RELIABLE SENSING AND SMART CONTROL TECHNIQUES IN AGRICULTURAL IRRIGATION

A Dissertation

by

YANXIANG YANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jiang Hu
Committee Members,	Dana Porter
	Peng Li
	Jean-Francois Chamberland-Tremblay
Head of Department,	Miroslav M. Begovic

August 2019

Major Subject: Computer Engineering

ABSTRACT

The freshwater resource is quite limited and is being stressed due to growing human population and climate fluctuations. Agricultural irrigation is a major consumer of fresh water and therefore plays a critical role in potential water savings. The limited water resource requires the irrigation water use efficiency to be much higher than ever before. Moreover, irrigation plays a pivotal role in producing good quality and yield of crops of which importance is extremely vital to the public's subsistence. However, conventional irrigation systems and scheduling methods are oversimplified often resulting in not only a huge amount of water loss but also a reduction in crop productivity.

By deploying soil moisture sensors in the field, the irrigation water application amount can be based on the real-time soil water content and therefore unnecessary irrigation can be avoided. By this way, the water use efficiency (WUE) of irrigation can be significantly improved. However, the success of such a mechanism heavily relies on the assumption that the sensors can reliably convey representative soil moisture information with acceptable accuracy. A reliability-driven soil moisture sensing methodology is developed and discussed in this dissertation. It includes a genetic algorithm based optimization technique and a fault detection technique. The results indicate that the proposed methodology considerably improves system reliability in terms of mean time to failure (MTTF).

To further improve the WUE and automate the irrigation management, a deep reinforcement learning based irrigation optimization approach and an automated scheduling method for fixed-zone irrigation systems were developed to automate the irrigation process and achieve precise water application. The deep reinforcement learning irrigation optimization approach automatically determines the optimal or near optimal water application for an individual irrigation zone, while the automated scheduling method arranges irrigation tasks of a large number of irrigation zones in a multi-zonal system. The scheduling method prevents water hammer and assures that the system operates under a set of pre-defined hydraulic constraints. It can save time and reduce errors as compared to solutions based on manual computation and control. Simulation results show that

the proposed irrigation optimization approach and the scheduling method can optimize irrigation control for each zone, increase water use efficiency and achieve fully automated scheduling for multi-zonal management scenarios.

A real-time data acquisition system and a web-based micro-irrigation controller applying these ideas are built. The data acquisition system can collect real-time environmental data. Based on these data, one can do manual control through our controller to turn on and turn off the specific irrigation management zone. The controller can also work in a hybrid mode doing automated scheduling given the water application amounts of zones and in a fully automated mode using the deep reinforcement learning to get the optimized water application amounts.

DEDICATION

To my parents, who raised me up with love.

To my wife, whom I love so much.

ACKNOWLEDGMENTS

First, I would like to thank Professor Jiang Hu for his guidance and help in my graduate study and research. I am greatly indebted to Professor Hu for all that I learned from him both in research and life. I would also like to thank Professor Dana Porter. She is a really kind person who is always willing to provide help. I am very grateful to her tremendous help and encouragement along the way. Another person who has a large impact on this work is Mr. Thomas Marek. By working with him, I learned a lot of precious practical skills. I would also like to give gratitude to my research partners Lijia Sun and Hongxin Kong for their friendship and many enlightening discussions and collaborations. Also, many thanks to Dr. Kevin Heflin for his help during the field trip. I would also like to express my thanks to committee members, Dr. Li and Dr. Chamberland for their suggestions and time. Finally, I would like to express my immense gratitude to my wife, Yangyang. Her love and support helped me go through all the difficulties during my graduate study.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Jiang Hu [advisor], Professor Dana Porter from the Department of Biological & Agricultural Engineering, Professor Jean-Francois Chamberland and Professor Peng Li at Department of Electrical & Computer Engineering.

Funding Sources

Graduate study was supported by water seed grant from Texas A&M University.

NOMENCLATURE

AR	Autoregressive
CER	Combined Experience Replay
DA	Detection Accuracy
DQN	Deep Q Networks
DRL	Deep Reinforcement Learning
E	Evaporation
ER	Experience Replay
ET	Evapotranspiration
FAR	False Alarm Rate
GIO	General Input and Output
HPC	Heterogeneous Point Coverage
IoT	Internet of Things
MAD	Management Allowed Depletion
MDPs	Markov Decision Processes
MTTF	System Mean Time to Failure
NNs	Neural Networks
PAW	Plant Available Water
P-MP	Point-to-Multipoint
RL	Reinforcement Learning
RSSI	Received Signal Strength Indicator
RTC	Real-time Clocks
SDI	Subsurface Drip Irrigation

SIDSS	Smart Irrigation Decision Support System
SS-VRI	Site-Specific Variable Rate Irrigation
T	Transpiration
WSNs	Wireless Sensor Networks
WUE	Water Use Efficiency

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Motivation

Although water covers most of our planet, 97% of water is saline and a little less than 3% is hard to access. Therefore, only 0.014% of the water on Earth is easily accessible and usable for human beings. The limited fresh water resource forces changes to irrigation strategies. In order to maximize the profit, farmers have to enhance the water use efficiency in irrigated agriculture. The goal is to use as less water as possible while maintaining yields.

Water use efficiency of agricultural irrigation can be greatly improved by incorporating information input using modern technologies, including wireless sensors. By using moisture sensors in the soil profile, irrigation event control can be based on the actual soil moisture status and thereby reduce unnecessary irrigation [5, 6]. The success of such a mechanism largely hinges on the assumption that the sensors can reliably convey representative soil moisture information with acceptable accuracy. This is particularly true for agricultural irrigation, where sensors stay in harsh outdoor environments with farmers either reluctant to or cannot afford additional production time on maintenance efforts.

There are numerous previous studies on sensor deployment [7, 8, 9, 10, 11], however the problem faced with a representative irrigation sensing system is far from being well addressed. Most of sensor based irrigation methods are demonstrations of the advantages of using sensors with actual system construction. Other works on sensing systems can be generally categorized into two ways. One is to minimize sensor cost while achieving a certain sensing coverage. The other is on how to get the most information from sensing area under a certain constraint. However, neither case fits the characteristics of sensing in classical agricultural irrigation. The soil moisture sensor deployment is based on terrain and soil type of crop field where terrain and soil type of a crop field are heterogeneous in general; with the sensor granularity being quite coarse. For example, a nominal quarter mile center pivot irrigation machine covers 120-125 acres (49-51 hectares) of field, where a

small number of sensors are sufficient to cover the relatively homogeneous soil types and minimal terrain variations. As such, the grower decision regarding coverage and sensor cost is not very difficult. Moreover, if the irrigation controller is installed at the base of center pivot machine, the small number of sensors can communicate directly with the controller through wireless technology (e.g. Zigbee) typically without a signal relay. Thus, the communication fault tolerance by the complicated sensor network topology does not help in this scenario.

Traditionally, fixed (or manual) irrigation scheduling is widely used by farmers. This strategy is irrigating a fixed amount of water per a given time interval. This simplified irrigation strategy often results in water loss (and hence reduction in efficiency) and a reduction in crop productivity. In recent years, more precise irrigation strategies based on sensor data have been extensively studied. However, most of these strategies apply thresholds or simplistic models for decision making, thus involving a lot of inaccurate or non-optimal irrigation events. Typically, for sensor-based irrigation scheduling, an expert is needed to interpret sensor data and convert the data to appropriate threshold values for use with a scheduling model [12, 13, 14, 15, 16, 17]. The process can become too complicated considering the number of zones, rapidly changing weather, soil variability, crop types, and the different water needs at various growth stages. In addition, the amount of sensing data makes scheduling in real time even more challenging due to possibly contradictory data from different types of sensors and other data sources. Another obvious drawback of manual computed thresholds or models is the time-consuming aspect and lack of scalability. To overcome these issues, machine learning techniques were explored to automate the process [18, 19, 20, 21, 22, 23]. Linear regression or neural networks is used to extract useful information from sensing data and calculate the scheduling model. However, it still requires manual oversight to analyze and manually control the irrigation events. A reinforcement learning-based irrigation control can achieve fully automated control [4]. However, traditional reinforcement learning can only cover limited state space and therefore is difficult to accurately capture the entire, realistic irrigation environment.

1.2 Dissertation Outline

The rest of this dissertation is structured as follows. In Chapter 2, we begin with a brief introduction about irrigation systems. We also review some important concepts in irrigation scheduling. Basic knowledge of reinforcement learning and deep reinforcement learning is also presented in this chapter.

In Chapter 3, we design and build a real microirrigation controller. We begin with the wireless sensor network system design and field experiments. We then describe the web-based microirrigation controller, which can remotely monitor environmental status and control the microirrigation in three different modes, namely manual mode, hybrid mode, and fully automated mode.

In Chapter 4, we study a realistic soil moisture sensing problem where reliability is the main objective. We concentrate on two aspects, faulty sensor detection and sensor placement. The basic idea is to efficiently use redundancy to make up for unexpected sensor failures and sustain a healthy system operation. A fault detection technique that exploits both spatial and temporal correlation is proposed. The reliability issue is further addressed with a genetic algorithm-based sensor placement approach where the system MTTF (Mean Time To Failure) is maximized for a given budget on the number of sensors. Simulations are subsequently performed to compare our proposed methodology with a naïve approach and a previous reported work. The results show that our method can extend the system MTTF by more than two fold and thereby effectively improve reliability of system performance. Previous related works are discussed in Chapter 4.1. The proposed faulty sensor detection and sensor placement techniques are elaborated in Chapter 4.2. Chapter 4.3 is to describe simulation results and finally conclusions are provided in Chapter 4.4.

In Chapter 5, a smart irrigation approach is proposed. It is composed of two parts, a deep reinforcement learning-based method to determine irrigation water application of an individual zone and an automated scheduling approach to determine start/stop times of multi-zonal microirrigation or other fixed-zone irrigation systems. By using this approach, fully automated control and high WUE can be achieved. Moreover, this approach is readily scalable and can handle high-dimensional sensory inputs. A series of simulations and experiments were conducted to assess

the effort. The deep reinforcement learning (DRL) method is compared with traditional reinforcement learning (RL), threshold-based irrigation, and fixed interval irrigation scheduling. The proposed deep reinforcement learning-based method could determine the optimal or near optimal water application for selected irrigation management zones. However, the scheduling of fixed-zone irrigation systems, such as with a subsurface drip irrigation (SDI) , landscape irrigation, or greenhouse/nursery systems, can be still very challenging considering the hydraulic constraints (flow, pressure, system capacity, etc.). An abrupt change in flow (and subsequent water hammer) may cause damage to the irrigation system. To protect the irrigation system from large fluctuations in flow and pressure, an automated scheduling method for multi-zonal micro-irrigation is proposed. It replaces the manual control computations generally required to determine start/stop times and makes the overall scheduling process more applicable and efficient than in the manual mode.

2. BACKGROUND

2.1 Agricultural Irrigation

2.1.1 Different Irrigation Systems

Most commonly used type of irrigation system keeps changing over the years. Today, the most popular one is the sprinkler system. It waters the majority of irrigated areas in the United States. However, more recently, the use of microirrigation has increased due to its higher water use efficiency.

2.1.1.1 Center Pivot Irrigation

Figure 2.1: Center pivot irrigation system at Bushland, TX.

Center pivot irrigation is a type of irrigation that sprinklers are attached to a long arm that rotates around a pivot as shown in Figure 2.1 and Figure 2.2. It is able to irrigate a circular area centered on the pivot. Center pivot irrigation can be used to achieve higher irrigation application efficiency due to relatively precise irrigation applications with lower deep percolation and runoff losses (with careful management). It can also be used with conservation practices, such as reduced

Figure 2.2: Center pivot irrigation system with water on.

tillage, field contouring, etc., to reduce erosion losses.

2.1.1.2 Microirrigation

Microirrigation is a kind of irrigation that applies a small amount of water frequently on or below the soil surface as drops, tiny streams, or miniature spray. Typically, it can be further classified into several categories such as drip, subsurface drip, bubbler, and mist or spray irrigations. Water is applied through a pipe distribution network under low pressure. Microirrigation is potentially the most water-efficient irrigation application method, as it directly applies water within the crop root zone and hence minimizes evaporation and runoff losses. Moreover, it can mitigate soil erosion, reduce weed growth and incurs relatively low energy consumption.

2.1.2 Irrigation Scheduling

The goal of irrigation scheduling is to determine how much water to apply and when and where to apply it. To achieve precise irrigation, these determinations should be based on observations of crop status, weather conditions, soil water content levels, etc.

The most commonly used irrigation scheduling is based on crop water use, soil water evaporation, precipitation and stored soil water. An important concept is evapotranspiration (ET). Evapotranspiration is used to describe water loss caused by soil evaporation (E) and plant transpiration

(T). Transpiration is the process that water enters the roots of plants from the soil, passes through the stems and leaves of plants, and then returns to the atmosphere. ET may differ in terms of crop species, the stage of crop growth, relative maturity, weather conditions, soil profile, and tillage systems. Conventionally, farmers determine when to apply irrigation according to ET value, which affects the amount of plant available water in the soil.

There are some important terms related to soil water content as shown in Figure 2.3.

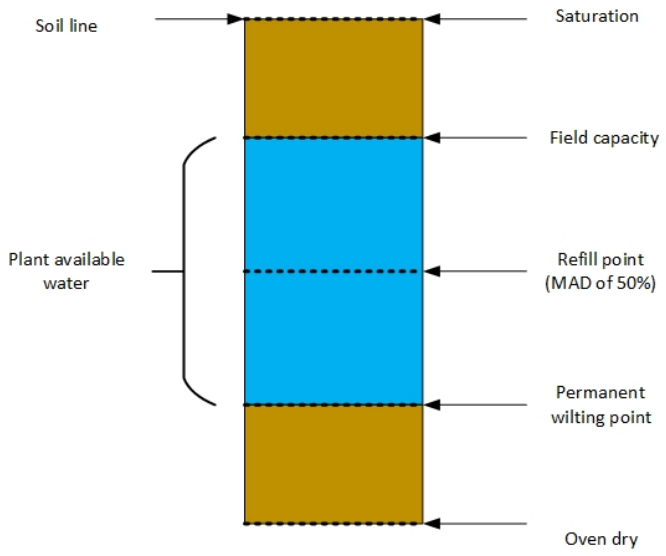

Figure 2.3: Soil water content.

- Plant available water (PAW), is the amount of water that crops or plants can get access to in soil.
- Field capacity, amount of water remaining in a soil when the downward water flow due to gravity becomes negligible.
- Permanent wilting point, is the soil water content at which the plant wilts and refers to 0% PAW.

Farmers tend to irrigate before the plant available water drop to a certain level at which plant stress occurs. This level is usually called allowable depletion level, which is also known as the management allowed depletion (MAD). The MAD may vary for different crops and different crop growth stage.

2.2 Deep Reinforcement Learning

2.2.1 Reinforcement Learning

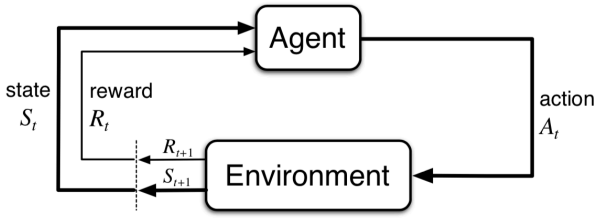

Figure 2.4: The agent-environment interaction in a Markov decision process reprinted from [1].

Reinforcement learning (RL) is a computational approach to learning from interaction with the environment to make good sequential decisions. The goal of reinforcement learning is to generate a mapping from experience to actions to maximize expected reward. The learning agent does not know which actions to take at the beginning and must discover which actions lead to the best reward by trying them. The action may affect the environment in some way and thereby all subsequent rewards. Researchers use Markov decision processes (MDPs) to mathematically formalize the reinforcement learning problem as shown in Figure 2.4. The basic idea is to capture only the essential components or aspects to model the problem that a learning agent tries to achieve its goal by interacting with the environment over time. Some important terms in MDPs and RL are stated as follows:

- State, a mathematical way to describe the current environment status or to distinct different environment status.

- Action, a way in which the learning agent can respond according to current environment status.
- Policy, a recipe that determines actions at each state.
- Reward, the goal of reinforcement learning indicating what is good in an immediate sense.
- Value function, a way to indicate what state or state-action pair is good in the long run.

Reinforcement learning differs from supervised learning which has been extensively studied in the field of machine learning. Supervised learning is learning from a labeled data set. For a concrete example, consider applying a supervised learning algorithm to the problem of predicting the house price, given features of the houses. If we are given a collection of house prices and features, then we are able to use a supervised learning algorithm to fit an approximate function to these data. This fitted function can be used as a model to predict the prices of houses in terms of features. However, in reinforcement learning, there is no labeled data and the reward signal is often delayed. Therefore, reinforcement learning learns from interaction with the environment while supervised learning learns from labeled data. Another significant difference is that reinforcement learning is to make a sequence of decisions and thereby involves multiple time steps. Although reinforcement learning has nothing to do with labeled data, it is different from so-called unsupervised learning. Typically, unsupervised learning learns from unlabeled data. The goal of it is to find the hidden structure of the data.

2.2.2 Deep Q Networks

The basic strategy for reinforcement learning is to perform the action that will eventually yield the highest cumulative reward at any given state. This cumulative reward is often referred to as Q value, and thereby we can formalize this strategy mathematically as follows:

$$Q(s,a) = r(s,a) + \gamma \max_{a} Q\left(s', a\right) \tag{2.1}$$

This equation is known as the Bellman Equation. It states that the Q value yielded from being at state s and selecting action a, is the immediate reward received, $r(s, a)$, plus the highest Q value possible from next state s'. γ is called the discount factor, which determines the importance of longterm rewards versus the immediate one. There is no need to really know what are the true Q values at the beginning. After a certain number of iterations, they will eventually converge to the real values. The strategy explained above is one popular reinforcement learning algorithm called

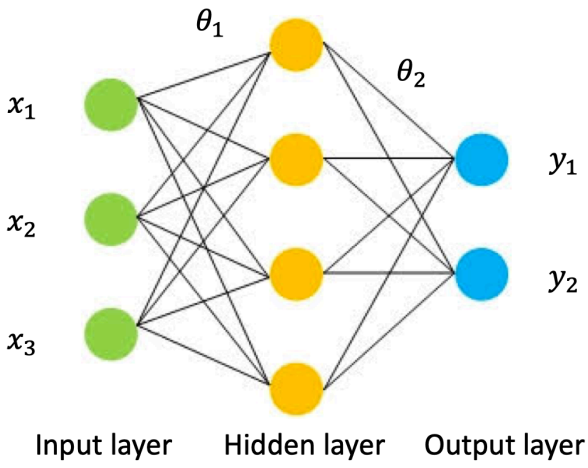

Figure 2.5: A typical neural network.

Q learning. It does a good job for relatively simple control. However, it is not practical to initialize and maintain the Q table if the Q table size is extremely large. To make Q learning fit to more complex problems that have a large number of states and actions, one can combine Q learning and neural networks. A typical neural network is shown in Figure 2.5. Each circle in the network is called a neuron. Each one is a computational unit that takes in inputs from other neurons in previous layer, processes them, and outputs the result to next layer. θ_i indicates a vector of weights connect neurons in layer $i-1$ and layer i. Deep Q networks (DQN) basically replace Q table in Q learning with a neural network. This neural network tries to approximate Q values and is referred to as the approximator and denoted as $Q(s, a; \theta)$, where θ represents the trainable weights of the

network. The detailed algorithm is stated as Algorithm 1.

Initialize replay memory D to capacity N;
Initialize action-value function Q with random weights;
for *episode* = $1, M$ **do**
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t;
    Otherwise select a_t that maximizes the Q value;
    Execute action a_t in emulator and observe reward r_t and s_{t+1};
    Store transition (s_t, a_t, r_t, s_{t+1}) in D;
    Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D;
    Set $y_j = r_j^S + \gamma \max_{a'} Q\left(s_{j+1}, a'; \theta\right)$ for non-terminal s_{j+1};
    Set $y_j = r_j^S$ for terminal s_{j+1};
    Perform a gradient descent step on $\left(y_j - Q\left(s_j, a_j; \theta\right)\right)^2$ according to equation;
  end
end

Algorithm 1: Deep Q learning with Experience Replay [24].

