into two types: geometry-based and fingerprinting-based. The geometry-based method includes multilateration (by measuring time-of-flight between the transmitter and multiple sensors) or triangulation (by measuring angle-of-arrival (AoA) of the transmitter at multiple sensors) [289]. The fingerprinting-based method [14] has a training stage that records the signal fingerprint for certain locations; Localization is then achieved by matching the real-time signal to the recorded fingerprints. Here, a fingerprint for a transmitter location may be a vector of received signal strengths (RSS) [299] at the sensors. Localization of simultaneously-active multiple transmitters is more challenging, and has been addressed in recent works [304, 305, 307].

Recently, there have been some works that have used quantum technology to investigate intruder/transmitter localization-related problems. E.g., [231] develops a scheme to improve the *size* of the fingerprints used in the above-described fingerprinting approach, by encoding classical sensor data into qubits through quantum amplitude encoding. In addition, [238] derives analytical equations to compute the AoA of an incoming RF signal using four entangled distributed quantum sensors, without any evaluations. [191] proposes a quantum sensor network using Mach-Zehnder interferometers to detect (not localize) intruders for surveillance purposes. Finally, [110, 308] investigate the optimization of the initial state in discrete-outcome quantum sensor networks and show that an entangled initial state yields higher measurement accuracy in some applications. In particular, QuCS lecture series [156] have successfully enhanced the visibility of quantum computing software and system level worldwide.

Parameter Estimation using Quantum Sensors. Prior works on parameter estimation using quantum sensors include: estimation of single [96] or multiple independent parameters [208], estimation of a single linear function over parameters [7], and estimation of multiple linear functions [221]. Our transmitter localization problem can be looked upon as a novel single parameter (TX location) estimation problem based on sensor measurements that are functions (based on signal propagation model and distance) of the parameter being estimated.

4.3 Methodology and Our Approach

Quantum State Discrimination (QSD). Given a quantum state $|\phi\rangle$ that is known to be equal to one of the states (known as *target states*) in the set $\{|\phi_1\rangle, |\phi_2\rangle, \ldots, |\phi_n\rangle\}$, the quantum state discrimination (QSD) problem is to determine which state $|\phi\rangle$ really is. In general, each target state $|\phi_i\rangle$ may be associated with a prior probability q_i; in this paper, we assume uniform prior. The QSD problem is typically solved using a series of measurements or a single measurement—as defined below. It is known that if the target states $\{|\phi_i\rangle\}$ are not mutually orthogonal, then there is no quantum measurement capable of perfectly (without error) distinguishing the states. Thus, a QSD solution may give an erroneous answer—i.e., guess the state to be in $|\phi_i\rangle$ when the state is really in $|\phi_j\rangle$ for some $i \neq j$. Thus, a QSD solution is associated with an overall *probability of error* (PoE), and the optimization goal of the QSD problem

is to determine the measurement (or a sequence of measurements) that minimizes the PoE. We note that in our developed schemes, we don't actually solve the QSD problems that arise due to the impracticality of implementing the general POVMs, as discussed later; instead, we just use the standard POVM known as pretty good measurement (PGM).

<u>General Measurements.</u> A general measurement [197] is defined by matrices $M_1, M_2, \ldots,$ M_n such that $\sum_i M_i^\dagger M_i = I$ where $M_i^\dagger$ is the conjugate transpose of M_i. If this general measurement is carried out on a pure state, we see the outcome "i" with probability $p(i) = \langle \phi | M_i^\dagger M_i | \phi \rangle$. Thus, if we associate the outcome "i" with the given state $|\phi\rangle$ being in the target state $|\phi_i\rangle$, the probability of error (PoE) for the given measurement $\{M_i\}$ is given by $\sum_i \sum_{j \neq i} \langle \phi_i | M_j^\dagger M_j | \phi_i \rangle$.

If we are only interested in the probability of outcomes (as in our context), the above general measurement can also be represented by the set of positive semi-definite matrices (PSD) $\{E_i = M_i^\dagger M_i\}$ where $\sum_i E_i = I$. This representation is called positive-operator valued measure (POVM); in this paper, we use this representation of measurement for simplicity.

Core Idea: TX Localization as QSD. Consider a geographic area where a transmitter can be at a set of potential locations $\{l_1, l_2, \ldots, l_n\}$. For simplicity, let us assume that the transmission power is constant. Let the initial state of the quantum system, composed of say m distributed quantum sensors, be $|\psi_0\rangle$. When the transmitter T is at a location l_i, let the impact of the T's transmission from location l_i evolve the overall state of the quantum system to $|\psi_i\rangle$ based on the model described in the previous section. Now, consider the set of target states $\{|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_n\rangle\}$ corresponding to the set of potential locations of the transmitter. Then, localizing the transmitters, i.e., determining the location l_i from where the transmission occurred, is tantamount to solving the QSD problem with the target states $\{|\psi_i\rangle\}$. Thus, determining the state of the quantum system yields the transmitter location.

<u>Selection of Initial State and Measurement.</u> In the above context, our goal is to select an initial state $|\psi_0\rangle$ and the POVM measurement (i.e., PSD matrices $\{E_1, \ldots, E_n\}$, one for each potential outcome/location) such that the overall PoE is minimized — for a given setting of transmitter location, quantum sensors, and signal propagation model. The optimization problem of selecting an optimal combination of initial state and POVM in our context is beyond the scope of this work. Here, we use a non-entangled uniform superposition pure initial state $|\psi_0\rangle = \sum_{i=0}^{2^m - 1} \frac{1}{\sqrt{2^m}} |i\rangle$. For a given initial state and target states, determining an optimal POVM can be shown to be a convex optimization problem and can be solved using an appropriate semi-definite program (SDP) [81]. However, due to scalability challenges in solving the SDP, whose size is exponential in the number of quantum sensors involved, in this paper, we use a well-known measurement known as *pretty-good-measurement* (PGM) which is known to perform well in general settings [107]. The PGM POVM is given by:

$$E_i = q_i \rho^{-1/2} \rho_i \rho^{-1/2} \tag{4.3}$$

where q_i is the prior probability and $\rho_i = |\psi_i\rangle \langle \psi_i|$ is the *density matrix* of the ith target state ψ_i, and $\rho = \sum_i q_i \rho_i$.

77
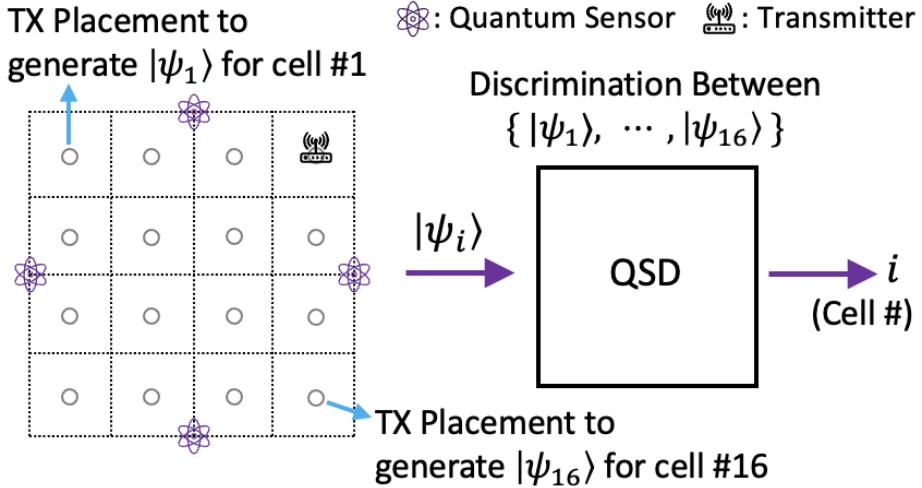
Figure 4.2: `OneLevel` Scheme.

Basic `OneLevel` Scheme; Key Challenges. The above-described methodology is essentially our basic `OneLevel` localization scheme, see Fig. 4.2. That is, the `OneLevel` scheme localizes the transmitter by first determining the set of target states $\{|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_n\rangle\}$ corresponding to the centers of the cells (a set of transmitter locations), and then, localizes the transmitter in real-time by performing QSD over the evolved quantum state using PGM measurement. Note that we use only the cells' centers to generate the target states, and also that the predicted location of the transmitter is always a cell's center in the QSD-based schemes, since the QSD-based schemes are fundamentally classification of the transmitter location into cells. *However, during evaluation, the actual location of the transmitter can be anywhere in the area*—presumably, non-center locations of the transmitter may incur higher localization errors.

The key challenges in the `OneLevel` scheme are twofold: (i) It is likely to incur a high probability of error due to a large number of target states (equal to the number of potential transmitter locations). (ii) A global POVM measurement over a large number of sensors can be difficult to implement in practice [295]; even ignoring the communication cost of teleporting the qubits to a central location, the main challenge arises due to the complexity of the circuit or hardware required to implement a POVM over a large number of qubit states. We address these challenges by designing a two-level localization scheme as described below; in the following section, we further address the above challenges by designing non-QSD based schemes.

`POVM-Loc` **Scheme**. `POVM-Loc` solves the above-mentioned challenges by localizing the transmitter by using two levels of POVMs, with each POVM requiring a measurement over a much fewer number of sensors and with a much fewer number of possible target states. We discretize the given area into a grid; each unit of the grid is called a *cell*. A *block* is a group of neighboring cells that form a rectangle. In Fig. 4.3 (a), a grid has 4×4 cells and 2×2 blocks. The thick dotted lines depict the blocks while the non-thick dotted lines depict the cells. In general, for a $N \times N$ grid with N^2 cells,
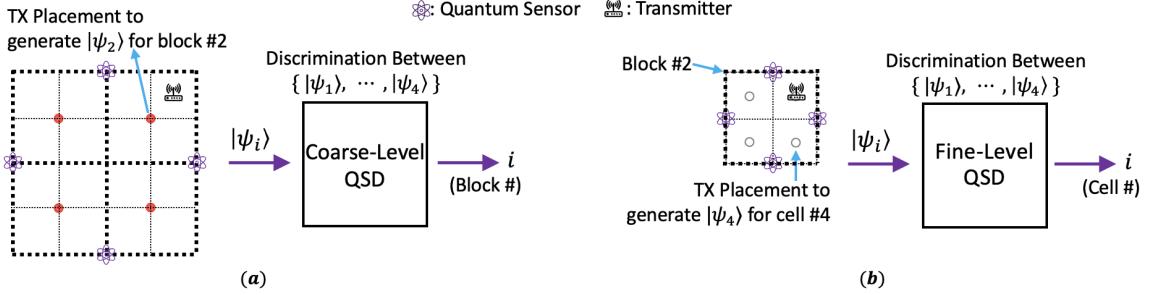
Figure 4.3: `POVM-Loc` Scheme. (a) Coarse-level localization phase, and (b) Fine-level localization phase.

we construct blocks by dividing the entire grid into $\sqrt{N} \times \sqrt{N}$ blocks — yielding N blocks in the whole area, with each block comprised of $\sqrt{N} \times \sqrt{N} = N$ cells. Without loss of generality, we assume $\sqrt{N}$ to be an integer in our discussion. The basic idea of the `POVM-Loc` scheme is to localize the transmitter in two stages: first, localize the transmitter at a block level (Fig. 4.3 (a)); and then, within that block, localize the transmitter at the cell level (Fig. 4.3 (b)). The sensors, target states, and POVMs used for localization at these two stages are different. Such a two-stage localization scheme naturally addresses the above-mentioned challenges by reducing both the number of sensors as well as target states required at each stage. We describe the scheme in more detail below.

Coarse-Level Localization. The coarse level concerns localizing the transmitter at the block level, and is done based on *coarse-level sensors* deployed over the entire given area. The target states for the coarse-level QSD/localization are the states corresponding to the location at the center of each block in the given area. As mentioned above, since the number of blocks is N, the number of target states for the Coarse-Level localization is N. The POVM measurement for the coarse-level localization is constructed using Eqn. 4.3 for the PGM measurement over the target states derived from the impact of the transmitter at coarse-level discrete locations (i.e., the center of the blocks) on the coarse-level sensors. Note that in reality, the transmitter is likely not at the center of the blocks—but, we stipulate that a block's center is a reasonable representative of the actual locations of the transmitter in that block. More formally, let $\{L_1, L_2, \ldots, L_N\}$ denote the centers of the blocks in the area, and S be the coarse-level sensors. Let $\hat{U}_i$ denote the impact on S when the transmitter is at location L_i. Then, the target states for the coarse-level localization are $\{\hat{U}_i |\psi_0\rangle\}$ where $|\psi_0\rangle$ is the initial state of S. These target states are used to determine the POVM measurements as per Eqn. 4.3, and thus, determine the block.

Fine-Level Localization. Once the transmitter has been localized within a block B via coarse-level localization, the transmitter is then localized at a cell level within B. For fine-level localization, each block B has a set of fine-level sensors $S(B)$ deployed within B (which need not be disjoint from the coarse-level sensors). The target states for fine-level localization within B correspond to the potential locations of the transmitter within B which are the centers of the cells within B, see Fig. 4.3 (b), and is derived from the impact of the transmitter's signal at the fine-level sensors
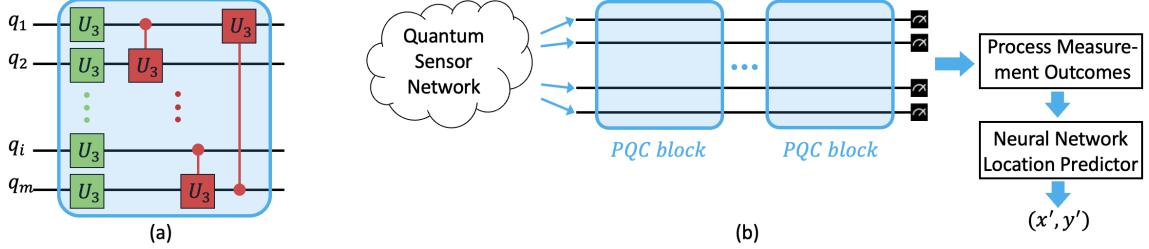
Figure 4.4: (a) Our parameterized quantum circuit (PQC) block, for the general case of m qubits. It contains m number of U_3 gates and m number of CU_3 gates. (b) The hybrid quantum-classical circuit to localize a transmitter. It consists of multiple PQC blocks, followed by classical processing of measurements, and finally, a neural network-based location predictor. We use only four blocks of PQCs in our hybrid circuit.

$S(B)$. Note that at the fine-level localization phase, only the sensors $S(B)$ where B is the block selected in the previous coarse-level localization are involved. Note that $S(B_1)$ and $S(B_2)$ from two different blocks need not be disjoint. More formally, let $\{l_1, l_2, \ldots, l_N\}$ denote the centers of the cells in the block B selected by the coarse-level localization phase, and $S(B)$ be the fine-level sensors. Let $\hat{U}_i$ denote the impact on $S(B)$ when the transmitter is at location l_i. Then, the target states for the fine level localization are $\{\hat{U}_i |\psi_0\rangle\}$ where $|\psi_0\rangle$ is the initial state of $S(B)$. These target states are used to determine the POVM measurement as per Eqn. 4.3, and thus, determine the cell within the block B, which is the TX location. As mentioned before in the one-level scheme, we note that, during evaluation, the location of the transmitter can be anywhere in the area, even though we have only use the cells' centers to generate the target states.

Multi-shot Discrimination. The quantum measurement is intrinsically probabilistic and the single-shot discrimination can incur a high probability of error. One way to reduce this probability of error is to repeat the discrimination process many times and pick the most frequent measurement outcome. Such repeated measurements are commonly done in quantum sensing [66] and computing [232]. In our context, the repetitions are done while the transmitter remains fixed.

4.4 Parameterized Quantum Circuit Based Localization

Motivation. The QSD-based method discussed in the previous section has a solid mathematical foundation, but its practical implementation is non-trivial and even infeasible for a large sensor network and/or a large number of potential transmitter locations. In particular, the POVM measurement operator (derived from the QSD problem or corresponding to the pretty-good measurement) can be infeasible to implement for a large number of outcomes/locations. The issue is somewhat mitigated by using a two-level approach as described above, but the PoE (probability of error)

in the first (coarser) level can still be high due to imperfect training (as we assign a single outcome to a *set* of target locations). A potential approach to address the above challenge is to "translate" a given POVM into an appropriate quantum circuit comprised of quantum gates and simple measurements (e.g., projective and/or computational basis) [295]. E.g., [124] presents a technique to convert POVM operators to such quantum circuits. However, the computational time incurred in translating a POVM operator into a quantum circuit is exponential to the number of qubits and is thus infeasible. In addition, the translated quantum circuits are also sub-optimal in terms of the number of CNOT gates used [124]. Also, note that the POVM computed in our QSD-based method is sub-optimal to begin with.

In this section, we develop a machine learning (ML) technique to actually learn a quantum circuit that represents the processing and measurement protocol needed to localize the transmitter from the evolved quantum state. The learned quantum circuit model maps the global evolved quantum state to the transmitter location. In essence, we avoid computing the POVM (from QSD, or using the pretty-good measurement) altogether (and thus, also avoid the challenge of translating it to a quantum circuit), and instead learn the required quantum circuit representing the measurement protocol. To facilitate learning the quantum circuit, we use an appropriate parameterized quantum circuit (PQC) and learn its parameters—as in [202] wherein a POVM is trained using parameterized quantum circuits. Our PQC-based localization method based on the above insights is described below. We start with a brief introduction to PQCs.

Parameterized Quantum Circuits (PQC). Parameterized quantum circuits have emerged as a powerful tool in quantum computing [18], providing an adaptable framework for tackling diverse computational tasks. Parameterized quantum circuits (PQCs) can be regarded as machine learning models with remarkable expressive power; just like classical ML models, PQC circuits/models are trained to perform data-driven tasks. PQC's security can be enhanced via distributing the circuits to multiple quantum cloud providers [265]. PQCs offer several advantages over fixed quantum circuits [18, 252, 253], including:

1. Adaptability. The parameters in PQCs can be adjusted to tailor the circuit for a specific problem, allowing a single circuit structure to be repurposed for various tasks.

2. Trainability. PQCs can be trained using classical optimization algorithms to solve optimization problems and machine learning tasks, making them a vital component of hybrid quantum-classical algorithms.

3. Noise Resilience. PQCs can be more robust against noise and errors in near-term quantum devices, as they allow shorter-depth circuits that reduce the impact of errors.

In essence, PQCs are quantum circuits comprised of parameterized gates and measurements. Commonly used parameterized gates in PQCs include rotation gates $R_x(\theta), R_y(\theta), R_z(\theta)$ which represents rotating about the X, Y, Z axis respectively with

angle θ. A more versatile gate is the $U_3(\theta, \phi, \lambda)$, which can be used to generate any single-qubit operation by setting the appropriate values for the parameters; U_3 gate can be decomposed into simpler R_x, R_y, R_z gates. The parameterized $CU_3(\theta, \phi, \lambda)$ gate is the controlled version of the U_3 gate; it applies the U_3 gate only when the control qubit is in the $|1\rangle$ state. We use a combination of U_3 and CU_3 gates in our parameterized quantum circuits.

PQC-based Localization Method. At a high level, in our PQC-based localization scheme, the QSN data is fed into a trained hybrid quantum-classical model, which represents the overall measurement strategy and thus outputs the transmitter location. Our hybrid quantum-classical model (see Fig. 4.4(b)) consists of the following three components. (i) Parameterized Quantum Circuit (PQC), (ii) Processing the measurement outcomes, (iii) Neural network location predictor, to convert the processed measurement outcomes to the transmitter location. We describe each of the above components below.

1. Parameterized Quantum Circuit (PQC) Design. The parameterized quantum circuit can be designed in many ways. We design our PQC component based on some common PQC-design patterns [154, 266] used in prior works. For example, in [164], a block of PQC contains one layer of ZZ gates and one layer of R_y gates. In [183], a block of PQC contains one layer each of R_x, R_y, R_z, CZ gates. In our scheme, the quantum circuit is composed of blocks, and each block is a combination of U_3, CU_3 gates; we use these two gates in our design as they form a universal gate set and are widely used in PQC circuits. Circuits consisting U_3 and CU_3 gates have a high expressive power as each gate has three trainable parameters. In particular, given N number of input qubits, a block consists of N number of U_3 and N number of CU_3 gates. See Fig. 4.4. In a block, each input qubit is first operated on by the unary U_3 gate in parallel, forming a layer of U_3 gates. Then, there is a series of CU_3 gates following a ring connection pattern, i.e., each CU_3 is executed over two "consecutive" qubits (with the first being the control qubit) except for the last CU_3 gate which is over the last and the first qubit. Thus, a single block has a circuit depth of $N + 1$. The overall PQC may have a series of above blocks—the expressive power of the model increases monotonically with the increase in the number of blocks. In our evaluations (§2.5), we used four blocks as we observed that four blocks provide good performance while having a modest circuit depth. After the blocks, the PQC ends with the measurement on the standard computational basis, i.e., the Pauli Z basis.

2. Process Measurement Outcomes. As in the QSD-based schemes, we will use the PQC to make repeated measurements. To use the repeated measurements effectively for location prediction, we characterize the set of repeated measurement results by expectation values, one for each qubit. In particular, we compute the expectation value $\langle Z \rangle$ of the Pauli Z operator (which represents the measurement in the computational basis), and feed as input to a neural network for final location prediction as described below. We note that, for a quantum state $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, the expectation value $\langle Z \rangle$ of the Pauli Z operator is given by $\langle \psi | Z | \psi \rangle = |\alpha|^2 - |\beta|^2$.

3. Neural Network to Predict Location. We consider two variants of our neural network predictor: (i) Classifier variant. which outputs a class/label corresponding to

the cell where the TX is located, and thus, predicts the location to be the cell's center. (ii) Regression variant, that outputs locations as the x and y coordinates.

Classifier Variant. In [274], a novel quantum label-based supervised learning approach is proposed to perform multi-classification. Our overall circuit with the Classifier component for the location prediction essentially equates to a circuit for quantum state discrimination (QSD), as the QSD problem also outputs a finite number of discrete outcomes. For the Classifier Variant, we use a simple neural network with only an input layer and an output later, having no hidden layers—i.e., a single fully connected layer. The input neurons are the expectation values of the Pauli Z operator from the measurements as described above, and the output neurons represent the cell labels. See Fig.4.5(a), which shows the fully connected layer for a network of 4 quantum sensors deployed in a 4×4 grid with 16 cells.

Regression Variant. The Classifier Variant outputs locations in a discrete space—which is fundamentally sub-optimal if the transmitter can be anywhere in the 2D space. To output the predicted location in the continuous 2D space, we use a Regression Variant that outputs the location as an (x, y) point. For the setting wherein the transmitter may be located anywhere in the 2D space, the Regression Variant should have a smaller localization error. Fig.4.5(b) shows the fully connected layer for the Regression Variant; the number of output neurons is always two, i.e., a X coordinate and a Y coordinate.

<u>4. Loss Function.</u> During training, the gradient of the loss function is back-propagated through the neural network and the quantum circuit parts, so that the parameters within these parts can be appropriately updated. The loss functions used for the Classifier Variant and the Regression Variant are different; for the Classifier variant, we use the cross-entropy loss function while for the Regression variant, we use the mean square error loss function.

`PQC-One` **and** `PQC-Two` **Schemes.** The above-described hybrid quantum-classical model is essentially our `PQC-One` localization scheme. By using `PQC-One` as a building block and using the same two-level (coarse, fine) idea described in §4.3, we design the `PQC-Two`, corresponding to the two-level QSD-based schemes described in the previous section. At the first coarse level, the output of the "coarse-level `PQC-One`" will determine the block the transmitter is in. Then at the second fine level, the output of the "fine-level `PQC-One`" tied to the block determined by the coarse level is the final location output. The PQC-based schemes essentially use the trained circuit in lieu of the POVM used in the QSD-based schemes. The PQC-based schemes can be used with either the Classifier or the Regression variant for the last predictor component.

4.5 Evaluation

In this section, we evaluate our developed schemes. We make two observations, which are as expected: (1) Performance of two-level methods is better than one-level methods in general, and (2) Performance of the PQC-based methods is superior to the QSD-based methods. In summary, our schemes are able to achieve meter-level
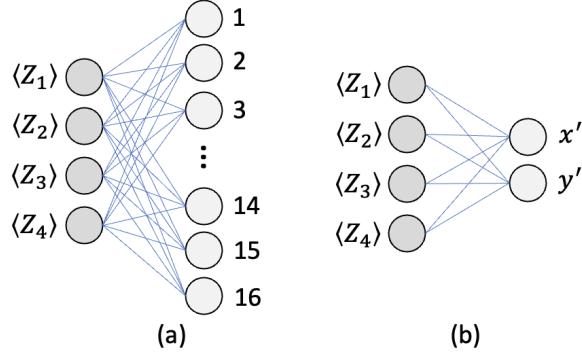
Figure 4.5: Neural network (a fully connected layer) for 4 quantum sensors to predict the location from processed measurements. (a) Classifier Variant, (b) Regression Variant.

(1-5m) localization accuracy, and near-perfect (99-100%) classification accuracy in the case of discrete locations.

4.5.1 Evaluation Settings

Algorithms Evaluated. We evaluate four algorithms: `OneLevel`, `POVM-Loc`, `PQC-One`, `PQC-Two`. As the name implies, `OneLevel` and `PQC-One` are one-level methods, while `POVM-Loc` and `PQC-Two` are two-level methods. Similarly, `OneLevel` and `POVM-Loc` are QSD-based methods, while `PQC-One` and `PQC-Two` are PQC-based methods. We use the Regression Variant in our PQC-based methods by default, since in our default setting the transmitter can be anywhere in the 2D space. Our code[3] is written in Python and uses Numpy and Scipy libraries to perform matrix-related operations.

QSD-based Method Implementation. To implement the QSD-based methods, we first determine the target states which are then used to construct the pretty-good-measurement POVM via Eqn. 4.3. To localize a transmitter, we first compute the evolved state and then, use the POVM to determine the target state or the TX location. This is done repeatedly as described in Section 4.3, and in two levels (coarse, fine) depending on the localization scheme. The target states and evolved states are both generated using the sensor model described in Section 4.2, i.e., using Eqns. 4.1-4.2, with the electric field strength (E) and phase shift range modeled as below.

Generating Sensor Readings. The crux of determining target states, computing the evolved states, and simulating the training datasets is to compute the phase shift picked up by the quantum sensors due to the signal during the sensing process. In Eqn. 4.2, we modeled the phase shift as a function of the electric field strength and the sensing time. Thus, we need a model for the electric field strength. In free space, the electric field strength produced by a transmitter with an isotropic radiator can

---

[3]https://github.com/caitaozhan/QuantumLocalization

be approximated as [271]

$$E = \frac{\sqrt{30 \cdot P}}{d} \times (1 + noise)$$

where E is the electric field strength in $V \cdot m^{-1}$, P is the transmitter power output in W (watt), and d is the distance from the radiator in m. Since in most quantum sensing applications, the signal to be sensed are weak signals, here we assume the power of the transmitter $P = 0.1\mu W$. Ideally, the strength of the electric field is inverse to the distance between the transmitter and the sensor. But in reality, the relationship is more complicated. So, we add a random uniform variable $noise \in [-0.05, 0.05]$ to incorporate reality in a simple way. The target states and thus the POVMs are computed assuming zero noise during training, while during localization, the signal received is assumed to contain noise. The simulated datasets for PQC-based methods are assumed to contain noise too.

Range of Phase Shift ϕ. We set the sensing time t' to 1 millisecond[4]. As mentioned later, our grid cells are $10m \times 10m$, and we assume 5 meters to be the minimum distance allowed between a transmitter and a quantum sensor. Thus, we choose the coupling constant γ to be such that a quantum sensor at 5 meters away from the transmitter accumulates a phase shift of 2π during the sensing time t'; this entails that the maximum phase shift is 2π and the minimum phase shift is as low as 0 (when the sensor is very far away from the transmitter).

PQC-Based Methods Implementation and Training. Different than the QSD-based methods, the PQC-Based methods involve quantum circuits. We use the publicly available TorchQuantum [252] library to implement and train the parameterized hybrid circuits. TorchQuantum's classes are inherited from a core class of PyTorch [201], which is used to implement the neural network predictor. Thanks to PyTorch, we are able to train the PQCs fast on a GPU. We use the Adam optimizer and train for 80 epochs for both `PQC-One` and `PQC-Two` methods.

The sensor readings are also used as the sensor data to train the PQC-based hybrid circuit models. Essentially, for a fixed initial global state of the sensors (say, ψ), each sample consists of the quantum state received from the quantum sensor network (input feature) and the location of the transmitter (ground truth target). More formally, each sample is of the kind: $(\bigotimes_{i=1}^{m} \hat{U}_i |\psi\rangle, L)$, where $\hat{U}_i$ is the evolution unitary operator for the ith quantum sensor (as per §4.2 and above paragraphs), $|\psi\rangle$ is the uniform superposition initial state, and L is the location of the transmitter in the field in all scenarios except for one, i.e., L is the block number for samples used to train a "coarse-level `PQC-One`" in the `PQC-Two` Classifier Variant. We use one hundred training examples/samples for each cell, with the transmitter's location randomly scattered over the cell. For example, consider a 4×4 grid with a block length of 2. The training dataset for `PQC-One` will have $16 \times 100 = 1600$ samples. And `PQC-Two` will have $16 \times 100 = 1600$ samples in the first level to train a "coarse-level `PQC-One`", and $4 \times 400 = 1600$ samples in the second level to train 4 blocks each requiring

---

[4]In principle, the sensing time period must be smaller than the decoherence time, which varies across different quantum technologies.

Performance of Localization Algorithms
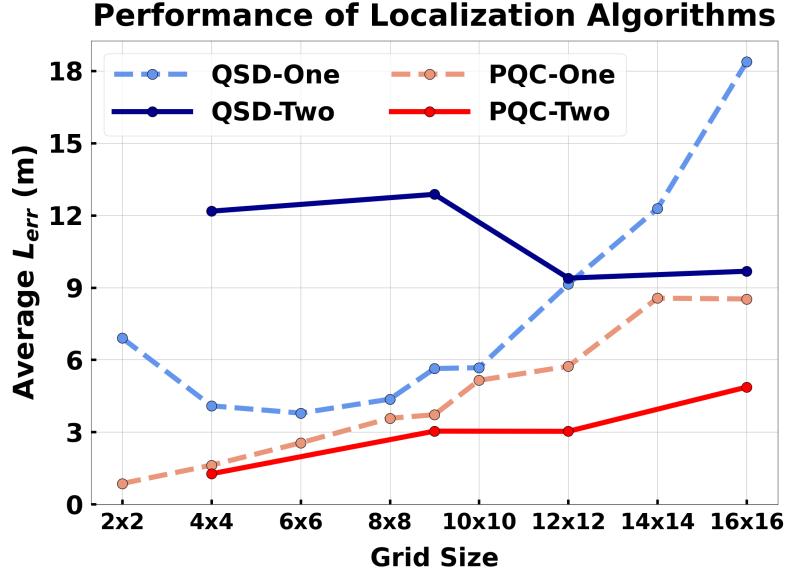
Figure 4.6: The performance of `OneLevel`, `POVM-Loc`, `PQC-One`, `PQC-Two` for varying grid size and 8 quantum sensors.

