University of California
Merced

Modeling and Optimization for Irrigation Control Using Wireless Sensor Networks

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering and Computer Science

by

Daniel A. Winkler

2019

The dissertation of Daniel A. Winkler is approved.

François Blanchette

Wan Du

Miguel Á. Carreira-Perpiñán

Alberto E. Cerpa, Committee Chair

University of California, Merced

2019

Dedicated to the friends and family who supported me through this adventure.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank my advisor Alberto Cerpa for his guidance both before and during my graduate studies. Without his support, my progress in this endeavor would not have been possible. In addition, I would like to thank my committee members Miguel Á. Carreira-Perpiñán, Wan Du, and François Blanchette for their insight across my various stages of research. Next, I must thank my many past and current colleagues, including Varick Erickson, Tao Liu, Alex Beltran, Niloofar Piroozi, Ashish Yadav, Claudia Chitu, and the undergraduates who have helped along the way. Finally, I want to thank Jim Brugger and the facilities management personnel who made my research possible.

Vita

2013–2019	Graduate Student Researcher, University of California, Merced
2013–2019	Teaching Assistant, University of California, Merced
2009–2013	B.S. Computer Science Engineering, University of California, Merced.

Publications

D. A. Winkler, M. A. Carreira-Perpiñán and A. E. Cerpa, "Plug-and-Play Irrigation Control at Scale", IPSN 2018

D. A. Winkler, A. Beltran, N. P. Esfahani, P. P. Maglio, and A. E. Cerpa., "FORCES: feedback and control for occupants to refine comfort and energy savings", UbiComp 2016

D. A. Winkler, R. Wang, F. Blanchette, M. A. Carreira-Perpiñán and A. E. Cerpa, "MAGIC: Model-Based Actuation for Ground Irrigation Control" IPSN 2016

(Poster) **D. A. Winkler**, R. Wang, F. Blanchette, M. A. Carreira-Perpiñán, A. E. Cerpa, "MICO: Model-Based Irrigation Control Optimization", SenSys 2015

D. A. Winkler, A. E. Cerpa, "Distributed independent actuation for irrigation control", BuildSys 2014

(Demo) V. L. Erickson, A. Beltran, **D. A. Winkler**, N. Esfahani, J. Lusby, A. E. Cerpa, "ThermoSense: Thermal Array Sensor Networks in Building Management", SenSys 2013

Abstract of the Dissertation

Modeling and Optimization for Irrigation Control Using Wireless Sensor Networks

by

Daniel A. Winkler

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Merced, 2019

Professor Alberto E. Cerpa, Chair

Lawns, also known as turf, cover an estimated 128,000km^2 [nas] in North America, consuming an estimated 7 billion gallons of freshwater each day. Despite recent developments in irrigation control and sprinkler technology, state-of-the-art irrigation systems are unable to consider localized water requirements across the irrigation system and deliver localized control, preventing efficient irrigation. Inspired by preliminary results in simulation, we introduce a distributed irrigation controller, allowing us to sense moisture data across the space, actuate each sprinkler independently, and perform computation in a distributed way. To efficiently schedule irrigation for these distributed devices, we introduce modeling techniques allowing us to predict future water movement through the space caused by runoff, leaching, and weather effects that will affect the moisture in the system. These models are then used as constraints in optimization to choose schedules for the distributed valves that minimize system water consumption while maintaining optimal plant health. Finally, we show through extensive deployment side by side with state-of-the-art control strategies that our proposed systems are capable of providing significant water savings while simultaneously providing a

higher quality of service to the turf compared to the baselines. Furthermore, we find that through clever system design we can achieve a perpetual system lifetime and virtually eliminate manual system configuration requirements, allowing us to bridge the technology gap to the end user to vastly improve system adoptability. In this way, we demonstrate the feasibility of wireless sensor use in turf irrigation systems.

CHAPTER 1

Introduction

Public and private lawns, also known as turf, make up the single largest irrigated crop by surface area in North America, covering approximately 128,000 km^2[nas]. In 2015, it was estimated that lawns consumed 9 billion gallons of fresh water each day for irrigation [epac]. As only 1% of water on the planet's surface is estimated to be fresh and available for our use [fre], there is significant environmental, economical, and political motivation to improve the efficiency of the irrigation systems.

We wish to improve the efficiency of turf irrigation systems, but as the primary purpose of turf installations is to be aesthetically pleasing, we must maintain a high quality of service to the plant. If we do not deliver sufficient water to the plant, it will eventually wilt and die. However, over-irrigation also has consequences. In addition to wasted water, over-irrigation over time will cause discoloration of the plant, erosion of the soil, and in extreme cases can carry fertilizer chemicals beyond the root zone of the plant and into the drinking water supply, as occurred in California's Salinas Valley and Tulare Lake Basin, investigated by [HL12]. To achieve our goal of reducing water consumption and maintaining a high quality of service, we must apply irrigation exactly where it is needed across the irrigated space.

To apply the correct amount of water to the surface, our irrigation controller must meet three requirements. (1) We must be able to track soil water conditions

across the irrigated space for system monitoring and as feedback for control. (2) We must be able to irrigate water where it is needed, not possible in standard irrigation systems due to centralized valve locations. (3) As water moves across and through the soil, we must be able to model and predict this movement beforehand, in order to apply it to the surface efficiently.

1.1 Proposed Contributions

In my thesis, I intend to demonstrate that use of wireless sensor networks can solve the ongoing limitations of turf irrigation systems, feasible not only in a research context but also as a potentially commercialized system. The primary contributions are as follows:

- We investigate the weaknesses of centralized actuation in standard turf irrigation systems, and introduce a wireless distributed sensor/actuator to allow significantly finer granularity of control for more efficient and higher-quality service to the turf.
- To prepare for or take advantage of future weather conditions when making control decision, we develop a method of predicting evapotranspiration losses in the near future using commonly-available weather predictions and historical data trends.
- We introduce and evaluate three methods of computing valve scheduling for distributed actuation by coupling fluid movement modeling techniques and optimization to find schedules that minimize system water consumption while maintaining quality of service constraints for the plant.
- Finally, we design a fully distributed irrigation controller, allowing the sys-

tem to control irrigation systems at any scale with robust system operation. In doing so, we design a distributed node which takes advantage of a co-processor architecture to enable significant computation and storage capabilities with the best-case energy consumption of a standard WSN device, and leverage energy harvesting to allow the system to operate indefinitely.

1.2 Proposal Organization

Chapter 2 discusses related work in the fields of irrigation control in Section 2.1, methods of plant health monitoring in Section 2.2, techniques for fluid modeling in Section 2.3, and others.

Chapter 3 investigates the weaknesses of the standard centrally-actuated irrigation system architecture, and hypothesizes the benefits of controlling at a much finer spatial granularity using distributed wireless sensor network actuators, the first work to do so. To quantify the effect such a system would have on system operation, we develop a model of water movement through the irrigated space using a cellular automata model that considers sprinkler fluid distribution, downhill surface runoff and infiltration subject to estimated soil flow rate, available soil storage, etc. This model is then used as a black-box for an optimization problem to find schedules for the distributed actuators that minimize system water consumption while maintaining minimum soil water levels across the space required for plant health. While the optimization of the proposed model was found to be too slow to control a large-scale irrigation system, results in simulation suggest that the proposed distributed actuation significantly reduce system water consumption in comparison to the centrally-controlled system.

Based on the promising results in Chapter 3, Chapter 4 introduces the DIC-

TUM system, aiming to improve the modeling and optimization techniques and put the system into practice to control a large-scale irrigation system. First, we describe the development of the wireless sensor/actuator node to be installed on each sprinkler. These devices have a solenoid to control a flow of water to each sprinkler, a soil moisture sensor to monitor moisture levels across the space, and wireless communication capabilities to transmit data with sister nodes and a basestation. We then propose a PDE model of water movement, built from first principles to be significantly more representative of the physical space than the cellular automata model of Chapter 3. This model describes all of the short-term water movement effects that occur during irrigation; application of sprinklers, surface runoff and diffusion, absorption into the soil, and sub-surface downhill flow and diffusion. Whereas the model in Chapter 3 was used as a black box in optimization, the DICTUM system uses the PDE model as constraints to an optimization problem to improve optimization performance. As using the non-linear PDE model as constraints will cause our optimization to be non-convex and introduce local optima as well as increasing time to solve optimization, we choose to first simplify this model by discretizing and linearizing it, resulting in a linear program that will be easier to solve in practice. Optimized schedules are then transmitted to their respective actuator device to be run. To test this system, we launch two side-by-side irrigation systems covering a total area of $\sim$10,000 ft^2 for 4 weeks, and find that water savings in the order of 10-25% are possible compared to existing control strategies, while simultaneously improving quality of service significantly.

Although the DICTUM system introduced in Chapter 4 had significant improvements on our primary metrics of success, it had practical limitations as an irrigation control strategy at scale. The proposed PDE model required manual parameter definition, which either requires broad assumptions in the homogene-

ity of the space which reduces model accuracy, or requires significant time and expense to measure these parameters across the space. In addition, we recognize that while the PDE model provides valuable information about water movement during irrigation, performance limitations prevent it from being useful to predict movement across the full 24-hour cycle, which can potentially cause reduced quality of service to the turf. In response to these limitations, Chapter 5 proposes data-driven modeling techniques across two time horizons, short-term and long-term. The long-term model describes how each individual location across the space tends to lose its moisture *between* irrigation cycles, and is used in conjunction with a weather prediction method we introduce to choose a goal state of moisture that should be reached by the end of irrigation to satisfy plant moisture requirements across the full 24-hour cycle. In addition, we employ a short-term model to describe how actuation of sprinklers will affect the soil moisture across the entire space. The short-term model is then used in optimization to choose sprinkler schedules to guide soil moisture towards our goal-state of moisture, computed using the long-term model, in the most efficient way possible. We find across two large deployments that while the proposed OPTICS system only slightly reduces water consumption in comparison to the DICTUM system, its ability to better estimate long-term losses between irrigation cycles allows the OPTICS system to provide improved quality of service. At the same time, the OPTICS data-driven system requires virtually no human input, automatically adjusts in response to changing and heterogeneous environmental conditions, and is capable of scaling to control very large irrigation systems.

Finally, Chapter 6 considers the limitations of the *centralized* decision-making of the previous chapters. Under centralized control, data for model generation and decision-making must be constantly moved across the network to and from a single controller, causing increased radio consumption across the network and

a single point-of-failure as discussed in Chapter 5. In addition, as the OPTICS model for short-term water movement is global, considering the effect of every sprinkler in the space on every sensor in the space, significant processing time is needlessly performed due to each sprinkler's limited area of influence. We recognize that these limitations make this application a prime candidate for a *distributed control system* system that we call WISDOM, in which different regions of the irrigation systems are able to make their own control decisions based on the conditions within a local neighborhood. As all low-power platforms within the WSN domain are severely restricted in terms of memory and computation, we design a wireless sensor node with a co-processor architecture that allows truly low-power operation when the system is sitting idle, and significant computation and storage capabilities during irrigation, allowing model training, optimization, and any future system requirements. To ameliorate the additional power consumption of the distributed device, we recognize that each node is collocated with a sprinkler, and equip each with a micro hydro turbine and charge controller to allow battery recharge during irrigation. To determine the WISDOM system operation on our end-to-end metrics introduced in previous chapters, we launch a 4-week system deployment and compare to the evapotranspiration and OPTICS systems to find that while we provide the added benefits of scalability, robustness, and perpetual operation, our system loses no performance in comparison to the OPTICS system. To determine whether the use of localized models results in less efficient schedules, we simulate an extremely large-scale irrigation system in several configurations to compare the schedules produced using global and local modeling, and demonstrate that in ordinary irrigation system and environmental characteristics, the WISDOM system will find identical schedules to the OPTICS system, despite drastic reduction in communication energy costs in the network.

CHAPTER 2

Related Work

2.1 Irrigation control strategies

In recent years, several advancements in irrigation control strategies have come forth to improve the operation of turf irrigation systems, but the vast majority of irrigation systems remain under antiquated control, with manually-defined valve scheduling, for which best practices are suggested by the Environmental Protection Agency [epaa, epab, epad]. These industry-standard irrigation systems suffer from two primary limitations, in the architecture of the irrigation system itself and in the control routine used to generate schedules for it.