The cost function of the neural network is stated as follows:

$$\text{cost } = \left[Q(s, a; \theta) - \left(r(s, a) + \gamma \max_a Q\left(s', a; \theta\right)\right)\right]^2 \tag{2.2}$$

where $r(s, a) + \gamma \max_a Q(s', a; t))$ is usually referred as Q target. In reinforcement learning, the training set is created as the learning agent interacting with the environment. The agent selects the best actions using a neural network. The states, actions, rewards and the next states are recorded as tuples and stored into memory. For each episode, while the new tuple is recorded, we randomly sample a certain number of records from the memory and train the neural network. This process is referred as Experience Replay (ER). Figure 2.6 shows one feasible way to approximate a Q table. Although it is correct, it is inefficient. This architecture requires multiple neural networks to approximate the Q table. There is a better architecture which is shown in Figure 2.7. Observation in the figure indicates the observation of environment status which is referred as state in reinforcement learning. Output is a vector of Q value corresponding to each action.

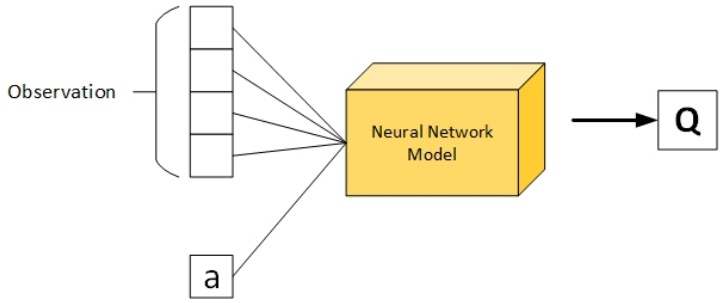

Figure 2.6: An architecture for Q function approximator.

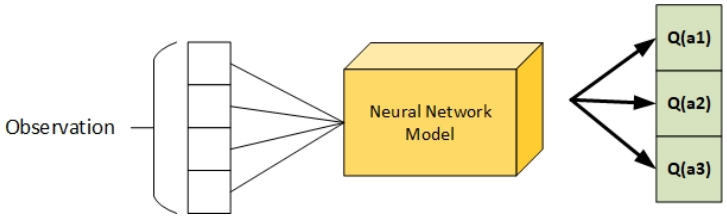

Figure 2.7: A better architecture for Q function approximator.

3. HARDWARE AND SOFTWARE SYSTEMS FOR SMART IRRIGATION

Typically, a smart irrigation system contains three subsystems, data acquisition system, controller, and actuator (irrigation machine). In this chapter, a real operational smart irrigation system was built. It includes a wireless sensor network that collects environmental data and a web-based controller that can be interfaced with microirrigation system or other fixed zone irrigation systems.

3.1 Data Acquisition System

3.1.1 Introduction

Agricultural irrigation is a major user of ground and surface water in the United States. Therefore, if we can improve the irrigation water use efficiency, we can greatly alleviate the water shortage problem. Thanks to the development of information technology, such as wireless sensor networks(WSNs), the real-time environmental data such as the condition of the field, the forecast of the weather, and status of the irrigation machine are now available for the irrigation controller. The controller can smartly manage its water use based on these data, thus, to achieve water savings. A data acquisition system is developed to get real-time environmental data and facilitate irrigation scheduling. Originally, the data acquisition system is built for the center pivot irrigation system which is located in the experiment field at Bushland, Texas. However, it can be also used in other irrigation systems. The high-level system view is shown in Figure 3.1. Thanks to the data acquisition system, irrigation control can be based on the real-time information of soil, weather status and position of the irrigation machine to achieve relatively less water use while maximizing the crop yield.

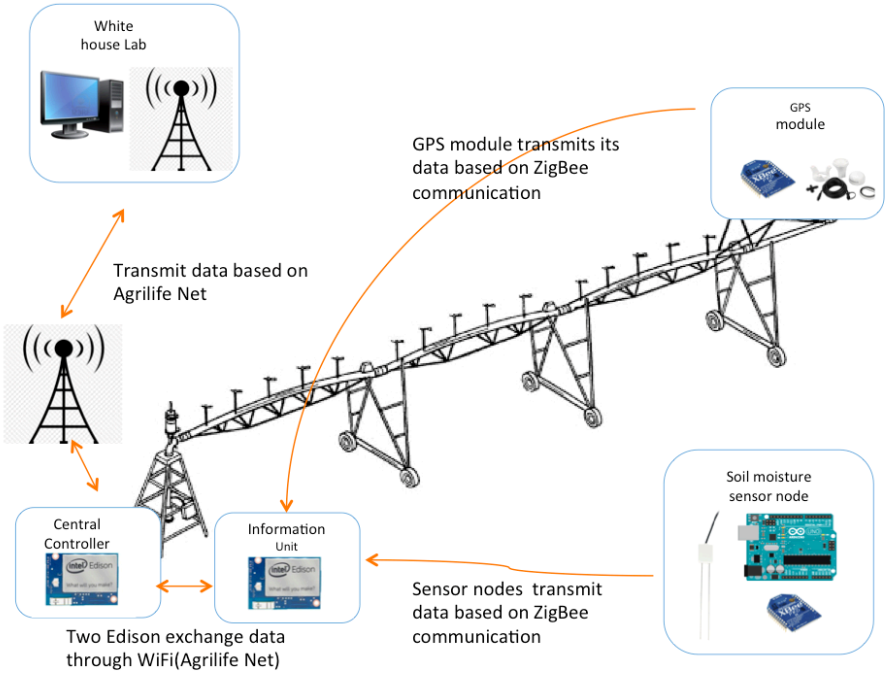

Figure 3.1: Irrigation system overview.

3.1.2 Wireless Sensor Network

Wireless communication is the key making it possible to get soil moisture data, weather data, and GPS data efficiently. Figure 3.2 shows the network topology of the data acquisition system. The networks of our data acquisition system consisting of a soil moisture sensor network, a GPS network, and the Internet are shown in Figure 3.3.

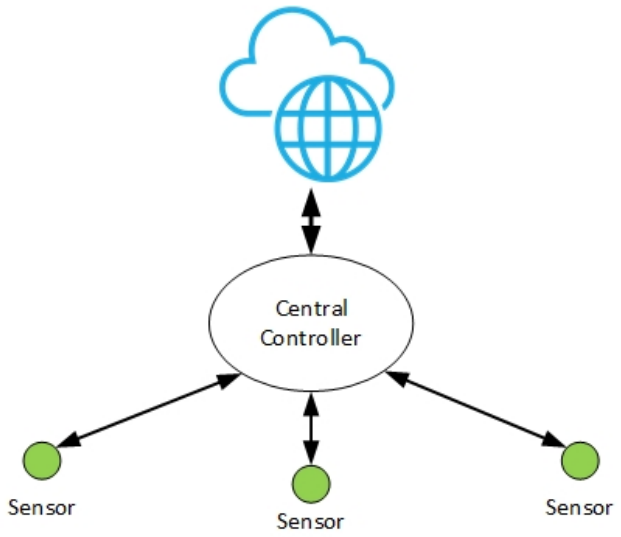

Figure 3.2: High level network topology.

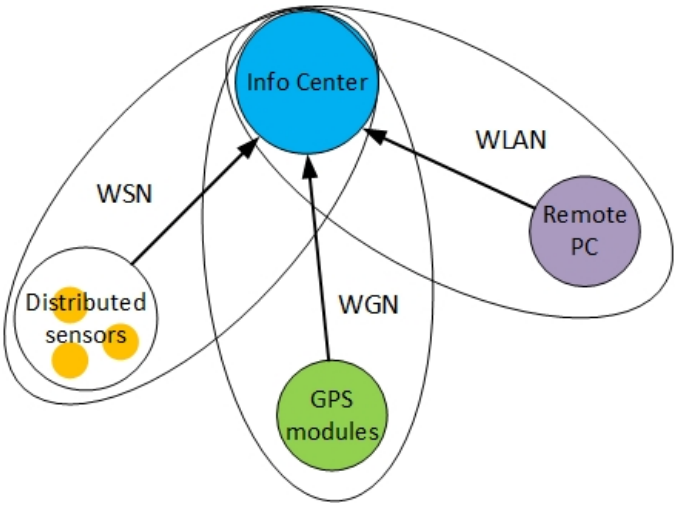

Figure 3.3: Networks in the data acquisition system.

Figure 3.4 shows the soil moisture sensor box we developed. Figure 3.5 and Figure 3.6 show the process of sensor deployment in the field. Soil moisture sensor network and GPS network is based on ZigBee and the internet access is through WiFi. The hardware system includes an Information Unit, Sensor Node, and GPS unit. The Information Unit is the module that handles

the soil moisture sensor data and GPS data and has them ready to be read by the central controller. Also, the controller on the center pivot that is based on the Intel Edison platform can access the Internet. The sensor node and GPS Unit are slaves while Information Unit plays the role of master. From the perspective of the irrigation controller, the Information Unit is a slave waiting for the commands. Based on the simulation and the real tests in the field, our wireless network system is one efficient and robust design to collect data for irrigation scheduling. The network system has the following three properties: a. Scalability: all the communication is based on the wireless network. This can not only provide dynamic mobility and cost-free relocation but also make it easier to scale the whole system. There is no doubt that the smart irrigation systems will be different depending on field sizes and node counts. This feature is going to be more and more important for our modern agricultural systems. b. Efficiency: because the end wireless nodes directly communicate with the Information Unit, our network system, which is based on point-to-multipoint (P-MP), is more efficient than the mesh network with relays. c. Robustness: the special topology of our networks increases the robustness of the whole system. The failure of one communication connection will never have an impact on the other because of the isolation.

Figure 3.4: The soil moisture sensor box we developed.

Figure 3.5: Digging holes in the field to deploy soil moisture sensors.

Figure 3.6: Deploy soil moisture sensors in the field.

For any embedded system, the developers always want to achieve high levels of functionality

and performance while simultaneously maximizing battery life. A common approach to reducing power dissipation is to wake up the system periodically. Modern microcontrollers often offer real-time clocks (RTC) that can run in low-power standby modes enabling the microcontroller to wake up automatically at specified time intervals. The remote sensing application uses this capability to wake up when it needs to read sensor data and sleep for the rest of time. By this method, we finally achieved 8 times of battery life (20 min sleep interval).

3.1.3 Communication Field Experiments

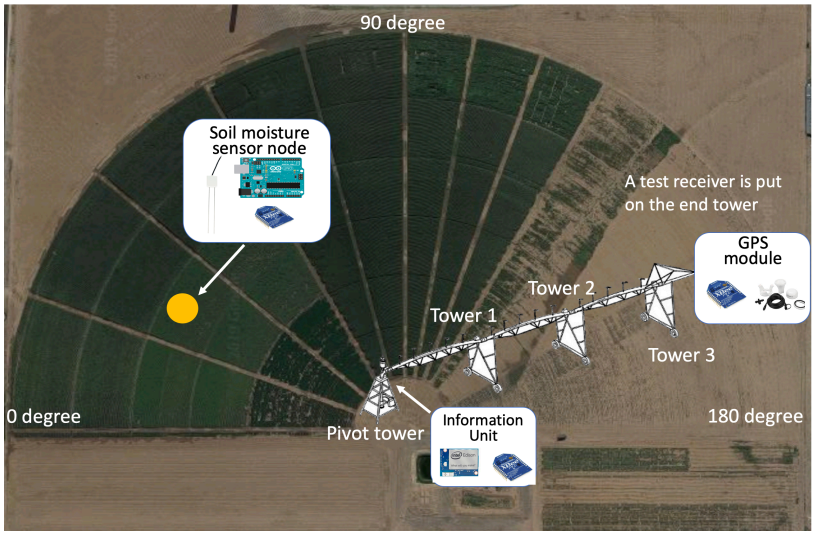

Figure 3.7: Field test of ZigBee communication.

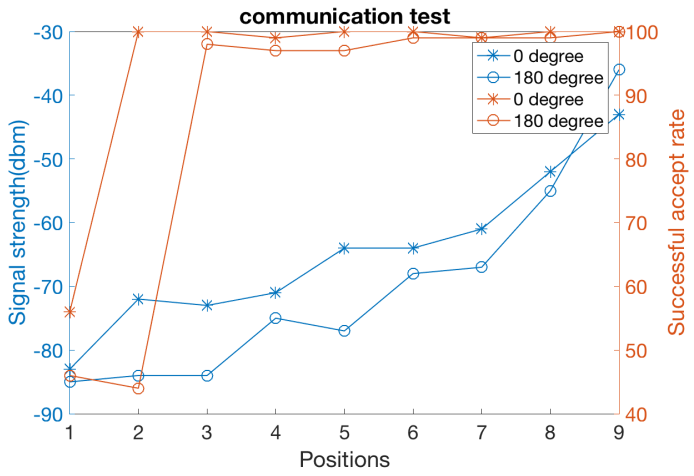

Figure 3.8: Communication test results.

The realistic environment for ZigBee communication in the field is harsh. Many factors may affect the communication quality. To achieve better communication and guarantee we get accurate real-time GPS, we need to identify these factors that may degrade the communication quality and evaluate our communication in different situations and different communication ranges. A series of experiments are conducted under different conditions to assess the ZigBee communication quality between the GPS module and the central control. The data are obtained from the experiments conducted in the field in Amarillo. In theory, there may be no problem for the communication between the ZigBee communication to cover quarter mile communication. However, in . However, in practice, because of the harsh environment, the communication may not be good as expected. We did experiments in two directions: 0 degrees and 180 degrees as shown in Figure 3.7. For the experiment on each direction, we selected 9 different locations to test communication quality. One receiver was put on tower 7 (the outermost tower of the center pivot irrigation system), and one transmitter is set up to communicate with this receiver on 9 different locations, namely under the pivot tower, near pivot tower, tower 1, tower 2, tower 3, tower 4, tower 5, tower 6, and tower 7. Based on the test data, the limited length of antenna extension cable did not affect the communication quality. However, communication quality differs a lot for different communication ranges. Moreover, based on the result which is shown in Figure 3.8, we can see that the communication

quality of 0 degrees is better than the quality of 180 degrees. This is because we put the antenna on the north side of the pivot tower facing 0 degrees and the metal tower causes significant interference to the communication. Based on the experiment data, communication quality at 0 degree is, in general, 8.23% better than 180 degrees. To improve the communication quality, we replaced previous transceivers with more powerful transceivers. We also applied extension cables to mount the antenna high enough to achieve clear line of sight communication. Figure 3.9 and Figure 3.10 show the upgrading process in the field. After this upgrading, the communication quality has been greatly improved. The comparison results are shown in the Figure 3.11 and Figure 3.12.

Figure 3.9: Communication system upgrade A, replacing previous Xbee module with more powerful one with external antenna.

Figure 3.10: Communication system upgrade B, using extension cable to mount antenna high enough to achieve clear line of sight communication.

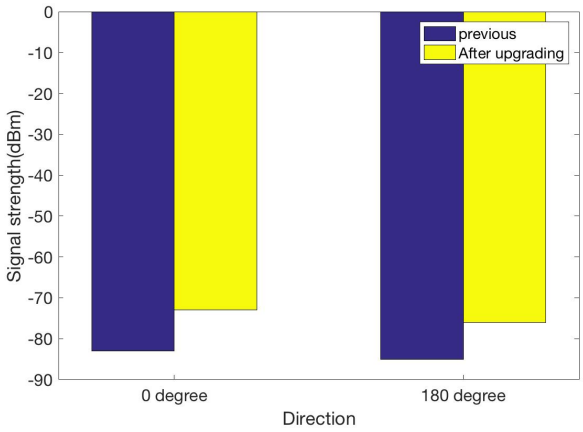

Figure 3.11: Comparison result of signal strength.

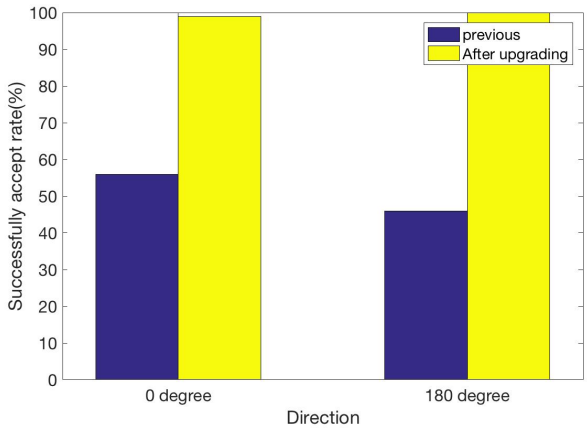

Figure 3.12: Comparison result of successfully accept rate.

3.2 Microirrigation Controller

3.2.1 Introduction

Microirrigation is becoming more and more popular due to its high application efficiency. Figure 3.13 shows a typical microirrigation system. It usually contains several components:

- control head
- mainlines, submains, and manifolds
- emitters
- flushing system

The control head typically contains four major components, pumping station, control and monitoring devices, fertilizer and chemical injectors, and filtration system. The control head distributes the water from the pump to the mainline. It is the part that controls the water application amount. It must balance the water pressure to avoid operational problems and add fertilizer and chemicals to the water. The mainline, submains, and manifolds deliver water that received from the control head to the lateral and emitters. Laterals and emitters further deliver water to the plant.

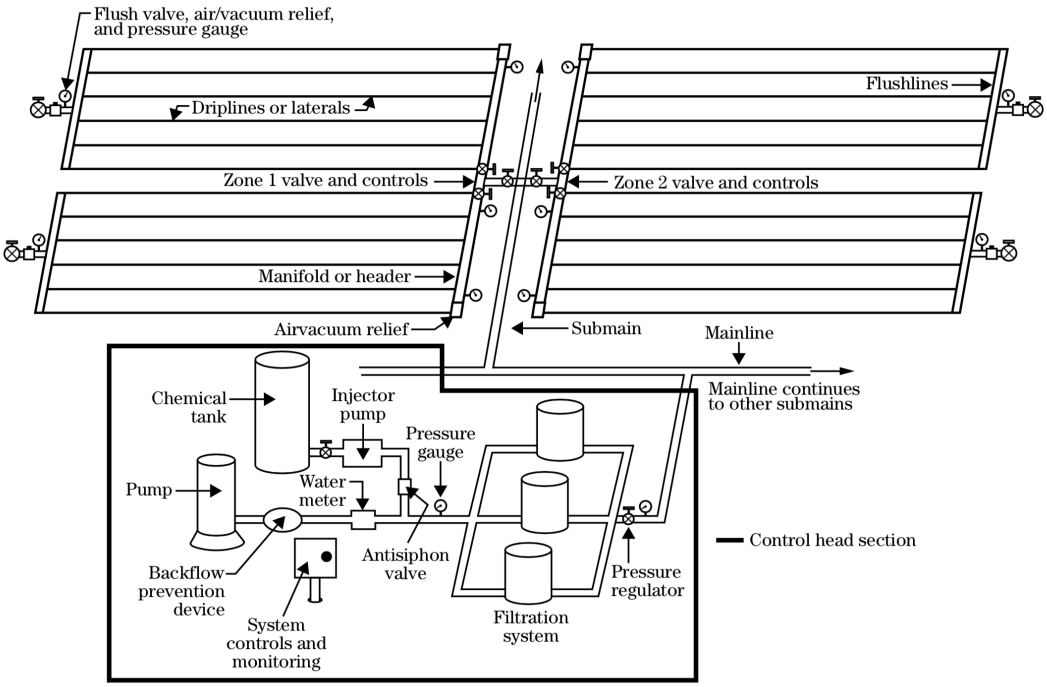

Figure 3.13: A typical microirrigation system layout reprinted from [2].