$4 \times 100 = 400$ to train a "fine-level `PQC-One`". Thus, there are a total of 3600 training samples used to train 5 models in a `PQC-Two` method.

Quantum Sensor Deployment. We deploy sensors uniformly over the area; for the `POVM-Loc` and `PQC-Two` schemes, we deploy the fine-level sensors along the block borders so that the sensors can be used by the two neighbor blocks, i.e. fine-level sensors for the blocks are not disjoint. We use a maximum of 8 quantum sensors for any single QSD instance—since the memory and computing requirements for storing and implementing a POVM become prohibitive beyond that. E.g., a POVM for 256 target-states over 12 sensors requires 69 GB of main memory storage.[5] The PQC-based methods have a bottleneck on the number of sensors due to POVM considerations, but we are still limited in practice nevertheless due to training time and GPU memory; thus, we use a maximum of 16 sensors in the first level or in any block of the second level. This limits the training time to at most several hours and GPU memory requirements to 8-16 GB. We discuss more details on the number of sensors used at various levels and blocks, below. Finally, each grid cell is of size $10m \times 10m$ in all settings, and the transmitter can be anywhere in the given area except that the minimum distance between any sensor and transmitter is 5m.

<u>Two-Level Schemes: Blocks and Sensors Used.</u> As described in §4.3, for a grid $N \times N$, if N as a perfect square, the grid is divided into $\sqrt{N} \times \sqrt{N}$ blocks—with the first-level localizing the transmitter into one of the blocks, and the second-level localizing the transmitter into a cell within the block. However, in this section, to get a better insight into the performance trends, in this section, we have also considered N values that are not perfect squares. For such N values, we have determined block sizes as

---

[5]We need 256 matrices each of size $2^{12} \times 2^{12}$, with each matrix element being a complex number requiring 16 bytes.

Localization Performance in 16x16 Grid
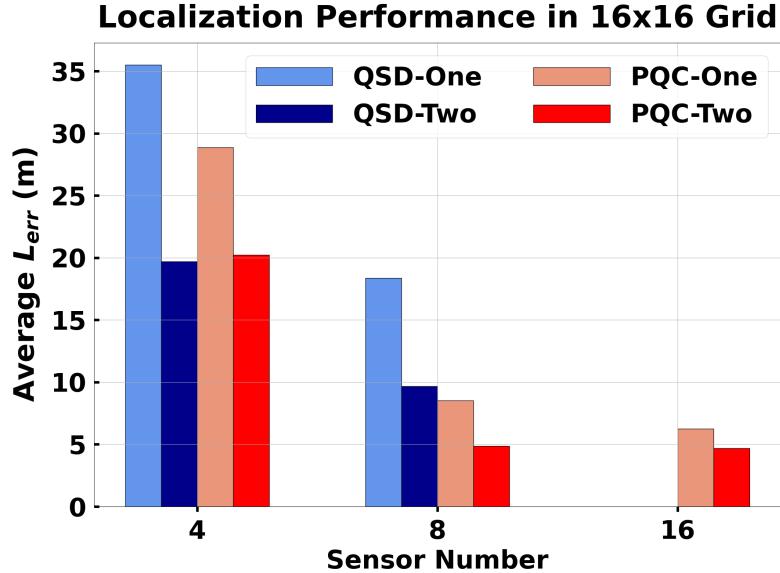
Figure 4.7: The performance of `OneLevel`, `POVM-Loc`, `PQC-One`, `PQC-Two` for varying sensor number and a 16×16 grid.

integers close to the $\sqrt{N}$; e.g., for a 12×12 grid, we divided the grid into 4×4 blocks each of 3×3 cells. In terms of the number of sensors at each level—we use up to 16 sensors in the first level of localization, but in the second level we always use exactly 4 sensors per block irrespective of the block/grid size.

Performance Metrics. We use the `Localization error` (L_{err}, in meters) as the main metric to evaluate our localization schemes. L_{err} is defined as the distance between the actual location of the transmitter and the predicated location. In all plots except the CDF plots, average L_{err} refers to the *average* localization error over many TX locations; in the CDF plots, the distribution is over many TX locations.

4.5.2 Evaluation Results

In our evaluation, we evaluate the performance of our proposed four localization algorithms' performances for varying grid size and number of quantum sensors. Note that, for one-level algorithms, the number of sensors is the total number of sensors used, while for the two-level algorithms, the number of sensors parameter is the number of sensors used in the first/coarse level (recall that, in the second level, we use only 4 sensors for each block).

Varying Grid Size. Fig. 4.6 shows the performance of all four algorithms with varying grid sizes when the number of quantum sensors is eight. We observe that the PQC-based methods have lower localization error than the QSD-based methods, and the two-level schemes generally perform better than one-schemes—except that the `POVM-Loc` performs worse than `OneLevel` for smaller grid sizes.[6] The results show

---

[6]This is likely because the QSD problem at the first/coarse level has a high error. The high error here is due to all cells being at the border edges, making the quantum state discrimination hard.

Cumulative Distribution Function of L_{err}
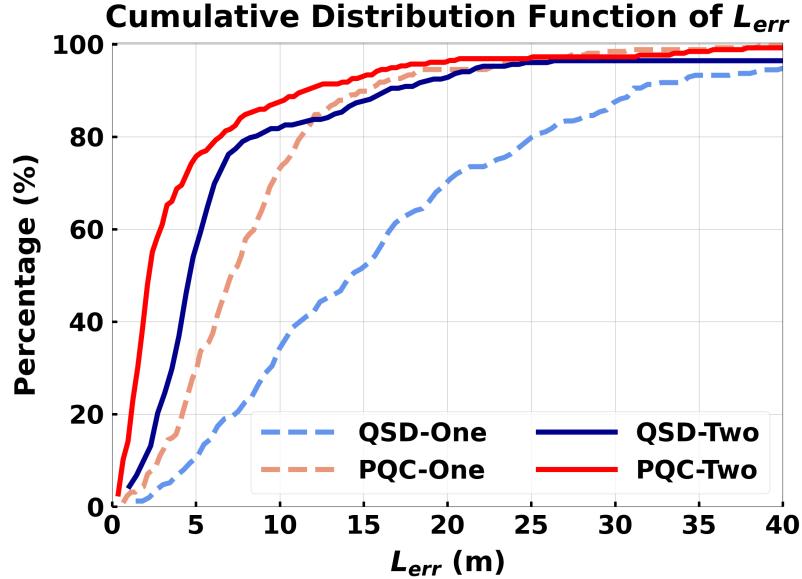
Figure 4.8: The cumulative probability of L_{err} of `OneLevel`, `POVM-Loc`, `PQC-One`, `PQC-Two` for a 16×16 grid and 8 quantum sensors.

the power of a well-trained parameterized hybrid circuit and the effectiveness of two-level schemes. More specifically, we observe that for a 16×16 grid, the average L_{err} of `PQC-Two` is $4.9m$, which is almost half of the average L_{err} of `PQC-One` at $8.5m$. Similarly, the L_{err} of `POVM-Loc` is also almost half of the L_{err} of `OneLevel`, i.e., $9.6m$ vs $18.3m$.

Varying Number of Sensors. Fig. 4.7 shows the average L_{err} in a 16×16 grid with a varying number of quantum sensors. As expected, we observe that the L_{err} improves with an increasing number of quantum sensors, for all four schemes. For the `PQC-Two` scheme, we observe that the L_{err} improvement from 8 sensors to 16 sensors is minimal, i.e. $4.9m$ vs $4.7m$. This is because having 8 sensors in the first/coarse level seems sufficient to determine the block, and then, in the second level, each block will always have 4 sensors associated with it (performance in the fine level is the same).

CDF. Fig. 4.8 shows the cumulative distribution function of L_{err} for four methods when the grid size is 16×16 and the number of sensors is 8. This plot gives insight into the *distribution* of L_{err} over different TX locations, compared with Fig. 4.7 which shows only average L_{err} across many TX locations. Here, the distribution is over 256 TX locations—one random TX location per cell for 256 cells in the 16×16 grid. We observe, as expected, that the two-level schemes are better than the one-level schemes, and the PQC-based methods are better than the QSD-based methods, except that `POVM-Loc` has a better CDF plot than `PQC-One` up to the 83-th percentile. The above exception implies that `POVM-Loc` has a higher number of locations with large L_{err} compared with `PQC-One`; this is likely due to `POVM-Loc` incurring errors in determining the block at the first/coarse level, which can lead to large localization

---

The neighboring two cells across the border of two blocks are close, thus hard to determine which block the cell is in.

Performance of Localization Algorithms
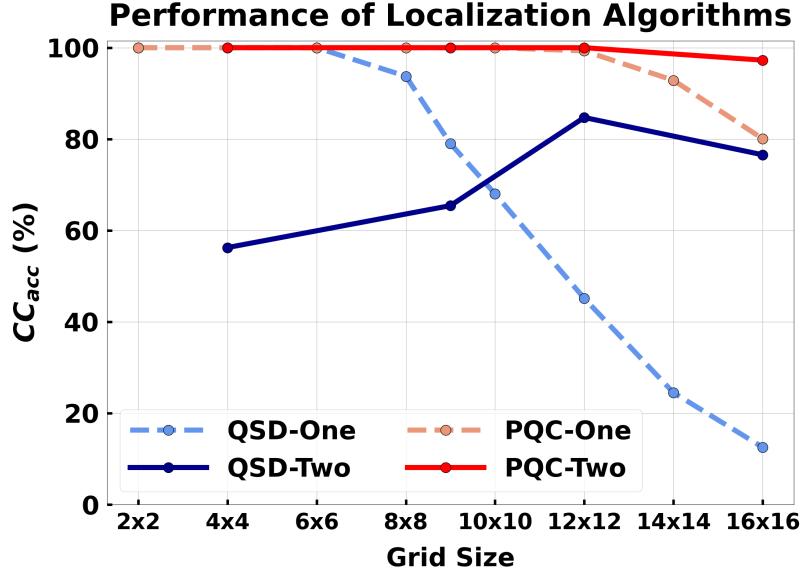
Figure 4.9: The performance of `OneLevel`, `POVM-Loc`, `PQC-One`, `PQC-Two` for varying grid size and 8 sensors.

errors.

Discrete Setting: In the previous evaluation results, we have considered the practical *continuous* setting wherein the transmitter can be anywhere in the area. To evaluate the true performance of our QSD-based methods, which are fundamentally classification strategies, we now evaluate the discrete setting wherein the transmitter is located only at the center of a cell and the predicted output of a localization method is the cell number of the transmitter. In this *discrete* setting, we evaluate the performance metric of *Classification Accuracy* L_{acc} which is the percentage of times the method is correct in predicting the *cell* number. Also, in this discrete setting, the PQC-based methods use the Classification variant in the location predictor component, while the QSD-based methods remain the same.

Fig. 4.9 shows the performance of the four algorithms with varying grid sizes when the number of quantum sensors is eight. We observe similar trend for each algorithm as well as similar relative trends among the algorithms as in the continuous setting. We make two important observations:

1. First, in the QSD-based methods, the `POVM-Loc` is a significant improvement over `OneLevel` (from 13% to 77% for grid side 16×16), which shows the effectiveness of our two-level approach.

2. Second, for the largest grid size of 16×16, the L_{acc} for QSD-based `POVM-Loc` is reasonable at 77% but is further improved impressively by `PQC-Two` at 97%; this shows the effectiveness of our PQC-based methods. The 3% error here in `PQC-Two` is mainly due to the errors in the first level of determining the block.

Also, we see that for lower grid sides, the `POVM-Loc` surprisingly performs worse than
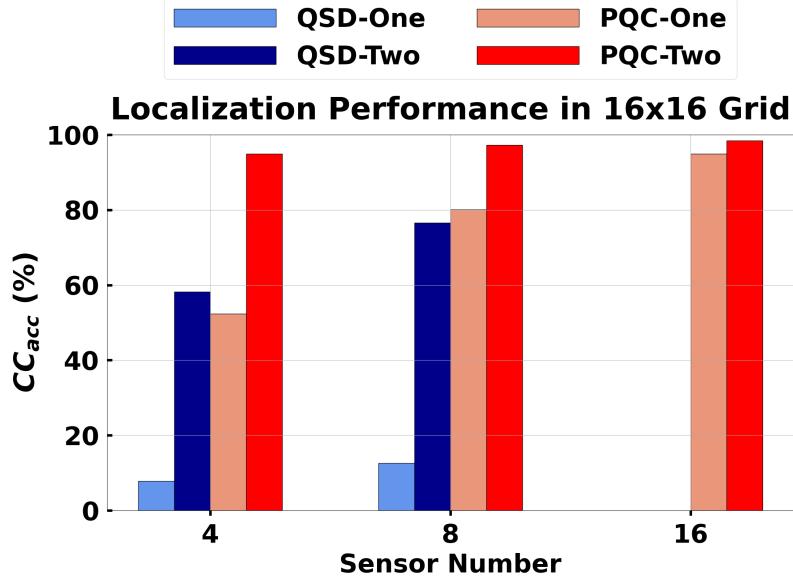
Figure 4.10: The performance of `OneLevel`, `POVM-Loc`, `PQC-One`, `PQC-Two` for varying sensor number and a 16×16 grid.

`OneLevel`; the reason for this is similar to the continuous case that determining the blocks at the first level becomes more erroneous when the grid size is small.

Fig.4.10 shows the L_{acc} in a 16×16 grid for varying number of quantum sensors. As expected, we observe that the L_{acc} improves with an increasing number of quantum sensors. More importantly, the L_{acc} for `PQC-Two` is *near-perfect* at 99% with 16 sensors; this shows the effectiveness of the two-level method as well as of the well-trained parameterized hybrid circuits. As in the continuous-domain setting, we don't show the QSD-methods for 16 sensors, as it was infeasible to implement the QSD-based methods for a large number of sensors.

4.6 Conclusion and Future Work

In this paper, we have developed effective schemes for an important quantum sensor network application, viz., localization of a wireless transmitter. The work demonstrates how a network of quantum sensors can collaborate to predict a parameter (here, the location of an event/transmitter) that is received/sensed differently at different sensor locations (e.g., depending on the distance from the event). In particular, this work shows the promise of quantum sensor networks in the localization of events in general—one of the most important applications of classical sensor networks.

Our work has significant opportunities for improvement. In particular, one can optimize the initial state of the sensors to further improve the localization performance (note that, here, we have only used a uniform superposition initial state). In this context, we are also interested in determining whether entangled initial states are helpful; recent works [92, 93, 95] have shown that entangled states can be efficiently distributed over a quantum network. In addition, one can consider general multi-

level approaches and restricted forms of measurement, design parameterized quantum circuits with noise [252], and develop techniques to distribute such circuits over a quantum (sensor) network as in [87, 239, 240]. We plan to explore some of these directions in our future work.

Chapter 5

Optimizing Initial State of Detector Sensors in Quantum Sensor Network

In this chapter, we consider a network of quantum sensors, where each sensor is a qubit detector that "fires," i.e., its state changes when an event occurs close by. The change in state due to the firing of a detector is given by a unitary operator which is the same for all sensors in the network. Such a network of detectors can be used to localize an event, using a protocol to determine the firing sensor which is presumably the one closest to the event. The determination of the firing sensor can be posed as a *Quantum State Discrimination* problem which incurs a probability of error depending on the initial state and the measurement operator used.

In this chapter, we address the problem of determining the optimal initial global state of a network of detectors that incur a minimum probability of error in determining the firing sensor. For this problem, we derive necessary and sufficient conditions for the existence of an initial state that allows for perfect discrimination, i.e., zero probability of error. Using insights from this result, we derive a conjectured optimal solution for the initial state, provide a pathway to prove the conjecture, and validate the conjecture empirically using multiple search heuristics that seem to perform near-optimally.

5.1   Introduction

Quantum sensors, being strongly sensitive to external disturbances, can measure various physical phenomena with extreme sensitivity. These quantum sensors interact with the environment and have the environment phenomenon or parameters encoded in their state, which can then be measured. Thus, quantum sensors can facilitate several applications, including gravitational wave detection, astronomical observations, atomic clocks, biological probing, target detection, etc. [206]. A study [10] has shown the advantages of microwave quantum radar in the detection of a target placed in a noisy environment by exploiting quantum correlations between two modes, probe and idler. Estimation of a single continuous parameter by quantum sensors can be further enhanced by using a group of entangled sensors, improving the standard deviation of

measurement by a factor of $1/\sqrt{N}$ for N sensors [96]. Generally, a network of sensors can facilitate spatially distributed sensing; e.g., a fixed transmitter's signal observed from different locations facilitates localization via triangulation. Thus, as in the case of classical wireless sensor networks, it is natural to deploy a network of quantum sensors to detect/measure a spatial phenomenon, and there has been recent interest in developing protocols for such quantum sensor networks (QSNs) [33, 82, 208, 210].

Initial State Optimization. Quantum sensing protocols typically involve four steps [66]: *initialization* of the quantum sensor to a desired initial state, transformation of the sensor's state over a *sensing* period, *measurement*, and *classical processing*. Quantum sensor networks would have similar protocols. In general, the initial state of the QSN can have a strong bearing on the sensing protocol's overall performance (i.e., accuracy). E.g., in certain settings, an entangled initial state is known to offer better estimation than a non-entangled state [82, 208]. If entanglement helps, then different entangled states may yield different estimation accuracy. Thus, in general, determining the initial state that offers optimal estimation accuracy is essential to designing an optimal sensing protocol. The focus of our work is to address this problem of determining an optimal initial state. Since an optimal initial state depends on the sensing and measurement protocol specifics, we consider a specific and concrete setting in this paper involving detectors. To the best of our knowledge, ours is the only work (including our recent work [110]) to address the problem of determining provably optimal initial states in quantum sensor networks with discrete outcome/-parameters.[1]

QSNs with Detector Sensors. We consider a network of quantum "detector" sensors. Here, a detector sensor is a qubit sensor whose state changes to a unique final state when an event happens. More formally, a sensor with initial state $|\psi\rangle$ gets transformed to $U|\psi\rangle$ when an event happens, where U is a particular unitary matrix that signifies the impact of an event on the sensor. Such detector sensors can be very useful in *detecting* an event, rather than *measuring* a continuous parameter representing an environmental phenomenon. More generally, we consider a network of quantum detector sensors wherein, when an event happens, exactly one of the sensors fires— i.e., changes its state as above. In general, a network of such detector sensors can be deployed to *localize* an event — by determining the firing sensor and, hence, the location closest to the event. Our paper addresses the problem of optimizing the initial global state of such QSNs to minimize the probability of error in determining the firing sensor.

Contributions. In the above context, we make the following contributions. We formulate the problem of initial state optimization in detector quantum sensor networks. We derive necessary and sufficient conditions for the existence of an initial state that can detect the firing sensor with perfect accuracy, i.e., with zero probability of error. Using the insights from this result, we derive a conjectured optimal solution for the problem and provide a pathway to proving the conjecture. We also develop multiple search-based heuristics for the problem and empirically validate the conjectured

---

[1]For estimation of continuous parameters, some works [82, 224] exist that have shown that certain initial states can saturate the quantum Cramer-Rao bound (also see §5.2.1).

solution through extensive simulations. Finally, we extend our results to the unambiguous discrimination measurement scheme, non-uniform prior, and considering quantum noise.

5.2   `ISO` Problem and Related Work
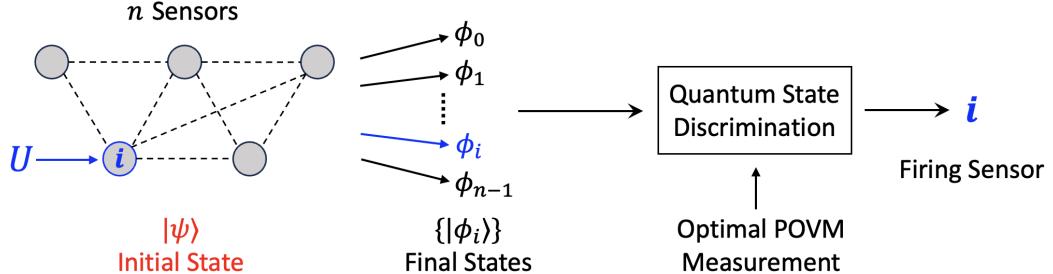
Figure 5.1: `ISO` Problem. Given n deployed quantum sensors, an event changes the state of one of the sensors (i^{th} sensor in the figure) by a unitary operator U. Quantum state discrimination with the optimal measurement is used to determine the firing sensor. The `ISO` problem is to determine the initial state (possibly, entangled) that minimizes the probability of error in discriminating the potential final states. The dashed lines connecting the sensors signify a potential entangled global state.

Setting. Consider n quantum sensors deployed across a geographical area, forming a quantum sensor network. See Fig. 5.1. Each sensor stores a qubit whose state may potentially change due to an event in the environment. Let $|\psi\rangle$ denote the initial (possibly entangled) state of the n sensors. Let U be a unitary operator that represents the impact of an event over a qubit in a sensor; here, U may describe the rotation of a spin caused by a magnetic field or a phase shift induced in a state of light by a transparent object. Let the two eigenvectors of U be $\{u_+, u_-\}$, and without loss of generality, let the corresponding eigenvalues be $\{e^{+i\theta}, e^{-i\theta}\}$ where $\theta \in (0, 180)$ degrees; thus, $U|u_\pm\rangle = e^{\pm i\theta}|u_\pm\rangle$. Let $|\phi_i\rangle = (I^{\otimes i} \otimes U \otimes I^{\otimes(n-i-1)})|\psi\rangle$, where U appears in the ith position and $i \in \{0, \cdots, n-1\}$, represents the system's state after the event affects the ith sensor. *We assume that events in our setting affect exactly one sensor with uniform probability.*[2] We refer to the n possible resulting states $\{|\phi_i\rangle\}$ as the **final states**; these final states have an equal prior probability of occurrence on an event.

Objective Function $P(|\psi\rangle, U)$. When an event results in the system going to a particular final state $|\phi_i\rangle$, we want to determine the sensor (i.e., the index i) that is impacted by the event by performing a global measurement of the system. For a given setting (i.e., $|\psi\rangle$ and U), let $M(|\psi\rangle, U)$ be the optimal *positive operator-valued*

---

[2]In essence, we assume that sensors are sparsely deployed such that an event affects at most one sensor, and that the event itself is uniformly likely to occur at the sensor locations. If there is no prior information about the event's location, then assuming uniform probability is reasonable. See §5.8, where we consider the generalization of non-uniform probabilities.

measure (POVM) measurement for discriminating the final states $\{|\phi_i\rangle\}$, i.e., the measurement that incurs the minimum probability of error in discriminating the final states $\{|\phi_i\rangle\}$ and thus determining the index i given an unknown final state. Let $P(|\psi\rangle, U)$ be the (minimum) probability of error incurred by $M(|\psi\rangle, U)$ in discriminating the final states $\{|\phi_i\rangle\}$.

`ISO` **Problem Formulation.** Given a number of sensors n and a unitary operator U, the `ISO` problem to determine the initial state $|\psi\rangle$ that minimizes $P(|\psi\rangle, U)$. In other words, we wish to determine the initial state $|\psi\rangle$ that yields the lowest probability of error in discriminating the final states when an optimal POVM is used for discrimination.

For clarity of presentation, we consider only the minimum error measurement scheme, till the last Section 5.8.1 where we extend our results to the unambiguous discrimination measurement scheme.

Potential Applications. One of the main applications of detector sensor networks is event localization. Assume we have some critical locations to monitor, and we place one quantum detector at each critical location. Then, a network of quantum detectors, wherein a detector's state changes (as represented by the unitary U), can be used to localize the event occurrence—as the location of the firing detector also gives the event's location. The event in the above scenario could be anything that can be represented by a unitary U, e.g., an event may represent the presence of a magnetic field, an acoustic event (e.g., an explosion), a signal transmission that can be detected, or movement of a detectable object.

Paper Organization. The rest of the paper is organized as follows. We end this section with a discussion on related work. In the following section (§5.3), we establish a necessary and sufficient condition for *three* final states to be orthogonal—and hence, the existence of an initial state such that $P(|\psi\rangle, U) = 0$. We generalize the result to an *arbitrary* number of sensors in §5.4, and give an optimal solution for the `ISO` problem when the orthogonality condition is satisfied. In §5.5, we use the insights from §5.4 to derive a conjectured optimal solution for an arbitrary U and the number of sensors; in the section, we also provide a pathway to proving the conjecture. In the following sections, we develop search-based heuristics for the problem (§5.6) and use these heuristics to empirically validate our conjectured solution through simulations (§5.7). In §5.8, we consider generalizations related to unambiguous discrimination measurement, non-uniform prior probabilities, and quantum noise. Finally in §5.9, we conclude and discuss some potential future work.

5.2.1   Related Work

Continuous Parameter Estimation using Quantum Sensors. In prior works [82, 221], protocols have been studied for estimation of a single parameter using multiple sensors [96], multiple parameters [149, 208], a single linear function over parameters [82, 208, 233], and multiple linear functions [7, 221]. Quantum state estimation considering nuisance parameters is reviewed in [241]. These and many other works [82, 90, 208] have also investigated whether the entanglement of sensors offers any advan-

tage in improving the estimation accuracy. Some of the above works have optimized the measurement protocols (e.g, [82, 210]) in the addressed settings, but none of the above works have addressed the problem of initial state optimization. To the best of our knowledge, all prior works have modeled the sensed parameters in the continuous domain, e.g., these parameters could be the strength of a magnetic field at different locations. In contrast, in some sense, our work focuses on the estimation of discrete-valued parameters.

Optimal State Discrimination. There has been a lot of work on quantum state discrimination [13, 16, 20, 21] – wherein the goal is to determine the optimal measurement protocol to minimize the probability of error in discriminating a set of given states. A closed-form expression is known only for two states and very specialized cases for a larger number of states. However, numerical techniques exist (e.g., SDP-based [81]). Our work differs in the following ways: (i) The set of final states we want to discriminate is very specialized. (ii) Our goal is to optimize the initial state—that minimizes the probability of error using an optimal POVM (in some sense, we implicitly assume that an optimal POVM for a given set of final states can be derived).

Initial State Optimization. Recent works have used variational circuits to seek an optimized probe state for a set of sensors, in the context of classical supervised learning [330] and (continuous) parameter estimation [136] under noise. In additional, a recent work [80] investigates estimation accuracy with different levels of entanglements for measuring a linear combination of field amplitudes. In contrast, we seek provably optimal initial state solutions. To the best of our knowledge, the only other work that has investigated the initial state optimization problem is our recent preliminary work [110] where we address the same problem as in this paper. In [110], we give an optimal solution for the case of $n = 2$ sensors, and, for the general case of n sensor, we derive close-form expressions for the probability of error for a heuristic solution for a restricted class of initial states, and investigate the benefit of entanglement in the initial state.

5.3 Orthogonality of Final States for Three Sensors

Note that an optimal solution for two sensors (i.e., $n = 2$) is known and is based on geometric ideas (See [110] and §5.5); however, the solution for two sensors doesn't generalize to higher n. For $n \geq 3$, instead of directly determining the optimal solution, we first focus on determining the the conditions (on U) under which the optimal initial state yields orthogonal final states. We start with the specific case of $n = 3$, as this gives us sufficient insight to generalize the results to an arbitrary number of sensors. Determining the conditions for orthogonality also helps us in conjecturing the optimal initial state for general settings.

The basic idea for deriving the condition on U that yields orthogonal final states (i.e., the below theorem) is to represent the final states on an orthonormal basis based on U's eigenvalues and eigenvectors; this allows us to derive expressions for the pairwise inner products of the final states, and equating these products to zero yields the desired conditions. We now state the main theorem and proof for three sensors.

Theorem 1. *Consider the* ISO *problem, with the unitary operator U, initial state $|\psi\rangle$, and final states $\{|\phi_i\rangle\}$ as defined therein. Recall that the eigenvalues of U are $\{e^{+i\theta}, e^{-i\theta}\}$. When the number of sensors n is three, the following is true.*

For any $\theta \in [60, 120]$ degrees, there exists a $|\psi\rangle$ such that $|\phi_0\rangle, |\phi_1\rangle, |\phi_2\rangle$ are mutually orthogonal. Also, the converse is true, i.e., for $\theta \in (0, 60) \cup (120, 180)$, there is no initial state that makes $|\phi_0\rangle, |\phi_1\rangle, |\phi_2\rangle$ mutually orthogonal.

PROOF: Let us first start analyzing the inner product of $|\phi_0\rangle$ and $|\phi_1\rangle$. Let $z_0 = \langle \phi_0 | \phi_1 \rangle$. We see that:

$$z_0 = \langle \psi | (U^\dagger \otimes I \otimes I)(I \otimes U \otimes I) | \psi \rangle$$
$$= \langle \psi | (U^\dagger \otimes U \otimes I) | \psi \rangle$$

Since U is unitary, its eigenvectors u_- and u_+ are orthogonal. It is easy to confirm that the following eight eigenvectors of the middle-part $(U^\dagger \otimes U \otimes I)$ form an *orthonormal basis*: $\{|u_- u_- u_-\rangle, |u_- u_- u_+\rangle, |u_- u_+ u_-\rangle, |u_- u_+ u_+\rangle, |u_+ u_- u_-\rangle, |u_+ u_- u_+\rangle, |u_+ u_+ u_-\rangle, |u_+ u_+ u_+\rangle\}$. We denote these eight eigenvectors as $\{|j\rangle \mid j = 0, \cdots, 7\}$, with the $|j\rangle$ eigenvector "mimicking" the number j's binary representation when u_- and u_+ are looked upon as 0 and 1 respectively (so, $|3\rangle$ is $|u_- u_+ u_+\rangle$).