The fundamental limitation of irrigation system architecture is in their use of centrally-located water valves. In these systems, it is common for a single valve to control tens of sprinklers, meaning if the valve is enabled, water will flow to all of the attached sprinklers. In practice, however, the irrigation needs of the soil are highly heterogeneous across an irrigation system. Changes in surface topography and solar exposure will affect how much of the irrigated water is lost to evaporation or runoff, and differences in soil type and depth will affect how much water must be absorbed to reach a particular level of moisture, all of which can vary significantly across an irrigated space. With centrally-located water valves, a single area within the turf requiring more water will force the entire irrigation system to be actuated extra. This commonly results in substantial overwatering

of the space, even with the most intelligent control strategy deciding the schedules run by the irrigation valves. The EPA best-practices guides recommend that in large deployments where more than one valve is required due to system pressure constraints, the individual valve schedules should be tailored to suit the different requirements of the irrigated regions. However, these systems are still highly centralized limiting the potential water savings, and this level configuration requirement is often ignored due to the additional burden it puts on the groundskeeper.

In addition to the physical limitations of the irrigation system, the majority of these systems employ a basic trial-and-error control strategy based on static schedule configuration. A groundskeeper will make an initial guess as to the irrigation needs of the space in length of irrigation time required, and set the irrigation system to irrigate for this length of time periodically. As time passes, the groundskeeper will visually inspect the health of the turf that has resulted from this irrigation. Any indication of dryness due to browning of the plant or wetness due to puddling or discoloration will be noted, and the irrigation period will be adjusted accordingly. This technique has 4 primary disadvantages: (1) As the groundskeeper's primary concern is the aesthetic quality of the turf and the worst aesthetic penalties occur when the grass does not receive enough water, the groundskeeper will almost always over-water intentionally to maintain sufficient moisture, resulting in the irrigation system consuming more water than necessary. (2) This manual re-configuration of irrigation schedules is highly imprecise and subjective as slow plant reaction to moisture conditions makes a poor visual feedback loop. (3) As these groundskeepers are often responsible for many irrigated areas and/or busy performing other duties, in practice this adjustment occurs rarely if at all. (4) Finally, due to long periods between adjustment, the trial-and-error method causes the statically-defined irrigation schedules to remain

unchanged across seasonal weather changes. Depending on the time of year, this often results in diminished quality of service due to underwatering or efficiency due to overwatering. The EPA best-practices guides all recommend that these systems be replaced with more sophisticated weather-based controllers so that manual scheduling is not required, but the simplicity, low cost and start-up time of these simple controllers make them the most commonly-used strategies to this day.

Both of these limitations affect the operation of these systems, so work has been done to correct these weaknesses. Although the physical limitations of these irrigation systems had not been addressed prior to our work, control techniques seeking to improve the operation of existing systems are discussed here.

2.1.1 Weather-based strategies

It is well understood that weather is a primary water source or sink in irrigation systems, and the most obvious water contributor is precipitation. In the trial-and-error control strategy mentioned above, the system has no understanding of weather, and will continue with its periodic irrigation even if rain is occurring, potentially causing significant waste of water. To prevent this source of waste, irrigation system manufacturers such as Hunter [raia] and Rain Bird [raic] have developed precipitation sensors that wire into the irrigation controller. Rain gathers within a sensor, and once it reaches a particular threshold, a signal is sent to the irrigation controller to disable irrigation until the moisture within the sensor has dried out. These systems do a good job to prevent the irrigation system from wasting water during precipitation, but the sensor itself still requires the user to manually calibrate the precipitation threshold at which irrigation is disabled. Set too low, the irrigation system may be disabled when irrigation is still

required, and set too high and water waste will occur. Although this system has clear benefits to system efficiency during precipitation, having a human directly in the loop is not desirable for these control strategies.

In addition to precipitation, there are other environmental factors that will influence the amount of water available within the soil for the turf to use. These factors are generally broken into two categories, evaporation and plant transpiration. Evaporation is the process in which the surface water is vaporized due to environmental conditions and transported away from the soil. Although it has the strongest effect on standing surface water, evaporation can also extract moisture from the soil that lies close to the boundary on the surface. Plant transpiration is the process in which plant roots extract water from the soil for use in its biological processes, and the amount of water uptake depends on how active the plant is. When a plant is young, the primary loss is caused by evaporation from solar radiance directly hitting the soil. As the plant grows taller and thicker and shades the soil, the majority of losses are from plant transpiration. Both evaporation and plant transpiration are directly affected by weather conditions, and so with the ability to estimate these losses based on measurable weather conditions, irrigation can then be scheduled in response.

To this end, work done in [JBA90] and [APR98] strived to find the relationship between weather conditions and the loss of water from the soil, with the latter known as the FAO-56 method now representing the standard. They develop a formulation that takes measured values of solar radiation, air temperature, humidity, and wind speed collected in controlled conditions, and outputs an estimated reference evapotranspiration value estimating the amount of water that has been lost subject to these environmental conditions. As evapotranspiration has become a common weather metric, many weather stations report the reference

evapotranspiration metric directly. For instance, the California Irrigation Management Information System [cim] provides an API for the retrieval of California weather data at 255 measuring sites across the state, many of which are available with evapotranspiration. In regions with no publicly-available weather station, an off-the-shelf evapotranspiration sensor such as Hunter's ET Sensor [hunb] or Weathermatic's weather sensors for the SmartLine irrigation controller [wea] can be locally installed.

An evapotranspiration irrigation controller begins operation by contacting the local weather station, which will provide the reference evapotranspiration losses. In our locale these losses are available at hourly, daily, monthly, or yearly intervals. Depending on the irrigation frequency configured into the irrigation controller, the losses will be summed since the time of last irrigation. As different types of plant provide different amounts of shade to the soil and require different amounts of water, they will result in different soil water losses even in identical environmental conditions. To correct for these variations, the reference evapotranspiration losses is then multiplied by a crop-specific evapotranspiration coefficient, enumerated for most standard crops including turf in [APR98]. This crop-specific estimate of water losses since the last irrigation period, in inches of water, can then be re-irrigated in a direct water replacement. Based on the known application rate of the sprinklers installed across the irrigation system in inches of water per minute, the lost water can be used to calculate the amount of time of irrigation required to replace the losses, and the irrigation system is activated.

Evapotranspiration controllers overcome some of the biggest weaknesses of the standard irrigation control, as they automatically adjust in response to changing weather conditions and require significantly less manual configuration of irriga-

tion schedules in comparison. Furthermore, as over-estimation is not needed, the marketing literature of these technologies boast a potential 30% of water savings in comparison to the standard trial-and-error irrigation control technique. As such, these systems are considered to be the best industry controllers available, and so we use the evapotransporation control strategy to compare to our proposed systems in Chapters 4 and 5. Despite the huge benefit of these controllers, this technique still has limitations. Although these controllers have knowledge of recent local weather patterns, they have no understanding of local environmental conditions causing surface runoff, leeching, site-specific changes in solar exposure due to tree occlusion or sloped land, etc. Without details of these conditions, the controller will be unable to take advantage of distributed sensing/actuating devices across the space, and will suffer from the efficiency and quality of service limitations of centralized valve control. Furthermore, as there is no sensor feedback in these systems to make adjustments, in practice these controllers irrigate an extra safety margin of water to ensure a high quality of service is provided across the space, causing additional water losses.

2.1.2 Sensor-based strategies

With recent improvements in soil moisture sensing technologies, some have suggested the use of sensors as primary input to an irrigation controller. These systems monitor the soil moisture levels in the soil over time, and when the soil is sensed to be too dry, irrigation is applied to raise moisture levels. In general, in systems such as that used by [ugm], the system is installed with a small number of buried soil moisture sensors in representative locations. However, this means the irrigation schedules for the entire space are dictated by the moisture conditions in specific soil regions, which in general are very heterogeneous. Over

time, this means that although the controller may be applying perfect irrigation to the region near the sensor burial, the rest of the system may be suffering due to differences in soil type and depth, solar coverage, and surface topography. If the controller performs no modeling to learn how moisture is lost over time, these systems may need to provide an extra safety margin of irrigated water as well to ensure healthy moisture levels are maintained, which may lead to inefficiencies over time. Or, alternatively, a sensor-based system may reactively actuate at any time during the day when moisture levels fall beneath the predescribed threshold, which can lead to plant sunburn and increased evaporation, wasting water. Finally, these systems are still under the physical limitation of centralized valve locations, which fundamentally limits the feasible efficiency of the irrigation system. Despite these limitations, the ability to sense the soil moisture conditions across the space is critical to provide efficient control, and so we choose to leverage spatially-dense soil moisture sensing in our system.

In [CDM08], the authors performed a side-by-side comparison of a sensor-based control strategy with timer-based controllers with and without rain sensors. The purpose of the study was to determine if control based on sensor input was sufficient to maintain a high irrigation quality to the plant, and determine what effect such a control strategy would have on system efficiency. The quality of irrigation was judged on a visual scale from 1-9, with 9 indicating the highest possible quality of health. The irrigation controllers using sensor input were configured to irrigate when moisture levels fell beneath a threshold representing the permanent wilting point, the minimum moisture level at which the plant will remain healthy. The study found that significant water savings were possible, from 8-56%. However, at the same time, the results found that some of the sensor-based systems that had the highest amount of savings also had slight reduction in plant quality, and would require a manual adjustment in control to

regain the plant health and appearance. Interestingly, as the study also tested the effect of irrigation frequency on system efficiency, it was found that irrigating every day, the maximum frequency tested, resulted in the most efficient irrigation operation.

As sensing of soil moisture conditions is crucial, we employ this into our system. We install soil moisture sensors at the same granularity as actuators, one per sprinkler. This gives our system the best understanding of the moisture conditions across the space, and minimizes the chance at miscalibration due to a misplaced sensor. By combining this feature with the ability to actuate the sprinklers independently across the space, our system can apply irrigation exactly where it is required. Finally, by modeling how water will move spatially and temporally, our system can work to irrigate not only to provide high quality of service at irrigation time, but at all time between irrigation cycles.

2.2 Quality of Service Metrics

The primary purpose of most turf installations is that it is aesthetically pleasant. Although we wish to reduce the water consumption of these turf irrigation systems, their primary purpose is to maintain the visual appeal of the plant. When searching for optimal schedules for valves in the irrigation system, we require the ability to monitor the plant to ensure we are providing a high quality of service. Some work has used the visual appeal of the plant [CDM08] as directly observed by humans to determine the effectiveness of various irrigation control strategies. However, this technique is highly subjective, as two humans are unlikely to provide aesthetic ratings that agree 100% of the time. Furthermore, if we wish to use this feedback as basis for control decisions, it is impractical to rely on humans constantly checking turf installations. A more quantitative option is

through the use of emerging optical solutions, such as those in [CS01, pho, cer]. By using an optical filter to isolate the wavelength region outside of the visible spectrum where plant reflectance changes the quickest upon plant stress, such systems are capable of identifying plants that are not healthy before it results in visible change. These techniques have started being used in large-scale aerial crop management by being fitted to airplanes. Although these visual techniques respond too slowly to irrigation changes to be used to make control decisions, in the future they may prove to be useful in verifying plant health at scale as a secondary sensor input to the control system.

Directly monitoring the plant aesthetics may be too slow to be used as feedback in a control system, but by maintaining the highest quality of health in the plant, we will be maximizing the plant's chance to achieve maximum visual appeal. Turf requires many things to stay healthy - an adequate level of solar exposure, the correct soil nutrients in appropriate amounts, and a sufficient level of moisture in the soil. Although we have no control over the solar exposure or soil nutrients, our irrigation system has direct control over the amount of moisture present in the soil. Furthermore, this soil moisture data is easily collected in our wireless sensor network, and will change quickly in response to irrigation that is applied to the surface, making it ideal as feedback for control. In plant physiology and soil physics, the moisture level where the capillary forces between the soil particles and water molecules overcomes the suction capabilities of the plant is known as the permanent wilting point [MD17, usd, Kir04]. Allowing the soil moisture to fall beneath this threshold will prevent the plant from collecting the moisture it requires from the soil, and will lead to degraded health of the plant. To maintain health then, our primary metric of quality of service is chosen to be the amount of time spent beneath this moisture threshold, which we wish to minimize. In practice, we choose to square this metric, as it indicates that

the farther beneath the minimum moisture threshold we allow the soil to go, the worse the penalty on plant health.

2.3 Modeling strategies

2.3.1 Industry

To choose optimal schedules, we must know how fluid will move across and through the soil. The first method of calculating soil water movement was through the use of Darcy's Law [Dar56], which describes flow of water through porous media at saturation using pressure differences within the soil. However, as soil behaves differently when it is unsaturated, this was later generalized in Richard's equation [Ric31] by considering the additional pressure produced by matric suction that occurs between soil particles and the water. These two formulations for fluid flow through porous media are still commonly used and serve as the basis for the following modeling tools, which solve these equations at various spatial and temporal discretization levels.