This chapter presents a real implementation of a microirrigation controller, which is the most important part of the control head determining water application amount and controlling water pressure.

3.2.2 Implementation

3.2.2.1 Requirements

Farmers and other irrigation managers (end-users) are often very busy and usually know little about the circuit board or computer programming. Therefore, one of the most important requirements is to make the control interface as simple as possible. Using techniques presented in previous chapters can achieve fully automated irrigation. However, manual control is still a necessity and plays an important role for trouble-shooting or in simple irrigation scheduling. Thus, several vital features of our controller are stated as follows:

- Web-based user interface and control. Modern responsive web design can significantly im-

prove the user experience and make it very easy for users to interact with our controller. Moreover, the web-based application together with the Internet enables remotely control. People can control through personal computers and mobile devices.

- Multiple control modes. There are three different control modes, namely manual mode, hybrid mode, and automatic mode. In manual mode, the farmer can have full control of irrigation management. The hybrid mode can automatically schedule the irrigation and turn on/off the valves, but it still needs the farmer to tell the controller how much water he wants to apply for each zone. In automatic mode, as the name suggests, the controller automatically schedules the irrigation based on the environmental data and algorithmic optimization results.

- High scalability. Our system is scalable with different fixed zone irrigation systems and different fields, but also can accommodate simple to complex systems with few to many zones.

3.2.2.2 Hardware Structure

The hardware structure is relatively simple. The major component is a programmable platform from Intel, called Up Squared Board. MCP23017 chip is used to extend the number of general IO ports (GIO). LED simulation panel is to show the running status of the irrigation machine and it can be also used to perform functionality test. The system structure is depicted in Figure 3.14.

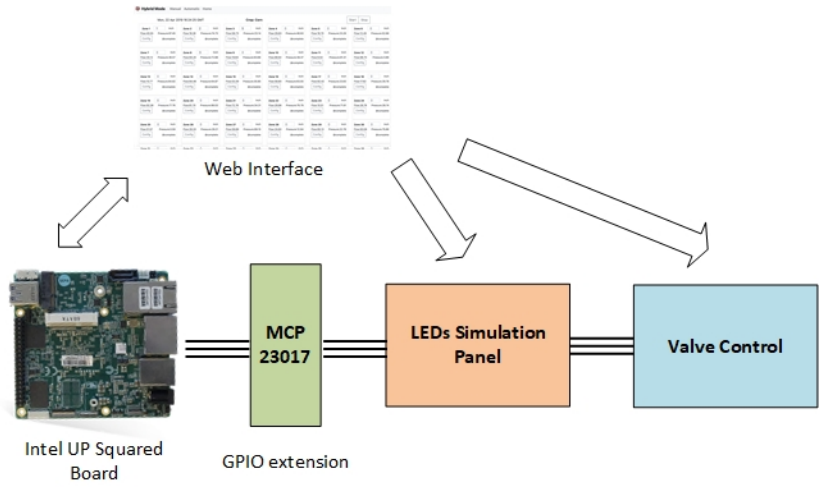

Figure 3.14: The system structure of microirrigation controller.

3.2.2.3 Software

Authentication is necessary to protect the irrigation system from malicious attacks. For our controller, there is a database to store usernames and hashed passwords. Only authorized users can get access to our controller. Moreover, the controller allows only one user to access it at any time. After login, one can see different control options, manual mode, hybrid mode, and automatic mode.

In manual mode, there are buttons to start and stop irrigation for the specific zones. On the bottom right, there is display for showing irrigation status information. The controller also collects water flow and pressure data. Figure 3.15 shows the interface of the manual mode. After clicking the start button of zone 1, the controller turns on the valves of zone 1 and start irrigation, as shown in Figure 3.16.

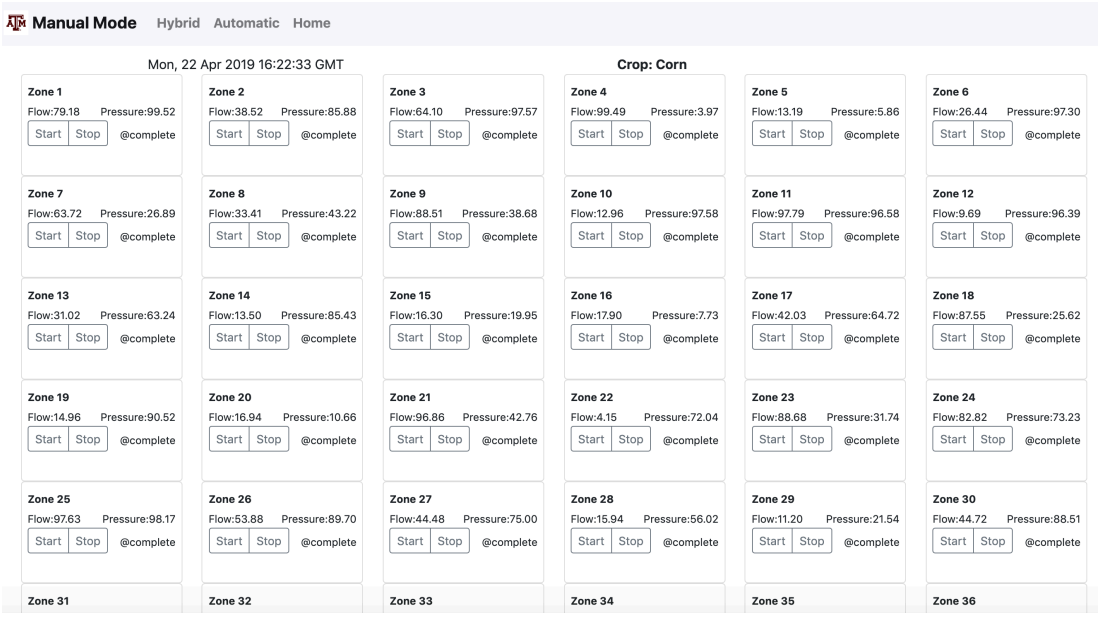

Figure 3.15: Manual mode control interface.

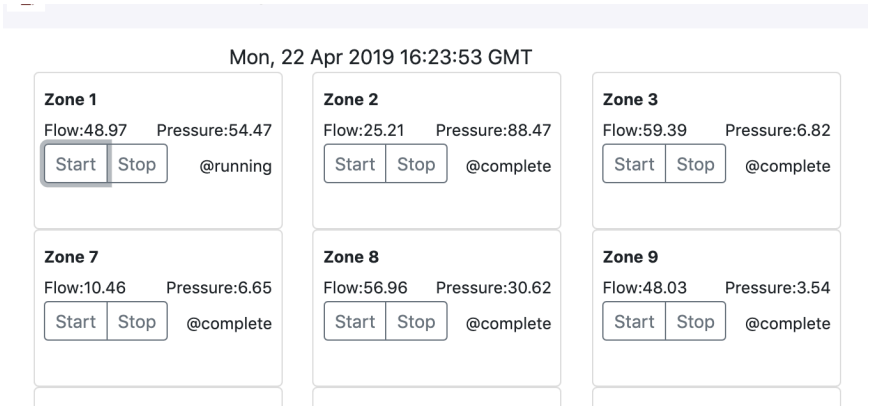

Figure 3.16: Irrigating zone 1 in manual mode.

In hybrid mode, there is just one start and stop button for all zones. The user needs to tell the controller how much he/she wants to irrigate for each zone. We want the controller automatically schedule the irrigation tasks, once the start button is clicked. The automated scheduling method presented in chapter 3 is used to achieve this goal. It can guarantee the controller runs safely.

There is also a progress bar to show the irrigation status and tell the user how much irrigation time left for each zone. Figure 3.17 and Figure 3.18 show the controller running in hybrid mode. Figure 3.19 shows the control flow of hybrid mode. In the beginning, the controller is in standby mode and waiting for the inputs. After receiving inputs and start command, the controller will check if the irrigation system is idle. If it is currently running, the controller will do nothing but wait. Otherwise, the controller will automatically schedule the irrigation based on the input data.

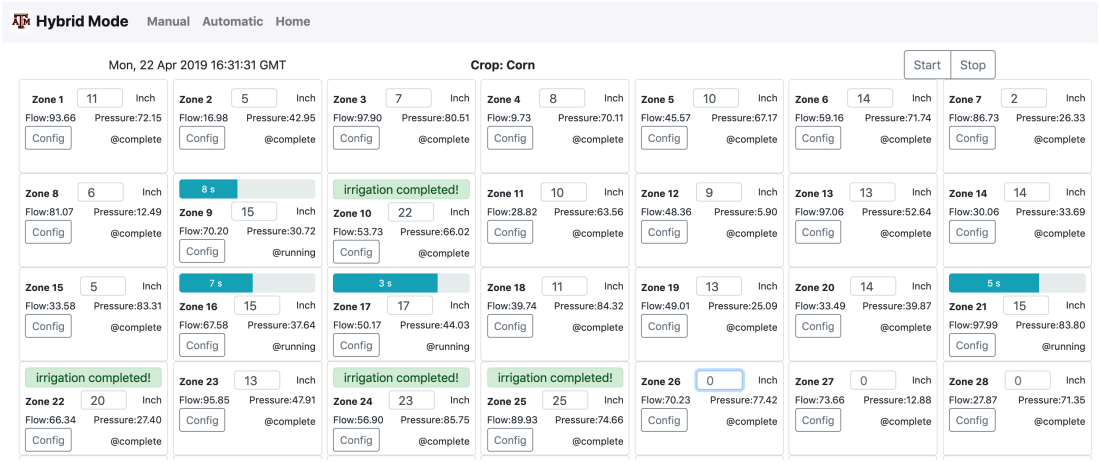

Figure 3.17: Running in hybrid mode i.

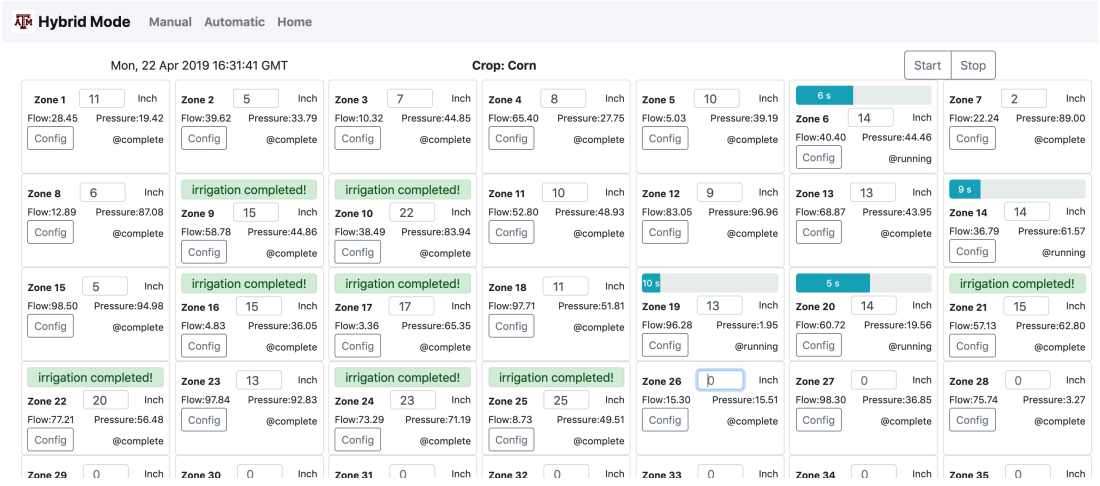

Figure 3.18: Running in hybrid mode ii.

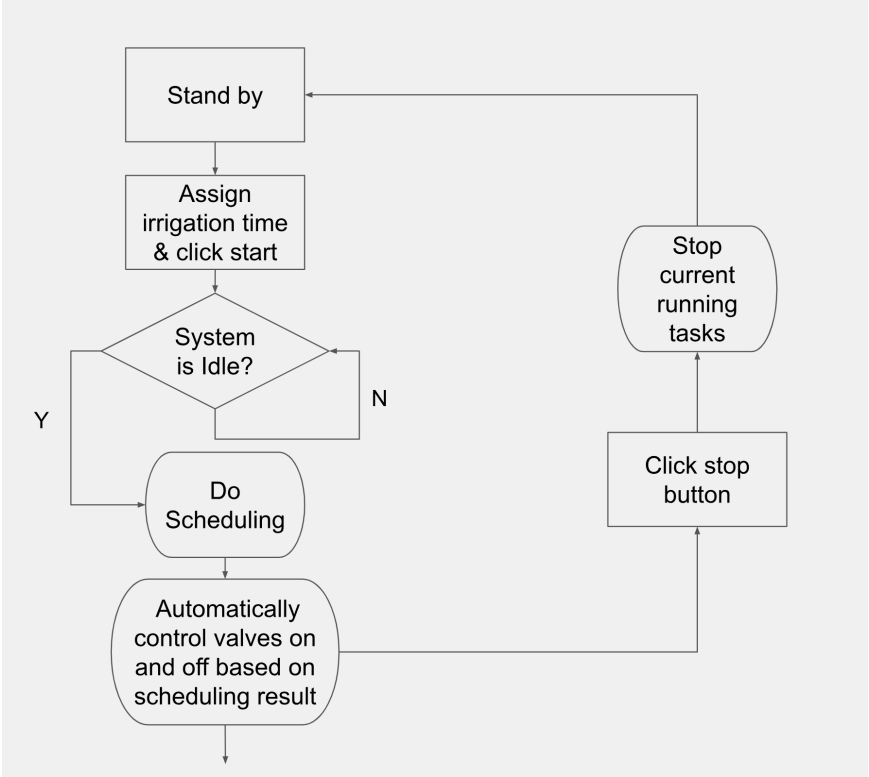

Figure 3.19: Control flow of hybrid mode.

In addition to basic control in hybrid mode, there are some other features. By clicking the configuration button, we can enter a configuration page as shown in Figure 3.20. We can input the zone size, soil type, and crop types to the controller to achieve more precise scheduling. Meanwhile, the real-time weather status and water flow information will be shown on this page.

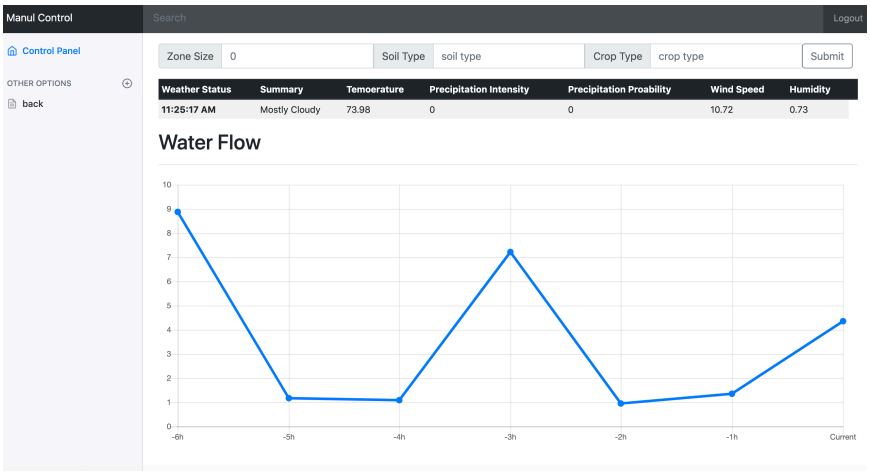

Figure 3.20: Configuration page of hybrid mode.

In automatic mode, the irrigation system runs automatically. There is no other action needed to take except clicking the start button. Figure 3.21 shows the interface of automatic mode. There is real-time weather status on the bottom. The pie chart is to show the status of the entire field (green = adequate soil moisture; yellow = needs irrigation soon; red = needs irrigation now).

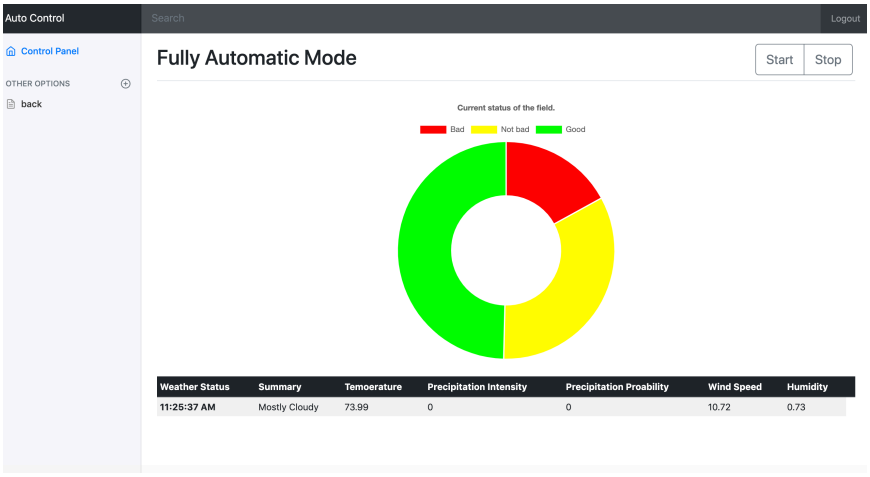

Figure 3.21: Running in automatic mode.

3.3 Conclusion

The hardware and software system is an essential infrastructure for validating the proposed irrigation scheduling techniques. We put our efforts on building a real irrigation system including a wireless sensor network to get environmental data and a controller that adopt advanced control techniques. This system is easy to use and able to run in multiple modes.

4. A RELIABLE SOIL MOISTURE SENSING METHODOLOGY *

Soil moisture sensing data can help make better decisions of irrigation. However, this conclusion heavily relies on the assumption that sensing data accurately convey soil water content information. In reality, this is not always true. In harsh outdoor environments, sensors tend to fail to convey the correct information. This will result in wrong decision makings and thereby cause water loss and production reduction. Therefore, the sensing reliability becomes critical for agricultural irrigation. In this chapter, a reliable sensing methodology is proposed. It includes a genetic algorithm based sensor placement method and a fault detection technique.

4.1 Previous Works

There are two categories of previous works that are related to our methodology. One is on fault detection in wireless sensor networks and the other is on sensor placement.

4.1.1 Fault Detection in Wireless Sensor Networks

In a wireless sensor network, fault detection is used to detect transient or permanent sensor or communication failures. It has been extensively studied, and only a few study samples are reviewed here. A comprehensive survey is provided in [25]. As stated in this survey, the fault detection methods can be basically classified intro three categories: centralized, distributed and hybrid. For a centralized approach, there is one central node or base station to monitor and analyze the status of the rest of sensor nodes. For a distributed approach, the fault detection task is usually assigned to each sensor node. The hybrid approach is a combination of the distributed and centralized approaches. Since the number of soil moisture sensors is typically limited (due to cost) for agricultural irrigation use, a centralized approach should make more sense. Shuo et al. [26] proposed a method, “FIND” to detect nodes with data faults. FIND ranks the nodes based on their sensing readings as well as the physical distance from the event. If there is a significant mismatch

*Part of this chapter is reprinted with permission from "A Reliable Soil Moisture Sensing Methodology for Agricultural Irrigation" by Y. Yang, L. Sun, J. Hu, D. Porter, T. Marek, and C. Hillyer, 16th IEEE International Conference on Ubiquitous Computing and Communications (UICC), 2017.

between the sensor data rank and the distance rank, then the node is considered to be at fault. Jin et al. [27] introduced a passive diagnosis approach based on the autoregressive (AR) and Kuiper test. The prediction error of an AR model can serve as a reliable measurement indicating the abnormal conditions of wireless sensor networks (WSNs), and the Kuiper test is used to indicate the health conditions of WSNs. A distributed fault detection algorithm is introduced in [28], where faulty sensor nodes are identified based on comparison of data from neighboring nodes. In this work, time redundancy is further assessed and applied to tolerate transient faults in both sensing and communication. The work of [29] discusses a distributed fault detection algorithm for identifying the faulty sensors based on node distance. It calculates the error between measurement data and distance weighted value of the neighbor nodes, and identifies faulty sensors by comparing this error with a certain threshold. In work [3], a weighted average value is exploited in a distributed fault detection manner. More specifically, making use of spatial redundancy, a faulty sensor can be diagnosed through comparing its own data with the average data from its neighboring sensors.