We can write the initial state $|\psi\rangle$ in the $\{|j\rangle\}$ basis as

$$|\psi\rangle = \sum_j \psi_j |j\rangle .$$

Thus, we get

$$z_0 = \langle \psi | (U^\dagger \otimes U \otimes I) \sum_j \psi_j |j\rangle$$
$$= \sum_j |\psi_j|^2 e_j$$

where $\{e_0, e_1, \ldots, e_7\}$ are the eigenvalues corresponding to the eight eigenvectors $\{|j\rangle\}$. As the eigenvalues are $1, 1, e^{+2i\theta}, e^{+2i\theta}, e^{-2i\theta}, e^{-2i\theta}, 1, 1$, we get:

$$z_0 = (|\psi_2|^2 + |\psi_3|^2)e^{+2i\theta} + (|\psi_4|^2 + |\psi_5|^2)e^{-2i\theta} + (|\psi_0|^2 + |\psi_1|^2 + |\psi_6|^2 + |\psi_7|^2) \quad (5.1)$$

Similarly, for $z_1 = \langle \phi_1 | \phi_2 \rangle = \langle \psi | (I \otimes U^\dagger \otimes U) | \psi \rangle$, we get the below. Note that, in the expression for z_1, the order of eigenvalues corresponding to the coefficients $|\psi_i|^2$ is $1, e^{+2i\theta}, e^{-2i\theta}, 1, 1, e^{+2i\theta}, e^{-2i\theta}, 1$ (see Observation 1 in §5.4). Thus, we get:

$$z_1 = (|\psi_1|^2 + |\psi_5|^2)e^{+2i\theta} + (|\psi_2|^2 + |\psi_6|^2)e^{-2i\theta} + (|\psi_0|^2 + |\psi_3|^2 + |\psi_4|^2 + |\psi_7|^2) \quad (5.2)$$

Similarly, for $z_2 = \langle \phi_0 | \phi_2 \rangle = \langle \psi | (U^\dagger \otimes I \otimes U) | \psi \rangle$, we get:

$$z_2 = (|\psi_1|^2 + |\psi_3|^2)e^{+2i\theta} + (|\psi_4|^2 + |\psi_6|^2)e^{-2i\theta} + (|\psi_0|^2 + |\psi_2|^2 + |\psi_5|^2 + |\psi_7|^2) \quad (5.3)$$

Now, for $|\phi_0\rangle, |\phi_1\rangle, |\phi_2\rangle$ to be mutually orthogonal, we need $z_0 = z_1 = z_2 = 0$. This yields the following seven Equations 5.4-5.10.

Imaginary Equations. For the imaginary parts of z_0, z_1, z_2 to be zero, we need the following to be true. We refer to these equations as the Imaginary equations.

$$|\psi_2|^2 + |\psi_3|^2 = |\psi_4|^2 + |\psi_5|^2 \tag{5.4}$$
$$|\psi_1|^2 + |\psi_5|^2 = |\psi_2|^2 + |\psi_6|^2 \tag{5.5}$$
$$|\psi_1|^2 + |\psi_3|^2 = |\psi_4|^2 + |\psi_6|^2 \tag{5.6}$$

Real Equations. For the real parts of z_0, z_1, z_2 to be zero, we need the following to be true. We refer to these equations as the Real equations.

$$-(|\psi_2|^2 + |\psi_3|^2 + |\psi_4|^2 + |\psi_5|^2)\cos(2\theta) = |\psi_0|^2 + |\psi_1|^2 + |\psi_6|^2 + |\psi_7|^2 \tag{5.7}$$
$$-(|\psi_1|^2 + |\psi_5|^2 + |\psi_2|^2 + |\psi_6|^2)\cos(2\theta) = |\psi_0|^2 + |\psi_3|^2 + |\psi_4|^2 + |\psi_7|^2 \tag{5.8}$$
$$-(|\psi_1|^2 + |\psi_3|^2 + |\psi_4|^2 + |\psi_6|^2)\cos(2\theta) = |\psi_0|^2 + |\psi_2|^2 + |\psi_5|^2 + |\psi_7|^2 \tag{5.9}$$

Above, the terms with $cos(2\theta)$ are on the left-hand side (LHS), and the remaining terms are on the right-hand side (RHS).

Finally, as ψ_j are coefficients of $|\psi\rangle$, we also have

$$\sum_j |\psi_j|^2 = 1 \tag{5.10}$$

Existence of $|\psi\rangle$ when $\theta \in [60, 120]$ that yields mutually orthogonal final states. Let us assume $|\psi_0|^2 = |\psi_7|^2 = y$ and $|\psi_i|^2 = x$ for $1 \le i \le 6$. These satisfy Equations 5.4-5.6, and the Equations 5.7-5.9 yield:

$$-4x\cos(2\theta) = 2x + 2y$$
$$-(2\cos(2\theta) + 1)x = y$$

The above has a valid solution (i.e., $x, y \ge 0$, and $2y + 6x = 1$ from Eqn. 5.10) when $\cos(2\theta) \le -\frac{1}{2}$ i.e., when $\theta \in [60, 120]$.

When $\theta \in (0, 60) \cup (120, 180)$, no existence of $|\psi\rangle$ that yields mutually orthogonal final states. Let $a = |\psi_0|^2 + |\psi_7|^2$. Then, by using Equation 5.4 in Equation 5.7 and so on, we get the following:

$$-2(|\psi_4|^2 + |\psi_5|^2)\cos(2\theta) = a + |\psi_1|^2 + |\psi_6|^2$$
$$-2(|\psi_2|^2 + |\psi_6|^2)\cos(2\theta) = a + |\psi_3|^2 + |\psi_4|^2$$
$$-2(|\psi_1|^2 + |\psi_3|^2)\cos(2\theta) = a + |\psi_2|^2 + |\psi_5|^2$$

Adding up the above equations and rearranging, we get:

$$(-2\cos(2\theta) - 1)\sum_{j=1}^{6} |\psi_j|^2 = 3a$$

Thus, we need $(-2\cos(2\theta) - 1) \ge 0$, as $a \ge 0$, i.e., we need $\cos(2\theta) \le -\frac{1}{2}$. Thus, for $\theta \in (0, 60) \cup (120, 180)$, there is no solution to the above equations. *Note that we have not used any symmetry argument here.* ∎

5.4 Orthogonality of Final States for n Sensors

In this section, we generalize the result in the previous section to an arbitrary number of sensors greater than 3.[3]

Theorem 2. *Consider the* ISO *problem, with the unitary operator U, initial state $|\psi\rangle$, and final states $\{|\phi_i\rangle\}$ as defined therein. Recall that the eigenvalues of U are $\{e^{+i\theta}, e^{-i\theta}\}$. Let $n \geq 3$ be the number of sensors. The following is true.*

For any $\theta \in [T, 180 - T]$ degrees, there exists a $|\psi\rangle$ such that the set of n states $\{|\phi_i\rangle\}$ are mutually orthogonal, where T is given by:

$$T = \frac{1}{2} \arccos\left(-\frac{\lceil \frac{n}{2} \rceil - 1}{\lceil \frac{n}{2} \rceil}\right)$$

Note that $T \in (45, 90)$ degrees. In particular, the values of T for increasing n are: 60 ($n = 4$), 65.9 ($n = 5, 6$), 69.3 ($n = 7, 8$), 71.6 ($n = 9, 10$).

The converse of the above is also true, i.e., for $\theta \in (0, T) \cup (180 - T, 180)$, there is no initial state $|\psi\rangle$ that makes $\{|\phi_i\rangle\}$ mutually orthogonal. ∎

Before we prove the theorem, we define the partitioning of coefficients and state an observation.

Partitioning the Coefficient-Squares $\{|\psi_j|^2\}$ into "Symmetric" Sets. Note that just renumbering the sensors does not change the optimization problem. Based on this intuition, we can group the eigenvectors $|j\rangle$ (and the corresponding coefficients ψ_j's) into equivalent classes. Let n be the number of sensors. Since only the coefficient-squares $\{|\psi_j|^2\}$ appear in the expression for pairwise inner-products of the final states, we just partition the coefficient-squares rather than the coefficients $\{\psi_j\}$ themselves—as only the coefficient-squares are relevant to our proposed solution and discussion. We partition the set of 2^n coefficient-squares into $n + 1$ symmetric sets $\{S_k\}$ as follows:

$$S_k = \{|\psi_j|^2 \mid |j\rangle \text{ has } k \text{ number of } u_+\} \quad \forall \, 0 \leq k \leq n$$

For each $0 \leq k \leq n$, let R_k be the number of coefficient-squares from S_k in the RHS of a `Real` equation, and L_k be the number of coefficient-squares from S_l in the LHS of `Real` equation. (Note that, by Observation 1 below, for any k, the number of coefficient-squares of S_k that are in the RHS (LHS) is the same for all `Real` equations.) **For the case of n = 3**, we have $S_0 = \{|\psi_0|^2\}, S_1 = \{|\psi_1|^2, |\psi_2|^2, |\psi_4|^2\}, S_2 = \{|\psi_3|^2, |\psi_5|^2, |\psi_6|^2\}, S_3 = \{|\psi_7|^2\}$. Also, we have $R_0 = R_1 = R_2 = R_3 = 1$, while $L_0 = L_3 = 0$, $L_1 = L_2 = 2$. We will use the above terms to prove the theorem.

---

[3]For two sensors, the single equation corresponding to the Equations 5.7-5.9 can be made equal to zero on both sides with $\theta = 45$ degrees and zeroing all coefficients on the RHS (which is possible due to lack of other equations).

Observation 1. *For a* `Real` *equation E corresponding to the inner-product of final states ϕ_i and ϕ_j (for $0 \leq i, j \leq n-1$), a coefficient-square $|\psi_r|^2$ appears in the RHS of the equation E iff the bit-representation of the number r has either both 0's or both 1's at the ith and j^{th} most-significant bits.* □

Lemma 2. *For $n \geq 3$,*

$$min_{1 \leq k \leq (n-1)} \frac{R_k}{L_k} = \frac{\lceil \frac{n}{2} \rceil - 1}{\lceil \frac{n}{2} \rceil}.$$

Thus, for the given T in Theorem 2, $L_k \cos(2T) + R_k = 0$ for some k, and $R_k + \cos(2T)L_k \geq 0$ for all k.

PROOF: For $n \geq 3$ and $0 \leq k \leq n$, from Observation 1 we get that:

$$R_k = \binom{n-2}{k-2} + \binom{n-2}{k}$$

$$L_k = 2\binom{n-2}{k-1}.$$

Above, we assume $\binom{x}{y} = 0$ if $y > x$ or $y < 0$. Now, a simple analysis shows that:

$$\left(min_{2 \leq k \leq (n-2)} \frac{\binom{n-2}{k-2} + \binom{n-2}{k}}{2\binom{n-2}{k-1}} \right) = \frac{\lceil \frac{n}{2} \rceil - 1}{\lceil \frac{n}{2} \rceil}$$

Since, for $n \geq 3$, $R_1 = R_{n-1} = n - 2$ and $L_1 = L_{n-1} = 2$, we get the lemma. ∎

Observation 2. *Let $\sum_i x_i = c$, for a set of variables $x_i \geq 0$ and a constant $c > 0$. The equation $\sum_i c_i x_i = 0$, where c_i are some constants, has a solution if and only if (i) at least one of the constants is positive and one of the constants is negative, or (ii) one of the constants is zero.* □

5.4.1 Proof of Theorem 2.

PROOF: If $\theta \in [T, 180 - T]$. Let the set of all coefficient-squares in each S_k to be equal to x_k, for each k. Then, each `Imaginary` equation becomes:

$$\sum_{k=0}^{n} (L_k/2)x_k = \sum_{k=0}^{n} (L_k/2)x_k \tag{5.11}$$

Each `Real` equation becomes:

$$-\cos(2\theta) \sum_{k=0}^{n} L_k x_k = \sum_{k=0}^{n} R_k x_k \tag{5.12}$$

$$\sum_{k=0}^{n} (R_k + \cos(2\theta)L_k)x_k = 0 \tag{5.13}$$

By Observation 2, the above equation (and thus, all `Real` equations) can be made true by appropriate choices of x_k since

1. $R_k + \cos(2\theta)L_k$ is positive for $k = 0$ as $L_0 = 0$ and $R_0 = 1$.

2. $R_k + \cos(2\theta)L_k$ is negative or zero for some k by Lemma 2 when $\theta \in [T, 180 - T]$.

If $\theta \in (0, T) \cup (180 - T, 180)$. Adding all the `Real` equations gives the following. Below, $f(j) = k$ such that $|\psi_j|^2 \in S_k$.

$$- \cos(2\theta) \sum_{j=0}^{2^n} \binom{n}{2} \frac{L_{f(j)}}{|S_{f(j)}|} |\psi_j|^2 = \sum_{j=0}^{2^n} \binom{n}{2} \frac{R_{f(j)}}{|S_{f(j)}|} |\psi_j|^2$$

The above gives:

$$\sum_{j=0}^{2^n} \frac{1}{|S_{f(j)}|} (R_{f(j)} + \cos(2\theta) L_{f(j)}) |\psi_j|^2 = 0$$

The above equation doesn't have a solution as $(R_k + \cos(2\theta)L_k) > 0$ for all k for $\theta \in (0, T)$ (and thus, for $\theta \in (180 - T, 180)$) for by Lemma 2. $\blacksquare$

Optimal Initial State under Theorem 2's Condition. Based on the above theorem, we can derive the optimal initial state under the condition of Theorem 2; the optimal initial state yields mutually-orthogonal final states.

Corollary 1. *Consider the* `ISO` *problem, with the unitary operator U, initial state $|\psi\rangle$, and final states $\{|\phi_i\rangle\}$ as defined therein. Recall that the eigenvalues of U are $\{e^{+i\theta}, e^{-i\theta}\}$. Let $n \geq 3$ be the number of sensors. When $\theta \in [T, 180 - T]$ degrees, where T is defined in Theorem 2, an optimal initial state $|\psi\rangle$ that yields mutually orthogonal final states n states $\{|\phi_i\rangle\}$ is given as follows.[4]*

Let S_l be the partition that minimizes the ratio R_l/L_l. It follows from Lemma 2's proof (we omit the details) that $l = \lfloor \frac{n}{2} \rfloor$, and R_l, L_l, and S_l are given by:

$$L_l = |S_l| \times \frac{\lceil \frac{n}{2} \rceil}{2\lceil \frac{n}{2} \rceil - 1}$$

$$R_l = |S_l| \times \frac{\lceil \frac{n}{2} \rceil - 1}{2\lceil \frac{n}{2} \rceil - 1}$$

$$|S_l| = \binom{n}{\lfloor \frac{n}{2} \rfloor}$$

Then, the coefficients of an optimal initial state $|\psi\rangle$, when $\theta \in [T, 180 - T]$ degrees with T defined in Theorem 2, are such that their coefficient-squares are as follows:

$$|\psi_j|^2 = \frac{1}{|S_l| - \cos(2\theta)L_l - R_l} \qquad \forall \, |\psi_j|^2 \in S_l$$

$$|\psi_j|^2 = \frac{-\cos(2\theta)L_l - R_l}{|S_l| - \cos(2\theta)L_l - R_l} \qquad \forall \, |\psi_j|^2 \in S_0$$

$$|\psi_j|^2 = 0 \qquad \forall \, |\psi_j|^2 \notin S_l \cup S_0$$

---

[4]We note that there are many optimal solutions.

PROOF: The proof of the above Corollary easily follows from the fact that each coefficient-square of the solution is positive (from Lemma 2), and that the coefficient-squares of the solution satisfy Eqn. 5.13 (and Eqn. 5.11 trivially) as well as the constraint in Eqn. 5.10. ∎

5.5 Conjectured Optimal `ISO` Solution

Provably Optimal Solution for Two Sensors. The above joint-optimization problem for the case of 2 sensors can be solved optimally as follows. First, we note that the minimum probability of error in discriminating two final states for a given initial state $|\psi\rangle$ is given by:

$$P_e = \frac{1}{2}\left(1 - \sqrt{1 - |\langle\psi|(U \otimes U^{-1})|\psi\rangle|^2}\right). \tag{5.14}$$

Now, when the eigenvalues of U are $\{e^{+i\theta}, e^{-i\theta}\}$, as in our `ISO` problem, then the initial state $|\psi\rangle$ that minimizes the above probability of error for $0 \le \theta \le \pi/4$ and $3\pi/4 \le \theta \le \pi$ can be shown to be the following entangled state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|u_+\rangle|u_-\rangle + |u_-\rangle|u_+\rangle). \tag{5.15}$$

For $\pi/4 \le \theta \le 3\pi/4$, there exists an initial state that yields orthogonal final states. The above follows from the techniques developed to distinguish between two unitary operators [63]; we refer the reader to our recent work [110] for more details. Unfortunately, the above technique doesn't generalize to n greater than 2, because for greater n, there is no corresponding closed-form expression for minimum probability of error.

Conjectured Optimal Solution For n Sensors. The main basis for our conjectured optimal solution is that an optimal initial state must satisfy the *symmetry of coefficients* property which is defined as follows: an initial state satisfies the *symmetry of coefficients* property, if for each k, the set of coefficient-squares in S_k have the same value. The *intuition* behind why an optimal initial state must satisfy the *symmetry of coefficients* property comes from the following facts:

1. The optimal initial state, under the condition of Theorem 2, satisfies the symmetry of coefficients property.

2. Since sensors are homogeneous, "renumbering" the sensors doesn't change the optimization problem instance fundamentally. Thus, if $|\psi\rangle$ is an optimal initial state, then all initial state solutions obtained by permuting the orthonormal basis $\{|j\rangle\}$ corresponding to a renumbering of sensors,[5] must also yield optimal initial states.[6] Now, observe that an initial state that satisfies the symmetry

---

[5]Note that renumbering the sensors is tantamount to renumbering the bits in the bit-representation of j integers of the orthonormal basis $\{|j\rangle\}$. See Theorem 3's proof for more details.

[6]Note that this fact doesn't imply that the optimal solution must satisfy the symmetry of coefficients property.

of coefficients property remains unchanged under any permutation of the orthonormal basis $\{|j\rangle\}$ corresponding to a renumbering of sensors.

3. Similarly, due to the homogeneity of sensors, an optimal initial state must yield "symmetric" final states—i.e., final state vectors that have the same pairwise angle between them. Now, from Observation 1, we observe that an initial state that satisfies the symmetry of coefficients yields final states such that their pairwise inner-product value is the same.

Finally, it seems intuitive that this common (see #3 above) inner-product value of every pair of final states should be minimized to minimize the probability of error in discriminating the final states. Minimizing the common inner-product value within the problem's constraints yields the below optimal solution conjecture.

Conjecture 1. *Consider the* `ISO` *problem, with the unitary operator U, initial state $|\psi\rangle$, and final states $\{|\phi_i\rangle\}$ as defined therein. Recall that the eigenvalues of U are $\{e^{+i\theta}, e^{-i\theta}\}$. Let $n \geq 3$ be the number of sensors. For a given $\theta \in (0, T] \cup [180 - T, 180)$, degrees, where T is from Theorem 2, the optimal initial state $|\psi\rangle$ for the* `ISO` *problem is as follows.*

Let S_l be the partition that minimizes $(R_l + \cos(2\theta)L_l)/(R_l + L_l)$, where R_l and L_l are as defined in the previous section. The coefficients of the optimal solution are such that their coefficient-squares are given by:

$$
\begin{aligned}
|\psi_j|^2 &= 1/|S_l| \quad \forall\ |\psi_j|^2 \in S_l \\
|\psi_j|^2 &= 0 \qquad \forall\ |\psi_j|^2 \notin S_l
\end{aligned}
$$

$\square$

We note the following: (i) The above conjecture optimal solution is provably optimal for $n = 2$, with $T = 45$ degrees; see Eqn. 5.15 above and [110]. (ii) The above conjectured optimal solution yields orthogonal final states for $\theta = T$. In particular, it can be easily shown that the above conjectured optimal solution is the same as the solution derived in Corollary 1 for $\theta = T$. (iii) The proposed state in the above conjecture is a Dicke State in the basis made up of $|u_-\rangle$ and $|u_+\rangle$. Dicke states can be prepared deterministically by linear depth quantum circuits in a single quantum computer [17], and be prepared in a distributed quantum network as well [219]. We now show that the above conjecture can be proved with the help of the following simpler conjecture.

Proving Symmetry of Coefficients. Based on the intuition behind the above Conjecture 1, one way to prove it would be to prove the symmetry of coefficients—i.e., the existence of an optimal solution wherein the coefficient-squares in any S_k are equal. Proving symmetry of coefficients directly seems very challenging, but we believe that the below conjecture (which implies symmetry of coefficients, as shown in Theorem 3) is likely more tractable. Also, *the below Conjecture has been verified to hold true in our empirical study (see §5.7).*

Conjecture 2. *For a given U, consider two initial states $|\psi\rangle = \sum_j \psi_j |j\rangle$ and $|\psi'\rangle = \sum_j \psi'_j |j\rangle$ such that (i) they are two unequal coefficient-squares, i.e., for some j, $|\psi_j|^2 \neq |\psi'_j|^2$, and (ii) they have the same objective function value, i.e., $P(|\psi\rangle, U) = P(|\psi'\rangle, U)$. We claim that the "average" state given by*

$$|\psi_{avg}\rangle = \sum_j \sqrt{\frac{|\psi_j|^2 + |\psi'_j|^2}{2}} |j\rangle$$

has a lower objective function value, i.e., $P(|\psi_{avg}\rangle, U) < P(|\psi'\rangle, U)$. □

We now show that the above Conjecture is sufficient to prove the optimal solution Conjecture 1.

Theorem 3. *Conjecture 2 implies Conjecture 1.*

PROOF: We start by showing that Conjecture 2 implies the symmetry of coefficients, and then minimize the common pairwise inner-product values of the final states.

Conjecture 2 implies Symmetry of Coefficients. First, note that for a given initial state $|\psi\rangle$, we can generate $(n! - 1)$ other "equivalent" initial states (not necessarily all different) by just renumbering the sensor (or, in other words, permuting the basis eigenvectors). Each of these initial states should yield the same objective value $P()$ as that of $|\psi\rangle$, as it can be shown that they would yield essentially the same set of final states. As an example, the following two initial states are equivalent (i.e., yield the same objective value $P()$); here, the sensors numbered 1 and 2 have been interchanged.

$$\psi_0 |0\rangle + \psi_1 |1\rangle + \psi_2 |2\rangle + \psi_3 |3\rangle + \psi_4 |4\rangle + \psi_5 |5\rangle + \psi_6 |6\rangle + \psi_7 |7\rangle$$

$$\psi_0 |0\rangle + \psi_2 |1\rangle + \psi_1 |2\rangle + \psi_3 |3\rangle + \psi_4 |4\rangle + \psi_6 |5\rangle + \psi_5 |6\rangle + \psi_7 |7\rangle$$

More formally, for a given initial state $|\psi\rangle = \sum_j \phi_j |j\rangle$, a permutation (renumbering of sensors) $\pi : \{0, 1, \ldots, n-1\} \mapsto \{0, 1, \ldots, n-1\}$ yields an equivalent initial state given by $|\psi'\rangle = \sum_j |\phi_{\Pi(j)}| |j\rangle$ where $\Pi : \{0, 1, \ldots, 2^n - 1\} \mapsto \{0, 1, \ldots, 2^n - 1\}$ is such that $\Pi(j) = i$ where the bits in the bit-representation of j are permuted using π to yield i. It can be shown that the set of final states yielded by $|\psi\rangle$ and $|\psi'\rangle$ are essentially the same (modulo the permutation of basis dimensions), and hence, they will yield the same probability of error and thus objective value $P()$.

Now, consider an optimal initial state $|\psi\rangle = \sum_j |\phi_j| |j\rangle$ that doesn't have symmetry of coefficients—i.e., there is a pair of coefficient-squares $|\phi_i|^2$ and $|\phi_j|^2$ such that they are in the same set S_k but are not equal. The numbers i and j have the same number of 1's and 0's in their binary representation, as $|\phi_i|^2$ and $|\phi_j|^2$ belong to the same set S_k. Let Π be a permutation function (representing renumbering of the n sensors) such that $\Pi(i) = j$. Consider an initial state $|\psi'\rangle = \sum_j \psi_{\Pi(j)} |j\rangle$, which has the same probability of error as $|\psi\rangle$. Now, applying Conjecture 2 on $|\psi\rangle$ and

$|\psi'\rangle$ yields a new initial state with a lower objective value $P()$, which contradicts the optimality of $|\psi\rangle$. Thus, all optimal initial-states must satisfy the symmetry of coefficients.

Maximizing the Pairwise Angle. Now, an optimal initial state with symmetry of coefficient will yield final states that have the same pairwise inner-product values (this follows from Theorem 2's proof). Also, we see that each pairwise inner-product value is (see Eqns.5.11 and 5.13 from §5.4):

$$\sum_{k=0}^{n}(R_k + \cos(2\theta)L_k)x_k \qquad (5.16)$$

with the constraint that

$$\sum_{k=0}^{n}(R_k + L_k)x_k = 1.$$

When $\theta \in (0, T]$. By Lemma 2, note that $(R_k + \cos(2\theta)L_k)x_k \geq 0$ for all k, for $\theta \in (0, T)$. We show in Lemma 3 below that, for states with equal and positive pairwise inner-products, the probability of error in discriminating them using an optimal measurement increases with an increase in the common inner-product value. Thus, the optimal initial state must minimize the above inner-product value expression in Eqn. 5.16. Now, from Observation 3 below, the inner-product value above is minimized when the coefficient-squares in the S_l that minimizes $(R_k + \cos(2\theta)L_k)/|S_l|$ are non-zero, while the coefficient-squares in all other S_k's where $k \neq l$ are zero. This proves the theorem.

When $\theta \in [180 - T, 180)$. Note that $\cos(2\theta) = \cos(2(180 - \theta))$, and since $(180 - \theta) \in (0, T]$ for $\theta \in [180 - T, 180)$, we can use the same argument as above for this case as well. ∎

Observation 3. *Let $\sum_i a_i x_i = 1$, for a set of positive-valued variables x_i and positive constants a_i. The expression $\sum_i c_i x_i$, where constants c_i's are all positive, has a minimum value of $\min_i c_i/a_i$ which is achieved by $x_i = 1/a_i$ for the i that minimizes $\min_i c_i/a_i$.* □

Minimizing Probability of Error in Discriminating "Symmetric" Final States. We now show, using prior results, that if the pairwise inner products (and hence, angles) of the resulting final states $|\phi_i\rangle$ are equal, then the probability of error in discriminating them is minimized when the pairwise inner-product values are minimized.

Lemma 3. *Consider n states to be discriminated $\phi_0, \phi_1, \ldots, \phi_{n-1}$ such that $\langle\phi_i|\phi_j\rangle = x$, for all $0 \leq i, j \leq n - 1$ and $i \neq j$. The probability of error in discriminating $\phi_0, \phi_1, \ldots, \phi_{n-1}$ using an optimal measurement increases with an increase in x when $x \geq 0$.*

PROOF: The optimal/minimum probability of error using the optimal POVM for a set of states with equal pairwise inner products can be computed to be [84]:

$$P_e = 1 - \frac{1}{n}\left(\sqrt{1 - \frac{(n-1)(1-x)}{n}} + (n-1)\sqrt{\frac{1-x}{n}}\right)^2$$

Let the inner term be y, such that $P_e = 1 - (y^2/n)$. The derivative of y with respect to x is given by:

$$\frac{n-1}{2\sqrt{n}} \left(\frac{1}{\sqrt{nx+1-x}} - \frac{1}{\sqrt{1-x}} \right).$$

The above is negative for $x \geq 0$. Thus, for a given number of sensors n and $x \geq 0$, the probability of error P_e increases with an increase in x. ∎

Summary. In summary, we propose the Conjecture 1 for the optimal solution for the `ISO` problem, based on the symmetry of coefficients. We also propose a Conjecture 2 which seems simpler to prove and provably implies Conjecture 1. We empirically validate these conjectures using several search heuristics in the following sections.

5.6   Search Heuristics

Algorithm 1: FindNeighbor($\mathbf{x}, i, stepSize$)

Input: The initial state $\mathbf{x}$, ith element of $\mathbf{x}$, step size
Output: A neighbor $\mathbf{x}'$ of $\mathbf{x}$

1 $\mathbf{x}' \leftarrow \mathbf{x}$;
2 $direction \leftarrow$ Generate a random unit 2D-vector ;
3 $direction' \leftarrow$ convert $direction$ to complex number ;
4 $\mathbf{x}'[i] \leftarrow \mathbf{x}'[i] + direction' \times stepSize$;
5 $\mathbf{x}' \leftarrow Normalize(\mathbf{x}')$;   // $\mathbf{x}^\dagger \mathbf{x} = 1$
6 **return** $\mathbf{x}'$;

In this section, we design three search heuristics to determine an efficient `ISO` solution, viz., hill-climbing algorithm, simulated annealing, and genetic algorithm. In the next section, we will evaluate these heuristics and observe that they likely deliver near-optimal solutions. We start with a numerical (SDP) formulation of determining an optimal measurement, and thus, develop a method to estimate the objective value $P(|\psi\rangle, U)$ of a given initial state $|\psi\rangle$.

Semi-Definite Program (SDP) for State Discrimination. We now formulate a semi-definite program (SDP) to compute the optimal measurement for a given initial state; this formulation allows us to determine the optimal measurement using numerical methods, and thus, facilitates the development of the search heuristics for the `ISO` problem as described below. Given a set of (final) quantum states, determining the optimal measurement that discriminates them with minimum probability of error is a convex optimization problem, and in particular, can be formulated as a semi-definite program [81]. Let the final states be $\{|\phi_i\rangle\}$ with prior probabilities p_i, where $\sum_i p_i = 1$. Let $\{\Pi_i\}$ be the POVM operator with Π_i being the element associated with the state $|\phi_i\rangle$, and let $Tr()$ be the trace operator. The SDP program to determine the optimal measurement operator can be formulated as below, where

the objective is to minimize the probability of error.

$$\min_{\Pi_i \in \mathcal{B}} 1 - \sum_{i=0}^{n-1} p_i Tr(\Pi_i \,|\phi_i\rangle\,\langle\phi_i|) \tag{5.17}$$

subject to the constraints:

$$\Pi_i \succeq 0, \quad 0 \leq i \leq n-1 \tag{5.18}$$

$$\sum_{i=0}^{n-1} \Pi_i = I \tag{5.19}$$

Above, Eqn. 5.18 ensures that every measurement operator is positive semidefinite, while Eqn. 5.19 ensures that the set of measurement operators is a valid POVM, i.e., the summation of all measurement operators is the identity matrix. Eqn. 5.17 minimizes the probability of error expression for a given POVM measurement and set of quantum states.