At a large scale, the water cycle can be modeled using SWAP (soil, water, atmosphere, and plant)[KDB17]. SWAP considers not only water movement within the unsaturated soil, but also the water uptake into the plant and environment, and can be configured to track the transport of nutrients or pesticides through the system. These factors make SWAP a useful tool in tracking the water balance across a large geographic area, allowing simulation of pesticide leaching, water drainage, salinity effects on plant growth, and other hydrological effects to be examined at large scale. Similarly, the SWAT (soil and water assessment tool) [swa] co-developed by the USDA Agricultural Research Service models river basin models water movement at a hydrological level for such applications as sed-

iment management modeling [BMG11], nutrient retention in Finland's Vantaanjoki watershed [GBG03], and modeling hydrological factors such as river discharge and nitrate transport of the entire European continent [ARV15]. However, due to the large scale of simulation in these tools, there is insufficient detail required to make irrigation control decisions at a smaller scale of each sprinkler.

Spanning between the large and small scale is the ParFlow hydrological modeling tool [JW01, AF96, KM06, Max13]. At the small scale, ParFlow models 3D unsaturated water flow through soil down to the ground water table and surface water flow using the shallow water equations at high precision. At the same time, ParFlow is capable of modeling weather conditions such as evapotranspiration and snow, and the coupling between surface and sub-surface flow. As this model is all-encompassing, it is designed to be run on very powerful machines that can take advantage of parallelization. As providing significant processing capabilities to an irrigation system controller will significantly raise the price of our system, such a system is not ideal for our use.

At the small scale, simulators are available that simulate water movement through soil using Richard's equation. Commonly used in the field of soil physics, Hydrus [hyd] includes finite element models for simulating saturated and unsaturated water flow, as well as solute and heat transport in 2- and 3-D. Similarly, the Comsol modeling suite has available a Subsurface Flow Module [com] which allows a physical model of the terrain to be built, and then fluid transport is simulated through the soil. The primary drawback of these systems is that they only model movement of water through the soil that already exists within the space, meaning there is no good way to simulate the application of irrigation onto the surface of the soil following an irrigation schedule. Furthermore, even if it were possible to configure the input of water onto the surface of the soil following

an irrigation schedule, these models do not track surface flow, so the effects of runoff can not be considered within the system. Finally, as these finite element models are calculating soil movement at very fine granularities, these models take significant time to solve.

In our use, we require the ability to manipulate an irrigation schedule at the sprinkler level, and track its effect on the movement of water through the soil. This model must be used in optimization, requiring many such irrigation schedules to be run through simulation. In addition to the details mentioned above, in general these modeling tools are designed to be as accurate as possible for simulations to be manually run. For this reason, they generally do not have a programmable interface that allows them to be tied in to an optimization library, and they tend to be very slow to solve each simulation, making optimization difficult. To overcome these limitations of performance and customization, we instead produce our own model that includes movement of water on the surface and within the sub-surface, allowing us to simulate sprinkler-specific irrigation schedules, distributing water onto the surface as would occur in reality. Furthermore, our model's improved performance and flexibility to accept any combination of soil parameters, topography, and irrigation system architecture allows it to be used in schedule optimization.

2.3.2 Academia

In [APN94], the authors develop a model describing water content at various soil depths at a single location in the field. This model follows the traditional loss balance function that accepts irrigation, rainfall and capillary rise as water inputs, and deep percolation, surface runoff/drainage, and evapotranspiration as water losses. These moisture changes subject to environmental and irrigation

factors will travel up and down the column as time progresses, and it is suggested that this information can be used to provide irrigation in such a way that the water is efficiently applied to the surface without excessive waste. The proposed model has the benefit of simplicity and intuition, and is surely useful to provide a rough estimate of how much water will be required to re-hydrate lost moisture. However, the proposed system has some limitations. Firstly, one of the most important environmental factors that would allow efficient localized irrigation scheduling is the prediction of runoff. Although the authors acknowledge that runoff is one of the required losses to accurately model water movement, they offer no strategy to predict it. Second, this model must make the assumption that the field is spatially homogeneous in water needs soil type and depth, etc., which will affect the model accuracy in many irrigated spaces at scale. For these reasons, it would be difficult to adapt such a system to provide sprinkler-specific irrigation schedules.

To optimize irrigation schedules, we must have a way to predict how water will move through the system in a computationally-efficient manner. For this reason, very complicated fluid movement models are difficult to integrate into a system such as ours, so alternative methods are considered to simplify complex systems. For instance, one could use moisture movement predicted by a complicated, accepted fluid flow model to train a simplified model through the use of Data Assimilation. Used to predict states of advanced systems given a particular input such as weather forecasting [Kal03] and large-scale hydrological patterns from satellite images [RME02], data assimilation is often used to create approximated system models, which can then be used for optimization. However, due to the immense size and discontinuity of the solution space in models such as one predicting fluid flow through soil can lead to an approximated model that is unable to reasonably predict outcomes of the system. Another option, called

Lumped Element Modeling, is a method of breaking complex problems into simpler "lumped" sub-components. This method has been used to create a simplified model of fluid jets for prototype analysis [GHN03], and to model aortic blood flow from arterial pressure in humans [WJS93] among other applications. Although such simplifications sacrifice accuracy, these approximations can greatly improve performance.

With use of automated irrigation systems, people maintaining the systems can fall victim to a set-and-forget mentality, which can lead to mis-configuration of irrigation systems and reduction of irrigation quality and/or efficiency. Researchers in [DMD13] build a one-dimensional soil water balance model to simulate the vertical movement of moisture in an irrigated space based on soil characteristics, irrigation schedule, real-time weather data, and irrigation infrastructure. With this simple model of moisture movement, an interactive tool was designed for homeowners and landscapists to allow quick evaluation of an installed system and provides key metrics such as expected occurrence of underwatering, amount of overwatering, and provide access to resources that may help configuration. The proposed feedback system provides some key insight on moisture changes, such as evapotranspiration that take place on a daily timescale, but over-simplify short-term effects like runoff, that can provide key insight into the movement of water across the surface. Although the tool provides projected evaluation of a particular irrigation schedule, the authors offer no strategies for schedule improvement.

2.4 Evapotranspiration prediction

In order to control intelligently, we must take into account *future* temperature shifts, precipitation, and other effects that affect irrigation requirements as later

discussed in Section 5.2.1. The standard weather metric for irrigation control has become evapotranspiration (ET), a measure of how much water is lost from the soil due to solar radiation, temperature, humidity, and wind. In [Oli97], a patent describes an irrigation controller that predicts future ET losses, but the main contribution is the combination of a reference ET estimate offset by predicted precipitation, to produce a system that will not over-irrigate with rain in the near future. Although the author recognizes the potential to use other forecasted weather metrics to predict future ET, they offer no implementation of this feature. An extensive study in [CLL07] finds that exceptional prediction of ET is possible when predictions for all four ET variables (solar radiation, temperature, humidity, and wind) are available in local data sources. The authors recognize that this is often not the case, and this holds true in our work, as wind speed and solar irradiance predictions are unavailable for our locale. As no suitable ET prediction could be found, we design our own.

The authors evaluate the effectiveness of two predictive methods for weekly evapotranspiration data based on recent historical weather trends for the use of planning and water resource management for agricultural use. The first is an autoregressive integrated moving average (ARIMA) model, and the second is an artificial neural net (ANN) approach. Both methods consider the recent evapotranspiration losses based on the FAO 56 use of the Penman-Monteith equation [APR98] equation for evapotranspiration, and predict the expected losses across the next week. These two methods are then compared to a "mean year model", where the evapotranspiration losses for week n of the year is estimated as the average of historical evapotranspiration losses for week n of all years of historical data. It is found that both ANN and ARIMA techniques have lower predictive error and variability of prediction than the "mean year model". As the predictions performed are at a weekly granularity, there is significantly less vari-

ance in weather trends in comparison to our application, where we must predict evapotranspiration losses across a 24-hour period. For this reason, it is unclear how using only past weather data will affect accuracy of prediction on the 24-hour time horizon, even using a more advanced model such as an ANN. For this reason, in our work we chose to use as input to our predictive models future weather trends, which are more likely to be indicative of the future environmental state.

A similar operation can be found here [MTT93], but for monthly evapotranspiration trends. This work considers 3 models for use in pre-emptive space planning; the first assumes the ET this month is the same as last year in this month. The second assumes the average of this month across all years in the historical record. The third is a time-series model, which will consider the statistical behavior or the system across the historical record. Similar to the previously-mentioned work, the prediction of weather across a monthly time horizon significantly reduces the variance of the weather patterns, making a time-series model particularly suitable. Surprisingly, the results of this study found that even on the month time-horizon, the averaged monthly model performed nearly as well as the more complicated time-series model. With this in mind, it is unlikely that such a model would work well on a daily time horizon, but future work is required to confirm this.

In [LTL15], 10 years of historical meteorological data is used to train four ANNs of different types (LR, PNN, MLP, GFF), with daily min, max, and mean temperature as input and resulting evapotranspiration losses as output. At the location of the study, forecasting provided min/max and average temperature values for up to 7 days into the future. These forecasted temperature values are fed into the ANNs trained on historical temp, and the predictive error is evaluated to determine the potential sources of error. It was found that primary

sources of error were in the forecasted temperature values provided by the public sources, and the lack of other environmental factors such as wind, humidity, and solar irradiance. The authors concluded that the ANNs provided ET predictions accurate enough for use by irrigation controllers, but as of yet the system has not been integrated into an irrigation controller. The methods presented are similar to those proposed in Section 5.2.1, in that forecasted weather trends are used as input to a predictive model. However, the forecasted data available in our locale was significantly more descriptive, as it contained both temperature and humidity, at hourly intervals. For this reason, we made the design decision to perform our predictions by finding the most similar daily weather trend in historical data using a k-nearest-neighbors regressor, but further analysis is required to consider the use of a more complicated model such as an ANN using the available data.

2.5 Co-processor architecture

As the devices used in wireless sensor networks are generally incapable of performing heavy computation, several approaches have been used to overcome these limitations. Firstly, it has become common for distributed applications to be configured for use on mobile [MPT17, JAC16, GLR14] or embedded [edi] devices with on the order of GBs of memory and storage, but they are both very energy-hungry and expensive at the scale of our application. Another approach is to reduce the required memory and computation by simplifying the models used, for instance in deep neural nets [YZZ17, BL16, YZS18]. However, in cases like ours where mobile phones are too expensive and power hungry and the problem can't be adequately simplified, the use of a co-processor comes to mind. Used in off-the-shelf devices like the iPhone [iPh] and Samsung Galaxy [gal], a co-processor can handle background tasks such as motion sensor querying with low power, or

highly specific tasks such as neural net computation for image processing and facial recognition without burdening the primary processor on the device. While co-processor architectures have been used in the WSN domain for chip reprogramming [HC02] and security-specific applications [POT10, WSC13, LKS10], no standard co-processor architecture has been developed allowing for low-power operation AND computational capabilities for model training and optimization as required in our application, so we design our own.

2.6 Miscellany

Work done in [BL94] seeks to investigate the effect of water droplet kinetic energy on the surface pooling that is allowed to occur. If water is allowed to pool on the surface, it will tend to run off a sloped surface, and can also be subject to evaporation. To prevent pooling from occurring, it is important to make sure the irrigated fluid is not applied more quickly than it can be absorbed by the soil. Additionally, the way the droplets strike the surface can affect infiltration rate, as high droplet kinetic energy has the potential to cause the soil surface to seal, preventing additional irrigation from infiltrating. To test this effect, the authors consider 45 samples of soil, and apply irrigation via carefully-measured droplets at various heights. The water application rates were chosen to emulate that of a center-pivot irrigation system, and the soil surface was arranged at a fixed angle to emulate a sloping hillside. The study revealed a correlation between the kinetic energy of the irrigated droplets and the amount of pooling that occurs, suggesting that reduced application rates are beneficial to the system and that water applied intermittently can improve the soil's ability to absorb surface water and reduce pooling.

These findings agree with the market emergence of low-flow sprinkler heads

such as the MP Rotator [hunc]. Traditional sprinklers throw water uniformly in all directions at the same time, with surface application rate decreasing linearly from its max right next to the sprinkler head to a minimum application rate of zero at the end of the sprinkler's reach. As the sprinkler must throw water in all directions, a relatively high water flow rate is required, and the radius of the sprinkler is relatively small. In contrast, the MP Rotator throws several thin streams of water in several directions at once, and the entire fixture rotates slowly to provide a fairly even coverage over time. In this way, the MP Rotator has a longer radius and a lower required flow rate than the traditional sprinkler, while the inherent intermittency of coverage improves the overall efficiency of the system by reducing the effect of pooling. For this reason, the MP Rotator has been used in all of our system deployments, as they represent the state of the art.