4.1.2 Sensor Placement

Most previous works on sensor placement are focused on minimizing sensor cost subject to the constraints of either spatial coverage or information acquisition. The coverage constrained methods ensure that a fixed sensing region is entirely represented by the sensors. One example is [7], which presents grid coverage strategies for effective surveillance and target location in a distributed sensor network. The work of [8] is a sensor placement scheme to solve the so-called Heterogeneous Point Coverage Problem, which is to cover a large number of target points with various coverage requirements using a minimal number of sensors. In order to avoid the fixed sensing region assumption, some works take different approaches. In particular, previous work [9] establishes a correlation-aware probabilistic model that selects the best sensor reading to acquire. The works of [10] and [30] attempt to find the most informative locations to place sensors by maximizing mutual information. The previous work of [11] was targeted to soil moisture sensing. In this work, it is observed that soil moisture data admits a coarse-grained monotonic ordering of locations in terms of their soil moisture content and this ordering is relatively stable over time.

Based on this observation and the assumption of Gaussian distribution, this work is a clustered sensor placement scheme, where locations are classified into clusters according to the ordering. There are additional works [31, 32] and [33] that use genetic algorithm for sensor placement with the objective of minimizing sensor cost.

4.1.3 Relation with Our Methodology

Although there are many previous works on fault detection in wireless sensor networks and sensor placement, they are mostly general techniques and there is almost no prior work specific for agriculture application or handling soil moisture sensors. For example, many previous works on fault detection utilize spatial and temporal redundancy. However, almost none of them exploits the spatial and temporal correlations with statistical approach like in our work. Perhaps the only the previous work on soil moisture sensing is [11]. However, this work is focused on reducing sensor cost, which is relatively easy to manage in agricultural irrigation, and does not address the reliability issue.

4.2 The Proposed Reliable Sensing and Sensor Deployment Methodology

Figure 4.1: A field applying center pivot irrigation machine.

Given a crop field, as shown in Figure 4.1, covered by a center pivot irrigation machine and

constraint on sensor expense, the goal of our methodology is to reliably obtain soil moisture information for a sufficiently long period of time considering sensor faults. This methodology consists of three parts: (1) statistically inferring soil moisture levels from functional sensors, (2) detecting sensor faults such that information inference is not confounded by faulty sensors, and (3) placing sensors such that high quality inference can be obtained even when there are a few sensor faults.

4.2.1 Soil Moisture Inference from Sensor Data

In a crop field covered by an irrigation machine, e.g., a circle area covered by a center pivot machine, one needs to obtain soil moisture level data within the pivot area so the amount of irrigation water can be accordingly decided. Inevitably, soil moisture levels vary at different locations due to different terrain (slope), soil type, the rotational time aspect of the pivot, etc. Usually, soil moisture levels are spatially continuous and the variations are spatially correlated. Therefore, it is practically effective to infer soil moisture levels of an entire field from data of several well placed sensors.

Kriging [34], [35], a theory developed by G. Matheron and D. G. Krige, is a systematic approach to the inference through statistical interpolation. More specifically, given sensor readings $\{z_1, z_2, ..., z_n\}$ at n locations, the moisture level z_t at any location t is inferred by

$$z_t = \lambda_1 z_1 + \lambda_2 z_2 + \ldots + \lambda_n z_n \tag{4.1}$$

where $\lambda_i, i = 1, 2, ..., n$ are constant kriging coefficients satisfying $\sum_{i=1}^{n} \lambda_i = 1$.

The values of kriging coefficients λ_i are obtained by solving the following linear system

$$\begin{aligned}
\lambda_1\gamma(d_{1,1}) + \lambda_2\gamma(d_{1,2}) + \ldots + \lambda_n\gamma(d_{1,n}) &= \gamma(d_{1,t}) \\
\lambda_1\gamma(d_{2,1}) + \lambda_2\gamma(d_{2,2}) + \ldots + \lambda_n\gamma(d_{2,n}) &= \gamma(d_{2,t}) \\
&\vdots \\
\lambda_1\gamma(d_{n,1}) + \lambda_2\gamma(d_{n,2}) + \ldots + \lambda_n\gamma(d_{n,n}) &= \gamma(d_{n,t})
\end{aligned} \tag{4.2}$$

where $d_{i,j}$ indicates the distance between locations i and j, and $\gamma(d_{i,j})$ is semivariance - half of the variance of the difference of moisture levels between location i and j. Figure 4.2 shows an example of semivariogram. A semivariance can be estimated as

$$\gamma(d) = \frac{1}{2n(d)} \sum_{i=1}^{n(d)} (z(x_i + d) - z(x_i))^2 \tag{4.3}$$

where z is the data at particular location, d is the distance between locations, and $n(d)$ is the total number of sample data of paired locations at a distance of d, x_i and $x_i + d$ refer to the ith pair locations that the distance between them is d.

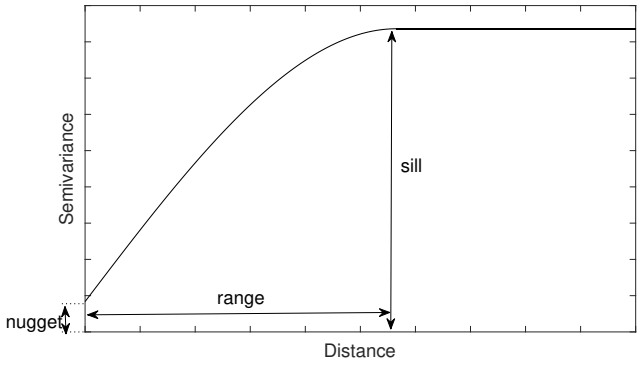

Figure 4.2: An example of semivariogram.

There are three important characteristics of semivariogram: range, sill and nugget [36], which are illustrated in Figure 4.2. The range is defined as farthest distance the two correlated locations can be away. The samples for distance within the range are spatially correlated. The sill is the value of semivariance when distance reaches the range. In theory, the semivariance should be zero when lag distance is zero. However, if it is some value significantly larger than zero for some tiny separation distance, then this value is referred to as the nugget. Estimating semivariance using Equation (4.3) usually requires a large number of actual data samples. A simpler yet effective

approach is to apply fitting of the spherical model

$$\gamma(d) = \begin{cases} C_0 + C_1(\frac{3d}{2r} - \frac{d^3}{2r^3}), & \text{if } d \leq r \\ C_0 + C_1, & \text{otherwise} \end{cases} \tag{4.4}$$

where r is the range in the semivariogram, C_0 and C_1 are two constant numbers that can be characterized according to experiment data in [37].

4.2.2 Fault Detection in Wireless Sensor Networks

Reliable sensing means to maintain appropriate system operation despite faults in a limited number of sensors. If a sensor fault can be detected, the kriging-based soil moisture inference can be continued without using data from the faulty sensor. As such, sensing grid quality can be maintained operationally even in presence of a few faults. In addition, the time interval for required field sensor repair or battery replacement can be extended significantly.

The faults considered in this work include transient faults, such as communication errors, battery exhaustion, and permanent electronic device failures in sensor nodes. Our fault detection is centralized. That is, the central irrigation controller collects data from all sensors and detects any faults from these data. The fault detection is based on both spatial correlation among multiple sensors and temporal correlation among samples of the same sensor. More specifically, a sensor's data are compared with its neighboring sensor data and its recent prior (historical) data. The data are regarded as faulty if significant inconsistency is observed from the comparisons.

Suppose there are n sensor nodes $\{s_1, s_2, ..., s_n\}$. For each sensor s_i, m of its most recent samples $z_{i,1}, z_{i,2}, ..., z_{i,m}$, among which $z_{i,m}$ is the latest one, are stored. Each data sample $z_{i,k}$ is associated with a binary indicator $v_{i,k}$, which is equal to 1 for valid data and 0 for faulty data. The soil moisture at location of sensor s_i inferred from its neighboring sensors is

$$z_{i,k}^{neighbor} = \frac{\sum_{s_j \in neighbor(s_i)} \lambda_j z_{j,k-1} v_{j,k-1}}{\sum_{s_j \in neighbor(s_i)} \lambda_j v_{j,k-1}}, \tag{4.5}$$

where λ indicates the kriging coefficient. Note the inference at time step k is based on the data at time step $k-1$, whose validity have already been evaluated. The prediction of current data at sensor s_i from its history is given by

$$z_{i,k}^{history} = \frac{\sum_{j=1}^{k-1} z_{i,j} v_{i,j}}{\sum_{j=1}^{k-1} v_{i,j}} \tag{4.6}$$

where index k is for the current time step. There are two special cases to be handled. The first one is when all historical data of sensor s_i are faulty, i.e., the denominator of Equation (4.6) is 0. Then, the historical average data are ignored in the fault detection. The other case is when all neighbor sensors are faulty and implies an overall system failure, which requires human actions such as sensor repair or battery replacement.

Except for the special cases, the fault detection is based on a consistency check with the weighted average between $z_{i,k}^{neighbor}$ and $z_{i,k}^{history}$ as follows:

$$\tilde{z}_{i,k} = \frac{\sum_{s_j \in neighbor(s_i)} v_{j,k-1}}{V} z_{i,k}^{neighbor} + \frac{\sum_{j=1}^{k-1} v_{i,j}}{V} z_{i,k}^{history} \tag{4.7}$$

where $V = \sum_{s_j \in neighbor(s_i)} v_{j,k-1} + \sum_{j=1}^{k-1} v_{i,j}$ is the total number of valid samples from neighbors and the history. Please note the weights for $z_{i,k}^{neighbor}$ and $z_{i,k}^{history}$ are dynamically determined by their numbers of valid samples, respectively. The validity $v_{i,k}$ is decided by comparing the actual measurement data $z_{i,k}$ and the data $\tilde{z}_{i,k}$ inferred from spatial and temporal correlation:

$$v_{i,k} = \begin{cases} 0, & \text{if } |\tilde{z}_{i,k} - z_{i,k}| > \theta \cdot z_{i,k} \\ 1, & \text{otherwise} \end{cases} \tag{4.8}$$

where θ is a small positive constant value, called sensitivity threshold. Although there are previous works using neighboring and historical data for fault detection, to the best of our knowledge, ours is the first one that is based on kriging. The kriging approach provides a statistical theory foundation and addresses the special problem of soil moisture sensing well.

4.2.3 Soil Moisture Sensor Placement

4.2.3.1 Problem Formulation

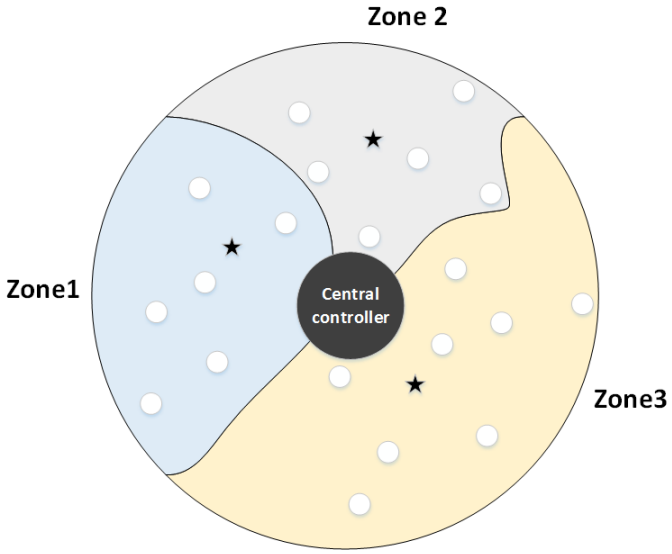

Figure 4.3: A field is divided into multiple zones.

Given a crop field covered by an irrigation machine and constraint on the number of sensors, one needs to place the sensors at locations such that sensing data reliably reflect soil moisture levels in the field. A crop field covered by a center pivot machine is of a circular shape as illustrated in Figure 4.3. In site-specific variable rate irrigation (SS-VRI), a field is usually divided into multiple zones and each zone has distinct soil type, terrain shape or crop type. One special case is variable speed irrigation, which requires all the zones intersect at the circle center. The center pivot is where the irrigation system is connected to the water source. The water source may actually be a well or reservoir that is located some distance away, but a pipeline transmits the water from the source to the pivot point. It is possible the water well is located near the pivot, but it is not necessarily the case. In many applications, the soil type, terrain shape and crop type within a zone are not very different from each other. As such, all locations in the same zone share the same irrigation actions, which are decided according to the soil moisture level at a representative location, such

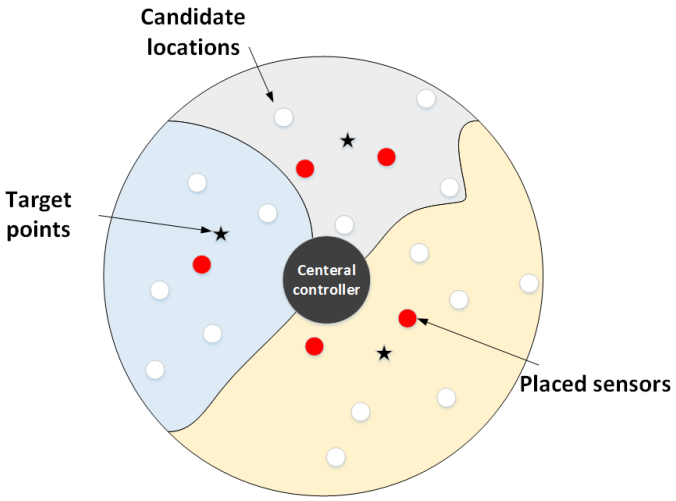

Figure 4.4: Sensors are placed according to genetic algorithm.

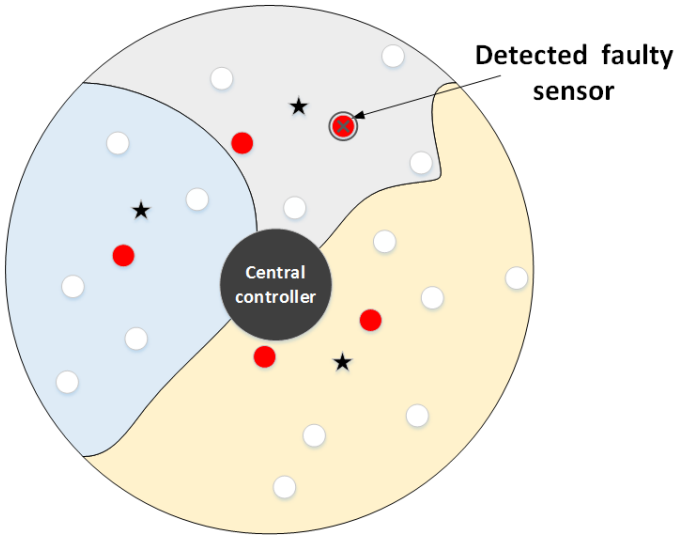

Figure 4.5: Faulty sensor is detected and excluded from information inference.

as the zone center. The set of zones and exchangeably their central points, which are called *target points*, are denoted by $Q = \{q_1, q_2, ...\}$. We assume there is a set of candidate sensor locations $P = \{p_1, p_2, ...\}$. Although a sensor can be placed anywhere in a field, a sufficiently large set P that is regularly distributed is effectively equivalent to considering all points in the field. In Figure 4.4, target points are indicated by stars and candidate sensor locations are illustrated by small circles. Figure 4.5 shows the sensor placement together with faulty sensor detection.

If the constraint of sensor cost allows up to n sensors, one needs to find a subset of locations $S \subset P, |S| \leq n$ to place sensors so as to minimize the error between the actual moisture levels at zone centers (target points) and those inferred from sensors. We consider the cases where $|Q| < n < 2|Q|$ such that a small degree of redundancy is allowed with limited sensor cost overhead. This placement optimization is mathematically described by

$$S^* = arg \min_{S \subset P, |S| \leq n} E[err(z_t, \tilde{z}_t(S))] \tag{4.9}$$

where z_t denotes the actual moisture levels at target points, $\tilde{z}_t$ is the estimate of moisture levels at target points according sensor data, $err()$ is a certain error function measuring the distance between the actual value and the estimate and $E[.]$ denotes expectation. This problem can be easily solved by placing sensors at zone centers. However, a reliable sensing requires that sensing result is still acceptable despite failures of a small number (k) of sensors. Let S' be a subset of sensors excluding at most k faulty sensors. Our sensor placement also seeks the objective of

$$S^* = arg \min_{S' \subset S, S \subset P, |S| \leq n, |S'| > n-k} E[err(z_t, \tilde{z}_t(S'))] \tag{4.10}$$

Simply placing sensors at zone centers would not necessarily work well for this concern. Directly optimizing both (4.9) and (4.10) is very difficult, as they require actual soil moisture values at target points. Trying different placements with actual sensor measurement is considered too costly. Moreover, sensor placement needs to consider communication quality. Since soil moisture sensors communicate directly to the central controller, they should be placed within range of the circle center so as to have reliable communication quality. To quantify the relation between the communication quality and distance, an experiment was conducted in the field using Xbee pro S2 transmitter. Figure 4.6 shows the measurement results on packet error rate and received signal strength indicator (RSSI) for different communication distances. One can see that the communication quality is excellent when the distance is within $150m$, is marginal for distance range of $150 - 250m$, and poor for distance beyond $250m$.

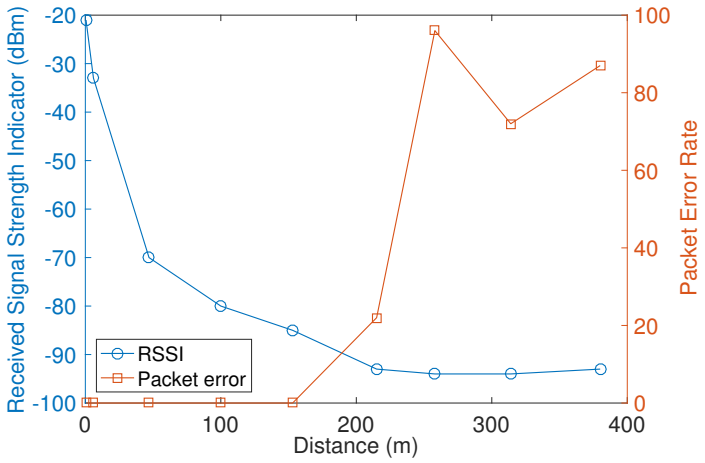

Figure 4.6: Field test results for sensor communication.

Our strategy is to solve a surrogate problem that is largely equivalent to simultaneous consideration of both (4.9), (4.10) plus communication quality. One observation is that a sensor s_i at location p_i contributes greatly to estimating the moisture level at target point q_j, if the absolute value of kriging coefficient $|\lambda_{i,j}|$ is large. Hence, it is preferred to choose locations with relatively large $|\lambda_{i,j}|$ with respect to target points. The reliability objective requires that the kriging coefficients of all selected locations are similar to each other. Otherwise, the overall estimation accuracy would have a large degradation when the sensor with the largest kriging coefficient fails. Kriging coefficients can be computed according to Section 4.2.1, which is much more efficient than repeating direct measurements at the field level.

4.2.3.2 Placement by Genetic Algorithm

The surrogate problem is generally a nonlinear integer programming and thus still difficult to solve. We solve it using a genetic algorithm, which is a very flexible framework. In fact, its flexibility allows us to easily take sensor-controller communication into consideration. Genetic algorithm emulates evolution in biological world, where each solution is like a living being. Starting with a set of arbitrary initial solutions or population, new solutions or a new generation of population can be obtained through reproduction. Throughout generations of reproductions, good solutions are retained like survival the fittest in evolution.

A fundamental step in genetic algorithm is to encode each solution as a chromosome. For sensor placement, each solution is encoded by a binary string $B = b_{|P|}b_{|P|-1}...b_2b_1$, where each bit b_i, $i = 1, 2, ..., |P|$ is 1 if a sensor is placed at candidate location $p_i \in P$ and otherwise 0. To satisfy the constraint that no more than n sensors are placed, we require $b_{|P|} + b_{|P|-1} + ... + b_2 + b_1 \leq n$.