The Objective Value of an Initial State. To design the search-based heuristics, we need a method to estimate an objective value for a given initial quantum state that evaluates its quality. In our context, for a given initial state $|\psi\rangle$, the ISO problem's objective function $P(|\psi\rangle, U)$ could also serve as the objective function in a search-based heuristic. $P(|\psi\rangle, U)$ can be directly estimated using the Eqn. 5.17 above.

$$P(|\psi\rangle, U) = 1 - \sum_{i=0}^{n-1} p_i Tr(\Pi_i \,|\phi_i\rangle\,\langle\phi_i|) \tag{5.20}$$

where $|\phi_i\rangle = (I^{\otimes i} \otimes U \otimes I^{\otimes(n-i-1)}) |\psi\rangle$ are the final states, and the optimal measurement $\{\Pi_i\}$ can be computed numerically using the SDP formulation given above.

Based on the above method to estimate the objective function $P()$, we can develop search heuristics for the ISO problem; at a high level, each heuristic searches for a near-optimal initial state by starting with a random initial state $\mathbf{x}$ and iteratively improving (not necessarily in every single iteration) by moving to $\mathbf{x}$'s better neighbor based on the objective value $P()$ of $\mathbf{x}$.

Hill-Climbing[7] (HC) Search Heuristic. The Hill-climbing (HC) heuristic starts with randomly picking an initial quantum state for the n-sensors, i.e., a 2^n length vector $\mathbf{x}$ of complex numbers with $\mathbf{x}^\dagger \mathbf{x} = 1$. During each iteration, we look into one element of the state vector $\mathbf{x}$ at a time. And for each element, we look into four random "neighbors" of the initial state (as described below), and pick the neighbor with the lowest objective value $P()$. We repeat the process until reaching the termination criteria, i.e., the improvement (if any) of moving to the best neighbor is smaller than a threshold (i.e., 10^{-6}). We also set a minimum number of 100 iterations.

To find a neighbor of a quantum state, we update one element of the state vector $\mathbf{x}$ at a time—by adding to it a random unit vector multiplied by a step size which

---

[7]In our context of a minimization problem, the heuristic actually *descends* into a valley of solutions, but we stick to the Hill-Climbing name because that's the common usage.

Algorithm 2: HillClimbing(U, n)

Input: Unitary operator U
Input: Number of sensor n
Output: Initial State **x**

1 $\mathbf{x} \leftarrow$ a random state vector with a length of 2^n;
2 $bestObjective \leftarrow P(\mathbf{x}, U)$;
3 $stepSize \leftarrow 0.1$;
4 $stepDecreaseRate \leftarrow 0.96$;
5 **while** *Termination Condition Not Satisfied* **do**
6   **for** $i = 1$ *to* 2^n **do**
7 $neighbors \leftarrow$ Find 4 neighbors, call FindNeighbor($\mathbf{x}, i, stepSize$) four times ;
8 $bestStep \leftarrow 0$;
9   **for** $j = 1$ *to* 4 **do**
10 $objective \leftarrow P(neighbors[j], U)$;
11   **if** $objective < bestObjective$ **then**
12 $bestObjective \leftarrow objective$;
13 $bestStep \leftarrow j$;
14   **end**
15   **end**
16   **if** $bestStep$ *is not 0* **then**
17 $\mathbf{x} \leftarrow neighbors[bestStep]$;
18   **end**
19   **end**
20 $stepSize \leftarrow stepSize \times stepDecreaseRate$;
21 **end**
22 **return x** ;

---

decreases with each iteration (a post-normalization step is done to maintain $\mathbf{x}^\dagger \mathbf{x} = 1$). For each element, we look into four random neighbors instead of one, to increase the chance of discovering better neighbors. See Algo. 1 for the neighbor-finding procedure and Algo. 2 for the overall Hill Climbing heuristic.

Simulated Annealing (SA) Heuristic. The above Hill-climbing heuristic can get stuck in a local optimal. Thus, we also develop a more sophisticated Simulated Annealing (SA) [134] metaheuristic which has a mechanism to jump out of a local minimum. By convention, SA applies the concept of energy E. In our context, the energy is the equivalent of the objective function value $P()$. In essence, SA allows itself to transition to solutions with worse objective values with a small (but non-zero) probability. In SA, the transition probability to a new neighbor state depends upon the improvement ΔE in the objective function and is given by:

$$P(\Delta E) = \min(1, e^{-\Delta E/T}), \tag{5.21}$$

where T is the temperature. We note that when the new state's objective value is

---

Algorithm 3: SimulatedAnnealing(U, n)

Input: Unitary operator U
Input: Number of sensor n
Output: Initial State **x**

1   **x** $\leftarrow$ a random state vector with a length of 2^n;
2   $stepSize \leftarrow 0.1$;
3   $T \leftarrow$ Standard deviation of some **x** neighbors' objective values ;
4   $stepDecreaseRate \leftarrow 0.96$;
5   $coolingRate \leftarrow 0.96$;
6   $stdRatio \leftarrow 1$;
7   **while** *Termination Condition Not Satisfied* **do**
8   **for** $i = 1$ *to* 2^n **do**
9   **for** $j = 1$ *to* 4 **do**
10           $\mathbf{x}' \leftarrow$ FindNeighbor($\mathbf{x}, i, stepSize$) ;
11           $E_1 \leftarrow P(\mathbf{x}, U)$;
12           $E_2 \leftarrow P(\mathbf{x}', U)$;
13           $\Delta E \leftarrow E_2 - E_1$;
14   **if** $\Delta E < 0$ **then**
15             $\mathbf{x} \leftarrow \mathbf{x}'$
16   **else**
17             $\mathbf{x} \leftarrow \mathbf{x}'$ with probability $e^{-\Delta E/T}$
18   **end**
19   **end**
20   **end**
21      $stepSize \leftarrow stepSize \times stepDecreaseRate$;
22      $std \leftarrow$ Standard dev. of **x** recent neighbors' scores;
23      $stdRatio \leftarrow stdRatio \times coolingRatio$;
24      $T \leftarrow min(T \times coolingRate, std \times stdRatio)$;
25   **end**
26   **return x** ;

---

lower, then ΔE is negative, and thus, $P(\Delta E)$ is 1, and the new state is readily transitioned to. Same as in [143], we set the initial temperature as the standard deviation of the objective value of several initial state's neighbors. As the SA algorithm iterates, the temperature T gradually decreases. In our context, the following works well and leads to fast convergence, compared to other standard equations used in other contexts [140].

$$T_n = min\{(1 - \epsilon)T_{n-1}, (1 - \epsilon)^n \sigma_{n-1}\}, \tag{5.22}$$

where σ_{n-1} is the standard deviation of objective values of the latest ten neighbors explored at the $(n-1)^{th}$ iteration. SA uses the same neighbor-finding method (Algo. 1) as in the previous Hill-climbing heuristic, with a similar termination condition as Hill-climbing except that we allow a few iterations for improvement. The pseudo-code of SA is shown in Algo. 3.

Algorithm 4: GeneticAlgorithm(U, n)

Input: Unitary operator U
Input: Number of sensor n
Output: Initial State $\mathbf{x}$

1 $N \leftarrow$ population size;
2 $\mathbf{x}_{pop} \leftarrow$ a size N population of length 2^n random state vectors;
3 **while** *Termination Condition Not Satisfied* **do**
4     $ranks \leftarrow computeRank(\mathbf{x}_{pop}, U)$;
5     $\mathbf{x}'_{pop} \leftarrow$ an empty children population;
6   **while** $length(\mathbf{x}'_{pop}) < size$ **do**
7        $parents \leftarrow$ get two states by $select(ranks, \mathbf{x}_{pop})$;
8        $children \leftarrow$ get two new states by $twoPointCrossover(parents)$;
9   Do mutation for $children$;
10   Add $children$ to $\mathbf{x}'_{pop}$;
11   **end**
12    $\mathbf{x}_{pop} \leftarrow$ the top N of states in $\mathbf{x}_{pop} + \mathbf{x}'_{pop}$;
13 **end**
14 $\mathbf{x} \leftarrow$ the best state in $\mathbf{x}_{pop}$;
15 **return** $\mathbf{x}$;

Genetic Algorithm (GA) Heuristic. The Genetic Algorithm (GA) is another popular metaheuristic algorithm for solving optimization problems. Inspired by the natural evolution of survival of the fittest [111], GA works by considering a "population" of candidate solutions and creating the next generation iteratively [309], until the best solution in a new generation does not improve from the best solution in the previous generation by at least a threshold. In our context, the initial population of candidate solutions is a set of random initial states. And candidate solutions are evaluated by a fitness function, which is conceptually the same as our objective function $P()$ (Eqn. 5.20) except that the fitness function is the higher the better while $P()$ is the lower the better. So, $1 - P()$ will serve as the fitness function for GA. The pseudocode for GA is shown in Algo. 4. To create a new generation, we pick a pair of candidate states as parents through the rank selection [129] and then generate a pair of children states by using the two-point crossover method [129]. Finally, we mutate the children in a way similar to finding neighbors in Algo. 1.

5.7 Validating the Conjectures Empirically

In this section, we evaluate our search heuristics for varying U operators (i.e., varying values of θ) and for $n = 2$ to 5 and observe that they likely deliver near-optimal initial state solutions to the ISO problem. Based on this observation, we show that our optimal solution Conjecture 1 is very likely true as well the Conjecture 2. Our
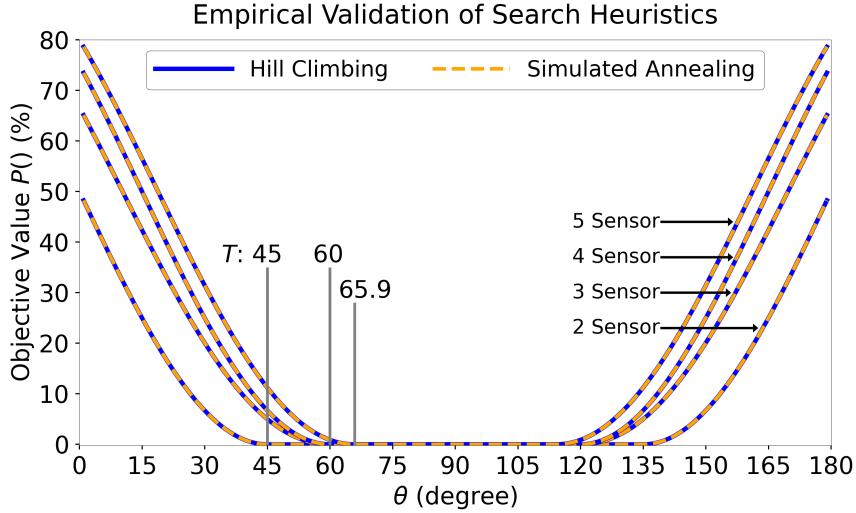
Figure 5.2: Performance of the three search heuristics for varying U's parameter θ, for different number of sensors in the network. Genetic Algorithm (GA) is not shown explicitly, for clarity, but it also performs almost exactly the same as Hill-Climbing and Simulated Annealing (SA) which are plotted above.

search heuristics implementation and experiment's raw data are open-source at[8].

Evaluation Setting. Recall that, without loss of generality, we assume the eigenvalues of U to be $\{e^{+i\theta}, e^{-i\theta}\}$ with $U|u_\pm\rangle = e^{\pm i\theta}|u_\pm\rangle$ where $u_\pm$ are the two eigenstates of U. In our evaluations, we vary the θ in the range of $(0, 180)$ degrees, and assume the prior probabilities of final states to be uniform. We consider four values of n, the number of sensors, viz., 2, 3, 4, and 5. Running simulations for much larger values of n is infeasible due to the prohibitive computational resources needed. E.g., the estimated computation time to run any of the search heuristics for $n = 10$ will take 10s of years, based on our preliminary estimates.[9]

Performance of Search Heuristics. Fig. 5.2 shows the performance of the search heuristics under varying θ and four values of $n = 2, 3, 4, 5$, in terms of the ISO objective function $P(|\psi\rangle, U)$ for the initial state solution $|\psi\rangle$. We make the following two observations:

1. All three heuristics perform almost exactly the same.

2. The heuristics deliver an initial state solution with $P(|\psi\rangle, U) = 0$ for the same range of θ given in Theorem 2.

---

[8]https://github.com/caitaozhan/QuantumSensorNetwork

[9]In our context, the Hill-Climbing heuristic goes through about 100 iterations and in each iteration, it needs to solve $4 \cdot 2^n$ instances of SDP formulations (Eqns 5.17-5.19) where n is the number of sensors. We use the Convex-Python CVXPY [69] package (which in turn used the Splitting Conic Solver [198]) to solve our SDP formulations, and observe that it takes more than an hour to solve a single SDP instance for $n = 10$; this suggests an estimate of 10s of years of computation time for $n = 10$.
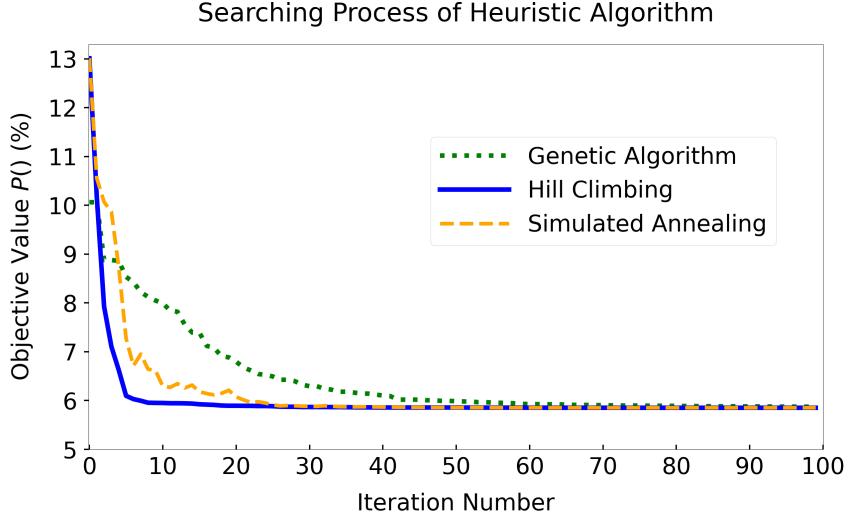
Figure 5.3: The objective value $P()$, probability of error, of the candidate solution over iterations of the three search heuristics for a special value of $\theta = 46$ degrees and $n = 4$ sensors.

We also observe that the heuristics perform the same for θ and $\pi - \theta$, i.e., symmetric along the $\theta = \pi/2$ line. Thus, in the remaining plots, we only plot results for $\theta \in (0, \pi/2]$. Fig. 5.3 shows the convergence rates of the three heuristics for a specific value of $\theta = 46$ degrees and $n = 4$ sensors. We observe that HC converges the fastest, followed by SA and GA. After 100 iterations, the HC and SA end at a probability of error of 5.85%, while GA ends at 5.86%.

Empirical Validation of Conjecture 2. Recall that Conjecture 2 states that an "average" solution of two `ISO` solutions with equal objective values have a lower objective value. To empirically validate Conjecture 2, we generate a random state $|\psi\rangle$, and then, generate $n! - 1$ additional states of the same objective value $P()$ by renumbering the sensors as discussed in Theorem 3's proof. Then, we take many pairs of these states, average them, and compute the objective value. Fig. 5.4 plots the objective value of the original state $|\psi\rangle$, and the range of the objective values of the averaged states. We observe that the objective values of the averaged states are invariably less than that of $|\psi\rangle$.

Empirical Validation of the Optimal Solution Conjecture 1. We now evaluate the performance of the initial state solution obtained by Conjecture 1 and compare it with the solution delivered by one of the search heuristics–Hill Climbing (HC). Here, we consider $\theta \in (0, T) \cup (180 - T, 180)$ degree, where T is as defined in Theorem 2. In Fig. 5.5, we observe that the HC heuristic and Conjecture 1 solutions have identical performance, suggesting that Conjecture 1's solution is likely optimal based on our earlier observation that the search heuristics likely deliver optimal solutions.

Symmetry "Index" vs. Objective Value (Probability of Error). In this final experiment, we investigate the impact of the symmetry of coefficients on the objective value of an initial state. Here, we only do experiments for $n = 3$ number of sensors.
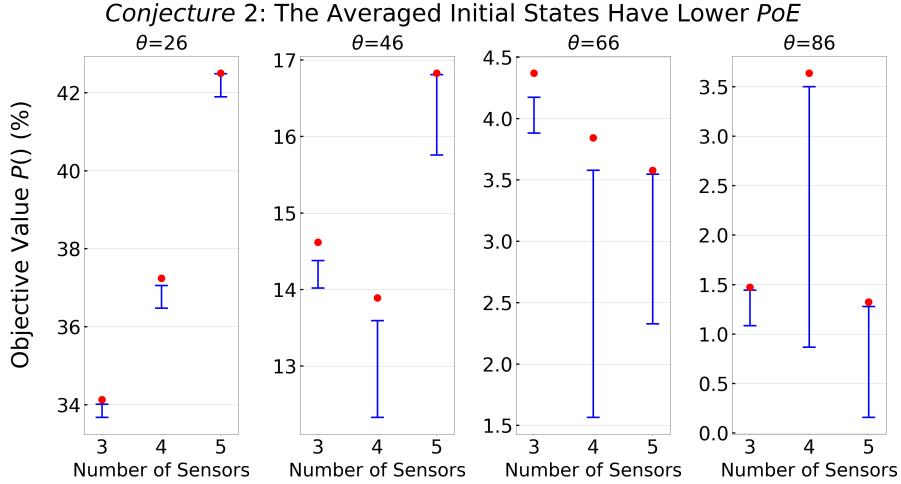
Figure 5.4: Empirical validation of Conjecture 2. For four different values of θ and three different values of n, we show that the objective value (Probability of Error) of the original initial state (the red circle) remains higher than the objective value of the many "averaged" states (range shown by blue the bar).

To this end, we define a notion of *symmetry index* which quantifies the symmetry of coefficients in a given initial state. In particular, we define the *symmetry index* for an initial state $|\psi\rangle = \sum_{j} \psi_j |j\rangle$ as:

$$\sum_{k=0}^{n} \sum_{|\psi_i|^2, |\psi_j|^2 \in S_k} (|\psi_i|^2 - |\psi_j|^2)^2 \tag{5.23}$$

where S_k is the kth symmetric set as defined in Theorem 2. The symmetric index being zero implies that within each symmetric set, all the coefficient-squares are equal. Fig. 5.6 shows that the search heuristic essentially generates solutions with lower and lower symmetry index, and finally, converges to a solution with zero symmetry index value. This is true for all three search heuristics (Figs. 5.6a) and for varying θ (Fig. 5.6b). Given Fig. 5.3 already shows that the objective value decreases as the searching iterations go on, we can conclude that the objective value and the symmetry index decrease simultaneously when the iterations go on. Furthermore in Fig. 5.7a, we show the correlation between symmetry index and objective value through a scatter plot—with the objective value generally decreasing with the decrease in symmetry index. Fig. 5.7b zooms into the later iterations of the heuristics wherein the symmetry index is very low (less than 0.08) to show a clearer view of the correlation.

5.8 Extensions

In this section, we consider the generalization to the unambiguous discrimination scheme, non-uniform prior probabilities, and quantum noise.
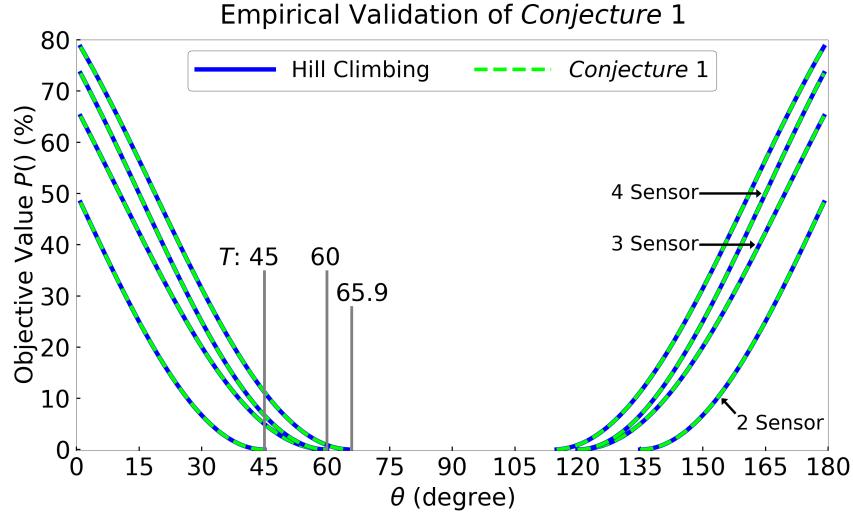
Figure 5.5: The Conjecture 1's solution performs almost exactly as the Hill-Climbing heuristic when $\theta \in (0, T] \cup [180 - T, 180)$, degrees, where T is from Theorem 2. For $n = 2$, Conjecture 1's solution matches with the provably optimal solution from [110] with T being 45 degrees.

5.8.1 Unambiguous Discrimination Measurement

Till now, we have only considered the minimum error measurement scheme wherein the measurement operator always outputs a state, though sometimes incorrectly and thus incurring a certain probability of error. We now consider an alternative measurement scheme of unambiguous measurement [16] where there are no errors, but the measurement can fail, i.e. giving an inconclusive outcome. The unambiguous measurement scheme thus may incur a probability of failure. Fortunately, our results for the minimum error measurement scheme also hold for the unambiguous discrimination measurement scheme and objective, as observed below.

1. The sufficient and necessary condition for orthogonality derived in Theorem 2 is a property of the states and the operator U, and is independent of the measurement scheme. Thus, Theorem 2 hold for an unambiguous discrimination scheme.

2. The intuition behind Conjecture 1 is based on the homogeneity of sensors and symmetry of the problem setting (e.g., symmetric eigenvalues of U, uniform probability of final states, etc.). Thus, we believe the optimal initial state solution for an unambiguous discrimination scheme is the same as in the case of the minimum error scheme. Thus, Conjecture 1 should hold.

3. Conjecture 2 is independent of the measurement scheme.

4. We prove the version of Lemma 3 corresponding to the unambiguous measurement below. Thus, Theorem 3 also holds for unambiguous measurement.

(a) The three search heuristics for $\theta = 46$ degrees.

(b) The HC heuristic for different values of θ.
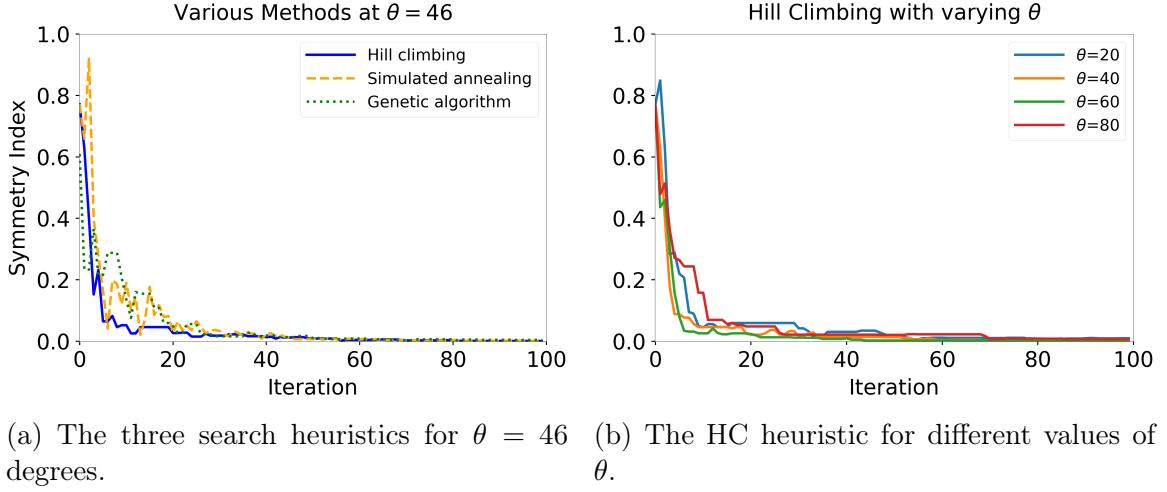
Figure 5.6: Symmetry-index of the candidate solutions over iterations.

5. The optimization problem of determining the optimal measurement $\{\Pi_i\}$ for an unambiguous discrimination scheme can also be formulated as an SDP [13], and thus can be computed numerically. Thus, the search heuristics from §5.6 will also work for unambiguous measurement with the corresponding SDP for an unambiguous discrimination scheme.

Lemma 4. *Consider n states to be discriminated $\phi_0, \phi_1, \ldots, \phi_{n-1}$ such that $\langle \phi_i | \phi_j \rangle = x$, for all $0 \leq i, j \leq n - 1$ and $i \neq j$. The probability of **failure** in discriminating $\phi_0, \phi_1, \ldots, \phi_{n-1}$ using an optimal measurement (for unambiguous discrimination) increases with an increase in x when $x \geq 0$.*

PROOF: The optimal/minimum probability of failure using the optimal POVM for a set of states with equal pairwise inner products is equal to x when $x \geq 0$ [84]. Thus, the lemma trivially holds. ∎

5.8.2 Non-uniform Prior Probability

Till now, we have implicitly assumed that the events (of affecting one sensor) occur with a uniform probability. Here, we consider the generalization of allowing for the events to occur with non-uniform probability. This could happen if different sensor locations can have different probabilities of the event occurrence.

Number of Sensors $n = 2$. When the number of sensors is 2, we observe that the optimal solution for the ISO problem actually remains unchanged. In particular, the expression for the minimum probability of error in discriminating the two final states, with non-uniform probabilities p_1 and p_2, for a given initial state $|\psi\rangle$ is given by (derived from [109]):

$$P_e = \frac{1}{2} \left(1 - \sqrt{1 - 4p_1 p_2 |\langle \psi | (U \otimes U^{-1}) | \psi \rangle|^2} \right). \tag{5.24}$$

(a) Probability of Error vs. Symmetry Index

(b) Probability of Error vs. Symmetry Index, zoomed-in for low values of symmetry index
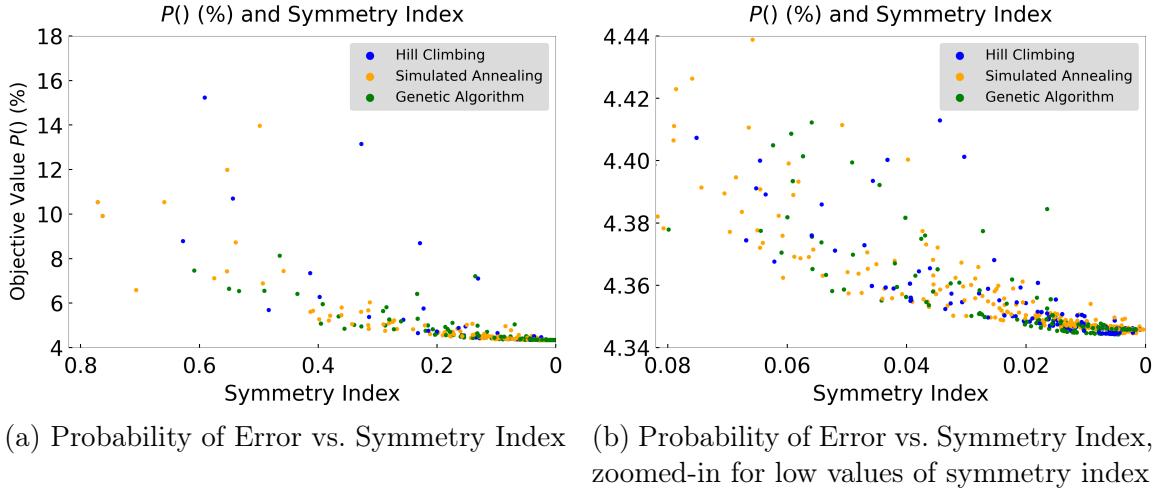
Figure 5.7: The correlation between the objective value (probability of error) and the symmetry index.

The above entails that, as for the case of uniform probabilities, we need to minimize $|\langle\psi|(U \otimes U^{-1})|\psi\rangle|$, which is independent of p_1 and p_2. Thus, the optimal initial state for $n = 2$ is independent of the probabilities associated with the final states/events.

Number of Sensors $n > 2$. For $n > 2$, it is easy to see that Theorems 1 and 2 that derive conditions for orthogonality of the final states remain unchanged since the probabilities of events/final-states do not affect the final states themselves. However, the optimal `ISO` solution for general values of θ is certainly different than that conjectured in Conjecture 1, since Conjecture 1 is fundamentally based on the symmetry of the final states, which is unlikely to be the case for non-uniform probabilities of events. On the other hand, it is easy to generalize the search heuristics for the case of non-uniform probability. See Fig. 5.8, which plots the objective value $P()$ for varying θ for the three search heuristics. We observe that (i) The heuristics return an optimal objective value (of zero) for the conditions in Theorems 2; (ii) All the heuristics perform almost the same. These observations suggest that the heuristics likely perform near-optimally even for the general case of non-uniform event probabilities. In addition, we note that, compared to the uniform probability case (i.e., Fig. 5.2), the optimal objective value $P()$ under non-uniform probabilities is lower than the $P()$ under uniform probabilities, for any particular θ.

5.8.3 Impact of Quantum Noise.

Till now, we have looked at the `ISO` problem from a theoretical perspective while ignoring the quantum noise. Since quantum noise is an essential aspect of quantum systems, we present a mitigation strategy to correct for quantum noise and evaluate it for two quantum noise models.