Work done in [PSL13] identify potential improvements to agriculture irrigation systems. On-site and remote control systems are discussed, as well as their learning curves, which may prevent unskilled farmers from adopting them. Bottlenecks are recognized in system development, standardization of technology, and business models, that must advance before control system alternatives become accepted. Although it is not tested, the authors propose a control strategy that seeks to bridge the learning gap for members in the farming community. The proposed strategy, with access to weather sensor inputs and expected soil behavior using simulate models, is able to provide a flexible control strategy that considers the environmental details to automate irrigation, while leaving the farmer in control via over-ride capabilities. In this way, the system will seek to minimize water consumption and water pump energy costs, with the ability to temporarily disable irrigation if faced with unfavorable environmental conditions such as high wind. This work goes to show that automation and simplicity is key to the adoption of new irrigation control equipment and strategies, and highly

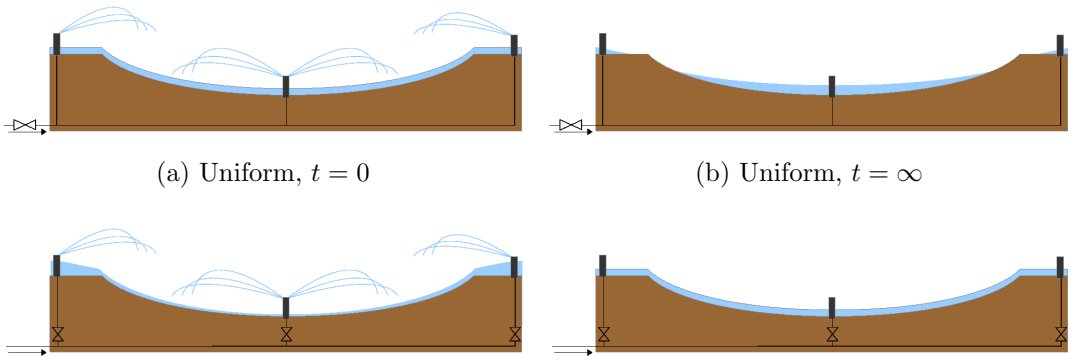
sought-after.

CHAPTER 3

Distributed Independent Actuation for Irrigation Control

Improper irrigation of lawns can lead to many problems. Too little water can lead to the death of the lawn and instability of the soil, and too much water can cause the roots of the plant to rot, leading to unattractive lawn coloration and death. Over-watering also leads to surface runoff causing evaporation and over-saturation, which can leech fertilizer chemicals into the surrounding soil. [HL12] investigates increasing nitrate levels in the water supply of California's Salinas Valley and Tulare Lake Basin. The authors attribute these levels, deemed "unsafe" by the California Department of Public Health, to agricultural fertilizers carried through heavily-saturated soils beyond the root zone into the groundwater. We identify the source of these problems as excessive moisture applied to the soil surface, and investigate the potential improvement of deploying an irrigation system that can perform distributed independent actuation to account for local variabilities in an irrigated area.

As an intuitive example, Figure 3.1a depicts a line of 3 sprinklers controlled by one central valve. As all sprinklers are activated for the same amount of time, a roughly uniform distribution of water across the surface is seen at the time of irrigation. However, as the diffusion of water into soil occurs much more slowly than surface water flow, the surface water flows across the non-homogeneous sur-



(a) Uniform, $t = 0$ (b) Uniform, $t = \infty$

(c) Non-Uniform, $t = 0$ (d) Non-Uniform, $t = \infty$

Figure 3.1: Expected results from various irrigation distributions

face causing an uneven surface coverage, as seen in Figure 3.1b. This uneven coverage will then be absorbed, causing a non-homogeneous sub-surface moisture distribution. In contrast, Figure 3.1c depicts an irrigation system where actuation is performed on each individual sprinkler. A schedule for each sprinkler is created that will provide optimal moisture distribution across the entire space. This schedule actuates the two uphill sprinklers for a longer period of time, and the downhill sprinkler for less. After irrigation, the surface water flows from areas of higher concentration and elevation, and so the runoff corrects for uneven coverage, seen in Figure 3.1d, where it will subsequently be absorbed as a more homogeneous sub-surface moisture distribution. Although the amounts of water used in both examples are similar, the centralized uniform actuation solution results in some locations receiving insufficient moisture content, requiring additional watering to ensure sufficent water is received.

In this chapter, we develop a water flow model that is computationally light enough to be evaluated by standard optimization techniques. This water flow model is then used to create optimal scheduling for different valve configurations. These schedules are then used in simulation with a physical model of a univer-

sity lawn to demonstrate the benefits of distributed independent actuation in irrigation systems.

3.1 Problem Formulation

An irrigation system that is capable of sprinkler-level actuation must be tested to see if it is a viable option as a control system. As such, we must have the ability to produce an optimal schedule such that the following is satisfied.

$$min \sum_{i} \vec{S_i} \text{ s.t. } \exists\, t > 0 \qquad \forall \vec{x} \in \vec{X} \quad a(\vec{x}, t, \vec{S}) \geq \theta_{pwp} \tag{3.1}$$

In this formula, a is the fluid flow model, which depends on the location within the space, timestep, and $\vec{S}$, the binary matrix of size $K \times T$ that defines the activation of each sprinkler k at each discreet time t. As $\vec{S}$ defines sprinkler activation across time, $\sum \vec{S}$ is proportional to system water consumption. This value is minimized such that at a future time, sub-surface moisture at each position in the space is above θ_{pwp}, a constant known in hydrology as the humidity level where plants can no longer absorb water from the soil. With these conditions satisfied, $\vec{S}$ contains the activation schedule for each sprinkler that will provide adequate moisture levels across the space with minimal system water consumption.

3.2 Water Flow Model

To accomplish the minimization described in Eq. 3.1, we must know how fluid moves across and through soil. This is well-studied in the field of soil physics; very accurate models including Hydrus[hyd] and Comsol Multiphysics' Subsurface

Flow Module[com] exist, which solve partial differential equations for pressures that exist between soil and water particles in the porous media. Although highly accurate, solving for these particle interactions require heavy processing, and take a tremendous amount of time to complete.

An overly-complicated fluid model makes optimization unreasonable, and so alternative methods are considered to simplify this complex system. One such method is to use data assimilation to simplify one of the accepted fluid flow models. Used to predict states of advanced systems given a particular input such as weather forecasting [Kal03] and large-scale hydrological patterns from satellite images[RME02], data assimilation is often used to create approximated system models, often for optimization. However, the size and discontinuity of the solution space can lead to an approximated model that is unable to reasonably predict outcomes of the system. Another technique, called Lumped Element Modeling, allows us to break a complex problems into simpler "lumped" sub-components. This method is used to create simplified models of fluid jets for prototype analysis[GHN03], and to model aortic blood flow from arterial pressure in humans [WJS93]. Although this simplification sacrifices accuracy, this method is more fitted to our problem, as it will allow an optimizer to explore a simplified solution space more quickly.

3.2.1 Water Flow Model Formulation

To approximate the behavior of moisture within an irrigated space, we define characteristics of the soil and expected flow patterns. These include soil absorption rate, soil depth, maximum flow rates, etc. The constants defined in Table 3.1 were chosen to resemble irrigated spaces at our university, but parameters can be found for any given space.

Table 3.1: Constants used in model

Constant	Shorthand	Value	Units
Max Absorbance Rate	MAR	1	mm/second
Max Storage	MS	300	mm
Max Slope	S_{max}	45	$^\circ$
Min Slope	S_{min}	0	$^\circ$
Max Water Movement Fraction	C_{max}	1	1
Min Water Movement Fraction	C_{min}	0.1	1

$$ssf(x, y, t) = ssf(x, y, t-1) + f_a(x, y, t)$$

The sub-surface fluid, ssf, is defined as the sum of the sub-surface moisture at the previous timestep and fluid absorbed, f_a at this time.

$$sf(x, y, t) = sf(x, y, t-1) + f_s + f_{ri} - f_{ro} - f_a(x, y, t)$$

Likewise, the surface fluid is defined as the sum of the surface fluid, sf, at the previous timestep, direct moisture contribution from sprinklers within range, f_s, incoming runoff, f_{ri}, minus lost water due to runoff, f_{ro}, and moisture absorbed into the soil.

$$f_a(x, y, t) = min \begin{cases} sf(x, y, t-1) \\ MAR * (1 - ssf(x, y, t-1)/MS) \end{cases}$$

To simplify the process of fluid absorption, we treat each patch of soil as a single storage unit with maximum storage MS, a reasonable simplification in irrigated spaces with thin sod layers. The rate of absorbance is defined between zero

and maximum absorption rate MAR, based on the moisture already contained in the soil. The amount absorbed is either the entire amount that lies on the surface, or the maximum amount the soil can absorb.

$$f_s = \sum_{k=1}^{K} \begin{cases} 0 & \text{if } dist(sloc(k),(x,y)) > MSR \\ \vec{S}(k,t) * MF * (1 - dist(sloc(k),(x,y))/MSR) & \text{otherwise} \end{cases}$$

At each timestep, each location (x, y) on the surface receives f_s units of water, the sum of contributions from all K sprinklers. Resembling water distribution of commercial sprinklers, each sprinkler contributes an amount scaled from MF, the maximum fluid application rate, at the sprinkler's location, to zero, at a distance of MSR, the maximum sprinkler radius or greater from the sprinkler. As university irrigation systems are professionally installed, we assume that no irrigated water will be delivered out of the irrigated space.

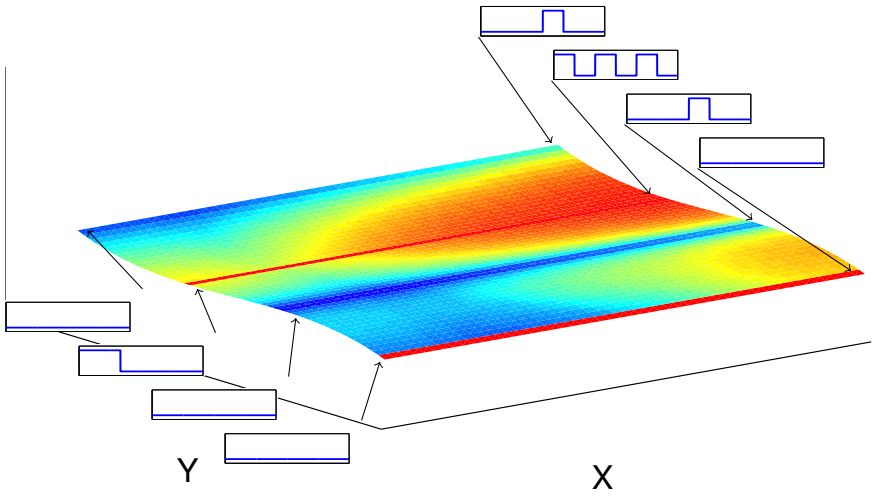
$$f_{ri} = C * \sum_{\text{neighbors}} \begin{matrix} dh(x_n, y_n) == (x, y) * sf(x_n, y_n, t-1) \\ -f_a(x_n, y_n, t) \end{matrix}$$

Each neighbor with its steepest downhill neighbor, or $dh(x_n, y_n)$, matching our location adds a fraction C of their previous moisture content to our location.