In the genetic algorithm, solutions are evaluated by a fitness function and a solution with a large fitness value is preferred. To account for the effect of communication quality, we associate each sensor location $p_i \in P$ with a communication quality factor c_i. The communication quality factor is decided according to the distance from sensor to central controller. If p_i is close to the central controller, the corresponding c_i has a relatively large value. For one sensor placement solution $\hat{B}$, we define its fitness function as

$$f(\hat{B}) = \frac{\sum_{i=1}^{|P|} c_i b_i (\sum_{j=1}^{|Q|} (\lambda_{i,j}^{P})^2)}{\sum_{i=1}^{n} \sum_{j=1}^{|Q|} (\lambda_{i,j}^{\hat{B}} - \frac{1}{n})^2} \tag{4.11}$$

Please note $\lambda_{i,j}^{P}$ and $\lambda_{i,j}^{\hat{B}}$ are quite different, although both indicate kriging coefficient between sensor location p_i and target point q_j. For $\lambda_{i,j}^{P}$, the kriging is based on all candidate locations in P. By contrast, the kriging for $\lambda_{i,j}^{\hat{B}}$ is from only the n sensor locations in solution $\hat{B}$. In the numerator of the fitness function, b_i has value of either 0 or 1 according solution $\hat{B}$. Hence, a solution involving large absolute values of kriging coefficients and high communication quality c_i tends to have large fitness value. The denominator is to make the kriging coefficients from the actual solution as uniform as possible, as the average kriging coefficient for a solution with n sensors is $\frac{1}{n}$. Overall, the fitness function encourages solutions with high inference quality (by large $|\lambda_{i,j}^{P}|$), high communication quality (by large c_i) and high reliability (by small $(\lambda_{i,j}^{|B|} - \frac{1}{n})^2$).

Given an initial population of solutions, new solutions are generated by selection, crossover and mutation. The selection operator is to select a subset of good solutions from the population of current generation according the fitness function, and then carry them over to the next generation. If there is a set $\mathcal{B} = \{B_1, B_2, ...\}$ solutions and each solution $B_i \in \mathcal{B}$ is evaluated by fitness function $f(B_i)$, the probability solution B_i is carried over to the next generation is $\frac{f(B_i)}{\sum_{j=1}^{|\mathcal{B}|} f(B_j)}$. An

important reason for using square in the fitness function definition in Equation (4.11) is to amplify the difference among solutions such that the chance of retaining good solutions is increased. Given a pair of selected solutions, there is a probability of crossover, which is then to combine parts of the chromosomes of both solutions to form a new solution. The probability is called the *crossover rate*. If crossover is not performed, the solutions stay the same as before. After the step of probabilistic crossover, mutation – a random change to the chromosome, is performed with a certain probability, which is the *mutation rate*. Please note new solutions must satisfy constraint $b_{|P|}+b_{|P|-1}+...+b_2+b_1 \leq n$ and are dropped out otherwise. Since "the better" solutions are kept in the selection, the overall solution quality increases over generations. The algorithm terminates when the maximum fitness solution converges and plenty of generations have been examined.

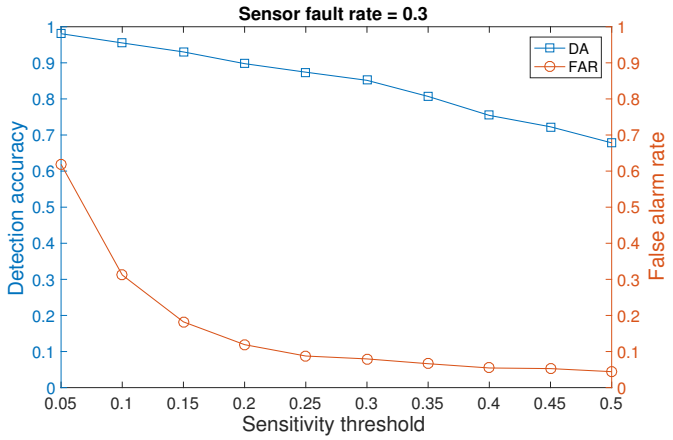

Figure 4.7: Detection accuracy and false alarm rate of our method over different sensitivity thresholds.

4.3 Simulations and Results

4.3.1 Results on Fault Detection

The quality of a faulty sensor detection method is evaluated by *detection accuracy* (DA) and *false alarm rate* (FAR). DA is defined as the ratio of the number of detected faulty sensors to the total number of faulty sensors and FAR is the ratio of the number of fault-free sensors identified

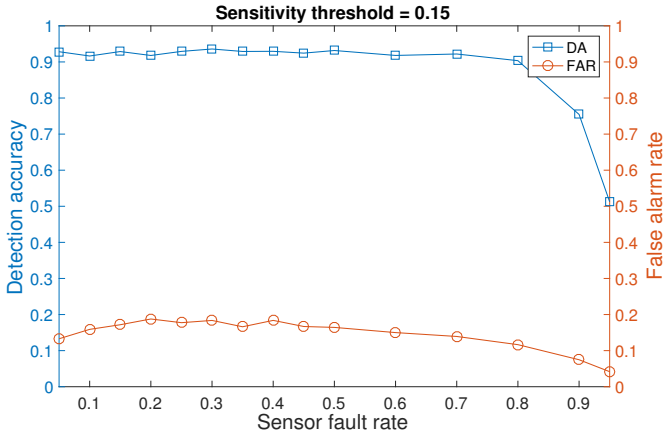

Figure 4.8: Detection accuracy and false alarm rate of our method over different sensor fault rates.

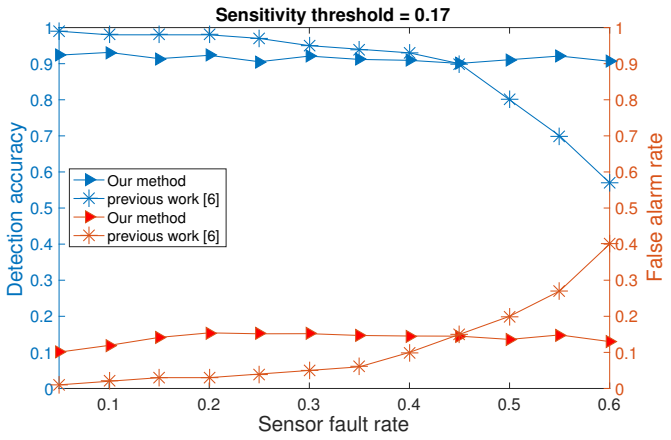

Figure 4.9: Comparison with previous work [3].

as faulty to the total number of fault-free sensors. In Figure 4.7 and Figure 4.8, DA and FAR of our method are evaluated for different sensitivity threshold θ values and different sensor fault rates (probability of fault) on a test case of 10 sensors. From the simulations, $\theta = 0.17$ is found to be a good choice as it leads to over 90% detection accuracy with about 10% false alarm rate. In Figure 4.9, we compared our method with the previous work [3]. As shown, one advantage of our method is that DA and FAR are still relatively good and acceptable even when sensor fault rate is very high, not like DA and FAR of work [3], which quickly becomes worse as the sensor fault rate increases.

4.3.2 Results on Sensor Placement

In sensor placement simulations, different cases, as shown in Table 4.1, are explored. The population of genetic algorithm is fixed at 20. The maximum allowable generation of GAs is 500.

Table 4.1: Test cases.

Case	Sensor budget	# candidate locations	# target points
1	3	10	2
2	5	10	3
3	7	20	4
4	8	20	5

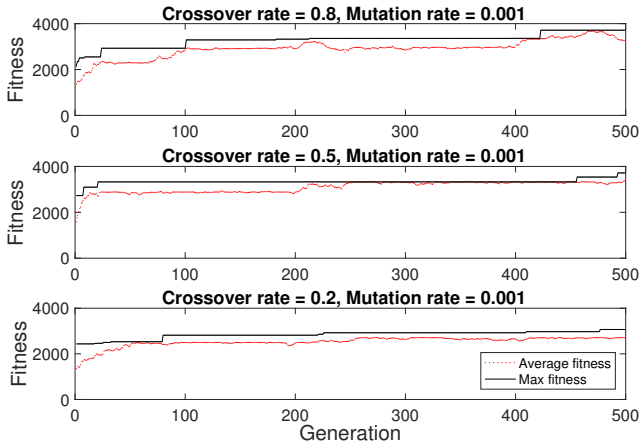

Figure 4.10: Fitness function value over iterations with different crossover rates.

For one test case, we studied the impact of parameters in the genetic algorithm and the results are shown in Figure 4.10 and Figure 4.11. They show that the quickest convergence occurs when the crossover rate is 0.8 and the mutation rate is 0.01.

We compared our sensor placement technique with two other methods:

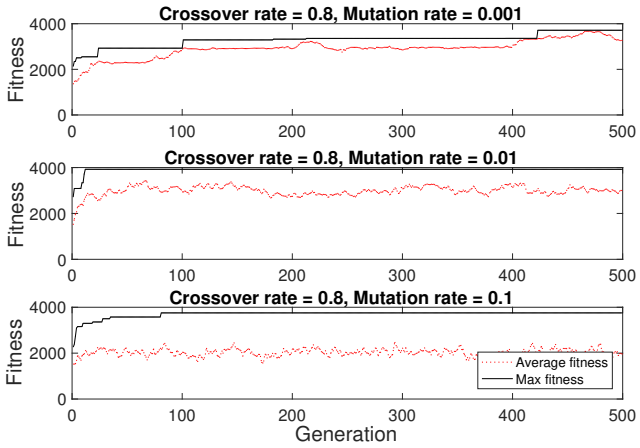

Figure 4.11: Fitness function value over iterations with different crossover rates.

- Naïve method: For each target point, find the nearest location, which is not blocked by crop or irrigation machine motion trace, to place a sensor.
- HPC (Heterogeneous Point Coverage) [8]: This previous work first assumes that each sensor can cover certain size of area. It finds a sensor placement such that each target point is covered by at least certain number of sensors.

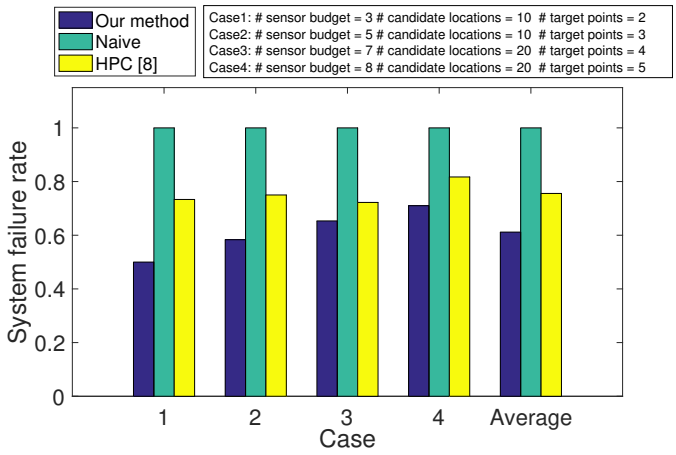

Figure 4.12: System failure rate in presence of single sensor failure without fault detection.

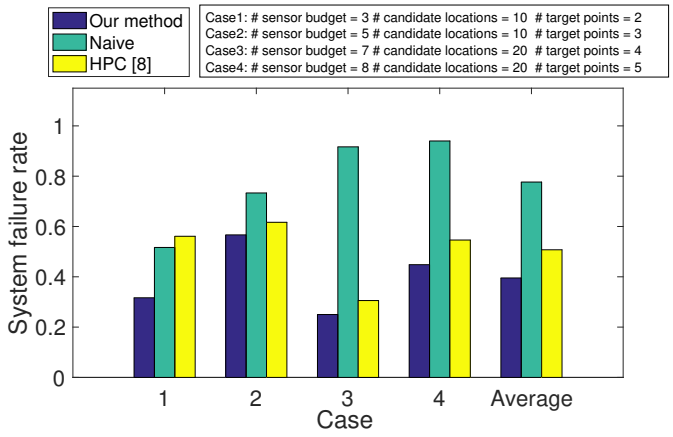

Figure 4.13: System failure rate in presence of single sensor failure with fault detection.

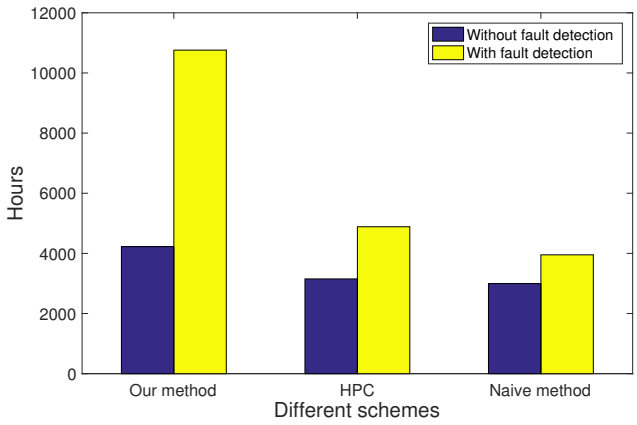

Figure 4.14: System MTTF (Mean Time To Failure).

Since the goal of our work is reliability, we evaluate system failure rate in presence of sensor faults, and particularly the case of a single sensor fault, which is generally the most common field case. If there are n sensors, n cases of single sensor fault are examined. A system failure means the sensing error for at least one target point is unacceptably large. System failure rate is the ratio of the number of system failures over n. The comparison results are shown in Figure 4.12 and Figure 4.13. Figure 4.12 is for cases without using the faulty sensor detection method described in Section 4.2.2 and indicates that our method leads to lower system failure rate than the other

two methods. Figure 4.13 is the result for cases with faulty sensor detection. Evidently, the fault detection can reduce the system failure rate, and again our method significantly outperforms the other two methods.

In reliability analysis, an important and intuitive metric is mean time to failure (MTTF). MTTF is the average time that a system or device functions properly from its "new" state to a state of permanent failure, i.e., thus, the system mean lifetime. The system MTTF results are obtained by averaging various cases of each method assuming different failure rates for sensors of different ages. From Figure 4.14, one can see that MTTF is more than doubled by utilizing our method together with our proposed fault detection technique.

4.4 Conclusions

In this work, faulty sensor detection and sensor placement are studied for soil moisture sensing in agricultural irrigation, where reliability is a critical concern distinguished from ordinary wireless sensor network designs. We show that spatial and temporal correlation of soil moisture can be effectively used in fault detection and sensor placement based on kriging technology, which has not been utilized in previous works. Accordingly, a faulty sensor detection technique and a genetic algorithm for sensor placement are proposed. Simulation results from various test cases demonstrate that our proposed techniques can significantly reduce system failure rate. Our techniques can improve system mean time to failure by more than two fold compared with naïve method and a previous work.

5. DEEP REINFORCEMENT LEARNING IRRIGATION PLANNING AND AUTOMATED MICROIRRIGATION SCHEDULING

Fixed irrigation scheduling and threshold irrigation scheduling are widely used by farmers. However, these naïve strategies often result in water loss and a reduction in crop productivity. In this chapter, a deep reinforcement learning irrigation planning strategy is proposed to algorithmically optimize water application amount of a certain zone. In addition, an automated microirrigation scheduling method is developed to manage multi-zonal microirrigation or other fixed-zone irrigation.

5.1 Previous Works

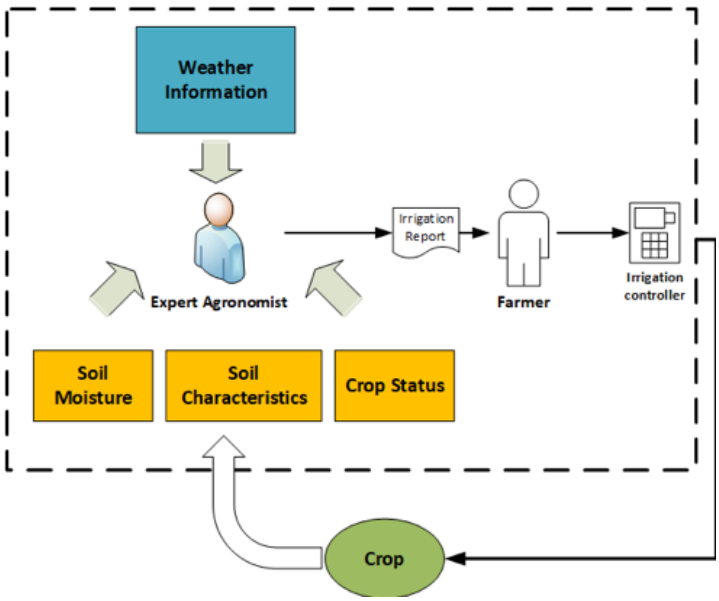

Figure 5.1: Traditional closed loop irrigation system.

By using wireless sensor technologies, the water use efficiency of agricultural irrigation can be greatly improved. Sensor-based irrigation approach [5, 6, 12, 13, 14, 15, 16, 17] can make irrigation decisions based on near term environment status and therefore reduce unnecessary irrigation

events. Usually, the prediction of irrigation water application amount is determined by an experienced farmer or an expert agricultural technician as per the system diagram in Figure 5.1. In this scenario, the expert needs to collect and analyze the information from different sources, such as weather conditions, soil characteristics, crop status and soil water levels. The manual calculation and analysis can be very tedious and time consuming. Moreover, it becomes unmanageable with an increasingly large number of sensory inputs. Capraro [38] introduced a neural network-based

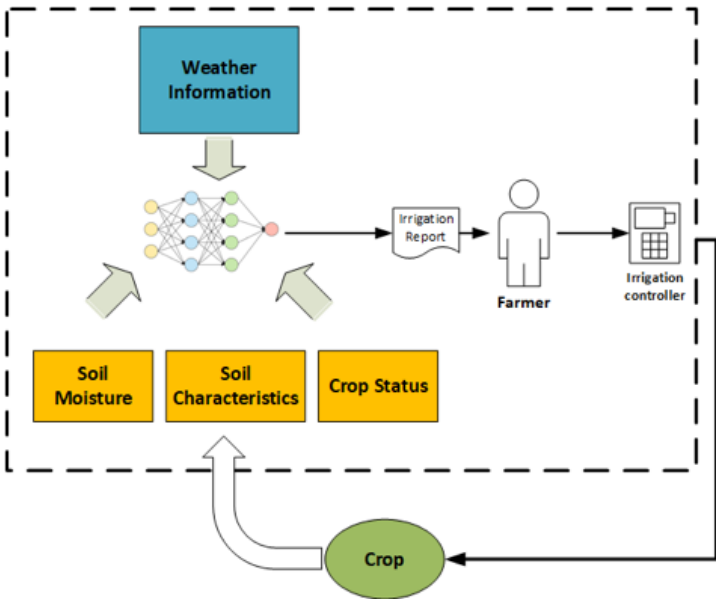

Figure 5.2: Machine learning-based irrigation advisor system.

irrigation approach. A more advanced one is proposed by Navarro-Hellín, et al. [39]. It is labelled as a Smart Irrigation Decision Support System (SIDSS), which is depicted in Figure 5.2. In this system, a machine learning-based model is used to replace the manual calculation and analysis. All the information or sensor inputs are handled by the machine learning model. Then, a farmer or crop consultant can determine an irrigation event decision based on the irrigation report, which is the output of the machine learning model. However, this system heavily relies on historical data and lacks the ability to "learn" from the changing environment. Moreover, it still requires human intervention to manually control the irrigation machine. [18, 19, 20, 21, 22, 23] proposed

a model-based irrigation strategy. This strategy determines event decisions based on either short-term (e.g. achieving/maintaining a set of soil-water deficit values) or predicted end-of-season effects (e.g. maximizing final yield). This strategy can potentially achieve more precise irrigation than offline approaches using traditional optimization techniques [40, 41, 42, 43]. However, this strategy heavily relies on the accurate mathematic model and lacks the ability to further adapt the model to different or multiple environments. Schütze [44] introduced a neuro-dynamic programming method where a model-based reinforcement learning technique is applied to determine irrigation decisions. However, the oversimplified model results in substantial inaccuracies. Sun, et al. [4] proposed a reinforcement learning-based irrigation control system as shown in Figure 5.3. The basic idea is to use a model-free reinforcement learning algorithm to determine both the decision making and irrigation control. Two neural networks (NNs) are also introduced to predict the DSSAT (The Decision Support System for Agrotechnology Transfer) [45] simulation results. One

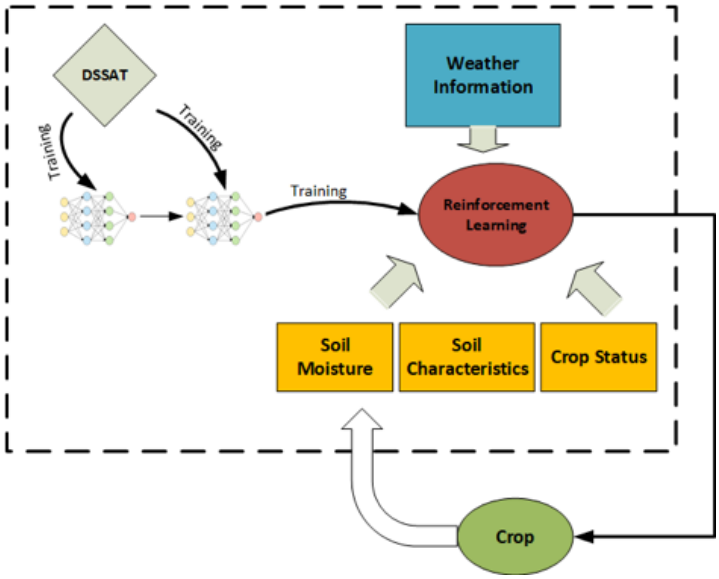

Figure 5.3: Reinforcement learning irrigation system structure.