Quantum Noise-Mitigation Strategy. In our context, the impact of the noise is that it essentially results in final states that are different (due to noise) than the

116
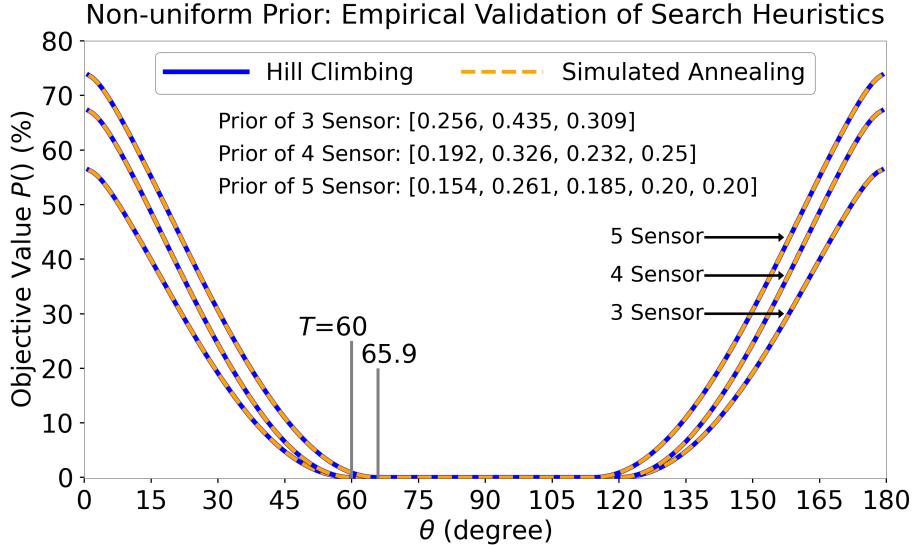
Figure 5.8: Performance of the three search heuristics with non-uniform prior for varying U's parameter θ, for a different number of sensors in the network. Genetic Algorithm (GA) is not shown explicitly, for clarity, but it also performs almost the same as Hill-Climbing and Simulated Annealing (SA), which are plotted above.

ones we try to discriminate. That is, consider an given initial state $|\psi\rangle$ which yields (noiseless) final states $\{|\phi_i\rangle\}$; let the optimal measurement to discriminate the final states $\{|\phi_i\rangle\}$ be the POVM with elements $\{E_i\}$. However, due to the noise, the actual noisy final states may actually be different than $\{|\phi_i\rangle\}$, which, when discriminated with the POVM $\{E_i\}$, will result in a higher probability of error than if there were no noise. Thus, to account for such quantum noise, we propose to modify the POVM measurement appropriately. In particular, we compute the POVM measurement to discriminate the expected noisy final states—which we represent by the density matrices of the mixed states representing the ensemble of potential final states. More formally, our strategy is as follows: For each final state $|\phi_i\rangle$, let ρ_i be the density matrix that represents the distribution/mixture of noisy final states that may result instead of $|\phi_i\rangle$. Then, we use SDP (Eqn. 5.17) to determine the optimal POVM $\{E'_i\}$ that optimally discriminates the density matrices $\{\rho_i\}$, and use it to discriminate the noisy final state.

Evaluation. We consider three popular noise models [197] for evaluation of our above mitigation technique.

1. *Amplitude damping* causes the quantum system to lose energy.

2. *Phase damping* describes the loss of quantum information without energy loss.

3. *Depolarizing channel* is probabilistically replacing the qubit by the completely mixed state, $I/2$.

All the above noise models can be characterized using the Kraus operators (K), which obey $\sum_i K_i^\dagger K_i = I$. In particular, the Kraus operators for the amplitude damping

117
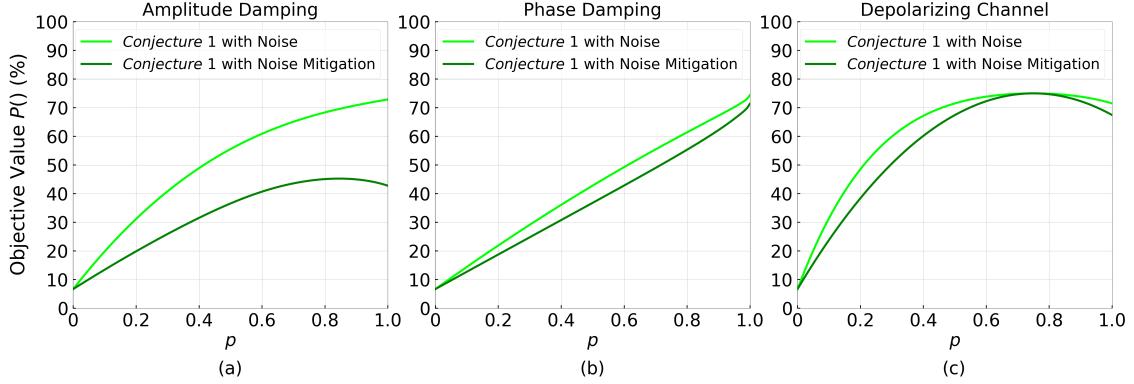
Figure 5.9: The improvement in the objective value $P()$ for the Conjecture 1's solution due to the noise-mitigation strategy, for the three noise models, for $\theta = 45$ degrees and four sensors.

are:

$$\mathcal{N}_{amp} = \{K_{a0}, K_{a1}\} = \{\begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}, \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix}\}$$

where p can be thought of as the probability of losing a photon [197]. The Kraus operators for the phase damping are:

$$\mathcal{N}_{pha} = \{K_{p0}, K_{p1}\} = \{\begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & \sqrt{p} \end{bmatrix}\}$$

where p can be interpreted as the probability that a photon from the system has been scattered (without loss of energy) [197]. Finally, the Kraus operators for the depolarizing channel are:

$$\mathcal{N}_{dep} = \{K_{d0}, K_{d1}, K_{d2}, K_{d3}\} = \{\sqrt{1-p} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \sqrt{\frac{p}{3}} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sqrt{\frac{p}{3}} \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sqrt{\frac{p}{3}} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\}$$

where p is the probability of a qubit being depolarized. For a given noise model, its Kraus operators give the operators by which the state's density matrix is transformed with a corresponding probability. For example, in our context, under the third noise model of depolarizing noise, for a given initial state $|\psi\rangle$, each final state $|\phi_i\rangle$ with a density matrix $\rho_i = |\phi_i\rangle\langle\phi_i|$ is transformed to $K_{d0}\rho_i K_{d0}^\dagger$ with a probability of $(1 - p)$ and to $K_{d1}\rho_i K_{d1}^\dagger$ or $K_{d2}\rho_i K_{d2}^\dagger$ or $K_{d3}\rho_i K_{d3}^\dagger$ with a probability of $p/3$ each. The above noise models are for a single sensor/qubit; for multiple qubits, we use a tensor product of the single-qubit noises. Fig. 5.9 shows the 1) impact of various quantum noise on the results, and 2) for the initial state (from Conjecture 1), how the objective value $P()$ improves due to the above-discussed noise-mitigation strategy for the three noise models, for the specific value of $\theta = 45$ degrees and the number of sensors equal to 4. We observe that the improvement is particularly significant in the case of amplitude-damping noise.

5.9   Conclusion and Future Directions

In this work, we formulate the problem of initial state optimization in detector quantum sensor networks, which has potential applications in event localization. We first derive the necessary and sufficient conditions for the existence of an initial state that can detect the firing sensor with perfect accuracy, i.e., with zero probability of error. Using the insights from this result, we derive a conjectured optimal solution for the problem and provide a pathway to proving the conjecture. Multiple search-based heuristics are also developed for the problem and the heuristics' numerical results successfully validate the conjecture. In the end, we extend our results to the unambiguous discrimination measurement scheme, non-uniform prior, and considering quantum noise.

Beyond proving the stated Conjectures in the paper, there are many generalizations of the addressed `ISO` problem of great interest in terms of: (i) More general final states (e.g, two sensors may change at a time, allowing for multiple impact operators U_1, U_2, etc.), (ii) Restricting the measurement operators allowed (e.g., allowing only the projective measurements and/or local measurements [37]), to incorporate practical considerations in the implementation of measurement operators. We are also interested in proving related results of interest, e.g., `ISO` initial-state solution being the same for minimum error and unambiguous discrimination.

Chapter 6

Efficient Quantum Network Communication using Balanced Entanglement Swapping Trees

Quantum network communication is challenging, as the No-cloning theorem in the quantum regime makes many classical techniques inapplicable; in particular, direct transmission of qubit states over long distances is infeasible due to unrecoverable errors. For long-distance communication of unknown quantum states, the only viable communication approach (assuming local operations and classical communications) is teleportation of quantum states, which requires a prior distribution of entangled pairs (EPs) of qubits. Establishment of EPs across remote nodes can incur significant latency due to the low probability of success of the underlying physical processes. The focus of this chapter is to develop efficient techniques that minimize EP generation latency. Prior works have focused on selecting entanglement *paths*; in contrast, we select *entanglement swapping trees*—a more accurate representation of the entanglement generation structure. [95] has developed a dynamic programming algorithm to select an optimal swapping-tree for a single pair of nodes, under the given capacity and fidelity constraints. For the general setting, [95] also developed an efficient iterative algorithm to compute a set of swapping trees. However, the dynamic programming algorithm has a high time complexity, and thus, may not be practical for real-time route finding in large networks. In this chapter, we focus on developing an *almost linear time* heuristic for the `QNR-SP` problem, based on the classic Dijkstra shorted path algorithm. The designed heuristic performs close to the DP-based algorithms in our empirical studies.

6.1 Introduction

Fundamental advances in physical sciences and engineering have led to the realization of working quantum computers (QCs) [9, 88]. Although there is some progress in the hardware to ensure the stability and scalability of quantum computing systems [50, 51], there are still significant limitations to the capacity of individual QC [36]. Quan-

tum networks (QNs) enable the construction of large, robust, and more capable quantum computing platforms by connecting smaller QCs. Quantum networks [234] also enable various important applications [49, 82, 137, 182, 227]. However, quantum network communication is challenging — e.g., physical transmission of quantum states across nodes can incur irreparable communication errors, as the No-cloning Theorem [70] proscribes making independent copies of arbitrary qubits. At the same time, certain aspects unique to the quantum regime, such as entangled states, enables novel mechanisms for communication. In particular, teleportation [19] transfers quantum states with just classical communication, but requires an a priori establishment of entangled pairs (EPs). This chapter presents techniques for the efficient establishment of EPs in a network.

The establishment of EPs over long distances is challenging. Coordinated entanglement swapping (e.g. DLCZ protocol [76]) using quantum repeaters can be used to establish long-distance entanglements from short-distance entanglements. However, due to the low probability of success of the underlying physical processes (short-distance entanglements and swappings), EP generation can incur significant latency—of the order of 10s to 100s of seconds between nodes 100s of kms away [225]. Thus, we need to develop techniques that can facilitate the fast generation of long-distance EPs. [95] solves the `QNR-SP` Problem: Given a single (s, d) pair, select a minimum-latency swapping tree under given constraints. In this chapter, we select near-optimal swapping trees by a heuristic at a much lower time complexity.

To the best of our knowledge, no prior work has addressed the problem of selecting an efficient swapping-tree for entanglement routing; they all consider selecting routing *paths* ([35] selects a path using a metric based on balanced trees; see §6.3.2). Almost all prior works have considered the "waitless" model, wherein all underlying physical processes much succeed *near-simultaneously* for an EP to be generated; this model incurs minimal decoherence, but yields very low EP generation rates. In contrast, we consider the "waiting" protocol, wherein, at each swap operation, the earlier arriving EP waits for a limited time for the other EP to be generated. Such an approach with efficient swapping trees yields high entanglement rates; the potential decoherence risk can be handled by discarding qubits that "age" beyond a certain threshold.

Our Contributions. We formulate the entanglement routing problem (§6.3) in QNs in terms of selecting optimal swapping *trees* in the "waiting" protocol, under fidelity constraints. In this context, we make the following contribution:

1. For the `QNR-SP` problem, the optimal algorithm in [95] has high time complexity; we aim to improve the time complexity of the algorithm without degrading its empirical performance. We thus design a near-linear time heuristic for the `QNR-SP` problem based on an appropriate metric that essentially restricts the solutions to balanced swapping trees (§6.4).

6.2 QC Background

Qubit States. Quantum computation manipulates *qubits* analogous to how classical computation manipulates *bits*. At any given time, a bit may be in one of two states, traditionally represented by 0 and 1. A quantum state represented by a *qubit* is a *superposition* of classical states, and is usually written as $\alpha_0 |0\rangle + \alpha_1 |1\rangle$, where α_0 and α_1 are *amplitudes* represented by complex numbers and such that $\|\alpha_0\|^2 + \|\alpha_1\|^2 = 1$. Here, $|0\rangle$ and $|1\rangle$ are the standard (orthonormal) *basis* states; concretely, they may represent physical properties such as spin (down/up), polarization, charge direction, etc. When a qubit such as above is *measured*, it collapses to a $|0\rangle$ state with a probability of $\|\alpha_0\|^2$ and to a $|1\rangle$ state with a probability of $\|\alpha_1\|^2$. In general, a state of an n qubit system can be represented as $\Sigma_{i=0}^{2^n-1} \alpha_i |i\rangle$ where "i" in $|i\rangle$ is i's bit representation. Qubit mapping refers to the problem of assigning logical qubits used in quantum algorithms to the physical qubits available on a quantum computer. [127] exploits the regular structure of modern quantum architecture and [126] leverages technical intuition (i.e., educated guess) to advance qubit mapping in domain specific problems.

Entanglement. Entangled pure[1] states are multi-qubit states that cannot be "factorized" into independent single-qubit states. E.g., the 2-qubit state $\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$; this particular system is a *maximally-entangled* state. We refer to maximally-entangled pairs of qubits as EPs. The surprising aspect of entangled states is that the combined system continues to stay entangled, even when the individual qubits are physically separated by large distances. This facilitates many applications, e.g., teleportation of qubit states by local operations and classical information exchange, as described next.

Teleportation. Direct transmission of quantum data is subject to unrecoverable errors, as classical procedures such as amplified signals or re-transmission cannot be applied due to quantum no-cloning [70, 272].[2] An alternative mechanism for quantum communication is *teleportation*, Fig. 6.1 (a), where a qubit q from a node A is recreated in another node B (*while "destroying" the original qubit q*) using only classical communication. However, this process requires that an EP already established over the nodes A and B. Teleportation can thus be used to reliably transfer quantum information. At a high level, the process of teleporting an arbitrary qubit, say qubit q, from node A to node B can be summarized as follows:

1. an EP pair (e_1, e_2) is generated over A and B, with e_1 stored at A and e_2 stored at B;

2. at A, a *Bell-state measurement* (BSM) operation over e_1 and q is performed, and the 2 classical bits measurement output (c_1c_2) is sent to B through the

---

[1]In this chapter, we largely deal with only pure qubit states. Entanglement of general mixed states is defined in terms of separation of density matrices [102].

[2]Quantum error correction mechanisms [68, 190] can be used to mitigate the transmission errors, but their implementation is very challenging and is not expected to be used until later generations of quantum networks.
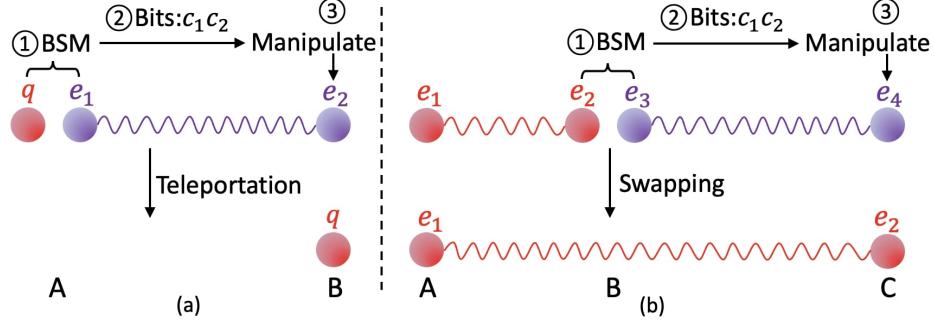
Figure 6.1: (a) Teleportation of $|q\rangle$ from A to B, while consuming an entangled pair (e_1, e_2). (b) Entanglement swapping over the triplet of nodes (A, B, C), which results in A's qubit entangled with C's qubit. This can be viewed as a teleportation of e_2 from node B to C.

       classical communication channel; at this point, the qubits q and e_1 at A are destroyed.

3. manipulating the EP-pair qubit e_2 at B based on received (c_1, c_2) changes its state to q's initial state.

Depending on the physical realization of qubits and the BSM operation, it may not always be possible to successfully generate the 2 classical bits, as the BSM operation is stochastic.

Entanglement Swapping (ES). Entanglement swapping is an application of teleportation to generate EPs over remote nodes. See Fig. 6.1 (b). If A and B share an EP and B teleports its qubit to C, then A and C end up sharing an EP. More elaborately, let us assume that A and B share an EP, and B and C share a separate EP. Now, B performs a BSM on its two qubits and communicates the result to C (teleporting its qubit that is entangled with A to C). When C finishes the protocol, it has a qubit that is entangled with A's qubit. Thus, an entanglement swapping (ES) operation can be looked up as being performed over a triplet of nodes (A, B, C) with EP available at the two pairs of adjacent nodes (A, B) and (B, C); it results in an EP over the pair of nodes (A, C).

Fidelity: Decoherence and Operations-Driven. Fidelity is a measure of how close a realized state is to the ideal. Fidelity of qubit decreases with time, due to interaction with the environment, as well as gate operations (e.g., in ES). Time-driven fidelity degradation is called *decoherence*. To bound decoherence, we limit the aggregate time a qubit spends in a quantum memory before being consumed. With regards to operation-driven fidelity degradation, Briegel et al. [32] give an expression that relates the fidelity of an EP generated by ES to the fidelities of the operands, in terms of the noise introduced by swap operations and the number of link EPs used. The order of the swap operations (i.e., the structure of the swapping tree) does not affect the fidelity. Thus, the operation-driven fidelity degradation of the final EP generated by a swapping-tree T can be controlled by limiting the number of leaves of T, assuming that the link EPs have uniform fidelity (as in [40]).

Entanglement Purification [32, e.g.] and Quantum Error Correction [218, e.g.] have been widely used to combat fidelity degradation. Our work focuses on optimally scheduling ES operations with constraints on fidelity degradation, without purification or error correction.

Quantum Memories. Multiple quantum memories have been recently proposed to bring quantum networks into realization. Types of quantum memories that support BSM measurements and gate unitary operations, and probably have a long decoherence time can be used in quantum communications. Most of them are matter-based which have photonic interface to produce matter-matter entanglement over two neighboring nodes (see below). At a high-level, there are three different quantum memory platforms: quantum dots, trapped atoms or ions, and colour centers in diamond. Each has its own physical characteristics and applications. While quantum dots have the ability to process quantum information very fast, they exhibit a very low decoherence time among others [207, 254]. To overcome the low efficiency of single atom-photon coupling process, atomic ensemble schemes have been proposed [76] where along with dynamic decoupling and cooling techniques, decoherence times of a few seconds have been achieved [67, 223, 249]. For trapped ion memories, decoherence times from several minutes to few hours have been demonstrated [141, 261]. To further increase the entanglement generation rate, [22] proposes a way to use a single silicon–vacancy (SiV) colour center in diamond to perform asynchronous photonic BSM at the node located in the middle of two adjacent quantum nodes.

6.2.1 Generating Entanglement Pairs (EPs)

As described above, teleportation, which is the only viable means of transferring quantum states over long distances, requires an a priori distribution of EPs. Thus, we need efficient mechanisms to establish EPs across remote QN nodes; this is the goal of our work. Below, we start with describing how EPs are generated between adjacent (i.e., one-hop away) nodes, and then discuss how EPs across a pair of remote nodes can be established via ESs.

Generating EP over Adjacent Nodes. The physical realization of qubits determines the technique used for sharing EPs between adjacent nodes. The *heralded entanglement* process [35, 230] to generate an atom-atom EP between adjacent nodes A and B is as follows:

1. Generate an entangled pair of atom and a telecom-wavelength photon at node A and B. Qubits at each node are generally realized in an atomic form for longer-term storage, while photonic qubits are used for transmission.

2. Once an atom-photon entanglement is locally generated at each node (at the same time), the telecom-photons are then transmitted over an optical fiber to a photon-photon/optical BSM device C located in the middle of A and B so that the photons arrive at C at the same time.

3. The device C performs a BSM over the photons, and transmits the classical result to A or B to complete ES.

Other entanglement generation processes have been proposed [190]; our techniques themselves are independent of how the link EP are generated.

Generating EP between Remote Nodes. Now, EP between non-adjacent nodes connected by a path in the network can be established by performing a sequence of ESs at intermediate nodes; this requires an a priori EP over each of the adjacent pairs of nodes in the path. For example, consider a path of nodes $x_0, x_1, x_2, x_3, x_4, x_5$, with an EP between every pair of adjacent nodes (x_i, x_{i+1}). Thus, each node x_i $(1 \leq i \leq 4)$ has two qubits, one of which is entangled with x_{i-1} and the other with x_{i+1}. Nodes x_0 and x_5 have only one qubit each. To establish an EP between x_0 and x_5, we can perform a sequence of entanglement swappings (ESs) as shown in Fig. 6.2. Similarly, the sequence of ES over the following triplets would also work: (x_2, x_3, x_4), (x_2, x_4, x_5), (x_0, x_1, x_2), (x_0, x_2, x_5).
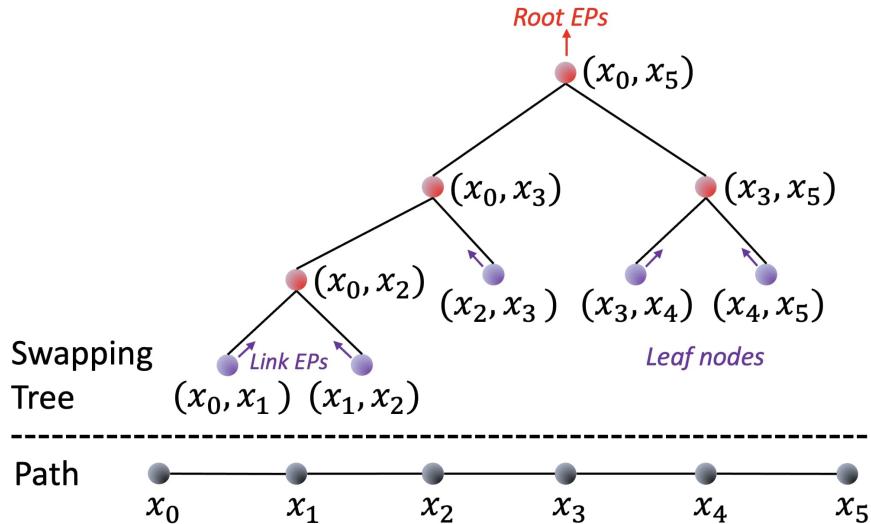
Figure 6.2: A swapping tree over a path. The leaves of the tree are the path-links, which generate link-EPs continuously.

Swapping Trees. In general, given a path $P = s \rightsquigarrow d$ from s to d, any complete binary tree (called a *swapping tree*) over the ordered links in P gives a way to generate an EP over (s, d). Each vertex in the tree corresponds to a pair of network nodes in P, with each leaf representing a link in P. Every pair of siblings (A, B) and (B, C) perform an ES over (A, B, C) to yield an EP over (A, C)—their parent. See Fig. 6.2. *Note that subtrees of a swapping tree execute in parallel.* Different swapping trees over the same path P can have different performance characteristics, as discussed later (see Fig. 6.4).

Expected Generation Latency/Rate of EPs. In general, our goal is to continuously generate EPs at some rate using a swapping tree, using continuously generated EPs at the leaves. The stochastic nature of ES operations means that an EP at the tree's root will be successfully generated only after many failed attempts and hence significant latency. We refer to this latency as the *generation latency* of the EP at the root, and in short, just the generation latency of the tree. EP generation rate is

125

the inverse of its generation latency. Whenever we refer to generation latency/rate, we implicitly mean *expected* generation latency/rate.

Two Generation Protocols: `WaitLess` and `Waiting` When a swapping tree is used to (continuously) generate EPs, there are two fundamentally different generation protocols [225, 246].

- `WaitLess` *Protocol.* In this model, all the underlying processes, including link EP generations and atomic BSMs are synchronized. If all of them succeed then the end-to-end EP is generated. If *any* of the underlying processes fail, then all the generated EPs are discarded and the whole process starts again from scratch (from generation of EP at links). In the `WaitLess` protocol, all swapping trees over a given path P incur the same generation latency—thus, here, the goal is to select an optimal path P (as in [40, 230]).

- `Waiting` *Protocol.* In `Waiting` protocol, a qubit of an EP may wait (in a quantum memory) for its counterpart to become available so that an ES operation can be performed. Using such storage, we preclude discarding successfully generated EPs, and thus, reduce the overall latency in generation of a root-level EP. E.g., let (A, B) and (B, C) be two siblings in a swapping tree and EP for (A, B) is generated first. Then, EP (A, B) may wait for the EP (B, C) to be successfully generated. Once the EP (B, C) is generated, the ES operation is done over the triplet (A, B, C) to generate the EP (A, C). If the EP (A, B) waits beyond a certain threshold, then it may decohere.

Hardware Requirement Differences. `WaitLess` protocols can generate EPs without quantum memories in a relay fashion if the EP generation among adjacent nodes can be tightly synchronized. In contrast, `Waiting` protocols benefit from memories with good coherence times (§6.5).

Why `Waiting`'s Entanglement Generation Rate is Never Worse. The focus of the `WaitLess` protocol is to avoid qubit decoherence due to storage. But it results in very low generation rates due to a very-low probability of *all* the underlying processes succeeding at the same time. However, since qubit coherence times are typically higher than the link-generation latencies[3], an appropriately designed `Waiting` protocol will always yield better generation rates *without significantly compromising the fidelity* The key is to bound the waiting time to limit decoherence as desired; e.g., in our protocol, we restrict to trees with high expected fidelities (§6.3), and discard qubits that "age" beyond a threshold. Both protocols use the same number of quantum memories (2 per node), though the `Waiting` protocols will benefit from low-decoherence memories; other hardware requirements also remain the same.

---

[3]Link generation latencies for 5 to 100km links range from about 3 to 350 milliseconds for typical network parameters [35], while coherence times of few seconds is very realistic (coherence times of several seconds [86, 165] have been shown long ago, and more recently, even coherence times of several minutes [222, 236] to a few hours [262, 320] have been demonstrated.
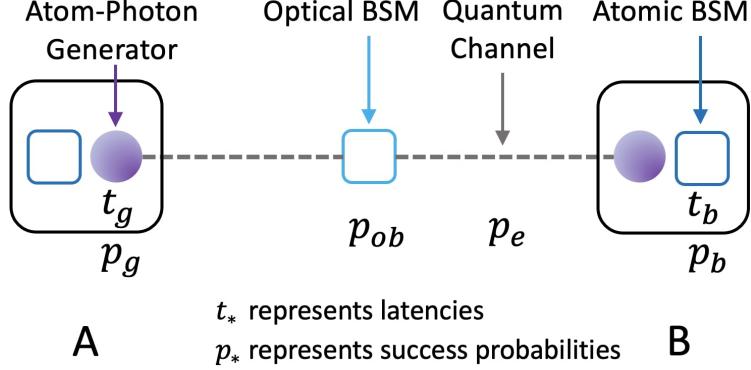
Figure 6.3: Key notations used.

6.3 Model, Problem, and Related Works

In this section, we discuss our network model, formulate the problem addressed, and discuss related work.

Network Model. We denote a quantum network (QN) with a graph $G = (V, E)$, with $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{(v_i, v_j)\}$ denoting the set of nodes and links respectively. Pairs of nodes connected by a link are defined as *adjacent* nodes. We follow the network model in [35] closely. Thus, each node has an atom-photon EP generator with generation latency (t_g) and probability of success (p_g). Generation latency is the time between successive attempts by the node to excite the atom to generate an atom-photon EP; this implicitly includes the times for photon transmission, optical-BSM latency, and classical acknowledgment. *For clarity of presentation* and without loss of generality, we assume homogeneous network nodes with the same parameter values. The generation rate is the inverse of generation latency, as before. A node's atom-photon generation capacity/rate is its aggregate capacity and may be split across its incident links (i.e., in the generation of EPs over its incident links/nodes). Each node is also equipped with a certain number of atomic memories to store the qubits of the atom-atom EPs.

A network link is a quantum channel (e.g., using an optical fiber or a free-space link), and, in our context, is used only for the establishment of link EP. In particular, a link $e = (A, B)$ is used to transmit telecom-photons from A and B to the photon-photon BSM device in the middle of e. Thus, each link is composed of two half-links with a probability of transmission success (p_e) that decreases exponentially with the link distance (see §6.5). The optical-BSM operation has a certain probability of success (p_{ob}). To facilitate atom-atom ES operations, each network node is also equipped with an atomic-BSM device with an operation latency (t_b) and probability of success (p_b). Finally, there is an independent classical network with a transmission latency (t_c); we assume classical transmission always succeeds.

Single vs. Multiple Links Between Nodes. For our techniques multiple links between a pair of adjacent nodes can be replaced by a single link of aggregated rate/capacity. Hence we assume only a single link between every pair of nodes. However, distinct

multiple links between nodes have been used creatively in [230] (which refers to them as multiple channels); thus, we will discuss multiple links further in §6.5 when we evaluate various techniques. We note that the all-photonic protocol in [12] is essentially a more sophisticated version of the multi-link `WaitLess` protocol in [230] to further minimize memory requirements, but it uses multipartite cluster states which are challenging to create. In either case, in terms of the selection of paths/trees, the path-selection techniques from [230] should also apply to the all-photonic protocol with certain modifications to account for how the cluster states are generated.

EP Generation Latency of a Swapping Tree. Given a swapping tree and EP generation rates at the leaves (network links), we wish to estimate the generation latency of the EPs over the remote pair corresponding to the tree's root with the `Waiting` protocol. Below, we develop a recursive equation. Consider a node (A, C) in the tree, with (A, B) and (B, C) as its two children. Let T_{AB}, T_{BC}, and T_{AC} be the corresponding (expected) generation latencies of the EPs over the three pairs of nodes. Below, we derive an expression for T_{AC} in terms of T_{AB} and T_{BC}; this expression will be sufficient to determine the expected latency of the overall swapping tree by applying the expression iteratively. We start with an observation. If two EP arrival processes X_1 and X_2 are exponentially distributed with a mean inter-arrival latency of λ each, then the expected inter-arrival latency of $\max(X, Y)$ is $(3/2)\lambda$. From above, if assume T_{AB} and T_{BC} to be exponentially distributed with the same expected generation latency of T, then the expected latency of both EPs arriving is $(3/2)T$. Thus, we have:

$$T_{AC} = (\frac{3}{2}T + t_b + t_c)/p_b, \tag{6.1}$$

Remarks. We make the following remarks regarding the above expression. First, when $T_{AB} \neq T_{BC}$, we are able to only derive an upper-bound on T_{AC} which is given by the above equation but with T replaced by $\max(T_{AB}, T_{BC})$.[4] Second, our motivation for the exponential distribution assumption stems from the fact that the EP generation latency at the *link level* is certainly exponentially distributed if we assume the underlying probabilistic events to have a Poisson distribution. Third, note that the resulting distribution is not exponential. Despite this, we apply the above equation recursively to compute the tree's generation latency. Finally, Eqn. 6.1 is conservative in the sense that each round of an EP generation of any subtree's root starts from scratch (i.e., with no link EPs from prior round) and ends with either a EP generation at the *whole swapping tree*'s root or an atomic-BSM failure at the subtree's root. We do not "pipeline" any operations across rounds within a subtree, which may lower latency; this is beyond this work's scope.