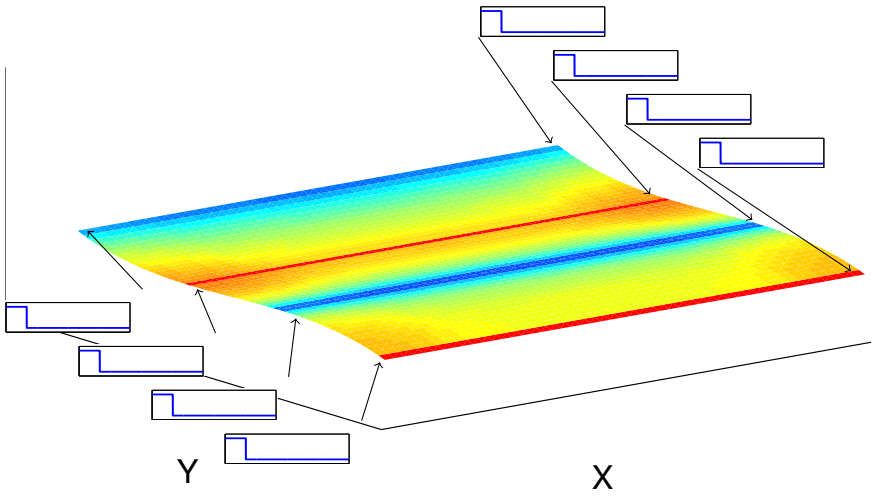
$$f_{ro} = \begin{cases} C * (sf(x,y,t-1) - f_a(x,y,t)) & \text{if } \exists \text{ dh(x,y)} \\ 0 & \text{otherwise} \end{cases}$$

If a downhill neighbor, dh(x,y), exists, meaning we have at least one downhill neighbor, a fraction C of our surface moisture is lost to that neighbor.

$$C = C_{min} + (C_{max} - C_{min}) \frac{(slope((x_d, y_d),(x,y)) - S_{min})}{(S_{max} - S_{min})}$$



(a) Fully-independent valve optimization



(b) Single-valve optimization

Figure 3.2: Sub-surface moisture distribution

In order to simplify our water flow model, we make the assumption that moisture moving across the surface moves in the direction of steepest descent, from (x, y) to location (x_d, y_d). To approximate this surface flow, we define bounds on surface slope, S_{min} & S_{max}, and on C as C_{max} & C_{min}, defined in Table 3.1. It should be noted that C_{min} is non-zero to ensure that surface flow does not slow to zero as the topography approaches a local minimum. Any slope between S_{min} & S_{max} is then linearly interpolated between C_{max} & C_{min}.

3.3 Optimization Solution

As the input to the simulation is a schedule of binary integers and the formulation is linear with respect to the schedule, the optimization problem is a non-continuous integer linear programming problem. Limitations in integer linear programming solvers found the Pattern-Search algorithm to be the best choice, as it does not require the function to be continuous or convex. Although it is incapable of solving integer problems, we were able to apply it by rounding all float inputs within [0,1] to the nearest integer. This is not optimal, as turning a discrete problem into a continuous one could harm our results, and in the future we will go into greater depth exploring further optimization routes.

$$min \sum_{i} \vec{S_i} + ssf_{unsatisfied}/ssf_{total} * \epsilon$$

To improve on the objective funtion defined in Eq. 3.1, the constraints ensuring that moisture levels are sufficiently high in all locations are converted to a single slack variable penalty function, which harshly punishes the objective function by a factor ϵ multiplied by the percentage of the sub-surface area that did not receive adequate irrigation. The reasons for using a slack variable penalty

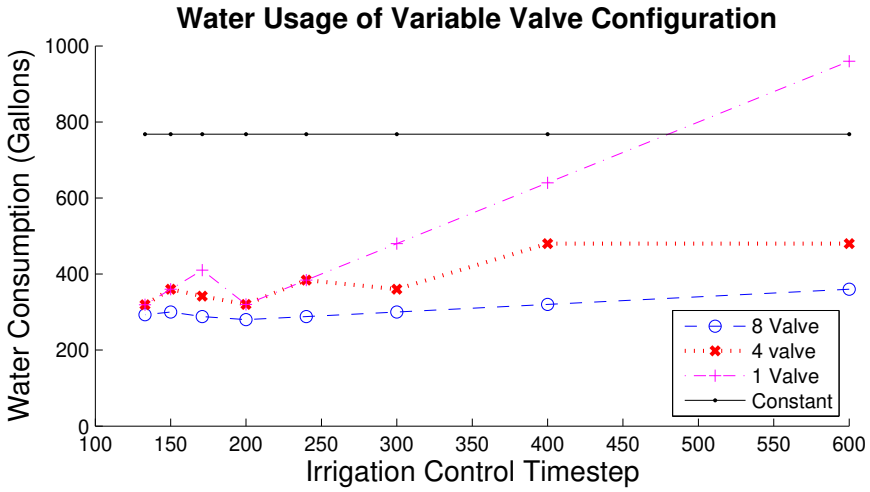


Figure 3.3: System efficiency of multiple configurations with varying control timesteps

function are two fold. For one, it allows the optimizer to explore the solution space where constraints would be exceeded. In addition, in the case that a solution that satisfies the constraints can not be found, a slack variable ensures that we will still return the best schedule possible, even if imperfect.

To make the calculation of optimal schedules feasible, each schedule is discretized in blocks of irrigation timesteps. These timesteps are varied and the effect on system efficiency is described in Section 3.4. Of course, this discretization limits the potential improvement of the optimized schedule, and so in future work improved optimization strategies will be used to more thoroughly solve the problem.

3.4 Results

To determine the viability of an irrigation system with distributed independent actuation, we compare its efficiency to that of a uniformly actuated irrigation

system. The system used in simulation has 8-sprinklers and covers an area of 1600 ft^2 with a non-homogeneous topography. Tests are run using a single valve with a constant watering time, chosen in an ad-hoc manner to closely match those used on our campus, as baseline. The 8 sprinklers are split evenly into 1, 4, and 8 valves for variable spatial granularity, and optimized schedules are found with control timesteps from 600 seconds to 130 seconds, to vary temporal granularity. Simulations were run using these optimized schedules, and the resulting system water consumption is plotted in Figure 3.3.

The time-independent constant watering strategy can be seen to consume significantly more water than all but the least granular optimization strategies. This schedule replicates irrigation timing chosen in an ad-hoc manner by the greenskeepers at our university. A large amount of savings seen with the other systems is due to the ability to water intermittently, which allows the soil to absorb the water before more is applied. The efficiency is shown to improve as the number of valves used increases, and generally as the timestep decreases. In this proof-of-concept experiment, the optimized schedule with a fully distributed independent actuation system provided sufficient water to the entire space, while using 64% less water than the constant watering schedule. As our university consumes approximately 32.5 M gallons each year on irrigation alone, these results suggest that using a distributed independent actuation system could save about 20 M gallons of water per year, costing $112,000.

The final sub-surface moisture distribution of different valve configurations with optimized scheduling can be seen in Figures 3.2a and 3.2b. The plots demonstrate the activation schedules chosen to be applied at the respective sprinkler locations. In Figure 3.2b, where the optimized schedule is calculated for a single valve, the schedule for each sprinkler is equivalent, and the coverage distributed

is fairly uniform. In the 8-valve configuration of Figure 3.2a, the optimized schedule delivers more water to the upper region of the figure, but less water to the lower region of the figure. Although this results in a distribution that is less uniform, the total water consumption has been improved by 13.6%. It should also be noted that the optimizer converged on valve schedules that take advantage of intermittent watering. Although surface flow causes the majority of flow in non-homogeneous terrain, this intermittent watering will reduce pooling even in level terrain, reducing the effects of evaporation to water loss. In future work, we will introduce an additional slack variable penalty function to weigh the optimizer towards solutions with more uniform moisture distribution, even if the total water consumption is slightly higher, and measure the benefits of intermittent watering on homogeneous terrain.

3.5 Conclusions

In this chapter, we developed a fluid flow model to determine if distributed independent actuation could benefit the efficiency of an irrigation system. In simulation, we determined that by moving from a system with one valve using a schedule chosen in an ad-hoc manner to a system with fully distributed independent actuation capabilities, savings of up to 64% can be realized. In addition, the water consumption of this new actuation system consumes 13.6% less water than a system with half the number of valves, suggesting that a fully distributed system can be maximally efficient. In the next chapter, we apply this knowledge to create a distributed independent actuation system to demonstrate that irrigation systems can be made much more efficiently.

CHAPTER 4

DICTUM: Distributed Irrigation Control with Turf hUmidity Modeling

As discussed in the previous chapter, inefficiencies in turf irrigation stem from the system's inability to properly apply irrigation where it is needed. Great improvements in irrigation system design have been made recently; new sprinkler heads apply water much more slowly to avoid runoff and leeching [hunc], and new irrigation controllers schedule irrigation using weather data to take into account the water lost each day due to evaporation and plant transpiration, known coupled as evapotranspiration. Even the best control strategies still behave as though all turf requires the same amount of water, when in fact there often exist large variations in soil type and depth, topography, and direct sunlight. If this information were utilized, every location throughout the irrigated space could be given the amount of water it needs. However, as the infrastructure of traditional irrigation systems is usually configured for each valve to actuate many sprinklers, such a system could not even make proper use of fine-grained water requirement information as all sprinklers must be actuated for the same amount of time.

Our contributions in this chapter address both of these limitations. First, we develop a computationally-light model that uses characteristics of the irrigated space to analyze the fundamental causes of fluid movement. This model is then integrated into an optimization framework to allow for optimal valve scheduling

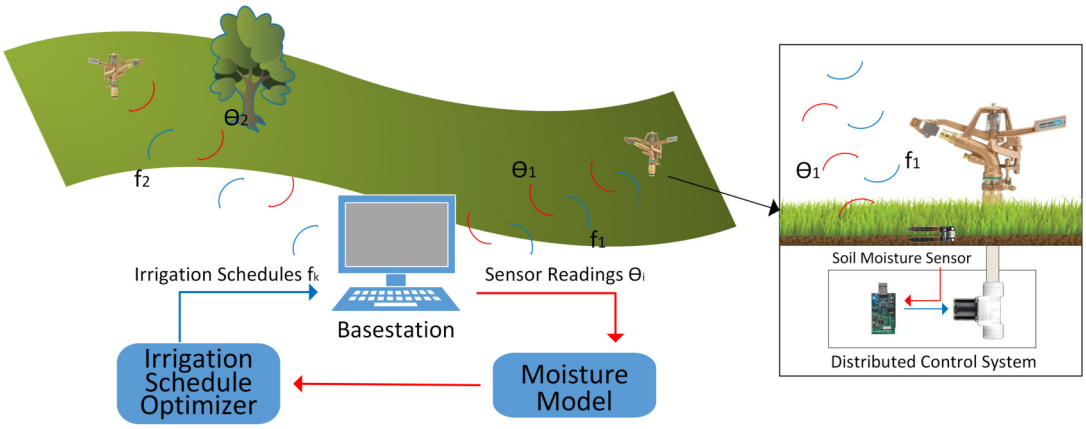


Figure 4.1: DICTUM System Architecture

to be computed. The second contribution is the development of the DICTUM sprinkler node, capable of actuating its attached sprinkler, sensing the moisture in its surrounding environment, and communicating wirelessly with its sister nodes in the environment. With our experience of two large-scale irrigation system deployments over a total of 7 weeks, with 4 weeks of fine-grained data collection and monitoring, we demonstrate that the model-based DICTUM system can help provide more precise irrigation control to turf areas, reducing water usage and substantially improving the quality of service over common-practice and state-of-the-art control strategies.

4.1 System Overview

Figure 4.1 shows an overview of the DICTUM system architecture. Our irrigation control system uses multiple modules to provide control to the space. To explain how these modules work together to create a fluid processing pipeline, we first describe their roles. This processing pipeline is described at irrigation time each day, when schedule generation occurs.

Our deployment consists of a distributed network of sensing and actuation (DICTUM) nodes, integrated into the plumbing infrastructure of the irrigation system. Each DICTUM node is equipped with a soil moisture sensor, a solenoid to control the flow of water, and a mote, to provide radio communication capabilities. To allow the DICTUM system to react to changing soil conditions, an attached volumetric water content (VWC) sensor is periodically sampled from the environment by each sprinkler node in the space. This collected data is then routed through the wireless sensor network to the *Basestation*, interfacing between the 802.15.4 network and another communication medium such as an ethernet or 4G network. Once received, this data is then incorporated with the sensor readings collected from other nodes to create a "snapshot" of the soil moisture across the entire space.

Once the current state of VWC in the soil is determined, it is fed into the *Moisture Model* for integration. The *Moisture Model*, described in detail in Section 4.2, contains a mathematical formulation for moisture movement throughout the system. This model is all-inclusive, modeling the characteristics of the irrigation system and soil characteristics within the space, providing means to calculate the conductivity through the soil. With the VWC collected, the model is passed to the *Irrigation Schedule Optimizer* for analysis.

The *Irrigation Schedule Optimizer* sets up and solves the following constrained optimization problem. The fluid flow model is incorporated as equality constraints at each spatial location and time, which must be satisfied for an optimal solution to result in a valid flow. To ensure adequate moisture levels across the space and thus a high quality of service, a goal water saturation level provides inequality constraints at each spatial location at the end of irrigation. Although in principle the PDEs defining the model are nonlinear, we linearize them in order

to make the optimization problem convex, without local optima, and easier to solve computationally. The final result is a linear program that, although large (due to the discretization), can be solved accurately in a reasonable time. As the objective function minimizes the total water consumption of the irrigation system, the solution provides optimal activation schedules for each actuation node in the space, while maintaining minimum moisture levels.