NN inputs irrigation and weather information and predicts total soil water content while the other NN predicts crop yield given daily total soil water content of an entire crop season. Subsequently,

prediction of the crop yield is used as the training data to train the reinforcement learning model. The basic structure is depicted in Figure 5.3. This approach achieved relatively precise irrigation and allows full automation of the irrigation process. However, its state space is restricted to limited size and difficult to scale to large problems. Therefore, it is difficult to accurately represent actual irrigation context, leading to loss of important information that may affect the needed irrigation application amount. Deep reinforcement learning has attracted a lot of attention since Mnih et al. [24] proposed their first deep reinforcement learning algorithm, deep Q-networks (DQN). DQN has been proven to surpass human experts in some Atari (computer based) games. To the best of our knowledge, researchers have not studied or adapted this advanced algorithm to agricultural irrigation control. In this work, a deep reinforcement learning-based irrigation approach is proposed, which can overcome the drawbacks of traditional reinforcement learning and/or other machine learning-based irrigation control. To the best of our knowledge, there is no similar work on fixed zone micro-irrigation scheduling. Some prior works [46, 47, 48] consider one zone and attempt to compute the start/stop irrigation time for that specific zone based on sensory inputs instead of handling many irrigation zones.

5.2 Deep Reinforcement Learning Irrigation and Automated Microirrigation Scheduling

The proposed irrigation approach contains two parts: deep reinforcement learning irrigation and automated scheduling. Deep reinforcement learning irrigation precisely determines the water application amount needed in terms of current environmental information. This achieves the greatest seasonal net economic return, while automated scheduling is desired to automatically determine the start/stop times of a large number of irrigation management zones.

5.2.1 Deep Reinforcement Learning Irrigation Planning

5.2.1.1 Problem Formulation

Reinforcement learning is typically used to deal with the problem of a goal-directed agent interacting with an uncertain environment as shown in Figure 5.4. In reinforcement learning, people use a set of states $S = \{s_1, s_2, ...\}$ to present the status of the environment. For agricultural

irrigation, the states are the observations of the real world, which are related to environmental information, such as soil water content, weather conditions, soil profile characteristics, crop growth status, etc. The entire irrigation system is treated as an agent. The different choices of water application amounts correspond to different actions $A = \{a_0, a_1, a_2, ..., a_k\}$. Seasonal net return is the reward referred as R. Agricultural irrigation scheduling requires a sequence of decisions of water application amounts and that is evaluated by the long-term net return results from the decisions. Thus, the deep reinforcement learning irrigation problem can be described as that of finding an optimal policy $\pi : S \mapsto A$ so that if in each state s the agent takes action $\pi(s)$, it will obtain maximum expected sum of rewards:

$$\mathbf{E}_\pi[R_0 + \gamma R_1 + \gamma^2 R_2 + \cdots] \tag{5.1}$$

where γ is the discount factor that determines the importance of future rewards. Therefore, the objective is to maximize the expected sum of rewards and the decision variable is the policy π. The constraint is that the choice of actions of each state is valid, $\pi(s) \in A$. In this work, the reward function is defined as follows:

$$Reward = Yield * Price_{crop} - Irrigation * Price_{water} \tag{5.2}$$

where $Yield$ is crop yield with unit ton/ha, $Irrigation$ is the total irrigation amount with unit ha-mm/ha, and $Price_{crop}$ and $Price_{water}$ represent the crop price and water price, respectively.

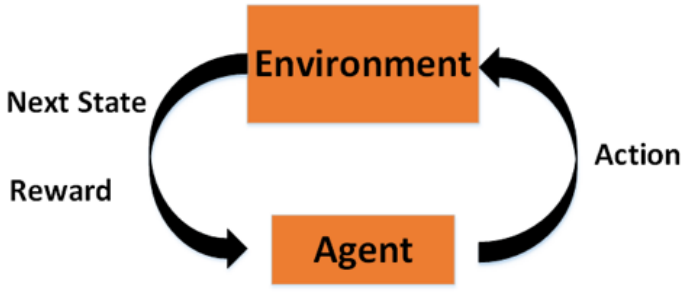

Figure 5.4: Interaction between agent and environment of reinforcement learning.

At a certain state s, if the agent takes some action a, the quality of state-action pair can be indicated by action-value function $Q(s, a)$. $Q^*(s, a)$ indicates the optimal action value given some state s.

$$Q(s, a) = \mathrm{E}\left[R_t | s_t = s, a_t = a\right] \tag{5.3}$$

$$Q^*(s, a) = \max_{\pi} \mathrm{E}\left[R_t | s_t = s, a_t = a, \pi\right] \tag{5.4}$$

A well-known approach to calculate the optimal Q is to iteratively update the Q value according to equation 5.5.

$$Q\left(s_t, a_t\right) = Q\left(s_t, a_t\right) + \alpha \left[r_t + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}\right) - Q\left(s_t, a_t\right)\right] \tag{5.5}$$

After certain number of iterations, the Q values can converge to the optimal. After obtaining the optimal Q for every state and action pair, which corresponds to the maximum $Q(s, a)$ value, the optimal policy can be easily derived. It tells the agent the best action a^* to take at given state s.

5.2.1.2 Deep Q-Networks Irrigation

Traditional reinforcement learning (Q learning) stores a Q table while learning from the data. Deep reinforcement learning treats the multi-dimensional sensor inputs as the observation of the real world and uses artificial neural networks to approximate the Q function. Artificial neural networks make it possible to deal with the large quantities of sensed data and makes the proposed irrigation approach scalable. Given an irrigation agent, at each time-step, the agent selects an action a_t from the set of legal actions, $A = \{a_0, a_1, a_2, ..., a_k\}$, where each action corresponds to a specific water application amount. The action is passed to an environment emulator interfaced with AquaCrop [49] to calculate the reward in terms of predicted yield and water consumption. In DQN, the current observation of the environment (using multi-sensor inputs) x_t is used to indicate the current state s_t. The goal of the irrigation agent is to interact with the real environment and the AquaCrop model by selecting actions in a manner that maximizes the long-term return. An important technique in DQN is experience replay [50]. The experience at each time-step,

$e_t = (s_t, a_t, r_t, s_{t+1})$, are stored into a replay memory, which has a limited size, where r_t is the accumulative reward at time step t. Once the maximum size is reached, the newest experience overwrites the oldest experience. The objective of experience replay is to apply minibatch training for the artificial neural networks that is used to approximate the Q function using samples from replay memory. To improve the training performance, a technique called combined experience replay (CER) [51] is also employed, which enforces the last experience contained in the samples. At each time step, the agent selects an action according to ε-greedy policy [52]. Based on the current state and action, the environment emulator computes the next state and the subsequent reward. The system structure is depicted in Figure 5.5 and the detailed algorithm process is shown in Algorithm 2.

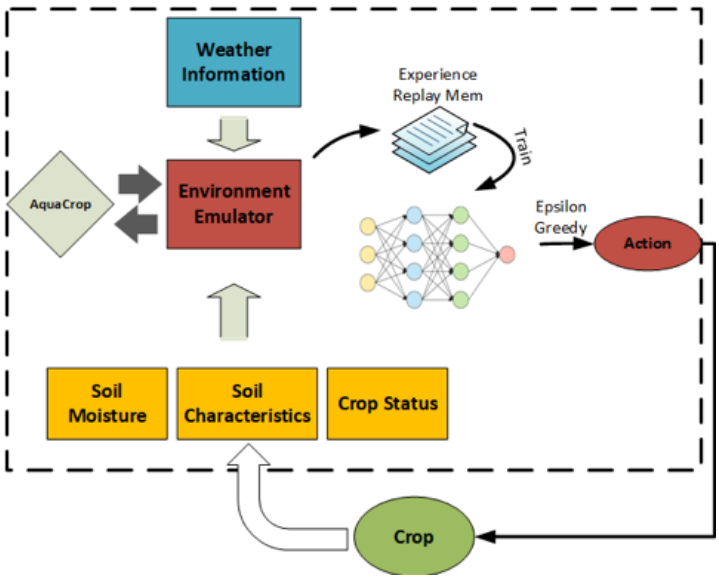

Figure 5.5: Proposed deep reinforcement learning irrigation system structure.

5.2.2 Automated Microirrigation Scheduling

5.2.2.1 Motivation

Although the proposed deep reinforcement learning irrigation optimization approach can determine the optimal or near optimal water application for a certain irrigation management zone,

Initialize replay memory M to capacity N;
Initialize artificial neural network with random weights;
for *episode = 1 to M* **do**
    initialize batch size;
    set current state to be the starting state;
    while *Crop growing season is not end* **do**
        With probability ϵ select a random action a_t;
        Otherwise select $a_t = max_a Q^*(s_t, a_t; \theta)$;
        Execute irrigation according to choice of the actions;
        Pass irrigation amount to Environment Emulator;
        Pass sensory data to Environment Emulator;
        Emulator modify corresponding files in AquaCrop;
        Emulator call AquaCrop Plugin to predict the yield and next state s_{t+1};
        Calculate the reward r_t;
        Store transition (s_t, a_t, r_t, s_{t+1}) into M;
        Sample minibatch of transition (s_i, a_i, r_i, s_{i+1}) from M;
        if s_i *is terminal state* **then**
            $y_i = r_i$
        else
            $y_i = r_i + \gamma max_{a'} * Q^*(s_{i+1}, a'; \theta)$
        end
        Perform a gradient descent step on $(y_i - Q(s_i, a_i; \theta))^2$;
        Set current state to be s_{t+1};
    end
end

Algorithm 2: Deep reinforcement learning irrigation.

the detailed scheduling for control use with the irrigation machine is still unclear. Generally, a center pivot irrigation system can adapt its speed to achieve differing water application rates; however, in the scheduling of fixed-zone systems, such as that of a subsurface drip irrigation system, a landscape irrigation system, or greenhouse/nursery system, the time of application can be varied but can still be challenging considering the given system hydraulic constraints. The proposed automated scheduling approach can overcome these limitations and fully automate the irrigation process by determining the timing of irrigation to provide the desired water application rate(s) for fixed-zone irrigation systems. It can also protect the irrigation system from large fluctuations in flow and pressure and assure system operations within the system hydraulic constraints.

5.2.2.2 *Problem Formulation*

An irrigation task is the continuous application of water in several time slots to a specific zone for partially fulfill the designated water amount for this zone. The maximum line pressure for drip tape or emitter lines needs to be balanced during irrigation events to avoid damage from over pressurization of the system. In addition, simultaneous operations of multiple zones (e.g. activating valves) must be avoided, since this can cause a large pressure fluctuation. Therefore, multi-zonal irrigation can be treated as a problem of scheduling several competing tasks that require exclusive use of a common resource. The total time of one irrigation iteration can be more than one day. However, we assume that the amount of water to be applied has already been optimized by the proposed deep reinforcement learning planning strategy. It is generally better to finish all the irrigation within a day or a limited time span. Sometimes it is impossible to schedule and complete all irrigation tasks within one day. In this case, we assume no planned prioritization and we wish to avoid scheduling bias, which means we do not want certain zones to preempt the resources of the irrigation system. Thus, the goal of the scheduling is to minimize the total irrigation time and balance the water needs of the zones subject to the system hydraulic constraints.

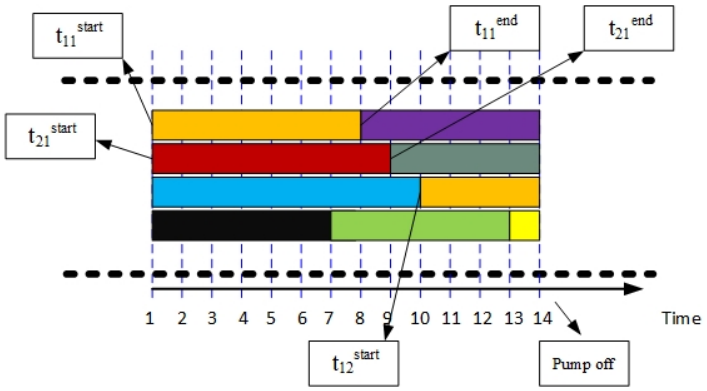

Figure 5.6: A scheduling for the irrigation under the system hydraulic constraints (total 8 zones and need to run 4 at a time). Different colors correspond to different zones. The rectangle indicates start/end time and the duration of each irrigation task.

The irrigation tasks require to use the shared resource of irrigation, such as well or pump capacity, which can serve only m tasks at a time. Each zone has a certain target irrigation time T_i, the time corresponding to the water application amount of zone i. Let ${t_{i,j}}^{start}$ be the j^{th} start time of irrigation of zone i and ${t_{i,j}}^{end}$ be the j^{th} end time of irrigation of zone i. Let k_i be the number of irrigation time segments for zone i and $K = \{k_1, k_2, \cdots, k_n\}$. Thus, ${t_{i,k_i}}^{end}$ is the last irrigation end time of zone i, which is the end time of the latest irrigation time segment. Please note that ${t_{i,j}}^{start} \leq {t_{i,j}}^{end}$. The number of management zones can be very large so the water needs of zones may not be fully met before the next needed irrigation event. In this scenario, time becomes critical and therefore an early irrigation end time for all zones is preferred. In addition, to address the end day soil profile water status in the worst scenario is still acceptable, it is a necessity to balance the water needs of different zones. Therefore, the problem is to find proper vector $K = \{k_1, k_2, ..., k_n\}$ and corresponding $\{{t_{1,k_1}}^{end}, {t_{2,k_2}}^{end}, \cdots, {t_{n,k_n}}^{end}\}$ to make $\max_i t_{i,k_i}^{end}$ and $\max_i \left(T_i - \sum_{j=1}^{k_i} \left(t_{i,j}^{end} - t_{i,j}^{\text{start}}\right)\right)$ as small as possible simultaneously. To protect the irrigation system from large pressure fluctuations, the program (and solution) must guarantee that no more than one operation happens at a time. Hydraulic constraints can be defined as $\forall_{i \neq i'}(|{t_{i,j}}^{end} - {t_{i',j'}}^{end}| \geq \alpha)$, where α indicates the minimum operational transitional time interval between two operations to avoid abrupt changes of water pressure and water flow. In addition, to avoid any waste of water, the scheduling solution also needs to satisfy the condition $\forall_i(T_i - \sum_{j=1}^{k_i} ({t_{i,j}}^{end} - {t_{i,j}}^{start}) \geq 0)$. The start/end times need to be valid, which means the start time cannot be lower than 0 and the end time cannot go beyond one day (Assuming the reinforcement learning approach computes a possible solution water application amount in one day period). Thus, this constraint can be described as $\forall_{i,j}(0 \leq {t_{i,j}}^{start} \leq {t_{i,j}}^{end} \leq 1440)$, where the unit is minutes. Figure 5.6 shows a valid scheduling solution determined within the given system hydraulic constraints.

5.2.2.3 Automated Micro-Irrigation Scheduling

For a fully automated irrigation system, the irrigation schedule needs to be computed within a short time using general embedded systems. Therefore, it is critical to find a fast algorithm. In this effort, a greedy heuristic is proposed to solve this problem in $O(nlogn)$ time complexity.

Typically, a greedy algorithm will go through a sequence of steps with a set of choices at each step. One pitfall of the greedy algorithms is that they do not always yield optimal solution.

Data: The operation time interval α, total time of one iteration of irrigation $totalTimeOfOneTurnIrrigation$, and irrigation time for each zone corresponding to water application amount $zone$
Result: start time and end time for every segment of each zone
initialization;
for *j = 1 to n* **do**
    Push $zone_j$ in to $maxheap$;
    it indicates the irrigation time needs;
    j indicates its zoneID;
end
for *i = 0 to m - 1* **do**
    Push 0 into a min heap $minheap$;
end
while *the peak of* $minheap < totalTimeOfOneTurnIrrigation$ **do**
    $zone_k$ = $maxheap$.pop();
    $startTime$ = $minheap$.pop(); ;
    $validEndTime$ = VALIDATE($startTme + processingTime, minheap$);
    $validEndTime$ = min($validEndTime$, $totalTimeOfOneTurnIrrigation$);
    $startTime[k]$.add($startTime$);
    $endTime[k]$.add($validEndTime$);
    $zone_k$ =$zone_k - (validEndTime - startTime)$;
    $minheap$.push($validEndTime$);
    $maxheap$.push($zone_k$);
end

Algorithm 3: Automated scheduling algorithm.

The basic strategy of determination is to make the best choice at each step without considering all the subsequent steps. To achieve the goal that minimizes the total irrigation time, the algorithm always starts a new task as long as it is feasible to do so. To achieve the goal that minimizes the maximum remaining time of all the tasks, the zone with the most water needs will be handled first. Also, the algorithm assures that the respective choice does not violate the system hydraulic constraints.

Guided by these general principles, the detailed algorithm process is described in Algorithm 3. The irrigation tasks with corresponding zone IDs are put into a max-heap, which is a data structure

Data: A sequence of irrigation end time and a new candidate end time
Result: Output a valid new end time
Assume $T_0, T_1, ..., T_{m-2}$ satisfy hydraulic constraint at the beginning in increasing order which can be easily built from the min heap;
T_{new} is the new end time for T_k;
α is the minimum operation time interval;
for *i = 0 to m - 2* **do**
  if $|T_{new} - T_i| < \alpha$ **then**
    if $T_{new} > T_i$ **then**
      $T_{new} = T_{new} + \alpha$;
    else
      $T_{new} = T_{new} + 2 * \alpha$;
    end
  end
end
return T_{new}

Algorithm 4: Hydraulic VALIDATE algorithm.

in computer science that enables retrieval of the maximum value in constant time complexity. The irrigation tasks of zones are required irrigation time corresponding to the needed water application amount. Certain irrigation systems only allow m zones to irrigate simultaneously due to the physical limit (an operational parameter). Thus, a min-heap of size m is initialized with value 0. To avoid one "heavy" task from occupying the resource for an extended time, the maximum irrigation processing time limit is enforced. Once the irrigation time of one zone exceeds the processing (operational) time, that zone is forced to release the resource to the controller. Obviously, the start time is always the minimum value in the min-heap. The decision-making aspect of the end time is subtler than the start time. A VALIDATE algorithm is developed to make sure that the end time does not conflict with other end times and start times. VALIDATE algorithm is described in Algorithm 4. If one ignores the system hydraulic constraints, then the end time is simply the start time plus the processing time, which is a predefined constant and indicates the default irrigation time. However, it is difficult to solve this problem optimally considering the system hydraulic constraints. The problem can thus be described as: given a value (operational end time) and an increasing sequence of numbers and the difference of each pair of numbers in the sequence is

larger than the minimum interval α, one needs to insert a new number and make the differences of every pair of numbers in this sequence still larger than the minimum interval by changing the new number as little as possible. After obtaining the irrigation end time, the zone is then updated with a new water need after this irrigation cycle and it is pushed back to max-heap.