6.3.1 Problem Formulation

We now formulate the central problem of selecting a *single* swapping trees for a single source-destination pair.

---

[4]The 3-over-2 formula as an upper bound has also been corroborated in a recent work [59] which derives analytical bounds on EP latency times in more general contexts.

QNR Single Path (QNR-SP) Problem. Given a quantum network and a source-destination pair (s, d), the QNR-SP problem is to determine a single swapping tree that maximizes the expected generation rate (i.e., minimizes the expected generation latency) of EPs over (s, d), under the following capacity and fidelity constraints:

1. *Node Constraints.* For each node, the aggregate resources used by $\bigcup_i \mathbf{T}_i$ is less than the available resources; we formulate this formally below.

2. *Fidelity Constraints.* Each swapping tree in $\bigcup_i \mathbf{T}_i$ satisfies the following: (a) Number of leaves is less than a given threshold τ_l; this is to limit fidelity degradation due to gate operations. (b) Total memory storage time of any qubit is less[5] than a given *decoherence threshold* τ_d.

Informally, the swapping-trees may also satisfy some fairness constraint across the given source-destination pairs. A special case of the above QNR problem is to select a single tree for a source-destination pair; we address this in the next section.

Formulating Node Constraints. Consider a swapping tree $\mathbf{T} \in \bigcup_i \mathbf{T}_i$ over a path P. For each link $e \in P$, let $R(e, \mathbf{T})$ be the EP rate being used by $\mathbf{T}$ over the link e in P. Let us define $R_e = \sum_{\mathbf{T}} R(e, \mathbf{T})$, and let $E(i)$ be the set of edges incident on i. Then, the node capacity constraint is formulated as follows.

$$1/t_g \geq \sum_{e \in E(i)} R_e/(p_g{}^2 p_e{}^2 p_{ob}) \quad \forall i \in V. \tag{6.2}$$

The above comes from the fact that to generate a single link EP over e, each end-node of e needs to generate $1/(p_g{}^2 p_e{}^2 p_{ob})$ photons successfully, since each photon (from each end-node) has a generation success of p_g and a transmission success rate of p_e, and the optical-BSM's success probability over the two successfully arriving photons is p_{ob}. Note that $1/t_g$ is a node's total generation capacity. Also, the memory constraint is that for any node i, the memory available in i should be more than $2x + y$ where x is the number of swapping trees that use i as an intermediate node and y is the number of trees that use i as an end node.

For homogeneous nodes and link parameters, it is easy to see that the best swapping tree is the balanced or almost-balanced tree over the shortest path. As described in §6.3.2, the QNR-SP problem has been addressed before in [35, 230] under different models. The problem of selecting *multiple* swapping trees for *multiple* source-destination pairs is solved in [95].

---

[5]We note that, in our context, the storage time as well as the memory coherence time are statistical quantities due to the underlying statistical mechanisms. However, for the purposes of *selecting* a swapping tree, we use a fixed decoherence threshold τ_d value equal to the mean of the distribution of the coherence time (recent work [144] computes optimal cut-offs/thresholds, and their techniques can be used to pick τ_d). When simulating a selected tree for generation of EPs, we can implement coherence time as a statistical measure.

6.3.2 Related Works

There have been a few works in recent years that have addressed generating long-distance EPs efficiently. All of these works have focused on selecting an efficient routing path [89, 230, 303] for the swapping process (Callefi [35] also selects a path, but using a metric based on balanced trees). In addition, all except [35] have looked at the `WaitLess` protocol of generating the EPs. Recall that in the `WaitLess` model, the selection of paths suffices, while in the `Waiting` model, one needs to consider the selection of efficient swapping trees with high fidelity. Selection of optimal swapping trees is a fundamentally more challenging problem than selection of paths—and has not been addressed before, to the best of our knowledge. We start with discussing how the `WaitLess` model works.

`WaitLess` **Approaches.** The most recent works to address the above problem are [230] and [40], both of which consider the `WaitLess` model. In particular, Shi and Qian [230] design a Dijkstra-like algorithm to construct an optimal path between a pair of nodes, when there are multiple links (channels) between adjacent nodes. Then, they use the algorithm iteratively to select multiple paths over multiple pairs of nodes. [89] is an improvement of [230] that proposes a synchronous multi-time-slot entanglement routing framework based on the idea of reusing the established but unused EP pairs in subsequent time slots. Chakraborty et al. [40] design a multi-commodity-flow like LP formulation to select routing paths for a set of source-destination pairs. They map the operation-based fidelity constraint to the path length (as in [32]) and use node copies to model the constraint in the LP. However, they explicitly assume that the link EP generation is deterministic—i.e., always succeeds. Among earlier relevant works, [200] proposes a greedy solution for grid networks, and [41] proposes virtual-path-based routing in ring/grid networks.

`Waiting` **Approach.** Due to photon loss, establishing long-distance entanglement between remote nodes at L distance by *direct* transmission yields EP rates that decay exponentially with L. DLCZ protocol [76, 225] broke this exponential barrier using 2^k equidistant intermediate nodes to perform entanglement-swapping operations, implicitly over a balanced binary tree, with a `Waiting` protocol; this makes the EP generation rate decay only polynomially in L. More recently, Caleffi [35] formulated the entanglement generation rate on a given path between two nodes, under the more realistic condition where the intermediate nodes in the path may not all be equidistant, but still considered only balanced trees. Their path-based metric was then used to select the optimal path by enumerating over the exponentially many paths in the network.

Our Approach (vs. [35]). Though [35] considers only balanced trees, its brute-force algorithm is literally impossible to run for networks more than a few tens of nodes. In our work, we observe that a path has many swapping trees, and, in general, imbalanced trees may even be better; see Figure 6.4. Thus, [95] design a polynomial-time dynamic programming (DP) algorithm that delivers an *optimal* high-fidelity swapping-tree; the approach effectively considers all possible swapping trees, not just balanced ones (note that, even over a single path, there are exponentially many trees). Our `Balanced-Tree` Heuristic (§6.4) is closer to [35]'s work, in that both consider only
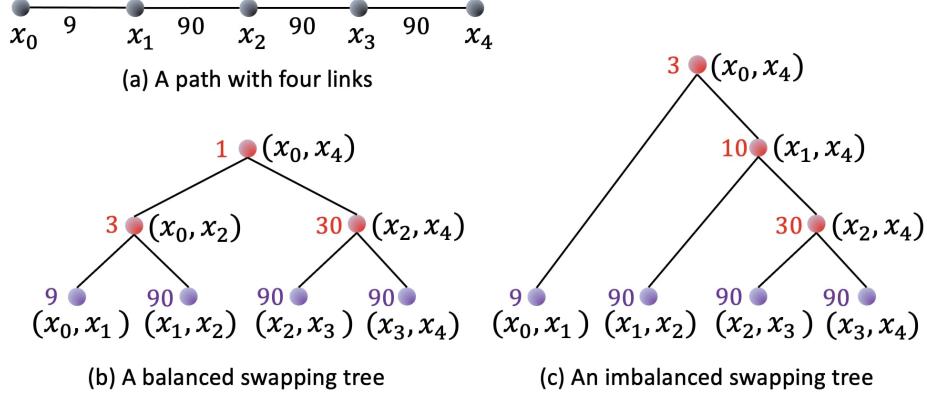
Figure 6.4: Consider the path in (a). The imbalanced tree of (b) has a higher EP generation rate than that of the balanced tree of (c). Here, the numbers represent the EP generation rates over adjacent links or node-pairs.

balanced trees; however, we use a heuristic metric that facilitates a polynomial-time Dijkstra-like heuristic to select the optimal path, while their recursive metric [6] (albeit more accurate than ours) is not amenable to an efficient (polynomial-time) search algorithm.

Other Works. In [125], Jiang et al. address a related problem; given a path with uniform link lengths, they give an algorithm for selecting an optimal sequence of swapping and purification operations to produce an EP with fidelity constraints. In other recent works, Dahlberg et al [62] design physical and link layer protocols of a quantum network stack, and [139] proposes a data plane protocol to generate EPs within decoherence thresholds along a *given* routing path. More recently, Bugalho et al. [34] propose an algorithm to efficiently distribute multipartite entanglement across over than two nodes. [101] proposes two entanglement scheduling and distribution algorithms in a buffered quantum network, where one is for commodities having deadlines and the other not having deadlines. [100] explores the fundamental trade-off between the throughput and entanglement fidelity in quantum networks and proposes an approximation scheme to approximate the achievable worst-case fidelity. We have extended this work of bipartite entanglement generation to multipartite entanglement (graph state) generation using linear programming [85].

6.4   Balanced-Tree **Heuristic for** QNR-SP

The DP-based algorithms presented in [95] for the QNR-SP problem have high time complexity, and thus, may not be practical for real-time route finding in large networks. In this section, we develop an almost-linear time heuristic for the QNR-SP problem, based on the classic Dijkstra shortest path algorithm; the designed heuristic

---

[6]We note that their formula (Eqn. 10 in [35]) is incorrect as it either ignores the 3/2 factor or assumes the EP generations to be synchronized **across all** links. In addition, their expression for "qubit age" ignores the "waiting for ES " time completely.

performs close to the DP-based algorithms in our empirical studies.

Basic Idea. The main reason for the high complexity of our DP-based algorithms in [95] is that the goal of the `QNR-SP` problem is to select an optimal swapping *tree* rather than a path. One way to circumvent this challenge efficiently while still selecting near-optimal swapping tree, is to restrict ourselves to only "balanced" swapping trees. This restriction allows us to think in terms of selection of paths—rather than trees—since each path has a unique[7] balanced swapping tree. We can then develop an appropriate path metric based on the above and design a Dijkstra-like algorithm to select an (s, d) path that has the optimal metric value. We note that Caleffi [35] also proposed a path metric based on balanced swapping trees, but their metric, though accurate, only had a recursive formulation without a closed-form expression—and hence, was ultimately not useful in designing an efficient algorithm. In contrast, we develop an approximate metric with a closed-form expression, based on the "bottleneck" link, as follows.

Path Metric M. Consider a path $P = (s, x_1, x_2, \ldots, x_n, d)$ from s to d, with links $(s, x1), (x1, x2), \ldots, (x_n, d)$ with given EP latencies. We define the path metric for path P, $M(P)$, as the EP generation latency of a balanced swapping over P, which can be estimated as follows. Let L be the link in P with maximum generation latency. If L's depth (distance from the root) is the maximum in a throttled swapping tree, then we can easily determine the accurate generation latency of the tree. However, in general, L may not have the maximum depth, in which case we can still estimate the tree's latency approximately, if the tree is balanced, as follows. In balanced swapping trees, *assuming* the maximum latency link L to be at the maximum depth, gives us a constant-factor approximation of the tree's generation latency. Thus, let us assume L to be at the maximum depth of a balanced tree over P; this maximum depth is $d = \lceil (\log_2 |P|) \rceil$. Let the generation latency of L be T_L. If we ignore the $t_b + t_c$ term in Eqn. 6.1 , then, the generation latency of a throttled swapping tree can be easily estimated to $T(\frac{3}{2p_b})^d$. The term $t_b + t_c$ can also be incorporated as follows. Let $T(i)$ denote the expected latency of the ancestor of L at a distance i from L. Then, we get the recursive equation: $T(i) = (\frac{3}{2}T(i-1) + t_b + t_c)/p_b$. Then, the path metric value $M(P)$ for path P is given by $T(d)$, the generation latency of the tree's root at a distance of d from L, and is equal to:

$$M(P) = T(d) = \bar{p}^d T_L + [(\bar{p}^d - 1)/(\bar{p} - 1)](t_b + t_c)/p_b$$

where $\bar{p} = 3/(2p_b)$ and $d = \lceil (\log_2 |P|) \rceil$. The above is a $(1 + 3/(2p_b))$-factor approximation latency of a balanced and throttled swapping tree over P.

Optimal Balanced-Tree Selection. The above path-metric $M()$ is a monotonically increasing function over paths, i.e., if a path P_1 is a sub-sequence of another path P_2, then $M(P_1) \leq M(P_2)$. Thus, we can tailor the classical Dijkstra's shortest path algorithm to select a (s, d) path with minimum $M(P)$ value, using the link's EP

---

[7]In fact, there can be multiple balanced trees over a path whose length is not a power of 2, but, since they differ minimally in our context, we can pick a unique way of constructing a balanced tree over a path.
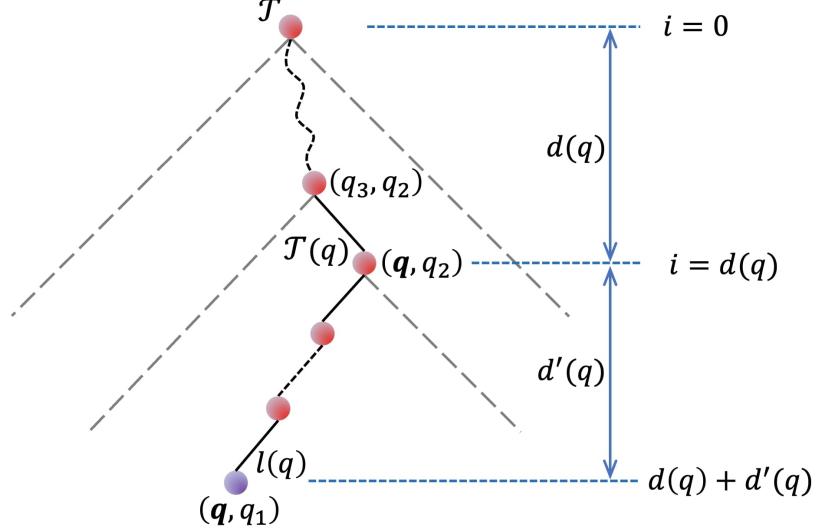
Figure 6.5: Qubit parameters in a swapping tree used to compute the *age* of a qubit q at a leaf node $l(q)$. Here, $l(q)$ is the left-most leaf of the subtree $\mathbf{T}(q)$.

generation latencies as their weights. We refer to this algorithm as `Balanced-Tree`, and it can be implemented with a time complexity of $O(m + n \log n)$ using Fibonacci heaps, where m is the number of edges and n is the number of nodes in the network.

Incorporating Fidelity Constraints.

Definition 1. (G)iven a swapping tree, the total time spent by a qubit in a swapping tree is the time spent from its "birth" via an atom-photon EP generation at a node till its consumption in a swapping operation or in the generation of the tree's root EP. We refer to this as a qubit's *age*. The maximum age over all qubits in a swapping tree is called the tree's (expected) *age*. See Fig. 6.5. □

Fidelity constraints in our path-metric-based setting can be handled by essentially computing the optimal path for each path length (number of hops in the path) up to τ_l, and then pick the best path among them that satisfies the fidelity constraints. This obviously limits the number of leaves to τ_l and addresses the operations-based fidelity degradation. The above also addresses the decoherence/age constraint, since it is easy to see that the age of a balanced swapping tree can be very closely approximated in terms of the latency and the number of leaves. Now, to compute the optimal path for each path length, we can use a simple dynamic programming approach that run in $O(m\tau_l)$ time where m is the number of edges and τ_l is the constraint on number of leaves.

6.5 Evaluations

The goal of our evaluations is to compare the EP generation rates, evaluate the fidelity of generated EPs, and validate our analytical models. We implement the various schemes over a discrete event simulator for QNs called NetSquid [60]. The NetSquid

simulator accurately models various QN components/aspects, and in particular, we are able to define various QN components and simulate swapping-trees protocols by implementing gate operations in entanglement swapping. Other existing simulators include open-source SeQUeNCe [278] and A^{2}Tango [323] that is based on SeQUeNCe for entanglement generation between two remote atom-ensemble nodes.
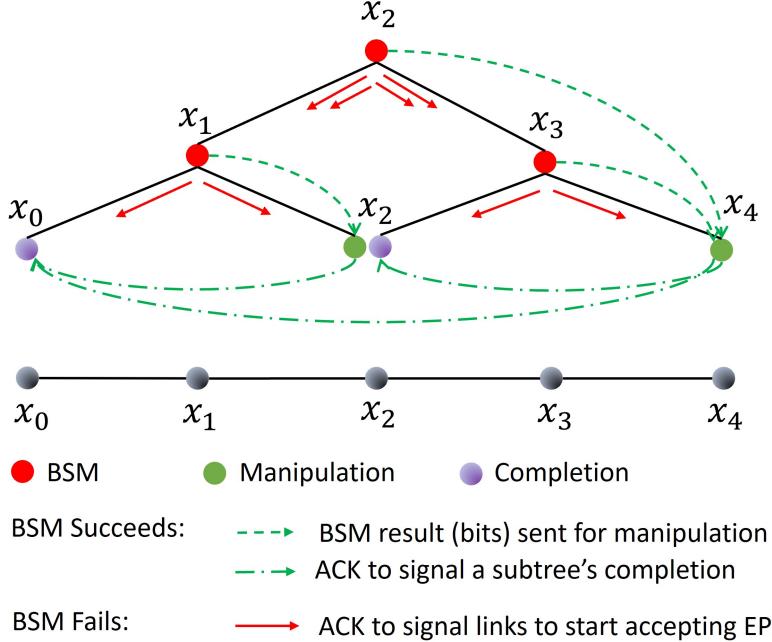
Figure 6.6: Swapping Tree Protocol Illustration. The shown tree is a certain hierarchy of nodes to illustrate the BSM operation in the swapping-tree protocol. A link-layer protocol continuously generates EPs over links (x_0, x_2) and (x_2, x_4). On receiving EP on links on either side, x_1 (x_3) attempts a BSM operation on the stored qubit atoms. If the BSM succeeds, x_1 (x_3) sends two classical bits (solid green arrows) to x_2 (x_4) for desired manipulation/correction after which x_2 (x_4) sends an ACK (dashed green arrows) to the other end-node x_0 (x_2) to complete the EP generation. If BSM at x_1 and x_3 are both successful, then x_2 attempts the BSM as above. If a BSM at say x_1 fails, that x_1 failure signals (red arrows) to all the descendant nodes of the subtree rooted at x_1 so that they can start accepting new EPs from the link layer protocol. Note that here node x_2 plays multiple roles and hence appears at multiple places in the figure.

Swapping Tree Protocol. Our algorithms compute swapping tree(s), and we need a way to implement them on a network. We build our protocol on top of the link layer of [62], which is delegated with the task of continuously generating EPs on a link at a desired rate (as per the swapping tree specifications). Note that a link (a, b) may be in multiple swapping trees, and hence, may need to handle multiple link-layer requests at the same time; we implement such link-layer requests by creating independent atom-photon generators at a and b, with one pair of synchronized generators for each link-layer request. As the links generate continuous EPs at desired rates, we need a

protocol to swap the EPs. Omitting the tedious bookkeeping details, the key aspect of the protocol is that the swap operation is done only when both the appropriate EP pairs have arrived. We implement all the gate operations (including, atomic and optical BSMs) within NetSquid to keep track of the fidelity of the qubits. On BSM success, the swapping node transmits classical bits to the end node which manipulates its qubit and sends the final ack to the other end node. On BSM failure, a classical ack is sent to all descendant link leaves, so that they can now start accepting new link EPs; note that in our protocol, a link l does not accept any more EPs, while its ancestor is waiting for its sibling's EP. See Fig. 6.6.

Simulation Setting. We use a similar setting as in the recent work [230]. By default, we use a network spread over an area of $100km \times 100km$. We use the Waxman model [267], used to create Internet topologies, to randomly distribute the nodes and create links; we use the maximum link distance to be 10km. We vary the number of nodes from 25 to 500, with 100 as the default value. We choose the two parameters in the Waxman model to maintain the number of links to 3% of the complete graph (to ensure an average degree of 3 to 15 nodes). For the `QNR-SP` problem, we pick (s, d) pairs within a certain range of distance, with the default being 30-40 kms.

Parameter Values. We use parameter values mostly similar to the ones used in [35] corresponding to a single-atom-based quantum memory platform and vary some of them. In particular, we use the atomic-BSM probability of success (p_b) to be 0.4 and latency (t_b) to be 10 μ secs; in some plots, we vary p_b from 0.2 to 0.6. The optical-BSM probability of success (p_{ob}) is half of p_b. We use atom-photon generation times (t_g) and probability of success (p_g) as 50 μsec and 0.33 respectively. Finally, we use photon transmission success probability as $e^{-d/(2L)}$ [35] where L is the channel attenuation length (chosen as 20km for an optical fiber) and d is the distance between the nodes. Each node's memory size is randomly chosen within a range of 15 to 20 units. Fidelity is modeled in NetSquid using two parameter values, viz., depolarization (for decoherence) and dephasing (for operations-driven) rates. We choose a decoherence time of two seconds based on achievable values with single-atom memory platforms [248]; note that decoherence times of even several minutes [222, 236] to hours [262, 320] has been demonstrated for other applicable memory platforms. Accordingly, we choose a depolarization rate of 0.01 such that the fidelity after a second is 90%. Similarly, we choose a dephasing rate of 1000 which corresponds to a link EP fidelity of 99.5% [40].

Algorithms and Performance Metrics. To compare our techniques with prior approaches, we implement the most recently proposed approaches, viz., (i) the `WaitLess`-based linear programming (LP) approach from [40] (called `Delft-LP` here), (ii) `Q-Cast` approach from [230] which is `WaitLess`-based but uses multiple links and requires memories. The `Waiting`-based algorithm by Caleffi [35] uses an exponential-time approach, and is thus compared only for small networks. The [200] and [41] approaches are not compared as they were found to be inferior to `Q-Cast`. Algorithms `DP-OPT` and `DP-Approx` in [95] are also being compared. We compare our `Balanced-Tree` with all the above-mentioned algorithms largely in terms of EP generation rates and the execution times.
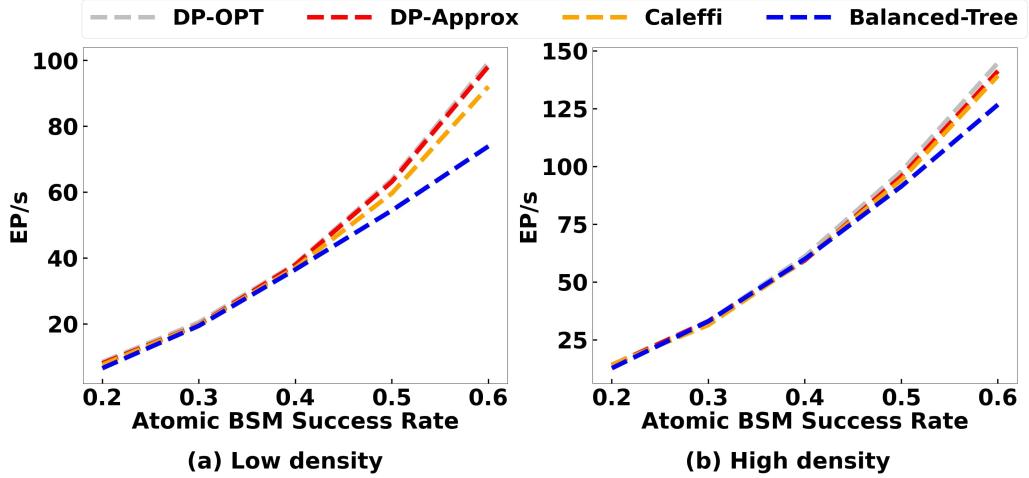
Figure 6.7: Compare the performance with Caleffi in (a) low-density network and (b) high-density network.

For all algorithms except for `Q-Cast`, we use only one link between adjacent nodes, since only `Q-Cast` takes advantage of multiple links in a creative way. In particular, for `Q-Cast`, we use $W = 1, 5$, or 10 sub-links ([230] calls them channels) on each link, with the node and link "capacity" divided equally among them. We note that in `Q-Cast` each node requires $2W$ memories (2 for each sub-link) with sufficient coherence time to allow for the entire swapping operation over the path to be completed. The `Delft-LP` approach explicitly assumes the generation of link EPs is deterministic, i.e., the value $p_g{}^2 p_e{}^2 p_{ob}$ is 1, and does not model node generation rates. We address these differences by extending their LP formulation: (i) We add a constraint on node generation rates, and (ii) add a $p_g{}^2 p_e(i,j)^2 p_{ob}$ factor to each link (i,j) in any path extracted from their LP solution.

Comparison with [35] for `QNR-SP` Problem. Note that [35] gives only an `QNR-SP` algorithm referred to as `Caleffi`; it takes exponential time making it infeasible to run for network sizes much larger than 15-20. In particular, for network sizes 17-20, it takes several hours, and our preliminary analysis suggests that it will take of the order of 10^{40} *years* on our 100-node network. See Table 6.1. Thus, we use a small network of 15 nodes over a 25km × 25km area; we consider average node degrees of 3 or 6. See Fig.6.7. We see that `DP-OPT` outperforms `Caleffi` by 10% on an average for the sparser graph and minimally for the denser graph. We see that `DP-Approx` performs similarly to `DP-OPT`, while `Balanced-Tree` is outperformed slightly by `Caleffi`; however, for this small network, since the `DP-OPT` and `DP-Approx` algorithms only take 10-100s of msecs (Table 6.1), `Balanced-Tree` need not be used in practice.

`QNR-SP` **Problem (Single Tree) Results.** We start with comparing various schemes for the `QNR-SP` problem, in terms of EP generation rate. We compare `DP-Approx`, `DP-OPT`, `Balanced-Tree`, `SP`, and `Q-Cast`; See Fig. 6.8, where we plot the EP generation rate for various schemes for varying number of nodes, (s, d) distance, p_b, and network link density. We observe that `DP-Approx` and `DP-OPT` perform very closely,
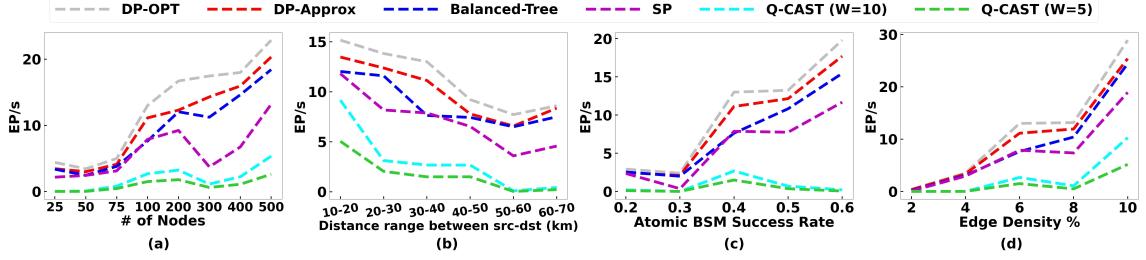
Figure 6.8: `QNR-SP` Problem: EP Generation Rates for varying parameters.
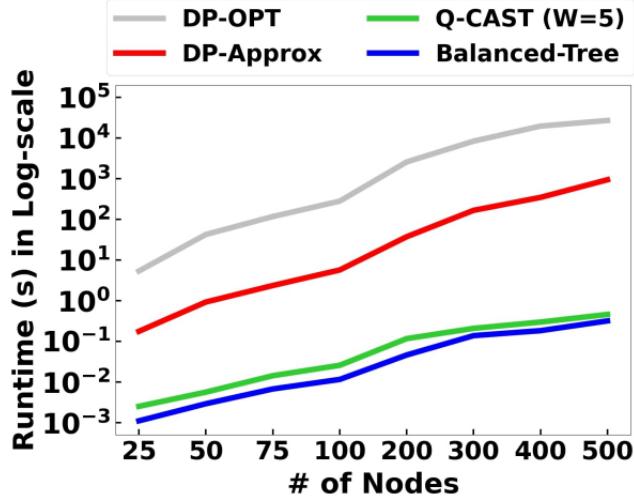
Figure 6.9: The execution time comparison of various algorithms for `QNR-SP` algorithms.

with the `Balanced-Tree` heuristic performing close to them; all these three schemes outperform the `Q-Cast` schemes (for $W = 5, 10$ sub-links) by an order of magnitude. We don't plot `Q-Cast` for $W = 1$ sub-links, as it performs much worse (less than 10^{-3} EP/sec). We note that `Q-Cast`'s EP rates here are much lower than the ones published in [230], because [230] uses link EP success probability of 0.1 or more, while in our more realistic model, the link EP success probability is $p_g{}^2 p_e{}^2 p_{ob} = 0.012$ for the default p_b value. We reiterate that our schemes require only 2 memory units per node, while the `Q-Cast` schemes require $2W$ units. The main reason for poor performance of `Q-Cast` (in spite of higher memory and link synchronization) is that, in the `WaitLess` model, the EP generation over a path is a very low probability event— essentially p^l where p is the link-EP success probability and l is the path length, for the case of $W = 1$ (the analysis for higher W's is involved [230]). Finally, our proposed techniques also outperform the `SP` algorithm, especially when the number of possible paths (trees) between (s, d) pair increases. In addition, we see that performance increases with an increase in p_b, number of nodes, or network link density, as expected due to the availability of better trees/paths; it also increases with a decrease in (s, d) distance as fewer hops are needed.

Execution Times. We ran our simulations on an Intel i7-8700 CPU machine, and

observed that the `WaitLess` algorithms as well our `Balanced-Tree` and `ITER-Bal` heuristics run in fraction of a second even for a 500-node network; thus, they can be used in real-time. Note that since our problems depend on real-time network state (residual capacities), the algorithms must run very fast. The other algorithms (viz., `DP-OPT`, `DP-Approx`, and `ITER-DPA`) can take minutes to hours on large networks, and hence, may be impractical on large network without significant optimization and/or parallelization. See Fig 6.9.

Table 6.1: Execution times of `QNR-SP` algorithm over small networks

Algorithm	Number of nodes					
	10	13	15	16	18	20
`Balanced-Tree`	239μs	360μs	373μs	492μs	530μs	552μs
`DP-Approx`	4ms	10ms	14.7ms	17.6ms	28ms	34ms
`DP-OPT`	148ms	363ms	572ms	706ms	1s	1.7s
Caleffi [35]	92ms	4.6s	14s	26mins	3.2hrs	12.8hrs

Here, we give execution times of different algorithms especially `Caleffi`'s for small networks of 10-20 nodes. See Table 6.1. We see that `Balanced-Tree` and `DP-Approx` take fractions of a second, while `DP-OPT` takes upto 2 seconds. However, as expected `Caleffi`'s execution time increases exponentially with increase in number of nodes – with 20-node network takes 10+ hours. Below, we further estimate `Caleffi`'s execution time for larger graphs.