Once the schedules are received by the *Basestation*, they are disseminated through the wireless sensor network to their respective DICTUM node. Upon reception of a schedule, the *Distributed Control System* routes power to the attached solenoid following the received schedule, allowing water to flow to the sprinkler.

4.2 Model Development

We wish to model a particular irrigation system, and use this information to find improved control techniques. This two-dimensional model incorporates water movement from the sprinkler heads to the ground, across the surface, through the sub-surface, as well as absorption into the soil. Although many models exist that describe one or more of these components, we present here the first model that efficiently combines them together.

4.2.1 Soil Characteristics

We first emphasize the differences between soil and typical porous media. Generally, a porous medium maintains constant characteristics across its entire range of saturation. However, in soils, two functional relationships govern the retention and movement of water through soil. First, an attraction exists between water

Table 4.1: Model Variable Reference

Variable	Usage	Variable	Usage
f_k	Actuation function of sprinkler k	ρ	Fluid density
c_k	Coverage of sprinkler k	ζ	Sub-surface boundary constant
θ	Volumetric soil moisture content	μ	Surface boundary constant
h	Surface fluid height	α_h	Surface flow parameter
$\vec{u}$	Velocity of water in soil	$K(\theta)$	Hydraulic conductivity
$\vec{v}$	Velocity of water on surface	$\psi(\theta)$	Matric suction
κ	Soil permeability	F_s	Fluid from sprinklers
κ_g	Grass permeability	$\vec{\tau}$	Tangential component of gravity
η	Fluid viscosity	θ_{pwp}	Minimum θ for Healthy Plants
L	Thickness of soil (m)	ϕ_s	Porosity of soil layer

and the soil particles, known as matric suction. When the soil experiences very low levels of saturation, the matrix exerts a strong suction, trying to pull water from the surrounding environment. The relationship between tension head and volumetric water content is known as the "water retention curve" a characteristic equation of the soil type as defined in [Gen80]. This relationship, as shown on the left axis of Figure 4.2, strongly impacts the movement of water through the soil, as dry soil will act as a sink, until increased saturation is reached. Second, the hydraulic conductivity of soil is dependent on the local volumetric water content. Due to the matric suction, the soil will increasingly resist the movement of moisture as the volumetric water content decreases. A typical relation for hydraulic conductivity is also given in [Gen80] and is shown on the right axis of Figure 4.2.

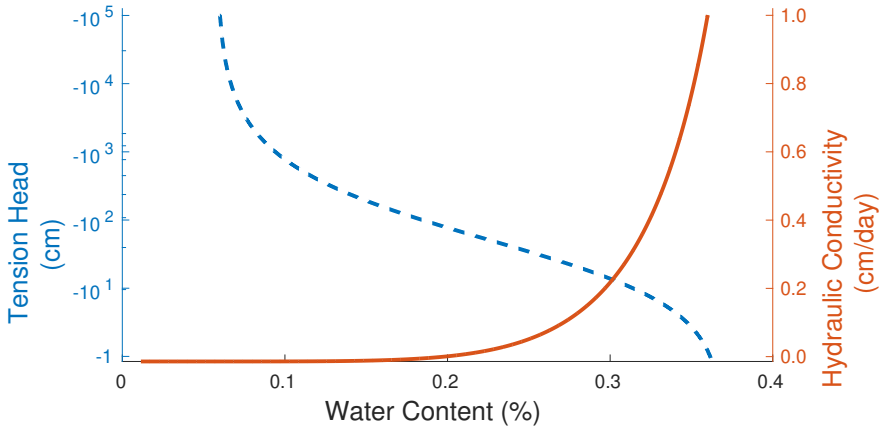


Figure 4.2: Sample Retention & Hydraulic Conductivity

4.2.2 Fluid Flow Model

We model water displacement above the soil surface and through the subsurface of the soil as flow through two different porous media. Fluid flow through porous media is well-studied, dating back to the work of Henry Darcy [Dar56], now known as Darcy's Law. To model the movement through the soil we use Darcy's Law for isotropic porous media (Eq. 4.1),

$$\vec{u} = \frac{\kappa}{\eta}(-\nabla P + \vec{\tau}) \tag{4.1}$$

where P is the pressure, $\vec{\tau}$ is the tangential component of gravity along the surface of the porous media, $\vec{u}$ is the fluid velocity averaged over the thickness of the soil, and other quantities as defined in Table 4.1. This model assumes that the porous media everywhere has the same dependence on water content. As our model tracks soil moisture at small scales, such a simplification is more practical than the alternative of collecting and analyzing samples across the entire space.

Darcy's Equation requires the determination of the pressure, which is generally linearly related to the amount of water above the point in question. In our model, see Figure 4.3, we compute the depth-averaged subsurface flow, by

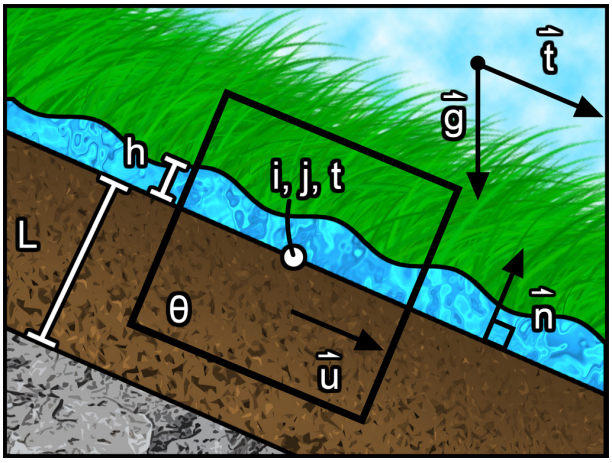


Figure 4.3: Physical Model Unit Diagram

considering a soil depth L. In this case the mean pressure driving flow includes the weight of water in the subsurface above a point, $\rho g L\theta$, the weight of water on the surface, $\rho g h$, where h is the height of water above the soil surface, and the matric suction of the soil $\rho g\psi(\theta)$. We thus express the mean pressure in soil as $P = \rho g(h + L\theta + \psi(\theta))$. We can therefore express the liquid velocity in the subsurface as

$$\vec{u} = -K(\theta)\nabla h + \frac{K(\theta)\vec{\tau}}{\rho g} - K(\theta)(L + \psi'(\theta))\nabla\theta \tag{4.2}$$

where we defined the hydraulic conductivity as $K(\theta) = \rho g\kappa(\theta)/\eta$.

To track the time-rate of change of the volumetric soil moisture content, θ, we use the divergence of the moisture flux, $\vec{u}\theta$, and the inflow from surface water, $\zeta h K(\theta)$, where $\zeta = 1/(L^2\phi_s)$ is a proportionality constant mapping surface water height to volumetric content in the subsurface based on soil porosity and depth, calculated by balancing the pressure gradient with the soil permeability. We thus have

$$\frac{\partial\theta}{\partial t} = -\nabla\cdot(\theta\vec{u}) + \zeta h K(\theta) \tag{4.3}$$

As the velocity, $\vec{u}$, is itself the gradient of the volumetric soil moisture content,

the movement of water will effectively behave as a diffusive process, moving sub-surface moisture towards areas of lower concentration. With the inclusion of a gravity term, $\vec{\tau}$, in the velocity equation, water will tend to move in the direction of steepest descent. Lastly, by allowing water to move through the boundary from the surface into the sub-surface, we allow irrigation on the surface of the soil to positively affect the amount of water moving through the sub-surface. Together, these terms result in sub-surface water movement that realistically depends on volumetric water content, local topography, and surface conditions.

In addition to tracking the soil moisture, we need to determine the height of the surface water, h. Depending on the species of lawn chosen to model, a square inch of turf can have tens to hundreds of blades of grass, each of which will impede the movement of water across the surface of the soil. By comparing the inertia of the fluid and the drag forces caused by the blades of turf, we find that for surface water velocities less than 1cm/s, the surface flow through the turf can be modeled as fluid flow through a (very) porous medium. Additional information and the calculations that support this approach can be found in Appendix A.1. The velocity of water through the turf averaged over the height h, denoted by $\vec{v}$, is therefore computed using Darcy's Equation for isotropic porous media, as was done for the sub-surface velocity in Equation 4.2:

$$\vec{v} = \frac{\kappa_g}{\eta}(-\rho g \nabla h - \vec{\tau}) = -\alpha_h \nabla h - \frac{\kappa_g}{\eta}\vec{\tau} \tag{4.4}$$

where the fluid density, permeability, and gravity terms have been absorbed into α_h. Similarly to sub-surface movement, the velocity through the layer of grass is dependent on the orientation of the surrounding topography, as well as the viscosity and density of the fluid.

The permeability used in Equation 4.4, κ_g, is not as well-defined as that of the sub-surface fluid flow. The shape, size, and density of the grass layer

is dependent on the species of grass, as well as its health. Research has been conducted [Cos06, RGV04] for the application of filter design that defines the permeability of a fibrous media based on its overall porosity and the size of the fibers. Using the proposed method, we find an approximate permeability of 10^{-5}m , a value similar to that of well-sorted gravel. This completes the description of the velocity of surface water, and we may use this velocity to keep track of the height of water on the surface:

$$\begin{aligned} \frac{\partial h}{\partial t} &= -\nabla \cdot (h\vec{v}) + F_s - \mu h K(\theta) \\ F_s &= \sum_{k=1}^{n} c_k f_k(t) \end{aligned} \tag{4.5}$$

where F_s is the rate of irrigation, determined using the activation $f_k(t)$ of sprinkler k at time t and coverage c_k of sprinkler k. The amount of water lost to soil is the same as that added to soil moisture, converted from soil moisture to pure water, where $\mu = \zeta L \phi_s / \phi_g$, with ϕ_s, ϕ_g as the porosities of soil and grass and L is the soil depth. We note that evaporation and leeching terms were not included in our formulation, due to the way our case study was conducted. At the request of campus authorities, all irrigation was performed in late evening, providing ample time to absorb into the soil, and allowing only minimal evaporation to occur. Although these terms are omitted in the present study, they could easily be introduced for an application that requires them.

4.2.3 Boundary and Initial Conditions

In order for the PDE-based model to accurately represent the movement of moisture through the turf and soil, we must also specify boundary conditions. Primarily, these boundary conditions must capture how moisture moves in and out of our domain on the sides and on the bottom. In our deployment location, the

turf is a sod layer with a thickness of approximately 1.5 inches. Below this sod layer lies a thick layer of clay. As the hydraulic conductivity of clay is 60–3700× smaller than the conductivity of a loamy soil [Gen80] and our optimization only includes movement of water within the first couple of hours following the start of irrigation, we consider that leeching into deeper soil is negligible, and thus omitted. The boundary conditions on the sides of the surface assume an expanse of identical surface, without fluid so that $h = 0$. With this constraint, any water on the surface located directly adjacent to the boundary will begin to lose moisture in the direction of the boundary subject to gravity. Likewise, the boundary conditions for the sides within the soil layer assume a surround of soil with a fixed volumetric water content of $\theta = \theta_{\text{pwp}}$, the minimum volumetric content to maintain turf health. Any moisture within the soil adjacent to the boundary will lose or gain moisture in the direction of the boundary subject to gravity or suction effects.

In addition to those boundary conditions, we must also provide an initial state of the system as a starting configuration. As we irrigate once daily, it is assumed that there is ample time for all surface water to be absorbed into the soil. As such, the initial condition for surface moisture is set to zero. The soil moisture content, however, must be measured from the soil to provide an accurate snapshot of the moisture distribution prior to irrigation. To provide this, current data from each sensing node is referenced and fed into the model. As the sensors are coarsely-distributed spatially throughout the area, the data is upsampled to the same granularity as the optimization problem using a bilinear interpolation.

4.2.4 Fluid Flow Model Simplification

We simplify the model in two ways for reasons of computational efficiency. Firstly, we linearize the model PDEs. Although this is not necessary if one only wants to solve the PDEs to obtain the flow over time given a schedule, it considerably facilitates the numerical optimization over the schedule. Nonlinear equality constraints make the feasible set nonconvex and give rise to local optima, which complicate finding a good optimum. They also require nonlinear optimization, which is slower. Second, we discretize the spatial and temporal domains and approximate the derivatives using finite differences. With these simplifications, since the objective function and inequalities are already linear in our application, the resulting optimization problem is a linear program, for which efficient solvers that can handle millions of variables and constraints are available. As seen later, this allows us to obtain a valid schedule in a relatively small amount of time.

4.2.4.1 Fluid Flow Model Linearization

The goal of the linearization is to characterize each equation in the model in terms of linear combinations of optimization variables. To remove non-linearities arising from optimization variables multiplied by each other, we make reasonable assumptions about the behavior of the system to substitute these non-linearities with linear counterparts.