Let's look at a simple example. Set m to be 2 and the processing time to be 20 minutes. In addition, assume that there are 3 irrigation tasks $t_1 = 20, t_2 = 30, t_3 = 40$ (minutes) corresponding to zone 1, zone 2, and zone 3. These tasks are put into a max-heap and a min-heap of size m is initialized with 0. First, pull one from max-heap and get irrigation task of zone 3 since t_3 is the maximum. Next, pull one start time from min-heap and get 0. By adding it with the processing time, a candidate end time 20 is obtained. Pass this number to Validate algorithm. Then, Validate algorithm will handle it and output the valid end time. Since there is no end time before, thus the valid end time is 20. After pushing back the valid end time into our min-heap and calculating the actual irrigation time using the valid end time and start time, the actual irrigation time is deducted from the irrigation task t_3. Push back updated t_3 into max-heap. Then next time when pulling from max-heap, t_2 will be obtained since $t_2 = 30$ is the maximum in the max-heap now. Keep doing this until all irrigation tasks are finished or time out.

5.3 Results

5.3.1 Results for Deep Reinforcecement Learning Irrigation Optimization

To evaluate the proposed deep reinforcement learning irrigation optimization approach, a series of simulations were conducted. The simulations are based on AquaCrop and the detail configurations are shown in Table 5.1. The proposed approach is compared with three other irrigation solution approaches, namely, reinforcement learning (Q learning) irrigation, threshold-based irrigation and fixed irrigation. Detailed configurations are provided in Table 5.2. In the simulations, irrigation water amount (unit mm) of actions are $\{0, 0.5, 1, \cdots, 9.5\}$. Fixed irrigation is to irrigate $50mm$ every ten days. The state definition used in the reinforcement learning irrigation is shown in Table 5.3. The states of DQN are the observations of actual field conditions. It can be

Table 5.1: Detail configurations of simulations.

Number of growing season days	108
Number of crops	3
Input crop models	MaizeGDD
	WheatGDD
	SoybeanGDD
Location of weather data	Hawzen
Year of weather data	1992
Sowing date	March 22, 1992
Maturity date	July 7, 1992
Soil profile	Deep uniform loamy sand

Table 5.2: Parameters for learning algorithms.

Replay memory size of DQN	10000
Learning rate for DQN	0.001
Learning rate for Q learning	0.25
Number of episodes	400
Action size	20
Time step	1 day
Initial epsilon	1
Epsilon decay	0.985
Number of hidden layers	2
Number of hidden units of each layer	20
Discount factor	0.9999

Table 5.3: State definition of reinforcement learning adapted from [4]. The header rows are ranges of water content level with unit mm. The header columns are time steps. Each entry in the table is a state ID.

State	<=320	(320,325]	(325,330]	(330,340]	>340
<=39	1	2	3	4	5
<=60	6	7	8	9	10
<=81	11	12	13	14	15
>81	16	17	18	19	20

a tuple of sensing data from multiple resources. In our evaluations, the state is fully defined by the date, the crop stage, precipitation, irrigation, reference ET, total water content in effective root

Table 5.4: Examples of three different states for DQN.

	Year	Month	Day	Crop Stage	Precipitation	Irrigation	Reference ET	WC	Stomatal WC
State1	1992	5	2	3	0	2	3.5	333.6	110.7
State2	1992	5	3	3	0	0	3.6	330.7	115.8
State3	1992	5	4	3	3	0	2.9	328.0	120.9

Table 5.5: Comparison of Different Irrigation Methods on Different Weather Conditions.

Irrigation Method	Weather Conditions	Dry Yield (ton)	Irrigation (mm)	Net Return (dollar)
DQN	Dry	4.131	624	391
	Moderate	4.068	358	642
	Wet	4.001	25	956
Q Learning	Dry	4.029	582	408
	Moderate	3.979	346	632
	Wet	4.199	283	749
80% PAW allowable depletion	Dry	4.181	666.5	361
	Moderate	4.182	437	591
	Wet	4.199	99.1	933
50% PAW allowable depletion	Dry	4.196	724.1	307
	Moderate	4.197	511.8	520
	Wet	4.199	191	841
30% PAW allowable depletion	Dry	4.199	830.6	201
	Moderate	4.199	572.5	460
	Wet	4.199	195	837
Fixed irrigation	Dry	2.424	390	205
	Moderate	3.528	390	477
	Wet	4.199	390	642

zone and water content in effective root zone at the upper threshold for stomatal closure, which means if any value of these features of two observations is different, then they are identified as two different states. Some examples of the states for DQN are shown in Table 5.4. Table 5.5 shows the results including dry yield, total irrigation amount and net return of wheat for one crop season. To better demonstrate the results, we depict all the results in a 3D figure. Figure 5.7, Figure 5.8 and Figure 5.9 are results for dry weather condition, moderate weather condition, and wet weather

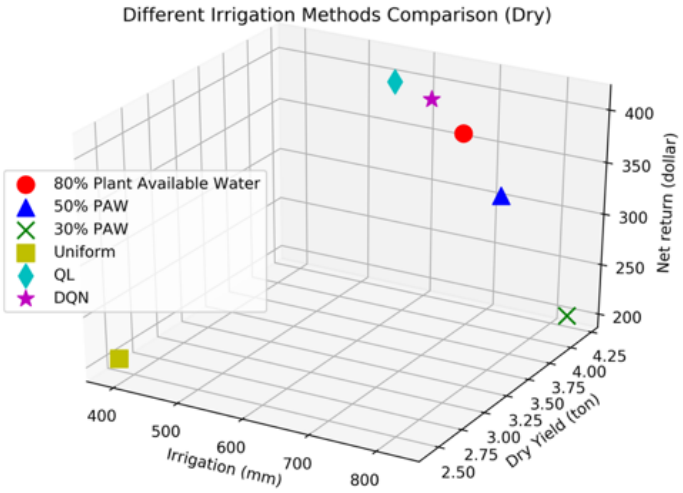

Figure 5.7: Comparison of different irrigation methods under dry weather condition.

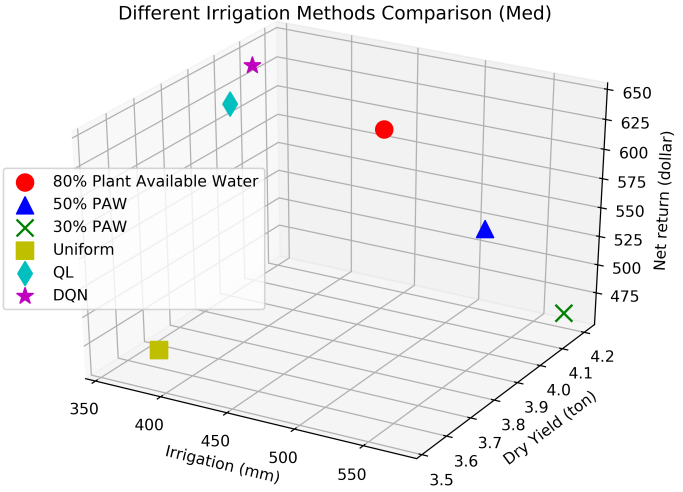

Figure 5.8: Comparison of different irrigation methods under moderate weather condition.

condition, respectively. From the figures, one can see that the proposed deep reinforcement learning (DQN) irrigation approach outperforms the other approaches for moderate and wet weather conditions and is very close to the best for dry weather condition.

Simulations using different crops mimic the same results as that of the simulations of the differing weather conditions. However, instead of changing weather conditions, we changed crop types

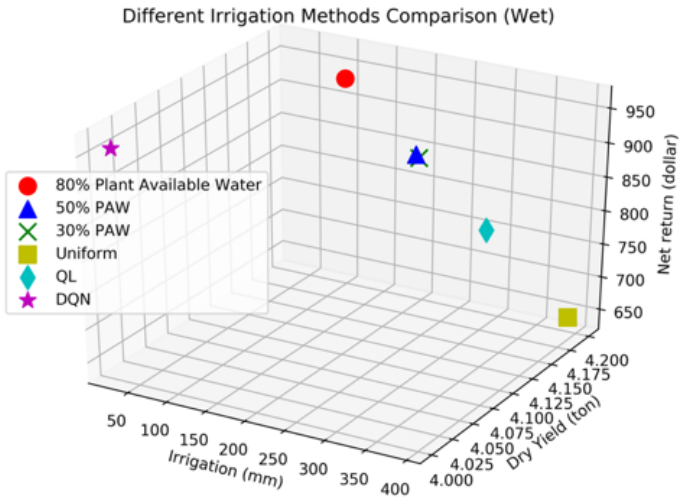

Figure 5.9: Comparison of different irrigation methods under wet weather condition.

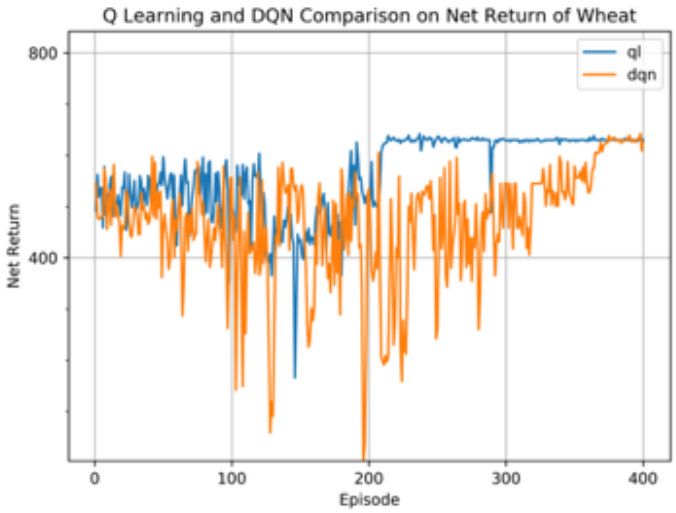

Figure 5.10: Learning curves for Q learning and Deep Q networks (Wheat).

while keeping the weather condition unchanged. The simulations were conducted on wheat, corn and soybean crops. Figure 5.10, Figure 5.11, and Figure 5.12 show the learning curves for both Q learning and deep Q networks, which indicate the learning process. Both methods converge to some values after a certain number of iterations of training. As shown in Table 5.6, Figure 5.13, Figure 5.14, and Figure 5.15, the proposed DQN method obtains highest net returns for corn and

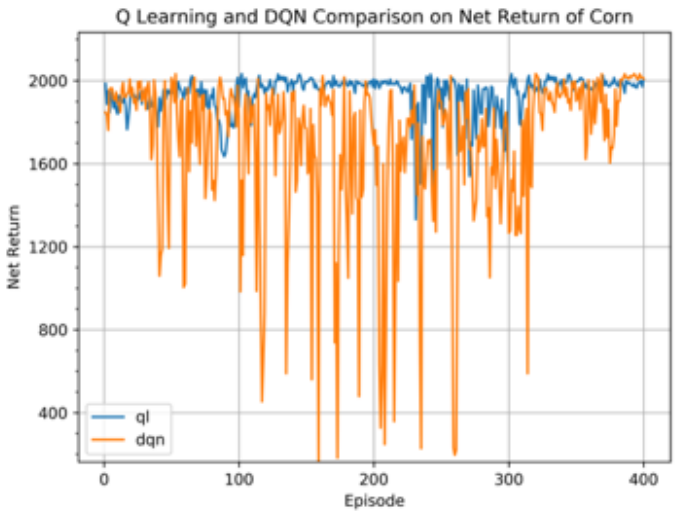

Figure 5.11: Learning curves for Q learning and Deep Q networks (Corn).

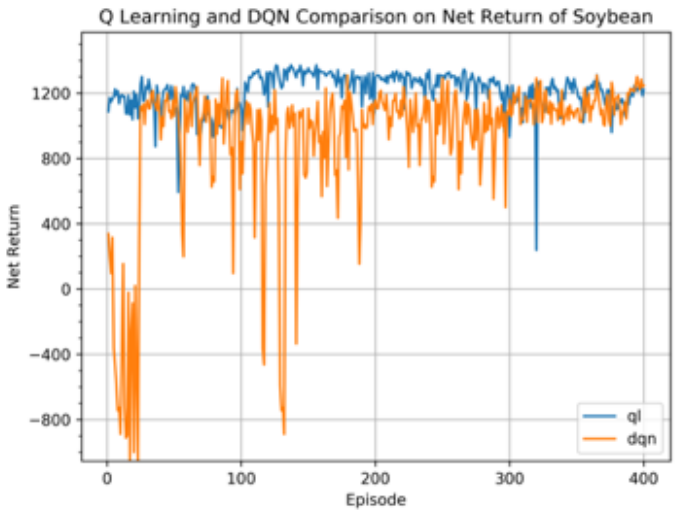

Figure 5.12: Learning curves for Q learning and Deep Q networks (Soybean).

wheat and net return for soybean is very close to the largest value.

5.3.2 Results for Automated Scheduling Method

The proposed scheduling method automates complex, multi-zone irrigations, and balances the water needs among the zones. This means within the same given irrigation time, the proposed method can maintain the irrigation requirement of all zones in an acceptable watered condition,

Table 5.6: Comparison of Different Irrigation Methods on Different Crop Types.

Irrigation Method	Crop Types	Dry Yield (ton)	Irrigation (mm)	Net Return (dollar)
DQN	Wheat	4.068	358	642
	Corn	10.79	286	2033
	Soybean	4.531	449	1301
Q Learning	Wheat	3.979	346	632
	Corn	10.741	301	2007
	Soybean	4.636	546	1244
80% PAW allowable depletion	Wheat	4.182	437	591
	Corn	10.506	311.1	1947
	Soybean	4.653	469	1328
50% PAW allowable depletion	Wheat	4.192	511.8	520
	Corn	10.656	432.8	1857
	Soybean	4.691	568.4	1243
30% PAW allowable depletion	Wheat	4.199	572.5	460
	Corn	10.774	536.2	1780
	Soybean	4.705	712	1105
Fixed irrigation	Wheat	3.528	390	477
	Corn	10.659	390	1901
	Soybean	3.167	390	833

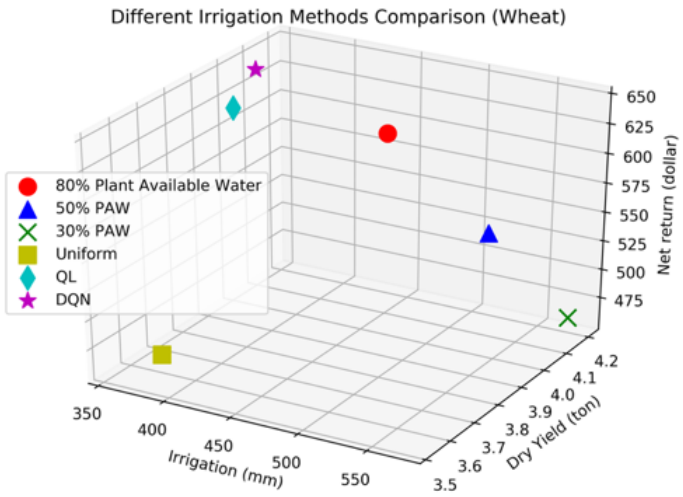

Figure 5.13: Comparison of different irrigation methods (Wheat).

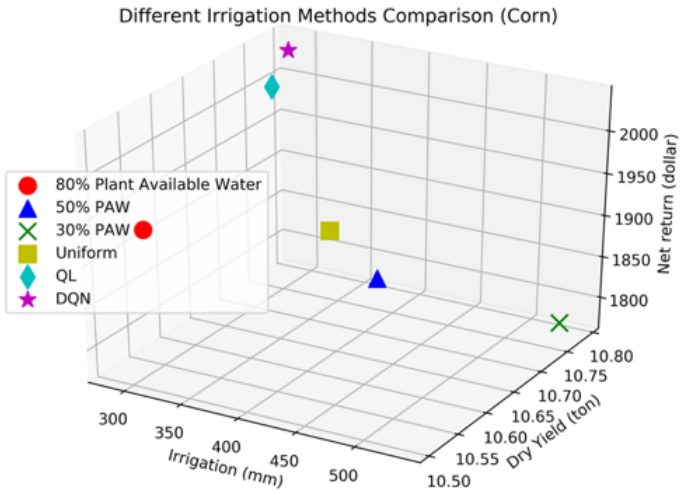

Figure 5.14: Comparison of different irrigation methods (Corn).

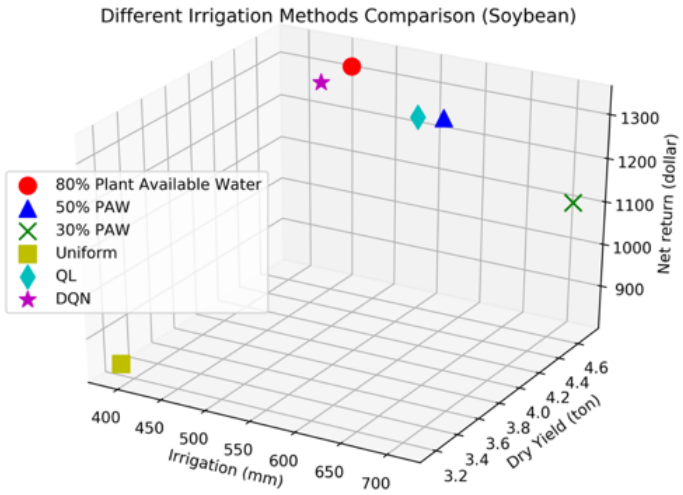

Figure 5.15: Comparison of different irrigation methods (Soybean).

while the other methods may fail to maintain an adequate soil water content for some zones. To further evaluate the outlined scheduling process, the proposed method is compared with a naïve scheduling approach which is close to manual computation. The naïve scheduling operates the maximum number of zones that can be operated during a period and schedules a constant time irrigation. This constant time is related to the total number of irrigation management zones, and

the maximum number of zones that can operate at a time plus the minimum time interval between operations. An intuitive example is shown in the Figure 5.16.

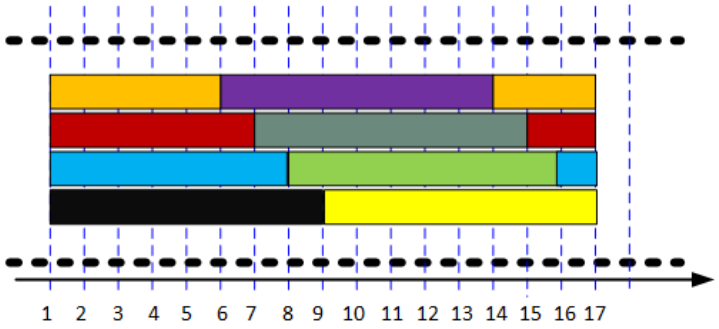

Figure 5.16: Naïve scheduling for the irrigation under the system hydraulic constraints (total 8 zones and need to run 4 at a time). Different colors correspond to different irrigation tasks of different zones. The rectangle indicates start/end time and the duration of the irrigation.