6.6 Conclusion

We have designed techniques for efficient generation of EP to facilitate quantum network communication, by selecting efficient swapping trees in a `Waiting` protocol. By extensive simulations, we demonstrated the effectiveness of our `Balanced-Tree`, i.e., compared to `DP-Approx`, it significantly decreases the time complexity while the performance drop is only minor.

Appendix

Use S number of sensors to determine the power for the hypothesis at location l^*, such that the hypothesis has the maximum probability. Let $N(\mu_1, \sigma_1^2), \cdots, N(\mu_S, \sigma_S^2)$ be the PDs of the S sensors, built during the training phase, when a transmitter with power p^* is transmitting at location l^*. Let $\mathbf{x} = \{x_1, x_2, \cdots, x_S\}$ be the observation vector of the S sensors during the localization phase. Then we predict the hypothesis at location l^* most likely has the power $p^* + \delta_p$, where

$$\delta_p = \frac{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}(x_j - \mu_j)}{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}}$$

The deduction relies on the assumption that the path loss between a transmitter and receiver is independent of transmit power, or unchanged. The likelihood of $\mathbf{x}$, given $\mathbf{N} = \{N(\mu_1, \delta_1^2), \cdots, N(\mu_S, \delta_S^2)\}$ and δ_p is,

$$P(\mathbf{x}|\mathbf{N}, \delta_p) = \prod_{j=1}^{S} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{\left(-\frac{(x_j - (\mu_j + \delta_p))^2}{2\sigma_j^2}\right)\right\}$$

$$log_e(P(\mathbf{x}|\mathbf{N}, \delta_p)) = log_e\left(\prod_{j=1}^{S} \frac{1}{\sqrt{2\pi\sigma_j^2}}\right) - \left(\sum_{j=1}^{S} \frac{(x_j - (\mu_j + \delta_p))^2}{2\sigma_j^2}\right)$$

$$\frac{\partial(log_e(P(\mathbf{x}|\mathbf{N}, \delta_p)))}{\partial \delta_p} = \sum_{j=1}^{S} \frac{x_j - \mu_j - \delta_p}{\sigma_j^2}$$

To maximize the likelihood of $\mathbf{x}$, let the differential equals to 0.

$$\sum_{j=1}^{S} \frac{x_j - \mu_j - \delta_p}{\sigma_j^2} = 0$$

$$\Rightarrow \delta_p = \frac{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}(x_j - \mu_j)}{\sum_{j=1}^{S} \frac{\gamma}{\sigma_j^2}} \quad , \text{ where } \gamma = \prod_{j=1}^{S} \sigma_j^2$$

Bibliography

[1] (2014). https://www.ettus.com/all-products/ub210-kit/

[2] (2014). https://greatscottgadgets.com/hackrf/one/

[3] (2022). https://www.nist.gov/programs-projects/rydberg-atom-based-quantum-rf-field-probes

[4] (2024). https://en.wikipedia.org/wiki/Wireless_sensor_network

[5] Adu-Manu, K. S., Tapparello, C., Heinzelman, W., et al. (2017). Water quality monitoring using wireless sensor networks: Current trends and future research directions. *ACM Trans. on Sensor Networks*, *13*.

[6] Alizadeh Kharazi, B., & Behzadan, A. H. (2021). Flood depth mapping in street photos with image processing and deep neural networks. *Computers, Environment and Urban Systems*.

[7] Altenburg, S., & Wölk, S. (2018). Multi-parameter estimation: Global, local and sequential strategies. *Physica Scripta*.

[8] Andrews, J. G., Buzzi, S., et al. (2014). What will 5G be? *IEEE Journal on Selected Areas in Communications*, *32*.

[9] Arute et al, F. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, *574*.

[10] Assouly, R., Dassonneville, R., Peronnin, T., Bienfait, A., & Huard, B. (2023). Quantum advantage in microwave quantum radar. *Nature Physics*, *19*.

[11] Ayyalasomayajula, R., Arun, A., et al. (2020). Deep learning based wireless localization for indoor navigation. *ACM MobiCom '20*.

[12] Azuma, K., Tamaki, K., & Lo, H.-K. (2015). All-photonic quantum repeaters. *Nature communications*, *6*.

[13] Bae, J., & Kwek, L.-C. (2015). Quantum state discrimination and its applications. *Journal of Physics A: Mathematical and Theoretical*, *48*.

[14] Bahl, P., & Padmanabhan, V. N. (2000). RADAR: An in-building RF-based user location and tracking system. *IEEE INFOCOM '00*.

[15] Bakshi, A., Mao, Y., Srinivasan, K., & Parthasarathy, S. (2019). Fast and efficient cross band channel prediction using machine learning. *ACM MobiCom '19*.

[16] Barnett, S. M., & Croke, S. (2009). Quantum state discrimination. *Adv. Opt. Photon., 1*.

[17] Bärtschi, A., & Eidenbenz, S. (2019). Deterministic preparation of dicke states. *Fundamentals of Computation Theory*.

[18] Benedetti, M., Lloyd, E., Sack, S., & Fiorentini, M. (2019). Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*.

[19] Bennett, C. H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., & Wootters, W. K. (1993). Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Phys. Rev. Lett., 70*.

[20] Bergou, J. (2007). Quantum state discrimination and selected applications. *Journal of Physics: Conference Series, 84*.

[21] Bergou, J. A., Herzog, U., & Hillery, M. (2004). 11 discrimination of quantum states. In *Quantum state estimation*. Springer.

[22] Bhaskar, M. K., Riedinger, R., Machielse, B., Levonian, D. S., Nguyen, C. T., Knall, E. N., Park, H., Englund, D., et al. (2020). Experimental demonstration of memory-enhanced quantum communication. *Nature, 580*.

[23] Bhattacharya, A., Zhan, C., Gupta, H., Das, S. R., & Djuric, P. M. (2020). Selection of sensors for efficient transmitter localization. *IEEE INFOCOM '20*.

[24] Bhattacharya, A., Abhishek, M., Champati, J. P., Gross, J., et al. (2022). Fast and efficient online selection of sensors for transmitter localization. *International Conference on COMmunication Systems & NETworkS*.

[25] Bhattacharya, A., Zhan, C., Maji, A., Gupta, H., Das, S. R., & Djurić, P. M. (2022). Selection of sensors for efficient transmitter localization. *IEEE/ACM Transactions on Networking, 30*.

[26] Bian, W. (2022). *Optimization-based deep learning methods for magnetic resonance imaging reconstruction and synthesis* (Doctoral dissertation). University of Florida.

[27] Bian, W. (2024). A review of electromagnetic elimination methods for low-field portable mri scanner.

[28] Bian, W., Jang, A., & Liu, F. (2024a). Improving quantitative MRI using self-supervised deep learning with model reinforcement: Demonstration for rapid T1 mapping. *Magnetic Resonance in Medicine*.

[29] Bian, W., Jang, A., & Liu, F. (2024b). Multi-task magnetic resonance imaging reconstruction using meta-learning.

[30] Bian, W., Jang, A., Zhang, L., Yang, X., Stewart, Z., & Liu, F. (2024). Diffusion modeling with domain-conditioned prior guidance for accelerated MRI and qMRI reconstruction. *IEEE Transactions on Medical Imaging*.

[31] Bian, W., & Tamilselvam, Y. K. (2024). A review of optimization-based deep learning models for MRI reconstruction.

[32] Briegel, H.-J., Dür, W., Cirac, J. I., et al. (1998). Quantum repeaters: The role of imperfect local operations in quantum communication. *Phys. Rev. Lett., 81.*

[33] Bringewatt, J., Boettcher, I., Niroula, P., Bienias, P., & Gorshkov, A. V. (2021). Protocols for estimating multiple functions with quantum sensor networks: Geometry and performance. *Phys. Rev. Res., 3.*

[34] Bugalho, L., Coutinho, B. C., Monteiro, F. A., & Omar, Y. (2023). Distributing multipartite entanglement over noisy quantum networks. *Quantum, 7.*

[35] Caleffi, M. (2017). Optimal routing for quantum networks. *IEEE Access.*

[36] Caleffi, M., Cacciapuoti, A. S., & Bianchi, G. (2018). Quantum Internet: From Communication to Distributed Computing! *ACM NANOCOM '18.*

[37] Cao, Y., & Wu, X. (2023). Distributed quantum sensing network with geographically constrained measurement strategies. *IEEE ICASSP '23.*

[38] Chakraborty, A., Rahman, M. S., Gupta, H., et al. (2017). SpecSense: Crowdsensing for efficient querying of spectrum occupancy. *IEEE INFOCOM '17.*

[39] Chakraborty, A., Bhattacharya, A., Kamal, S., Das, S. R., Gupta, H., & Djuric, P. M. (2018). Spectrum patrolling with crowdsourced spectrum sensors. *IEEE INFOCOM '18.*

[40] Chakraborty, K., Elkouss, D., Rijsman, B., & Wehner, S. (2020). Entanglement distribution in a quantum network: A multicommodity flow-based approach. *IEEE Transactions on Quantum Engineering, 1.*

[41] Chakraborty, K., Rozpedek, F., Dahlberg, A., & Wehner, S. (2019). Distributed routing in a quantum internet.

[42] Chamberlin, K., & Luebbers, R. (1982). An evaluation of longley-rice and gtd propagation models. *IEEE Transactions on Antennas and Propagation, 30.*

[43] Chen, J., Lan, T., & Choi, N. (2023). Distributional-utility actor-critic for network slice performance guarantee. *ACM MobiHoc '23.*

[44] Chen, J., Lan, T., & Joe-Wong, C. (2023). Rgmcomm: Return gap minimization via discrete communications in multi-agent reinforcement learning.

[45] Chen, J., Wang, Y., & Lan, T. (2021). Bringing fairness to actor-critic reinforcement learning for network utility optimization. *IEEE INFOCOM '21.*

[46] Chen, J., Zhang, L., Riem, J., Adam, G., Bastian, N. D., & Lan, T. (2023a). Explainable learning-based intrusion detection supported by memristors. *IEEE Conference on Artificial Intelligence (CAI '23).*

[47] Chen, J., Zhang, L., Riem, J., Adam, G., Bastian, N. D., & Lan, T. (2023b). Ride: Real-time intrusion detection via explainable machine learning implemented in a memristor hardware architecture. *IEEE DSC '23.*

[48] Chen, J., Zhou, H., Mei, Y., Adam, G., Bastian, N. D., & Lan, T. (2023). Real-time network intrusion detection via decision transformers.

[49] Chen, T.-Y., Liang, H., Liu, Y., Cai, W.-Q., Ju, L., Liu, W.-Y., Wang, J., Yin, H., Chen, K., Chen, Z.-B., et al. (2009). Field test of a practical secure communication network with decoy-state quantum cryptography. *Optics express*, *17*.

[50] Chen, T., Kim, J., Kuzyk, M., Whitlow, J., Phiri, S., Bondurant, B., Riesebos, L., Brown, K. R., & Kim, J. (2022). Stable turnkey laser system for a yb/ba trapped-ion quantum computer. *IEEE Transactions on Quantum Engineering*.

[51] Chen, T., Zhang, W., Fang, R., et al. (2017). Multi-path photon-phonon converter in optomechanical system at single-quantum level. *Optics Express*, *25*.

[52] Chen, W., Sabharwal, A., Taylor, E., Patel, A. B., & Moukaddam, N. (2021). Privacy-preserving social ambiance measure from free-living speech associates with chronic depressive and psychotic disorders. *Frontiers in Psychiatry*, *12*.

[53] Chen, Z. (2023). *Algorithms and applications of explainable machine learning* (Doctoral dissertation). Stony Brook University.

[54] Chen, Z., Silvestri, F., Tolomei, G., Wang, J., Zhu, H., & Ahn, H. (2022). Explain the explainer: Interpreting model-agnostic counterfactual explanations of a deep reinforcement learning agent. *IEEE Transactions on Artificial Intelligence*, *5*.

[55] Chen, Z., Silvestri, F., Wang, J., Zhang, Y., & Tolomei, G. (2023). The dark side of explanations: Poisoning recommender systems with counterfactual examples. *ACM SIGIR '23*.

[56] Chen, Z., Silvestri, F., Wang, J., Zhu, H., Ahn, H., & Tolomei, G. (2022). Relax: Reinforcement learning agent explainer for arbitrary predictive models. *ACM CIKM '22*.

[57] Cheng, L., Wu, C., Zhang, Y., Maple, C., et al. (2012). A survey of localization in wireless sensor network. *IJDSN*.

[58] Chipara, O., Lu, C., Bailey, T. C., & Roman, G.-C. (2010). Reliable clinical monitoring using wireless sensor networks: Experiences in a step-down hospital unit. *ACM SenSys '10*.

[59] Coopmans, T., Brand, S., & Elkouss, D. (2022). Improved analytical bounds on delivery times of long-distance entanglement. *Phys. Rev. A*, *105*.

[60] Coopmans, T., Knegjens, R., Dahlberg, A., Maier, D., Nijsten, L., Oliveira, J., Wehner, S., et al. (2021). NetSquid, a discrete-event simulation platform for quantum networks. *Communications Physics*, *4*.

[61] Corbalan, P., Picco, G., et al. (2019). Chorus: UWB concurrent transmissions for GPS-like passive localization of countless targets. *ACM/IEEE IPSN '19*.

[62] Dahlberg, A., Skrzypczyk, M., Coopmans, T., Wubben, L., et al. (2019). A link layer protocol for quantum networks. *SIGCOMM '19*.

[63] D'Ariano, G. M., Presti, P. L., & Paris, M. G. A. (2002). Improved discrimination of unitary transformations by entangled probes. *Journal of Optics B: Quantum and Semiclassical Optics*, *4*.

[64]   Dasari, M., Atigue, M. B., Bhattacharya, A., et al. (2019). Spectrum protection from micro-transmissions using distributed spectrum patrolling. *PAM '19*.

[65]   de Almeida, I. B. F., Chafii, M., et al. (2021). Blind transmitter localization in wireless sensor networks: A deep learning approach. *IEEE PIMRC '21*.

[66]   Degen, C. L., Reinhard, F., & Cappellaro, P. (2017). Quantum sensing. *Rev. Mod. Phys.*, *89*.

[67]   Deutsch, C., Ramirez-Martinez, F., Lacroûte, C., Reinhard, F., Schneider, T., Fuchs, J. N., Piéchon, F., et al. (2010). Spin self-rephasing and very long coherence times in a trapped atomic ensemble. *Phys. Rev. Lett.*, *105*.

[68]   Devitt, S. J., Munro, W. J., & Nemoto, K. (2013). Quantum error correction for beginners. *Reports on Progress in Physics*, *76*.

[69]   Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, *17*.

[70]   Dieks, D. (1982). Communication by EPR devices. *Physics Letters A*, *92*.

[71]   Dober, B., Ahmed, Z., Arnold, K., Becker, D. T., Bennett, D. A., Connors, J. A., Cukierman, A., D'Ewart, J. M., et al. (2021). A microwave SQUID multiplexer optimized for bolometric applications. *Applied Physics Letters*, *118*.

[72]   Dong, G. (2022). *Deep graph learning in mobile health* (Doctoral dissertation). University of Virginia.

[73]   Dong, G., Boukhechba, M., Shaffer, K. M., Ritterband, L. M., & Gioeli, D. G. (2021). Using graph representation learning to predict salivary cortisol levels in pancreatic cancer patients. *Journal of Healthcare Informatics Research*.

[74]   Dong, G., Tang, M., Cai, L., Barnes, L. E., & Boukhechba, M. (2021). Semi-supervised graph instance transformer for mental health inference. *20th IEEE International Conference on Machine Learning and Applications (ICMLA)*.

[75]   Dong, G., Tang, M., Wang, Z., Gao, J., Guo, S., Cai, L., Gutierrez, R., Campbel, B., Barnes, L. E., & Boukhechba, M. (2023). Graph neural networks in IoT: A survey. *ACM Trans. Sen. Netw.*, *19*.

[76]   Duan, L.-M., Lukin, M. D., Cirac, J. I., & Zoller, P. (2001). Long-distance quantum communication with atomic ensembles and linear optics. *Nature*, *414*.

[77]   Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

[78]   Dutta, A., & Chiang, M. (2016). "see something, say something" crowdsourced enforcement of spectrum policies. *IEEE Trans. on Wireless Comm.*, *15*.

[79]   Egerstrom, C. (2023). A mathematical introduction to quantum sensing [https://cklixx.people.wm.edu/teaching/math300/ChrisE.pdf].

[80]   Ehrenberg, A., Bringewatt, J., & Gorshkov, A. V. (2023). Minimum-entanglement protocols for function estimation. *Phys. Rev. Res.*, *5*.

[81] Eldar, Y. C., Megretski, A., et al. (2003). Designing optimal quantum detectors via semidefinite programming. *IEEE Trans. Information Theory.*

[82] Eldredge, Z., Foss-Feig, M., Gross, J. A., Rolston, S. L., & Gorshkov, A. V. (2018). Optimal and secure measurement protocols for quantum sensor networks. *Phys. Rev. A, 97.*

[83] *Electromagnetic spectrum superiority strategy* (tech. rep.). (2020). US Department of Defence.

[84] Englert, B.-G., & Řeháček, J. (2010). How well can you know the edge of a quantum pyramid? *Journal of Modern Optics, 57.*

[85] Fan, X., Zhan, C., Gupta, H., & Ramakrishnan, C. R. (2024). Optimized distribution of entanglement graph states in quantum networks.

[86] Fraval, E., Sellars, M. J., & Longdell, J. J. (2005). Dynamic decoherence control of a solid-state nuclear-quadrupole qubit. *Phys. Rev. Lett., 95.*

[87] G Sundaram, R., Gupta, H., & Ramakrishnan, C. (2021). Efficient distribution of quantum circuits. *35th International Symposium on Distributed Computing.*

[88] Gambetta, J. (2020). IBM's Roadmap For Scaling Quantum Technology.

[89] Gan, Y., Zhang, X., Zhou, R., Liu, Y., & Qian, C. (2023). A routing framework for quantum entanglements with heterogeneous duration. *IEEE QCE '23.*

[90] Ge, W., Jacobs, K., Eldredge, Z., Gorshkov, A. V., & Foss-Feig, M. (2018). Distributed quantum metrology with linear networks and separable inputs. *Phys. Rev. Lett., 121.*

[91] Ghaderibaneh, M., Dasari, M., & Gupta, H. (2019). Multiple transmitter localization under time-skewed observations. *IEEE DySpan '19.*

[92] Ghaderibaneh, M., Gupta, H., Ramakrishnan, C., et al. (2022). Pre-distribution of entanglements in quantum networks. *IEEE QCE '22.*

[93] Ghaderibaneh, M., Gupta, H., & Ramakrishnan, C. (2023). Generation and distribution of GHZ states in quantum networks. *IEEE QCE '23.*

[94] Ghaderibaneh, M., Zhan, C., & Gupta, H. (2024). Deepalloc: Deep learning approach to spectrum allocation in shared spectrum systems. *IEEE Access, 12.*

[95] Ghaderibaneh, M., Zhan, C., Gupta, H., & Ramakrishnan, C. R. (2022). Efficient quantum network communication using optimized entanglement swapping trees. *IEEE Transactions on Quantum Engineering (TQE), 3.*

[96] Giovannetti, V., Lloyd, S., & Maccone, L. (2011). Advances in quantum metrology. *Nature Photonics, 5.*

[97] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *IEEE CVPR '14.*

[98] Goswami, A., Ortiz, L., & Das, S. (2011). Wigem: A learning-based approach for indoor localization. *ACM CoNEXT '11*.

[99] Grobwindhager, B., Stocker, M., et al. (2019). SnapLoc: An ultra-fast UWB-based indoor localization system for an unlimited number of tags. *ACM/IEEE IPSN '19*.

[100] Gu, H., Li, Z., Yu, R., Wang, X., Zhou, F., Liu, J., & Xue, G. (2023). FENDI: Toward high-fidelity entanglement distribution in the quantum internet.

[101] Gu, H., Yu, R., Li, Z., Wang, X., & Zhou, F. (2023). ESDI: Entanglement scheduling and distribution in the quantum internet. *ICCCN '23*.

[102] Gühne, O., & Tóth, G. (2009). Entanglement detection. *Physics Reports, 474*.

[103] Gupta, H., Curran, M., Longtin, J., Rockwell, T., Zheng, K., & Dasari, M. (2022). Cyclops: An FSO-based wireless link for VR headsets. *SIGCOMM '22*.

[104] Gupta, H., Curran, M., & Zhan, C. (2020). Near-optimal multihop scheduling in general circuit-switched networks. *ACM CoNEXT '20*.

[105] Hao, Z., Huang, Q., Wang, C., Wang, J., Zhang, Y., Wu, R., & Zhang, C. (2023). Pinolo: Detecting logical bugs in database management systems with approximate query synthesis. *USENIX ATC '23*.

[106] Hartung, L., & Milind, M. (2015). Policy driven multi-band spectrum aggregation for ultra-broadband wireless networks. *IEEE DySpan '15*.

[107] Hausladen, P., & Wootters, W. K. (1994). A 'pretty good' measurement for distinguishing quantum states. *Journal of Modern Optics*.

[108] He, T., Huang, C., Blum, B. M., Stankovic, J. A., et al. (2003). Range-free localization schemes for large scale sensor networks. *ACM MobiCom '03*.

[109] Helstrom, C. (1976). *Quantum detection and estimation theory*. Academic Press.

[110] Hillery, M., Gupta, H., & Zhan, C. (2023). Discrete outcome quantum sensor networks. *Phys. Rev. A, 107*.

[111] Holland, J. H. (1992). Genetic algorithms. *Scientific American, 267*.

[112] Hu, J., McManus, M., Moorthy, S. K., Cui, Y., Guan, Z., Mastronarde, N., et al. (2022). NeXT: A Software-Defined Testbed with Integrated Optimization, Simulation and Experimentation. *IEEE FNWF '22*.

[113] Hu, J., Moorthy, S. K., Harindranath, A., Guan, Z., Mastronarde, N., Bentley, E. S., & Scott, P. (2021). SwarmShare: Mobility-Resilient Spectrum Sharing for Swarm UAV Networking in the 6 GHz Band". *IEEE SECON '21*.

[114] Hu, J., Moorthy, S. K., Harindranath, A., Zhaoxi, Z., Zhiyuan, Z., Nicholas, M., et al. (2023). A Mobility-Resilient Spectrum Sharing Framework for Operating Wireless UAVs in the 6 GHz Band. *IEEE/ACM TON, 31*.

[115] Hu, J., Zhao, Z., McManus, M., et al. (2023). NeXT: Architecture, prototyping and measurement of a software-defined testing framework for integrated RF network simulation, experimentation and optimization. *Computer Communications*.

[116] Hua, Y., Xie, Z., & Ye, F. (2023). A study of practical radar-based nighttime respiration monitoring at home. *IEEE RadarConf23*.

[117] Huang, B., & Wang, J. (2023). Adaptive static equivalences for active distribution networks with massive renewable energy integration: A distributed deep reinforcement learning approach. *IEEE Trans. Netw. Sci. Eng.*

[118] Huang, B., Zhao, T., Yue, M., & Wang, J. (2023). Bi-level adaptive storage expansion strategy for microgrids using deep reinforcement learning. *IEEE Transactions on Smart Grid, 15*.

[119] Huang, C., Bandyopadhyay, A., Fan, W., Miller, A., & Gilbertson-White, S. (2023). Mental toll on working women during the covid-19 pandemic: An exploratory study using reddit data. *PloS one, 18*.

[120] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*.

[121] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[122] Islam, B., Luo, Y., Lee, S., & Nirjon, S. (2019). On-device training from sensor data on batteryless platforms: Poster abstract. *ACM/IEEE IPSN '19*.

[123] Islam, B., Luo, Y., & Nirjon, S. (2023). Amalgamated intermittent computing systems. *IoTDI '23*.

[124] Iten, R., Reardon-Smith, O., Malvetti, E., Mondada, L., Pauvert, G., Redmond, E., Kohli, R. S., & Colbeck, R. (2021). Introduction to UniversalQ-Compiler.

[125] Jiang, L., Taylor, J. M., Khaneja, N., & Lukin, M. D. (2007). Optimal approach to quantum communication using dynamic programming. *Proceedings of the National Academy of Sciences, 104*.

[126] Jin, Y., Gao, X., Guo, M., Chen, H., Hua, F., Zhang, C., & Zhang, E. Z. (2023). Quantum fourier transformation circuits compilation.

[127] Jin, Y., Hua, F., Chen, Y., Hayes, A., Zhang, C., & Zhang, E. Z. (2023). Exploiting the regular structure of modern quantum architectures for compiling and optimizing programs with permutable operators. *ACM ASPLOS '23*.

[128] Karanam, C., Korany, B., et al. (2019). Tracking from one side – multi-person passive tracking with wifi magnitude measurements. *ACM/IEEE IPSN '19*.

[129] Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications, 80*.

[130] Khaledi, M., Khaledi, M., Sarkar, S., Kasera, S., Patwari, N., et al. (2017). Simultaneous power-based localization of transmitters for crowdsourced spectrum monitoring. *ACM MobiCom '17*.

[131] Kim, C. W., Ryoo, J., & Buddhikot, M. M. (2015). Design and implementation of an end-to-end architecture for 3.5 GHz shared spectrum. *IEEE DySPAN '15*.

[132] Kim, S., Jeon, H., Lee, H., & Ma, J. (2007). Robust transmission power and position estimation in cognitive radio. *IEEE MILCOM '07*.

[133] Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization.

[134] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*.

[135] Kocakulak, M., & Butun, I. (2017). An overview of wireless sensor networks towards internet of things. *IEEE Annual Computing and Communication Workshop and Conference (CCWC '17)*.

[136] Koczor, B., Endo, S., Jones, T., Matsuzaki, Y., & Benjamin, S. C. (2020). Variational-state quantum metrology. *New Journal of Physics*, *22*.

[137] Komar, P., Kessler, E. M., Bishof, M., Jiang, L., Sørensen, A. S., Ye, J., & Lukin, M. D. (2014). A quantum network of clocks. *Nature Physics*, *10*.

[138] Kour, H., Jha, R. K., & Jain, S. (2018). A comprehensive survey on spectrum sharing: Architecture, energy efficiency and security issues. *Journal of Network and Computer Applications*.

[139] Kozlowski, W., Dahlberg, A., & Wehner, S. (2020). Designing a quantum network protocol. *ACM CoNEXT '20*.

[140] Laarhoven, P. J. M., & Aarts, E. H. L. (1987). *Simulated annealing: Theory and applications*. Kluwer Academic Publishers.

[141] Langer, C., Ozeri, R., Jost, J. D., Chiaverini, J., DeMarco, B., Ben-Kish, A., Blakestad, R. B., Britton, J., et al. (2005). Long-lived qubit memory using atomic ions. *Phys. Rev. Lett.*, *95*.

[142] Lee, S., Islam, B., Luo, Y., & Nirjon, S. (2020). Intermittent learning: On-device machine learning on intermittently powered system. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*

[143] Lewis, R. (2007). Metaheuristics can solve sudoku puzzles. *Journal of Heuristics*.

[144] Li, B., Coopmans, T., & Elkouss, D. (2020). Efficient optimization of cut-offs in quantum repeater chains. *IEEE QCE '20*.

[145] Li, D., Jin, R., Gao, J., & Liu, Z. (2020). On sampling top-k recommendation evaluation. *KDD '20*.

[146] Li, D., Jin, R., Liu, Z., Ren, B., Gao, J., & Liu, Z. (2023). Towards reliable item sampling for recommendation evaluation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*.

[147] Li, D., Jin, R., Liu, Z., Ren, B., Gao, J., & Liu, Z. (2024). On item-sampling evaluation for recommender system. *ACM Trans. Recomm. Syst.*, *2*.

[148] Li, H., Liu, J., Zhang, X., Bai, Y., Wang, H., & Mueller, K. (2023). Transforming the latent space of stylegan for real face editing. *The Visual Computer*.

[149] Li, X., Cao, J.-H., Liu, Q., Tey, M. K., & You, L. (2020). Multi-parameter estimation with multi-mode ramsey interferometry. *New Journal of Physics*, *22*.

[150] Li, X., Wang, X., Chen, X., Lu, Y., Fu, H., & Wu, Y. C. (2024). Unlabeled data selection for active learning in image classification. *Scientific Reports*, *14*.

[151] Li, Y. (2023). *Improving inspector training and understanding through human-centric intelligence methods for drone-assisted bridge inspection* (Doctoral dissertation). Stony Brook University.

[152] Li, Z., Zhu, H., Liu, H., Song, J., & Cheng, Q. (2024). Comprehensive evaluation of mal-api-2019 dataset by machine learning in malware detection. *International Journal of Computer Science and Information Technology*.

[153] Li, Z., Xiao, Z., Wang, B., Zhao, B. Y., & Zheng, H. (2019). Scaling deep learning models for spectrum anomaly detection. *ACM MobiHoc '19*.

[154] Liang, Z., Cheng, J., Yang, R., Ren, H., Song, Z., Wu, D., Qian, X., Li, T., & Shi, Y. (2023). Unleashing the potential of LLMs for quantum computing: A study in quantum architecture design.

[155] Liang, Z., Liu, G., Liu, Z., Cheng, J., Hao, T., Liu, K., Ren, H., Song, Z., Liu, J., Ye, F., & Shi, Y. (2024). Graph learning for parameter prediction of quantum approximate optimization algorithm. *ACM/IEEE DAC '24*.

[156] Liang, Z., & Wang, H. (2023). QuCS: A lecture series on quantum computer software and system. *IEEE QCE '23*.

[157] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., et al. (2014). Microsoft COCO: common objects in context.

[158] Linder-Norén, E. (2019). *Open source yolov3 implementation*. https://github.com/eriklindernoren/PyTorch-YOLOv3

[159] Liu, C., Zhu, N., Sun, H., Zhang, J., Feng, X., Gjerswold-Selleck, S., Sikka, D., et al. (2022). Deep learning of MRI contrast enhancement for mapping cerebral blood volume from single-modal non-contrast scans of aging and alzheimer's disease brains. *Frontiers in Aging Neuroscience*, *14*.

[160] Liu, L., Ouyang, W., Wang, X., Fieguth, P. W., Chen, J., Liu, X., & Pietikäinen, M. (2018). Deep learning for generic object detection: A survey.