We break each optimization variable into a base value, with subindex 0, and a small deviation, denoted with a hat. For example, the volumetric moisture content, θ, is rewritten in the form $\theta = \theta_0 + \hat{\theta}$. Each occurrence of the original four optimization variables is replaced with a similar representation, and simplified to achieve the following four linear equations, where we define a function $\varphi(\theta) =$

$K(\theta)(L + \psi'(\theta))$ to simplify notation:

$$\frac{\partial h}{\partial t} = -\nabla \cdot (\underline{\hat{h}\hat{\vec{v}}} + \hat{h}\vec{v}_0 + h_0\hat{\vec{v}} + \underline{h_0\vec{v}_0}) + F_s - \eta\Big(h_0K(\theta_0) + h_0K'(\theta_0)\hat{\theta} + \hat{h}K(\theta_0) + \underline{\hat{h}K'(\theta_0)\hat{\theta}}\Big) \tag{4.6}$$

$$\frac{\partial \theta}{\partial t} = -\nabla \cdot (\underline{\hat{\theta}\hat{\vec{u}}} + \hat{\theta}\vec{u}_0 + \theta_0\hat{\vec{u}} + \underline{\theta_0\vec{u}_0}) + \zeta\Big(h_0K(\theta_0) + h_0K'(\theta_0)\hat{\theta} + \hat{h}K(\theta_0) + \underline{\hat{h}K'(\theta_0)\hat{\theta}}\Big) \tag{4.7}$$

$$\hat{\vec{u}} = -K(\theta_0)\nabla\hat{h} - K(\theta_0)\nabla h_0 - K'(\theta_0)\hat{\theta}\nabla h_0 - \underline{K'(\theta_0)\hat{\theta}\nabla\hat{h}} + \frac{K(\theta)\vec{\tau}}{\rho g} - \varphi(\theta_0)\nabla\theta_0 \tag{4.8}$$

$$- \varphi(\theta_0)\nabla\hat{\theta} - \varphi'(\theta_0)\hat{\theta}\nabla\theta_0 - \underline{\varphi'(\theta_0)\hat{\theta}\nabla\hat{\theta}} - \vec{u}_0$$

$$\hat{\vec{v}} = -\alpha_h\nabla h + \frac{\kappa_g}{\eta}\vec{\tau} - \vec{v}_0 \tag{4.9}$$

As we assume that the deviations are small relative to the base value, the underlined terms that represent the nonlinearities of the original formulation should be much smaller than other terms appearing in the equations, and are ignored. As $h_0\vec{v}_0$ in Equation 4.6 and $\theta_0\vec{u}_0$ in Equation 4.7 do not change spatially, the divergence of these terms is zero as well, so they are omitted.

4.2.4.2 Fluid Flow Model Discretization

The derivatives appearing in our modeling equations are approximated with finite differences. We use forward differences for the time derivatives for θ and h, for example $d\theta/dt \approx \frac{\theta_{i,j,t+1}-\theta_{i,j,t}}{\Delta t}$, where Δt is the time interval size and t is the temporal index, with $t = 0,\ldots,N_t$. We use centered differences for the spatial derivatives for θ and h, for example $d\theta/dx \approx \frac{\theta_{i+1,j,t}-\theta_{i-1,j,t}}{2\Delta x}$, where Δx is the spatial grid size and i,j are the indices of a spatial cell, for i=0,...,N_x and j=0,...,N_y . Consider the discretization on a continuous range $[0, L_x]$ where length $L_x = N_x * \Delta x$ in the x direction. The entire range is broken into N_x segments of length Δx. This discretization is performed on all variables of our linear model equations, resulting in the 6 equations found in Appendix A.2.

4.2.5 Model Accuracy

Across four weeks of DICTUM system deployment, we track the evolution of soil moisture in the system in response to the valve actuations chosen by the model-based optimization of the DICTUM system. As later described in Section 4.3, each day an optimization problem is solved that finds valve schedules that will minimize system water consumption while maintaining adequate moisture levels everywhere in the space. In addition to the optimized valve schedules, this optimization also produces the predicted sequence of soil moisture levels that will occur during irrigation using the PDE model. To determine the accuracy of the PDE model, each day we compare the final moisture level *predicted* by the model in optimization to the *true* final moisture levels experienced across the space after irrigation following the optimized schedules. Each day, we track the root mean square error (RMSE) between the predicted and true final moisture values, and normalize it across the range of moisture levels experienced in this dataset, a range of 17% volumetric content in this particular experiment. These errors, tracked each day, can be seen in Figure 4.4, where we can see the error in the first days of the deployment can reach as high as 25%. This is in part due to the coarseness of the initial moisture distribution measurements. Moreover, as we show in Section 4.5, the first days of an experiment tend to be where the control strategy must correct for poor initial moisture conditions of the space, and in this case, we suspect that these atypical initial conditions tend to reduce the accuracy of the model. This is supported by the fact that as the system runs and guides moisture conditions into a more typical range, the daily model error reduces to the order of $\sim$10% in the last several days of Figure 4.4.

In its current operation, the DICTUM system is used once at irrigation-time to optimize valve schedules for the day's irrigation. However, if the optimization

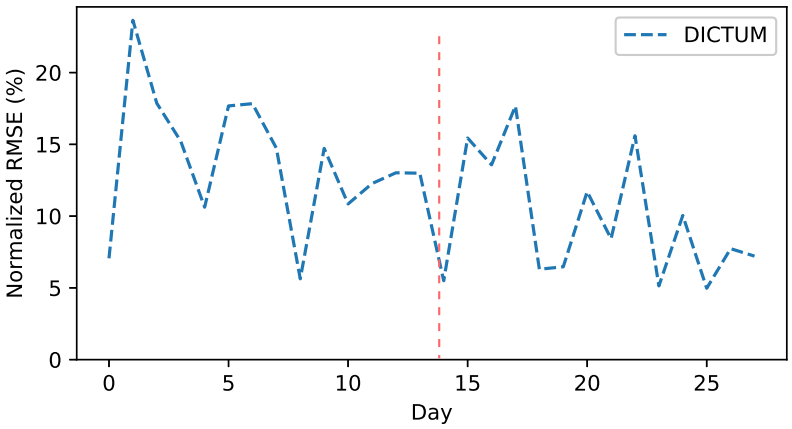


Figure 4.4: Accumulated model error across irrigation period across deployment 1 (left of red line) and deployment 2 (right of red line)

performance allows for it, it would be preferable for the system to also perform *intermediate* schedule optimization using freshly-sensed moisture values across the space in the middle of irrigation, to correct for error in the PDE model. To determine the effect this re-correction would have on model accuracy, on the same historical data as the previous error analysis, we consider a system that uses fresh data and re-computes optimization at *every* 60 second control timestep. Then, considering the same metric as before, we compute the average model error that is able to occur within this 60 second interval for each day in the experiment. Figure 4.5 shows that the average error tends to be between 1-2% of the true value indicating that using re-correction in the DICTUM system, on average, the predicted moisture level will be no more than 2% away from the true moisture level in the space, a significant accuracy improvement in comparison to the 10-25% error possible when no re-correction is done as shown in Figure 4.4.

As we discuss at length in Section 2.3, there are no existing models that have the features required for a model-based optimization in irrigation control;

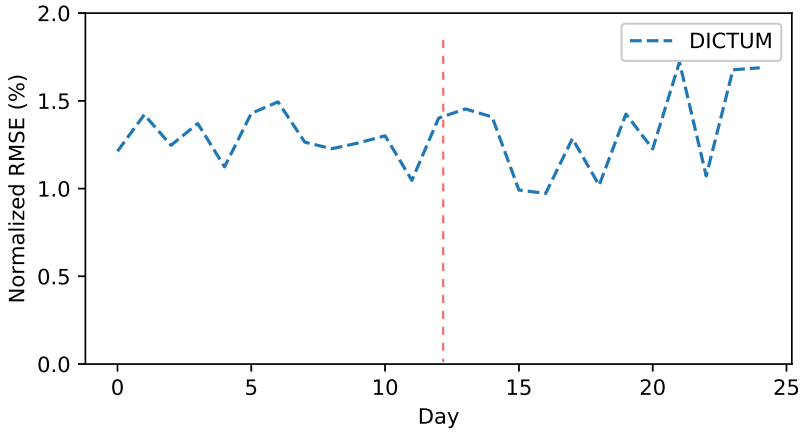


Figure 4.5: Average model error that accumulates over a 60 second window across deployment 1 (left of red line) and deployment 2 (right of red line)

for optimization we require a programmatic interface, the ability to model both sub-surface and surface fluid as well as the movement between the two, and relatively high performance to make schedule optimization tractable. Although the model used in the DICTUM system has some error in operation, it meets all the requirements for the system to function. Despite these imperfections, we later demonstrate in Section 4.5 that the DICTUM system is able to achieve significant system benefits in both efficiency and quality of service.

4.3 Optimization Over the Schedule

For use as an irrigation control system, we must now use our model from Section 4.2 to produce optimal sprinkler scheduling for the system. The objective of this optimization is to produce a schedule that provides enough moisture at all points in the space to maintain health, while minimizing system water consumption.

Table 4.2: Optimization Variables

Variable	Description
i, j	Spatial index $\in \{0, \ldots, N_x\}, \{0, \ldots, N_y\}$
t	Temporal index $\in \{0, \ldots, N_t\}$
k	Sprinkler location index $\in \{1, \ldots, K\}$
f_{kt}	Sprinkler k actuation at time $t \in \{0, 1\}$
$h_{(ijt)}$	Height of water on surface
$\theta_{(ijt)}$	Soil volumetric water content
$\theta_{\text{initial}(ij)}$	Sensed moisture at $t = 0$, upsampled to PDE grid $N_x \times N_Y$
$u^x_{(ijt)}$ $u^y_{(ijt)}$	Soil water velocity, x, y direction
$v^x_{(ijt)}$ $v^y_{(ijt)}$	Surface water velocity, x, y direction

Our optimization problem is the following linear program (LP) with variables defined in Table 4.2: The objective function is total water spent over a period N_t of time. We define $f_k(t)$ as a binary function that equals 0 if sprinkler k is off at time t and 1 otherwise, then the water spending is proportional to $\sum_{k=1}^{K} \int_0^{N_t} f_k(t)dt$. Discretizing over time gives as objective function $\sum_{k,t=0}^{K,N_t} f_{kt}$, where f_{kt} are $K \times N_t$ optimization variables (the sprinklers' schedule).

We have as additional optimization variables the values of $h, \theta, u^x, u^y, v^x, v^y$, (6 variables) at each spatiotemporal cell, with a total of $N_t \times N_x \times N_y \times 6$ variables. So the complete set of *optimization variables* is $\{f_{kt}, h_{ijt}, \theta_{ijt}, u^x_{ijt}, u^y_{ijt}, v^x_{ijt}, v^y_{ijt}\}_{i,j,t=0}^{N_x,N_y,N_t}$.

The equality constraints arise from the necessity of the joint values of these variables to satisfy the fluid flow PDEs everywhere in time and space. There are 6 PDEs, hence we have 6 equality constraints for each spatiotemporal cell. They are given by the linearized, discretized PDEs of Appendix A.2. As each constraint involves only 4 variables because of the spatial neighborhood relation

induced by the finite differences, the matrix of equality constraints is sparse.

As discussed in Section 4.2.3, we initialize our soil fluid levels at time $t = 0$ with the most recent soil moisture levels sampled across the space with our distributed sensors, upsampled to the spatial discretization level of our PDE model with a bilinear interpolation. This upsampled soil moisture snapshot is defined in Table 4.2 as θ_{initial} and is used to constrain the soil moisture levels at time $t = 0$ as shown in Equation 4.10e. Similarly, we constrain the surface water height to be zero at all locations at the beginning of irrigation as shown in Equation 4.10f, as irrigated moisture will have been absorbed into the soil well before the next irrigation is to occur.