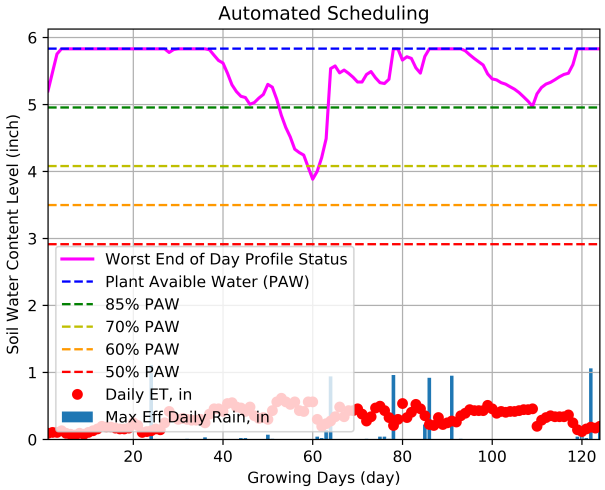

Figure 5.17: Soil water content level during the entire growing season under the automated schedule.

Figure 5.17 and Figure 5.18 show the simulation results and one can see that the proposed automated scheduling method outperforms the naïve method. It can better balance the water needs

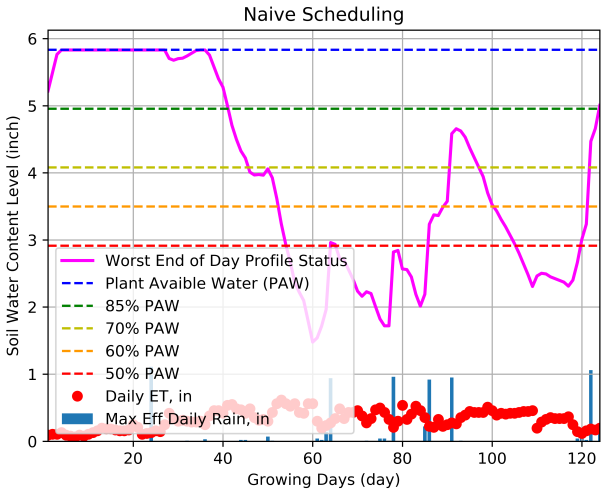

Figure 5.18: Soil water content level during the entire growing season under the naïve schedule.

Table 5.7: System constraints.

Field	Number of irrigation management zones	32
	Irrigated field acreage	10.6 acres
	Area per zone	0.3312 acre
Irrigation System Constraints	Well output	80 gpm
	Max line pressure	30 psi
	Desired operating tape pressure range	20 to 26 psi
	Application output rate per zone	11.4 gpm
	Max number of zones to operate at a time	8
Crop	Planting date	May 15, 2013
	Mature date	Sep 15, 2013
	Starting water level in the soil profile	4.93 inches
	Plant available water (PAW)	5.83 inches

among the respective zones and maintain a more agronomically desired overall root zone soil profile water status. This simulation configuration and input data are derived from actual operational field data collected at the Texas A&M AgriLife Research field site at Bushland, Texas. The detailed information of configuration is stated in Table 5.7. In the simulation, different zones may plant different crops thus having differing water needs. To simulate this scenario, 32 different crop ET data are required corresponding to the different zones and are generated from the historical

(actual) ET data multiplied a uniform distribution random number between 1 to 1.7.

5.4 Conclusion

In this work, a deep reinforcement learning irrigation control approach was proposed and evaluated. Results indicate it can accurately represent complicated irrigation context and determine precise water application amounts, thus achieving a high WUE. By combining with the AquaCrop model, it was shown to overcome insufficient data issues and enables the system to determine acceptable, profitable solutions while continuously learning during the production season. The evaluations of differing crop types and multiple weather conditions illustrate that the proposed method outperforms all other methods in most cases. To fully automate the irrigation process for micro-irrigation or other fixed-zone irrigation systems, an automated scheduling method was proposed. The automated scheduling method can replace manual computations that are required to determine start/stop times to provide the desired water application rate and timing for fixed-zone irrigation systems and can be more than 20 thousand times faster than manual computation. The irrigation scheduling can be done while protecting the system from large fluctuations in flow and pressure and assures system operation within the given hydraulic constraints (practical constraints of flows, pressures, system capacity, etc.). Additionally, it can handle many zones and balance the water needs among these multiple zones. The simulation results illustrate that the proposed automated scheduling outperforms the naïve automatic scheduling method.

6. CONCLUSIONS AND FUTURE WORKS

Agricultural irrigation plays a pivotal role in saving fresh water and determining crop yield. Precision irrigation is to irrigate an exact amount of water that crops need, no more and no less. To achieve this goal, researchers have spent a lot of efforts. Yet, many of them are restricted to construction and use of wireless sensor networks. Some of them consider algorithmic approaches to optimize water application, but they are either oversimplified or hard to implement and program in an embedded system board. In this dissertation, we presented methods to improve the reliability of wireless soil moisture sensor networks and developed real soil moisture sensor box deployed in the field at Bushland, Texas. We also apply modern machine learning techniques to achieving fully automated and optimized irrigation.

Accurate soil moisture measurement is extremely important in the proper management of an automated irrigation system. Ideally, the scheduling method implemented is based on the input of real-time soil moisture data. A sensor placement method is studied with a high degree of reliability and fault-tolerance and is based upon soil information (soil physical characteristics, spatial soil distribution), sensor performance, and crop-based information. The goal is to solve for an optimal or near-optimal solution in regards to a highly informative and fault-tolerant placement configuration with reasonable cost and effort. The algorithm finds the placement solution as to how many sensors are needed and locations as to where the sensors should be deployed. The concept of target is to select a small number of potential locations to deploy soil sensors where the sensors will acquire the most descriptive and representative locational information and provide reliable data for the long run.

A deep reinforcement learning based irrigation is proposed in Chapter 5. This approach can not only automate the irrigation process but also achieve highly precise water application that results in high water use efficiency. By utilizing this approach, the irrigation controller can automatically determine the optimal or near optimal water application for individual irrigation zone management in a multi-zone system. Traditional reinforcement learning can be superior to traditional periodic

and threshold-based irrigation scheduling. However, traditional reinforcement learning still fails to accurately represent a real-world irrigation environment due to its limited state space. Thus, the traditional reinforcement learning approach may neglect important information that affects actual water demand by plants/crops. Therefore, the traditional reinforcement learning approach is limited in scope and does not result in an optimal or near optimal solution. Compared with the traditional reinforcement learning approach, the deep reinforcement learning method can better model a real-world environment based on multi-dimension observations. By utilizing this method, an irrigation system can accurately and precisely respond to its environment. To fully automate multiple zones irrigation, a scheduling method for micro-irrigation and other fixed multi-zone irrigation systems is developed. It has the ability to prevent water hammer effect and assures that a system operates under a set of user-defined hydraulic constraints. In addition, because of the automated scheduling, the total labor cost can be considerably less than that from solutions based on manual computation and control, not to mention the time saving and reduced errors. The input required for this method is the water amount to be applied for each irrigation zone. The method gives a higher priority to the zone where irrigation is most urgent. As the amount of irrigation water is optimized, the method can achieve highly precise irrigation control.

This dissertation introduced how to build reliable and robust sensing system and how to apply state-of-the-art deep reinforcement learning algorithm to agricultural irrigation. There is still a long way to go. Some potential future works are stated as follows:

- We proposed a way to deploy soil moisture sensors in the field to achieve high reliability. In this method, we assume that we know the most representative points and they are the central points of management zones. However, in reality, this is not always true. Therefore, how to find the most representative points still remains unknown.

- In this dissertation, we proposed an automatic scheduling method and it is able to arrange irrigation tasks of a large number of zones. This method can fully automate the scheduling process and thereby reduce time and labor cost. It tries to balance the water needs of multiple zones. Yet, this is not always the right thing to do. In some cases, focusing just one or some

zones and ignoring the rest zones will result in higher net return. Therefore, we need to further decide to balance the water needs of zones or focus on a certain number of zones. This is a difficult trade-off problem and related to crops and their prices.

- One limitation of our deep reinforcement learning irrigation control is that it heavily hinges on a crop model. Typically, it takes several months to get crop yield data. Therefore, it is very hard for us to get enough data to train the deep reinforcement learning model. We have to rely on some mathematic crop model that can simulate crop growth in a very short time. Therefore, the accuracy of the proposed deep reinforcement leaning irrigation is limited by the accuracy of the crop model. One way to solve this problem is imitation learning. The idea of imitation learning is giving an agent prior information about how to irrigate correctly by mimicking a human expert. This will not only help us solve the limited data issue but also make the training process faster and safer.

- Internet of Things plays a pivotal role in the smart irrigation system. Typically, the IoT device has limit energy consumption, computing capacity, and memory, making it hard to apply traditional security countermeasures. Moreover, the architectures of IoT systems are highly dynamic and heterogeneous. Plus, usually, the IoT system is highly scalable and physically unprotected. Thus, the IoT system is quite vulnerable and facing a lot of security issues. For our current irrigation system, it contains two different networks, ZigBee networks and WiFi. For ZigBee wireless sensor network, there are three potential targets for attacks including sensors, ZigBee network connections, and micro-controllers. The possible attacks include taking control of existing sensor nodes, sending fake sensing data, sniffing sensing data, adding fake nodes, disconnecting sensor nodes, and physical attacks. If the confidentiality and integrity of the sensing data cannot be guaranteed, there is nothing to talk about the benefits of the IoT based irrigation system. One good thing is that ZigBee communication has limit communication range, not like the Internet, the attackers can sit anywhere to do attacks, they have to be within a certain range of our system. For a WiFi network, because it is part of

our TAMU network system, it requires a higher security level. The possible attacks include disconnecting the system from the Internet, sniffing irrigation status, modifying status and sensing data, taking control of the controller, and physical attacks.

This dissertation has provided methods for reliable soil moisture sensing and smart irrigation control that will be helpful to enhance the water use efficiency, save labor cost, and increase the net return. And, despite the many commercialized smart irrigation systems, a lot of interesting problems in adaptive and learning irrigation control remain. Since irrigation is closely related to crop quantity and quality and water savings, we believe that further efforts to study these problems will contribute to our society.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] E. A. Ross and L. A. Hardy, "Irrigation: national engineering handbook." `https://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=17837.wba`, 1997. Accessed: 2018-09-30.

[3] S. Ji, S.-F. Yuan, T.-H. Ma, and C. Tan, "Distributed fault detection for wireless sensor based on weighted average," *International Conference on Networks Security Wireless Communications and Trusted Computing*, vol. 1, pp. 57–60, 2010.

[4] L. Sun, Y. Yang, J. Hu, D. Porter, T. Marek, and C. Hillyer, "Reinforcement learning control for water-efficient agricultural irrigation," *IEEE International Conference on Ubiquitous Computing and Communications*, pp. 1334–1341, 2017.

[5] Y. Kim, R. G. Evans, and W. M. Iversen, "Remote sensing and control of an irrigation system using a distributed wireless sensor network," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 7, pp. 1379–1387, 2008.

[6] Y. Kim, R. Evans, and W. Iversen, "Evaluation of closed-loop site-specific irrigation with wireless sensor network," *Journal of Irrigation and Drainage Engineering*, vol. 135, no. 1, pp. 25–31, 2009.

[7] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1448–1453, 2002.

[8] R. Akhlaghinia, S. Hashemi, and B. Shadgar, "Sensor placement for heterogenous point coverage," *International Conference on Computer and Network Technology*, pp. 13–17, 2010.

[9] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," *International Conference on Very Large Data Bases*,

pp. 588–599, 2004.

[10] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, pp. 235–284, 2008.

[11] X. Wu, M. Liu, and Y. Wu, "In-situ soil moisture sensing: Optimal sensor placement and field estimation," *ACM Transactions on Sensor Networks*, vol. 8, no. 4, pp. 33–33, 2012.

[12] S. K. Ooi, I. Mareels, N. Cooley, G. Dunn, and G. Thoms, "A systems engineering approach to viticulture on-farm irrigation," *International Federation of Automatic Control Proceedings Volumes*, vol. 41, no. 2, pp. 9569–9574, 2008.

[13] Z. Li, N. Wang, T. Hong, A. Franzen, and J. Li, "Closed-loop drip irrigation control using a hybrid wireless sensor and actuator network," *Science China Information Sciences*, vol. 54, no. 3, pp. 577–588, 2011.

[14] C. M. Angelopoulos, S. Nikoletseas, and G. C. Theofanopoulos, "A smart system for garden watering using wireless sensor networks," *ACM International Symposium on Mobility Management and Wireless Access*, pp. 167–170, 2011.

[15] L. Pfitscher, D. Bernardon, L. Kopp, M. Heckler, J. Behrens, P. Montani, and B. Thome, "Automatic control of irrigation systems aiming at high energy efficiency in rice crops," *International Caribbean Conference on Devices, Circuits and Systems*, pp. 1–4, 2012.

[16] S. A. O'Shaughnessy, S. R. Evett, P. D. Colaizzi, and T. A. Howell, "Grain sorghum response to irrigation scheduling with the time-temperature threshold method and deficit irrigation levels," *Transactions of the American Society of Agricultural and Biological Engineers*, vol. 55, no. 2, pp. 451–461, 2012.

[17] S. A. O'Shaughnessy, S. R. Evett, P. D. Colaizzi, and T. A. Howell, "Wireless sensor network effectively controls center pivot irrigation of sorghum," *Applied Engineering in Agriculture*, vol. 29, no. 6, pp. 853–864, 2013.

[18] A. C. McCarthy, N. H. Hancock, and S. R. Raine, "Simulation of irrigation control strategies for cotton using model predictive control within the variwise simulation framework," *Computers and Electronics in Agriculture*, vol. 101, pp. 135–147, 2014.

[19] C. Lozoya, C. Mendoza, L. Mejía, J. Quintana, G. Mendoza, M. Bustillos, O. Arras, and L. Solís, "Model predictive control for closed-loop irrigation," *International Federation of Automatic Control Proceedings Volumes*, vol. 47, no. 3, pp. 4429–4434, 2014.

[20] L. Maton, D. Leenhardt, M. Goulard, and J. Bergez, "Assessing the irrigation strategies over a wide geographical area from structural data about farming systems," *Agricultural Systems*, vol. 86, no. 3, pp. 293–311, 2005.

[21] H. Krupakar and A. Jayakumar, "A review of intelligent practices for irrigation prediction," *arXiv preprint arXiv:1612.02893*, 2016.

[22] Y. Zhao, C. Bai, and B. Zhao, "An automatic control system of precision irrigation for city greenbelt," *IEEE International Conference on Industrial Electronics and Applications*, pp. 2013–2017, 2007.

[23] K. L. Moore and Y. Chen, "Iterative learning control approach to a diffusion control problem in an irrigation application," *International Conference on Mechatronics and Automation*, pp. 1329–1334, 2006.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *Neural Information Processing Systems Deep Learning Workshop*, 2013.

[25] T. Muhammed and R. A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 78, pp. 267–287, 2017.

[26] S. Guo, H. Zhang, Z. Zhong, J. Chen, Q. Cao, and T. He, "Detecting faulty nodes with data errors for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 3, pp. 40–40, 2014.

[27] X. Jin, T. W. Chow, Y. Sun, J. Shan, and B. C. Lau, "Kuiper test and autoregressive model-based approach for wireless sensor network fault diagnosis," *Wireless Networks*, vol. 21, no. 3, pp. 829–839, 2015.

[28] M.-H. Lee and Y.-H. Choi, "Fault detection of wireless sensor networks," *Computer Communications*, vol. 31, no. 14, pp. 3469–3475, 2008.

[29] Z. Feng, J. Q. Fu, and Y. Wang, "Weighted distributed fault detection for wireless sensor networks based on the distance," *Chinese Control Conference*, pp. 322–326, 2014.

[30] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Robust sensor placements at informative and communication-efficient locations," *ACM Transactions on Sensor Networks*, vol. 7, no. 4, pp. 31–31, 2011.

[31] H. Guo, L. Zhang, L. Zhang, and J. Zhou, "Optimal placement of sensors for structural health monitoring using improved genetic algorithms," *Smart Materials and Structures*, vol. 13, no. 3, pp. 528–528, 2004.

[32] S. Spanache, T. Escobet, and L. Travé-Massuyès, "Sensor placement optimisation using genetic algorithms," *International Workshop on Principles of Diagnosis*, pp. 179–184, 2004.

[33] K. S. Yildirim, T. E. Kalayci, and A. Ugur, "Optimizing coverage in a k-covered and connected sensor network using genetic algorithms," *World Scientific and Engineering Academy and Society International Conference on Evolutionary Computing*, pp. 21–26, 2008.

[34] G. Mathereon, "La théorie des variables régionalisées et ses applications," *Cah Cent Morphol Math*, vol. 5, pp. 1–212, 1970.

[35] D. G. Krige, "A statistical approach to some basic mine valuation problems on the witwatersrand," *Journal of the Southern African Institute of Mining and Metallurgy*, vol. 52, no. 6, pp. 119–139, 1951.

[36] G. Bohling, "Introduction to geostatistics and variogram analysis," *Kansas Geological Survey*, pp. 1–20, 2005.

[37] V. Pandey and P. K. Pandey, "Spatial and temporal variability of soil moisture," *International Journal of Geosciences*, vol. 1, no. 2, pp. 87–87, 2010.

[38] F. Capraro, D. Patino, S. Tosetti, and C. Schugurensky, "Neural network-based irrigation control for precision agriculture," *IEEE International Conference on Networking, Sensing and Control*, pp. 357–362, 2006.

[39] H. Navarro-Hellan, J. Martinez-del Rincon, R. Domingo, F. Soto-Valles, and R. Torres, "A decision support system for managing irrigation in agriculture," *Computers and Electronics in Agriculture*, vol. 124, pp. 121–131, 06 2016.

[40] M. E. Jensen, "Scheduling irrigation with computers," *Journal of Soil and Water Conservation*, vol. 24, no. 5, pp. 193–195, 1969.

[41] T. A. Howell, *Optimization of grain sorghum water use efficiency under high frequency irrigation by system simulation and stochastic dynamic programming*. Ph.D. dissertation, College Station, Texas, USA, 1974.

[42] B. J. D. Sunantara and J. A. Ramirez, "Optimal stochastic multicrop seasonal and intraseasonal irrigation control," *Journal of Water Resources Planning and Management*, vol. 123, no. 1, pp. 39–48, 1997.

[43] R. Wardlaw and K. Bhaktikul, "Application of genetic algorithms for irrigation water scheduling," *The Journal of International Commission on Irrigation and Drainage*, vol. 53, no. 4, pp. 397–414, 2004.

[44] N. Schutze and G. Schmitz, "Neuro-dynamic programming as a new framework for decision support for deficit irrigation systems," *International Congress on Modelling and Simulation*, pp. 2271–2277, 2007.

[45] J. Jones, "Decision support systems for agricultural development," *Systems Approaches for Agricultural Development*, pp. 459–471, 1993.

[46] R. J. Stirzaker, "When to turn the water off: scheduling micro-irrigation with a wetting front detector," *Irrigation Science*, vol. 22, no. 3-4, pp. 177–185, 2003.

[47] T. A. Howell and M. Meron, "Irrigation scheduling," *In Developments in Agricultural Engineering*, vol. 13, pp. 61–130, 2007.

[48] X. C. Yuan, Y. W.and Zhang and H. P. Zhao, "When to turn the water off: scheduling micro-irrigation with a wetting front detector," *IEEE International Conference on Computer-Aided Design and Computer Graphics*, pp. 618–622, 2009.

[49] E. Vanuytrecht, D. Raes, P. Steduto, T. C. Hsiao, E. Fereres, L. K. Heng, M. G. Vila, and P. M. Moreno, "Aquacrop: Fao's crop water productivity and yield response model," *Environmental Modelling & Software*, vol. 62, pp. 351–360, 2014.

[50] L.-J. Lin, *Reinforcement learning for robots using neural networks*. Ph.D. dissertation, Pittsburgh, PA, USA, 1992.

[51] S. Zhang and R. S. Sutton, "A deeper look at experience replay," *Neural Information Processing Systems Deep Reinforcement Learning Symposium*, December 2017.

[52] C. J. C. H. Watkins, *Learning from delayed rewards*. Ph.D. dissertation, Cambridge, England, 1989.