[161] Liu, L., Zhang, Y., Li, Z., et al. (2021). Distributed quantum phase estimation with entangled photons. *Nature Photonics*, *15*.

[162] Liu, S., Wu, K., Jiang, C., Huang, B., & Ma, D. (2023). Financial time-series forecasting: Towards synergizing performance and interpretability within a hybrid machine learning approach.

[163] Liu, Y., Zhang, S., & Gowda, M. (2021). When video meets inertial sensors: Zero-shot domain adaptation for finger motion analytics with inertial sensors. *IoTDI '21*.

[164] Lloyd, S., Schuld, M., Ijaz, A., Izaac, J., & Killoran, N. (2020). Quantum embeddings for machine learning.

[165] Longdell, J. J., Fraval, E., Sellars, M. J., & Manson, N. B. (2005). Stopped light with storage times greater than one second using electromagnetically induced transparency in a solid. *Phys. Rev. Lett.*, *95*.

[166] Luo, W., Li, Y., Urtasun, R., & Zemel, R. (2017). Understanding the effective receptive field in deep convolutional neural networks.

[167] Luo, X., Ma, X., Munden, M., Wu, Y.-J., & Jiang, Y. (2022). A multisource data approach for estimating vehicle queue length at metered on-ramps. *Journal of Transportation Engineering, Part A: Systems*, *148*.

[168] Luo, Y. (2023). *Scaling up task execution on resource-constrained systems* (Doctoral dissertation). University of North Carolina at Chapel Hill.

[169] Luo, Y., & Huang, Y. (2017). Text steganography with high embedding rate: Using recurrent neural networks to generate chinese classic poetry. *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*.

[170] Luo, Y., & Nirjon, S. (2021). Smarton: Just-in-time active event detection on energy harvesting systems. *17th DCOSS '21*.

[171] Luo, Y., Zhang, L., Wang, Z., & Nirjon, S. (2023). Efficient multitask learning on resource-constrained systems.

[172] Lyu, W., Dong, X., Wong, R., Zheng, S., et al. (2022). A multimodal transformer: Fusing clinical notes with structured ehr data for interpretable in-hospital mortality prediction. *AMIA Annual Symposium Proceedings*.

[173] Lyu, W., Zheng, S., Ma, T., & Chen, C. (2022). A study of the attention abnormality in trojaned BERTs. *NAACL '22*.

[174] Lyu, W., Zheng, S., Pang, L., Ling, H., & Chen, C. (2023). Attention-enhancing backdoor attacks against BERT-based models. *EMNLP '23*.

[175] Ma, X. (2022). *Traffic performance evaluation using statistical and machine learning methods* (Doctoral dissertation). University of Arizona.

[176] Ma, X., Karimpour, A., & Wu, Y.-J. (2020). Statistical evaluation of data requirement for ramp metering performance assessment. *Transportation Research Part A: Policy and Practice*, *141*.

[177] Ma, X., Karimpour, A., & Wu, Y.-J. (2023). Eliminating the impacts of traffic volume variation on before and after studies: A causal inference approach. *Journal of Intelligent Transportation Systems*.

[178] Ma, X., Yan, F., Yang, L., Foster, I., Papka, M. E., Liu, Z., & Kettimuthu, R. (2024). Malletrain: Deep neural networks training on unfillable supercomputer nodes. *ACM/SPEC International Conference on Performance Engineering*.

[179] Magliacane, J. A. (2008). SPLAT! a terrestrial RF path analysis application for Linux/Unix.

[180] Mao, Y., & Venkatakrishnan, S. B. (2023a). Less is more: Understanding network bias in proof-of-work blockchains. *Mathematics*, *11*.

[181] Mao, Y., & Venkatakrishnan, S. B. (2023b). Topiary: Fast, scalable publish/-subscribe for peer-to-peer (D)apps.

[182] Marcozzi, M., & Mostarda, L. (2021). Quantum consensus: An overview.

[183] McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., & Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. *Nature Communications*, *9*.

[184] Meyer, D. H., Kunz, P. D., & Cox, K. C. (2021). Waveguide-coupled rydberg spectrum analyzer from 0 to 20 GHz. *Phys. Rev. Applied*, *15*.

[185] Min, Y., He, J., Wang, T., & Gu, Q. (2022). Learning stochastic shortest path with linear function approximation. *ICML '22*.

[186] Min, Y., He, J., Wang, T., & Gu, Q. (2023). Cooperative multi-agent reinforcement learning: Asynchronous communication and linear function approximation. *ICML '23*.

[187] Min, Y., Wang, T., Xu, R., Wang, Z., Jordan, M., & Yang, Z. (2022). Learn to match with no regret: Reinforcement learning in markov matching markets. *NeurIPS '22*.

[188] Min, Y., Wang, T., Zhou, D., & Gu, Q. (2021). Variance-aware off-policy evaluation with linear function approximation. *NeurIPS '21*.

[189] Monjur, M., Luo, Y., Wang, Z., & Nirjon, S. (2023). Soundsieve: Seconds-long audio event recognition on intermittently-powered systems. *ACM MobiSys '23*.

[190] Muralidharan, S., Li, L., Kim, J., Lütkenhaus, N., Lukin, M. D., & Jiang, L. (2016). Optimal architectures for long distance quantum communication. *Scientific reports*, *6*.

[191] Nagy, M., & Nagy, N. (2022). Intrusion detection quantum sensor networks. *Sensors*, *22*.

[192] Narayanan, A., Zhang, X., et al. (2021). A variegated look at 5G in the wild: Performance, power, and QoE implications. *ACM SIGCOMM '21*.

[193] Nelson, J., Hazen, M., & Gupta, M. (2006). Global optimization for multiple transmitter localization. *IEEE MILCOM '06*.

[194] Nelson, J. K., et al. (2009). A quasi EM method for estimating multiple transmitter locations. *IEEE Signal Processing Letters*, *16*.

[195] Ni, H., Meng, S., Chen, X., Zhao, Z., Chen, A., Li, P., Zhang, S., Yin, Q., Wang, Y., & Chan, Y. (2024). Harnessing earnings reports for stock predictions: A QLoRA-Enhanced LLM approach.

[196] Ni, H., Meng, S., Geng, X., Li, P., Li, Z., Chen, X., Wang, X., & Zhang, S. (2024). Time series modeling for heart rate prediction: From ARIMA to transformers.

[197] Nielsen, M. A., & Chuang, I. L. (2011). *Quantum computation and quantum information: 10th anniversary edition* (10th). Cambridge University Press.

[198] O'Donoghue, B., Chu, E., Parikh, N., & Boyd, S. (2016). Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications, 169*.

[199] Panda, S. K., Lin, M., & Zhou, T. (2023). Energy-efficient computation offloading with dvfs using deep reinforcement learning for time-critical iot applications in edge computing. *IEEE Internet of Things Journal, 10*.

[200] Pant, M., Krovi, H., Towsley, D., Tassiulas, L., Jiang, L., Basu, P., Englund, D., & Guha, S. (2020). Routing entanglement in the quantum internet. *NPJ Quantum Information, 5*.

[201] Paszke, A., Gross, S., Massa, F., Lerer, A., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32*.

[202] Patterson, A., Chen, H., Wossnig, L., Severini, S., Browne, D., & Rungger, I. (2021). Quantum state discrimination using noisy quantum neural networks. *Phys. Rev. Res., 3*.

[203] Patwari, N., et al. (2005). Locating the nodes: Cooperative localization in wireless sensor networks. *IEEE Signal processing magazine, 22*.

[204] Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*.

[205] Penumarthi, P. K., et al. (2017). Multirobot exploration for building communication maps with prior from communication models. *Intl. Symp. on Multi-Robot and Multi-Agent Systems*.

[206] Pirandola, S., Bardhan, B. R., Gehring, T., Weedbrook, C., & Lloyd, S. (2018). Advances in photonic quantum sensing. *Nature Photonics, 12*.

[207] Press, D., De Greve, K., McMahon, P. L., Ladd, T. D., Friess, B., Schneider, C., Kamp, M., Höfling, S., Forchel, A., & Yamamoto, Y. (2010). Ultrafast optical spin echo in a single quantum dot. *Nature Photonics, 4*.

[208] Proctor, T. J., Knott, P. A., & Dunningham, J. A. (2018). Multiparameter estimation in networked quantum sensors. *Phys. Rev. Lett., 120*.

[209] Qi, Y., Luo, Y., Huang, Y., & Li, X. (2023). Blockchain-based privacy-preserving group data auditing with secure user revocation. *Computer Systems Science & Engineering, 45*.

[210] Qian, T., Bringewatt, J., Boettcher, I., Bienias, P., & Gorshkov, A. V. (2021). Optimal measurement of field properties with quantum sensor networks. *Phys. Rev. A, 103*.

[211] Rajendran, S., Calvo-Palomino, R., Fuchs, M., Van den Bergh, B., Cordobes, H., Giustiniano, D., Pollin, S., & Lenders, V. (2018). Electrosense: Open and big spectrum data. *IEEE Communications Magazine*, *56*.

[212] Rajendran, S., Meert, W., Giustiniano, D., Lenders, V., et al. (2018). Deep learning models for wireless signal classification with distributed low-cost spectrum sensors. *IEEE Transactions on Cognitive Communications and Networking*, *4*.

[213] Rappaport, T. (2001). *Wireless communications: Principles and practice* (2nd). Prentice Hall PTR.

[214] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *IEEE CVPR '16*.

[215] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement.

[216] Reynders, B., Minucci, F., Perenda, E., et al. (2020). Skysense: Terrestrial and aerial spectrum use analysed using lightweight sensing technology with weather balloons. *ACM MobiSys '20*.

[217] Rizk, H., et al. (2019). MonoDCell: A ubiquitous and low-overhead deep learning-based indoor localization with limited cellular information. *ACM SIGSPATIAL*.

[218] Roffe, J. (2019). Quantum error correction: An introductory guide. *Contemporary Physics*, *60*.

[219] Roga, W., Ikuta, R., Horikiri, T., & Takeoka, M. (2023). Efficient dicke-state distribution in a network of lossy channels. *Phys. Rev. A*, *108*.

[220] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. *MICCAI 2015*.

[221] Rubio, J., et al. (2020). Quantum sensing networks for the estimation of linear functions. *Journal of Physics A: Mathematical and Theoretical*.

[222] Saeedi, K., Simmons, S., Salvail, J. Z., Dluhy, P., Riemann, H., et al. (2013). Room-temperature quantum bit storage exceeding 39 minutes using ionized donors in silicon-28. *Science*, *342*.

[223] Sagi, Y., Almog, I., & Davidson, N. (2010). Process tomography of dynamical decoupling in a dense cold atomic ensemble. *Phys. Rev. Lett.*, *105*.

[224] Saleem, Z. H., Perlin, M., Shaji, A., & Gray, S. K. (2023). Achieving the heisenberg limit with dicke states in noisy quantum metrology.

[225] Sangouard, N., Simon, C., De Riedmatten, H., et al. (2011). Quantum repeaters based on atomic ensembles and linear optics. *Reviews of Modern Physics*, *83*.

[226] Sarkar, S., Buddhikot, M., Baset, A., & Kasera, S. K. (2021). DeepRadar: A deep-learning-based environmental sensing capability sensor design for CBRS.

[227] Scarani, V., Bechmann-Pasquinucci, H., Cerf, N. J., Dušek, M., Lütkenhaus, N., & Peev, M. (2009). The security of practical quantum key distribution. *Reviews of modern physics, 81.*

[228] Schmidt, M., et al. (2017). Wireless interference identification with convolutional neural networks. *IEEE Intl. Conf. on Industrial Informatics (INDIN).*

[229] Sheriff, J., Wang, P., Zhang, P., Zhang, Z., Deng, Y., & Bluestein, D. (2023). In vitro measurements of shear-mediated platelet adhesion kinematics as analyzed through machine learning. *Annals of Biomedical Engineering.*

[230] Shi, S., & Qian, C. (2020). Concurrent entanglement routing for quantum networks: Model and designs. *SIGCOMM '20.*

[231] Shokry, A., & Youssef, M. (2022). A quantum algorithm for rf-based fingerprinting localization systems. *IEEE LCN '22.*

[232] Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SICOMP.*

[233] Sidhu, J. S., & Kok, P. (2020). Geometric perspective on quantum parameter estimation. *AVS Quantum Science, 2.*

[234] Simon, C. (2017). Towards a global quantum network. *Nature Photonics, 11.*

[235] Srivastava, S., Huang, C., Fan, W., & Yao, Z. (2023). Instance needs more care: Rewriting prompts for instances yields better zero-shot performance.

[236] Steger, M., Saeedi, K., Thewalt, M., Morton, J., Riemann, H., et al. (2012). Quantum information storage for over 180 s using donor spins in a 28si "semiconductor vacuum". *Science, 336.*

[237] Sun, H., Liu, X., Feng, X., Liu, C., et al. (2020). Substituting gadolinium in brain MRI using deepcontrast. *IEEE ISBI '20.*

[238] Sun, X., Li, W., Tian, Y., Li, F., Tian, L., et al. (2022). Quantum positioning and ranging via a distributed sensor network. *Photonics Research, 10.*

[239] Sundaram, R. G., & Gupta, H. (2023). Distribution of quantum circuits using teleportations. *IEEE QSW '23.*

[240] Sundaram, R. G., Gupta, H., & Ramakrishnan, C. (2022). Distribution of quantum circuits over general quantum networks. *IEEE QCE '22.*

[241] Suzuki, J., Yang, Y., & Hayashi, M. (2020). Quantum state estimation with nuisance parameters. *Journal of Physics A: Mathematical and Theoretical, 53.*

[242] Tang, M., Dong, G., Zoellner, J., Bowman, B., et al. (2022). Using ubiquitous mobile sensing and temporal sensor-relation graph neural network to predict fluid intake of end stage kidney patients. *ACM/IEEE IPSN '22.*

[243] The LIGO Scientific Collaboration et al. (2015). Advanced LIGO. *Classical and Quantum Gravity, 32.*

[244] Tian, J. (2023). *Advances in statistical gene networks* (Doctoral dissertation). Carnegie Mellon University.

[245] Tian, J., Zhou, K., Wang, M., Zhang, Y., Yao, B., et al. (2023). UseClean: Learning from complex noisy labels in named entity recognition. *Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD)*.

[246] Tittel, W., Afzelius, M., Cone, R. L., Chanelière, T., Kröll, S., Moiseev, S. A., & Sellars, M. (2008). Photon-echo quantum memory.

[247] Üreten, O., & Willink, T. J. (2011). Joint estimation of emitter power and location in cognitive radio networks. *IEEE International Workshop on Signal Processing Advances in Wireless Communications*.

[248] van Loock, P., Alt, W., Becher, C., Benson, O., Boche, H., Deppe, C., Eschner, J., Höfling, S., Meschede, D., Michler, P., Schmidt, F., & Weinfurter, H. (2020). Extending quantum links: Modules for fiber- and memory-based quantum repeaters. *Advanced Quantum Technologies, 3*.

[249] Vetsch, E., Reitz, D., Sagué, G., Schmidt, R., Dawkins, S. T., & Rauschenbeutel, A. (2010). Optical interface created by laser-cooled atoms trapped in the evanescent field surrounding an optical nanofiber. *Phys. Rev. Lett., 104*.

[250] Wang, F., Kapse, S., Liu, S., Prasanna, P., & Chen, C. (2021). TopoTxR: A topological biomarker for predicting treatment response in breast cancer. *Information Processing in Medical Imaging*.

[251] Wang, F., Liu, H., Samaras, D., & Chen, C. (2020). TopoGAN: A topology-aware generative adversarial network. *ECCV '20*.

[252] Wang, H., Ding, Y., Gu, J., Li, Z., Lin, Y., Pan, D. Z., Chong, F. T., & Han, S. (2022). Quantumnas: Noise-adaptive search for robust quantum circuits. *HPCA*.

[253] Wang, H., Gu, J., Ding, Y., et al. (2022). Quantumnat: Quantum noise-aware training with noise injection, quantization and normalization. *ACM DAC '22*.

[254] Wang, H., He, Y.-M., Chung, T.-H., Hu, H., Yu, Y., Chen, S., Ding, X., Chen, M.-C., Qin, J., Yang, X., et al. (2019). Towards optimal single-photon sources from polarized microcavities. *Nature Photonics, 13*.

[255] Wang, J. (2023). *Performant, scalable, and efficient deployment of network function virtualization* (Doctoral dissertation). University of Southern California.

[256] Wang, J., Gupta, S., Vieira, M. A. M., Raghavan, B., & Govindan, R. (2023). Scheduling network function chains under sub-millisecond latency slos.

[257] Wang, J., Lévai, T., Li, Z., Vieira, M. A. M., Govindan, R., & Raghavan, B. (2021). Galleon: Reshaping the square peg of NFV.

[258] Wang, J., Lévai, T., Li, Z., Vieira, M. A. M., Govindan, R., & Raghavan, B. (2022). Quadrant: A cloud-deployable nf virtualization platform. *Proceedings of the 13th Symposium on Cloud Computing*.

[259] Wang, M., Zhao, H., Zhou, X., Ren, X., & Bi, X. (2021). Variance and distribution models for steering tasks. *ACM UIST '21*.

[260] Wang, P., Sheriff, J., Zhang, P., Deng, Y., & Bluestein, D. (2023). A multiscale model for shear-mediated platelet adhesion dynamics: Correlating in silico with in vitro results. *Annals of Biomedical Engineering*.

[261] Wang, P., Luan, C.-Y., Qiao, M., Um, M., Zhang, J., Wang, Y., Yuan, X., Gu, M., Zhang, J., & Kim, K. (2021a). Single ion qubit with estimated coherence time exceeding one hour. *Nature communications*, *12*.

[262] Wang, P., Luan, C.-Y., Qiao, M., Um, M., Zhang, J., Wang, Y., Yuan, X., Gu, M., Zhang, J., & Kim, K. (2021b). Single ion qubit with estimated coherence time exceeding one hour. *Nature Communications*, *12*.

[263] Wang, X., Gao, L., Mao, S., & Pandey, S. (2017). CSI-Based fingerprinting for indoor localization: A deep learning approach. *IEEE Transactions on Vehicular Technology*.

[264] Wang, Y., Zheng, K., Atkins, Z., Austermann, J., Bhandarkar, T., et al. (2022). Simons observatory focal-plane module: In-lab testing and characterization program. *Journal of Low Temperature Physics*, *209*.

[265] Wang, Z., Li, J., Hu, Z., Gage, B., et al. (2023). QuMoS: A framework for preserving security of quantum machine learning model. *IEEE QCE '23*.

[266] Wang, Z., Liang, Z., Zhou, S., Ding, C., et al. (2021). Exploration of quantum neural architecture by mixing quantum neuron designs. *IEEE/ACM ICCAD '21*.

[267] Waxman, B. (1988). Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, *6*.

[268] Wen, F., Qin, M., Gratz, P., et al. (2022). Software hint-driven data management for hybrid memory in mobile systems. *ACM Trans. Embed. Comput. Syst.*

[269] Wen, F., Qin, M., Gratz, P., & Reddy, N. (2021a). An fpga-based hybrid memory emulation system. *FPL '21*.

[270] Wen, F., Qin, M., Gratz, P., & Reddy, N. (2021b). Openmem: Hardware/software cooperative management for mobile memory system. *ACM/IEEE DAC '21*.

[271] Wikipedia. (2023). Field strength meter [https://en.wikipedia.org/wiki/Field_strength_meter].

[272] Wootters, W. K., & Zurek, W. H. (1982). A single quantum cannot be cloned. *Nature*, *299*.

[273] Wright, D., & Clegg, A. (2016). The 3.5 GHz citizens broadband radio service (CBRS).

[274] Wu, J., Hu, T., & Li, Q. (2023). More: Measurement and correlation based variational quantum circuit for multi-classification. *IEEE QCE '23*.

[275] Wu, J., Ye, X., & Man, Y. (2023). Bottrinet: A unified and efficient embedding for social bots detection via metric learning. *11th International Symposium on Digital Forensics and Security (ISDFS)*.

[276] Wu, J., Ye, X., & Mou, C. (2023). Botshape: A novel social bots detection approach via behavioral patterns. *12th International Conference on Data Mining & Knowledge Management Process*.

[277] Wu, K., & Chi, K. (2023). Enhanced e-commerce customer engagement: A comprehensive three-tiered recommendation system. *Journal of Knowledge Learning and Science Technology*.

[278] Wu, X., Kolar, A., Chung, J., Jin, D., Zhong, T., Kettimuthu, R., & Suchara, M. (2021). SeQUeNCe: A customizable discrete-event simulator of quantum networks. *Quantum Science and Technology, 6*.

[279] Wu, Y., & He, K. (2018). Group normalization.

[280] Xia, Y., Li, W., Clark, W., Hart, D., Zhuang, Q., & Zhang, Z. (2020). Demonstration of a reconfigurable entangled radio-frequency photonic sensor network. *Phys. Rev. Lett., 124*.

[281] Xiao, Y., & Varvello, M. (2022). FIAT: Frictionless authentication of IoT traffic. *ACM CoNEXT '22*.

[282] Xie, Z., Hua, Y., & Ye, F. (2022). A measurement study of fmcw radar configurations for non-contact vital signs monitoring. *IEEE RadarConf22*.

[283] Xie, Z., Nederlander, A., Park, I., & Ye, F. (2023). Short: Rf-q: Unsupervised signal quality assessment for robust rf-based respiration monitoring. *ACM/IEEE CHASE '23*.

[284] Xie, Z., Wang, H., Han, S., Schoenfeld, E., & Ye, F. (2022). Deepvs: A deep learning approach for rf-based vital signs sensing. *ACM BCB '22*.

[285] Xie, Z., & Ye, F. (2022). Self-calibrating indoor trajectory tracking system using distributed monostatic radars for large scale deployment. *ACM BuildSys '22*.

[286] Xie, Z., Zhou, B., Cheng, X., Schoenfeld, E., & Ye, F. (2021). Vitalhub: Robust, non-touch multi-user vital signs monitoring using depth camera-aided uwb. *IEEE ICHI '21*.

[287] Xie, Z., Zhou, B., Cheng, X., Schoenfeld, E., & Ye, F. (2022). Passive and context-aware in-home vital signs monitoring using co-located uwb-depth sensor fusion. *ACM Trans. Comput. Healthcare*.

[288] Xie, Z., Zhou, B., & Ye, F. (2021). Signal quality detection towards practical non-touch vital sign monitoring. *ACM BCB '21*.

[289] Xiong, J., & Jamieson, K. (2013). ArrayTrack: A Fine-Grained indoor location system. *USENIX NSDI '23*.

[290] Xu, R., Min, Y., & Wang, T. (2023). Noise-adaptive thompson sampling for linear contextual bandits. *NeurIPS '23*.

[291] Xu, Y., Chen, Y., Zhang, X., Lin, X., Hu, P., Ma, Y., Lu, S., Du, W., Mao, Z., Zhai, E., & Cai, D. (2023). CloudEval-YAML: A practical benchmark for cloud configuration generation.

[292] Yang, K., Chen, Y., Chen, X., & Du, W. (2023). Link quality modeling for lora networks in orchards. *ACM/IEEE IPSN '23*.

[293] Yang, K., Chen, Y., & Du, W. (2024). OrchLoc: In-Orchard Localization via a Single LoRa Gateway and Generative Diffusion Model-based Fingerprinting. *ACM MobiSys '24*.

[294] Yen, J., Wang, J., Supittayapornpong, S., Vieira, M. A. M., Govindan, R., et al. (2020). Meeting slos in cross-platform NFV. *ACM CoNEXT '20*.

[295] Yordanov, Y. S., & Barnes, C. H. W. (2019). Implementation of a general single-qubit positive operator-valued measure on a circuit-based quantum computer. *Phys. Rev. A, 100*.

[296] You, C., Dai, W., Min, Y., Liu, F., Clifton, D., Zhou, S. K., Staib, L., & Duncan, J. (2023). Rethinking semi-supervised medical image segmentation: A variance-reduction perspective. *NeurIPS '23*.

[297] You, C., Dai, W., Min, Y., Staib, L., & Duncan, J. S. (2023). Implicit anatomical rendering for medical image segmentation with stochastic experts. *MICCAI '23*.

[298] You, C., Min, Y., Dai, W., Sekhon, J. S., Staib, L., & Duncan, J. S. (2024). Calibrating multi-modal representations: A pursuit of group robustness without annotations.

[299] Yu, T., Haniz, A., Sano, K., et al. (2018). A guide of fingerprint based radio emitter localization using multiple sensors. *IEICE Trans. on Communications*.

[300] Zafari, F., Gkelias, A., & Leung, K. K. (2019). A survey of indoor localization systems and technologies. *IEEE Communications Surveys Tutorials, 21*.

[301] Zafer, M., Ko, B. J., & Ho, I. W.-H. (2010). Transmit power estimation using spatially diverse measurements under wireless fading. *IEEE/ACM TON, 18*.

[302] Zeng, Y., Chandrasekaran, V., Banerjee, S., et al. (2019). A framework for analyzing spectrum characteristics in large spatio-temporal scales. *ACM MobiCom*.

[303] Zeng, Y., Zhang, J., Liu, J., Liu, Z., & Yang, Y. (2023). Entanglement routing design over quantum networks. *IEEE/ACM TON, 32*.

[304] Zhan, C., Ghaderibaneh, M., Sahu, P., & Gupta, H. (2021). DeepMTL: Deep learning based multiple transmitter localization. *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '21)*.

[305] Zhan, C., Ghaderibaneh, M., Sahu, P., & Gupta, H. (2022). DeepMTL Pro: Deep learning based multiple transmitter localization and power estimation. *Elsevier Pervasive and Mobile Computing, 82*.

[306] Zhan, C., & Gupta, H. (2023). Quantum sensor network algorithms for transmitter localization. *IEEE International Conference on Quantum Computing and Engineering (QCE '23).*

[307] Zhan, C., Gupta, H., Bhattacharya, A., & Ghaderibaneh, M. (2020). Efficient localization of multiple intruders for shared spectrum system. *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN).*

[308] Zhan, C., Gupta, H., & Hillery, M. (2024). Optimizing initial state of detector sensors in quantum sensor networks. *ACM Transactions on Quantum Computing.*

[309] Zhan, C., & Li, C. (2016). Shape formation in games: A probability-based evolutionary approach. *12th International Conference on Computational Intelligence and Security (CIS).*

[310] Zhang, J., Wang, A., Ma, X., Carver, B., Newman, N. J., Anwar, A., Rupprecht, L., Tarasov, V., Skourtis, D., Yan, F., & Cheng, Y. (2023). Infinistore: Elastic serverless cloud storage. *Proc. VLDB Endow.*

[311] Zhang, L., Luo, Y., & Nirjon, S. (2022). Demo abstract: Capuchin: A neural network model generator for 16-bit microcontrollers. *ACM/IEEE IPSN '22.*

[312] Zhang, O., Qian, Z., Mao, Y., Srinivasan, K., & Shroff, N. B. (2019). ERSCC: Enable efficient and reliable screen-camera communication. *ACM MobiHoc '19.*

[313] Zhang, R., Liu, J., Du, X., et al. (2018). AOA-based three-dimensional multitarget localization in industrial WSNs for LOS conditions. *Sensors, 18.*

[314] Zhang, X., Cheng, S., & Mueller, K. (2023). Graphical enhancements for effective exemplar identification in contextual data visualizations. *IEEE Transactions on Visualization and Computer Graphics, 29.*

[315] Zhang, Z., & Zhuang, Q. (2021). Distributed quantum sensing. *Quantum Science and Technology, 6.*

[316] Zhang, Z., Zhang, P., Wang, P., Sheriff, J., Bluestein, D., & Deng, Y. (2021). Rapid analysis of streaming platelet images by semi-unsupervised learning. *Computerized Medical Imaging and Graphics, 89.*

[317] Zhao, H., Gu, S., Yu, C., & Bi, X. (2022). Bayesian hierarchical pointing models. *ACM UIST '22.*

[318] Zhao, S., Dong, Z., Cao, Z., & Douady, R. (2024). Using deep learning technique to enhance the portfolio construction based on PolyModel theory. *12th Bachelier World Congress of the Bachelier Finance Society.*

[319] Zhao, S., Wang, D., & Douady, R. (2023). PolyModel for hedge funds' portfolio construction using machine learning. *Research in Options.*

[320] Zhong, M., Hedges, M. P., Ahlefeldt, R. L., Bartholomew, J. G., Beavan, S. E., Wittig, S. M., Longdell, J. J., & Sellars, M. J. (2015). Optically addressable nuclear spins in a solid with a six-hour coherence time. *Nature, 517.*

[321] Zhou, B., Xie, Z., & Ye, F. (2019). Multi-modal face authentication using deep visual and acoustic features. *IEEE ICC '19*.

[322] Zhou, B., Xie, Z., Zhang, Y., Lohokare, J., Gao, R., & Ye, F. (2022). Robust human face authentication leveraging acoustic sensing on smartphones. *IEEE Transactions on Mobile Computing, 21*.

[323] Zhou, R., Lai, X., Gan, Y., Obraczka, K., et al. (2023). A simulator of atom-atom entanglement with atomic ensembles and quantum optics. *IEEE QCE '23*.

[324] Zhou, T., & Lin, M. (2022). Deadline-aware deep-recurrent-q-network governor for smart energy saving. *IEEE Trans. on Network Science and Engineering*.

[325] Zhou, T., & Lin, M. (2023). Cpu frequency scheduling of real-time applications on embedded devices with temporal encoding-based deep reinforcement learning. *Journal of Systems Architecture, 142*.

[326] Zhou, Y., Zeng, Z., Chen, A., Zhou, X., Ni, H., Zhang, S., Li, P., Liu, L., Zheng, M., & Chen, X. (2024). Evaluating modern approaches in 3d scene reconstruction: NeRF vs gaussian-based methods.

[327] Zhu, Y. (2023). *Machine learning and high-performance computing for accelerating multiscale modeling of thrombi* (Doctoral dissertation). Stony Brook University.

[328] Zhuang, J., & Hasan, M. A. (2023). Robust node representation learning via graph variational diffusion networks.

[329] Zhuang, J., & Kennington, C. (2024). Understanding survey paper taxonomy about large language models via graph representation learning.

[330] Zhuang, Q., & Zhang, Z. (2019). Physical-layer supervised learning assisted by an entangled sensor network. *Physical Review X, 9*.

[331] Zubow, A., Bayhan, S., et al. (2020). Deeptxfinder: Multiple transmitter localization by deep learning in crowdsourced spectrum sensing. *ICCCN '20*.