The inequality constraints define the goal state of the system, namely a schedule must provide volumetric water content in the soil exceeding the minimum water content plants need, θ_{pwp}, and be within prescribed lower and upper limits θ_{l} and θ_{u}. These constraints are of the bound type, i.e., they have the form "variable $\leq$ constant" for each variable. The LP is defined as follows:

$$\min_{\{f_{kt},h_{ijt},\theta_{ijt},u^x_{ijt},u^y_{ijt},v^x_{ijt},v^y_{ijt}\}_{i,j,t=0}^{N_x,N_y,N_t}} \sum_{k=1}^{K}\sum_{t=0}^{N_t} f_{kt} \quad \text{s.t.} \tag{4.10a}$$

$$0 \leq f_{kt} \leq 1 \qquad {\scriptstyle k=1,\ldots,K,\ \ t=0,\ldots,N_t} \tag{4.10b}$$

$$\theta_{\text{l}} \leq \theta_{ijt} \leq \theta_{\text{u}} \qquad {\scriptstyle i=0,\ldots,N_x,\ \ j=0,\ldots,N_y,\ \ t=0,\ldots,N_t} \tag{4.10c}$$

$$\theta_{\text{pwp}} \leq \theta_{ijN_t} \qquad {\scriptstyle i=0,\ldots,N_x,\ \ j=0,\ldots,N_y} \tag{4.10d}$$

$$\theta_{ij0} = \theta_{\text{initial},ij} \qquad {\scriptstyle i=0,\ldots,N_x,\ \ j=0,\ldots,N_y} \tag{4.10e}$$

$$h_{ij0} = 0 \qquad {\scriptstyle i=0,\ldots,N_x,\ \ j=0,\ldots,N_y} \tag{4.10f}$$

PDE model equations (A.2)–(A.7)

The PDE model equations are as calculated in Section 4.2.4.2, and can be found in Appendix A.2.

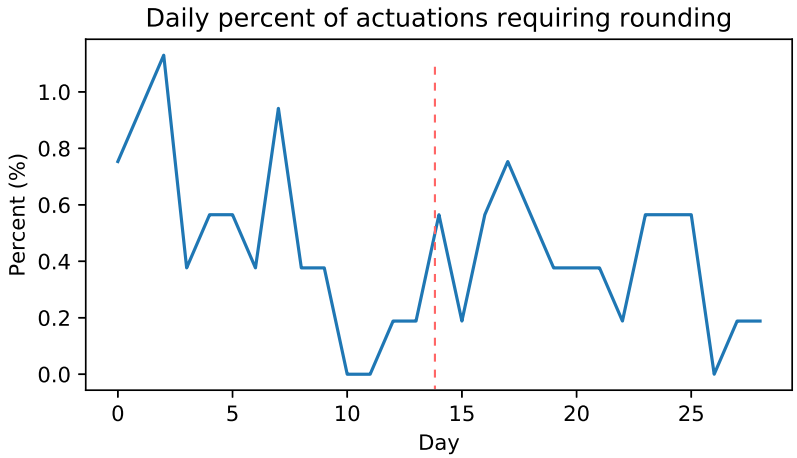


Figure 4.6: Percent of actuations requiring rounding across deployment 1 (left of red line) and deployment 2 (right of red line)

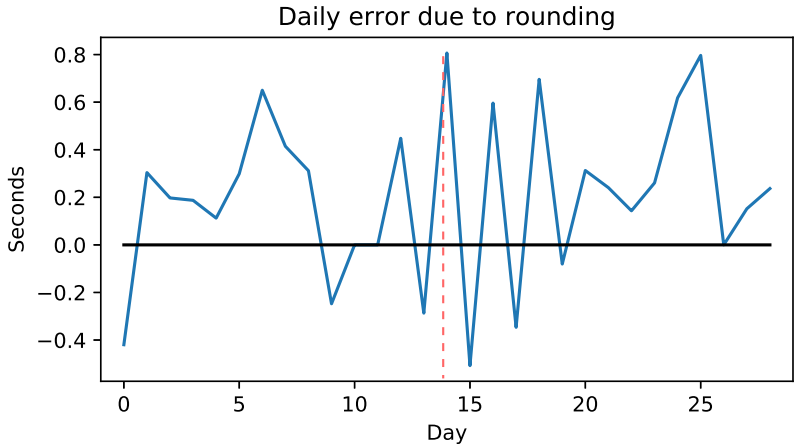


Figure 4.7: Daily effect of rounding on sprinkler on-time across deployment 1 (left of red line) and deployment 2 (right of red line)

The optimization variables f_{kt} represent the binary actuation of a physical water valve, which makes the problem an integer linear program (ILP). ILPs are NP-complete and must be approximated in practice. Here, we simply relax them to the continuous LP by letting each variable be real in [0,1] and then rounding them to integer values for use in the physical system. Although other approaches exist that can give better approximations (such as branch-and-bound), they are impractical for the size of our problem. We found across our two deployments that the optimal schedules tend to leave the sprinklers on for long periods of time, with optimal values of f_{kt} on the 0/1 boundary. Figure 4.6 shows the percent of f_{kt} values that do NOT lie on the 0/1 boundary, and thus require rounding; it can be seen that on the worst day, just over 1% of actuations requiring rounding. By then comparing the optimal schedules before and after their values are rounded, we can determine the effect this rounding has on the schedules used for irrigation. Figure 4.7 shows that on the worst day of our experiments, this rounding caused less than 1 second of deviation in our schedules, which is insignificant as irrigation takes nearly an hour each day. Interestingly, a higher percent of actuations require rounding in the first deployment, but does not lead to an increased effect on the sprinkler on-time.

The discretization in space and time results in a large number of variables and constraints. For example, using a coarse spatial grid of 10×10 with 100 timesteps results in 10000 cells and so 60000 variables (plus $100\times K$ schedule variables for K sprinklers), 60000 equality constraints and 10000 inequality constraints. Fortunately, the equality constraints are sparse and the inequality constraints are simple bounds. In the initial use of this system [WWB16], Stanford's CVX convex optimization library [GBY08] was used to solve the LP due to its convenient programmatic interface. However, we quickly realized that its performance suffered for large optimization problems such as ours, taking on the order of an hour

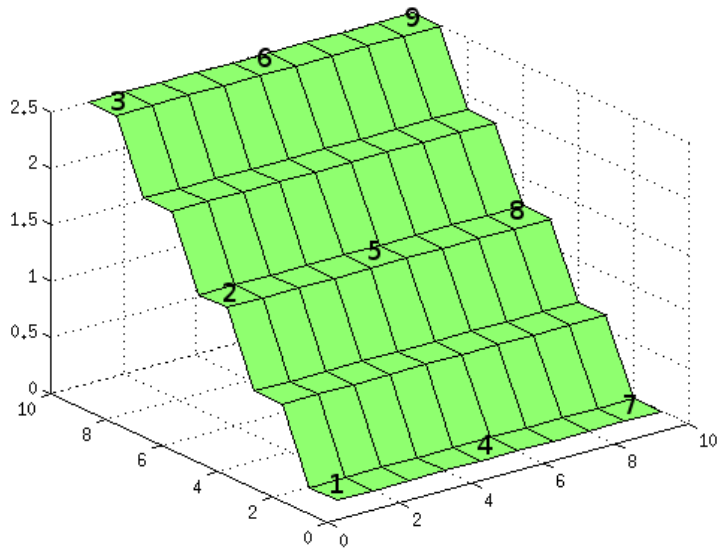
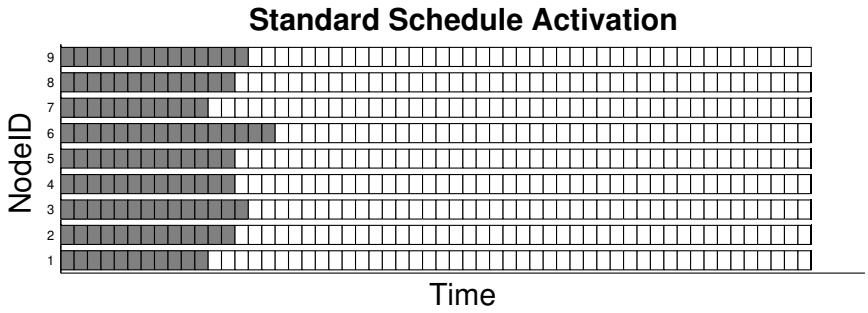


Figure 4.8: Sensing and Actuation Node Locations

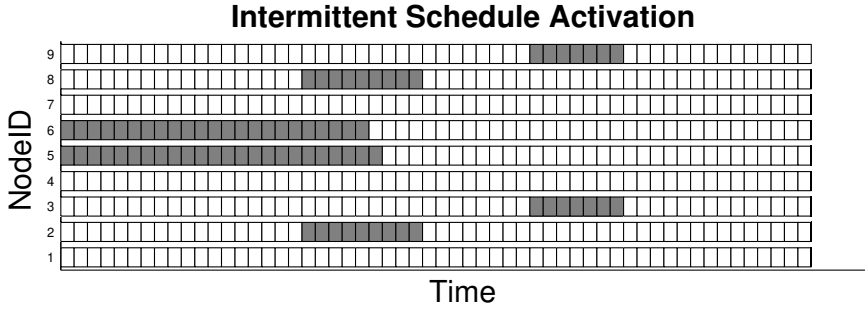
to find optimal schedules even at a very low spatial/temporal discretization level. Most of this runtime was spent setting up the constraints and variables. The optimization codebase was later ported to the Julia programming language [BEK14] for its similar ease of use and much improved speed (both in setting up the data and in solving the actual LP). A performance comparison between these two tools can be found in Section 4.3.2.

4.3.1 Proof-of-concept Simulation

As the model is integrated into the optimization framework, we can perform a proof-of-concept experiment to ensure optimization produces schedules that follow intuition. Figure 4.8 shows the topography used to test and node locations, made to resemble the hillside used in our case study, as shown in Figure 4.11. The hillside was modeled with soil characteristics as would be found in the deployment



(a) Optimized Schedule without Intermittency



(b) Optimized Schedule with Intermittency

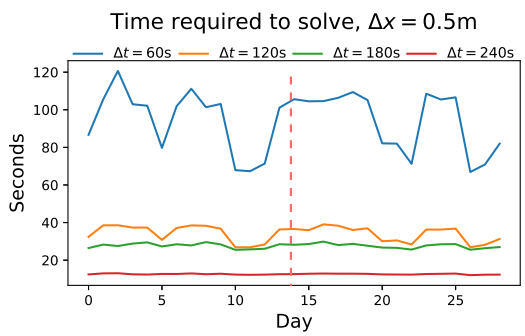
Figure 4.9: Example Optimized Schedules

location, and the optimization was performed. The schedule produced can be seen in Figure 4.9a, where the dark blocks correspond to active sprinklers as time progresses on the x axis. As this is an example problem, the time is purposefully unitless as it does not represent a real scenario. This schedule can be observed to favor irrigation at the top of the hill, and favor Nodes 1 and 7 least, as they are located in the bottom corners. This follows intuition, as any unused water at the top of the hill will move away as runoff, and benefit the downhill turf.

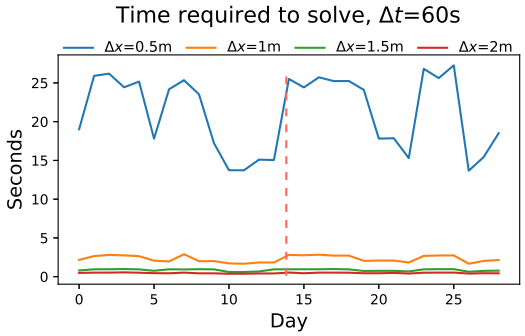
To continue the example, the soil was adjusted to mimic a less-absorbant clay, and reduce its thickness. The thin soil requires less water, but as the clay is less absorbant, watering all at once would cause most water to be lost as runoff. The schedule produced in response to this environmental change is shown in Figure 4.9b, with the dark blocks corresponding to active sprinklers. The optimizer finds a solution that causes actuation to occur intermittently, making irrigation non-continuous. As the less-absorbant soil causes runoff to occur much more dramatically, the optimal solution prevents the lower sprinklers from actuating at all, allowing the runoff from above to provide adequate moisture to the region below.

4.3.2 Effect of Model Granularity

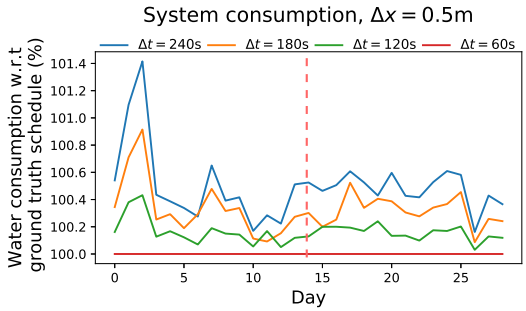
As discussed in Section 4.2.4.2, we must select both a temporal and spatial discretization level for our discretized moisture movement model, which will affect both the tractability of the problem and the accuracy of the model. Although the use of the slower CVX during our physical deployments prevented a thorough analysis, with the initial moisture conditions of each day of our deployments we can re-create the identical optimization problem that would have been run on each day of our deployment in the new Julia optimization framework and vary



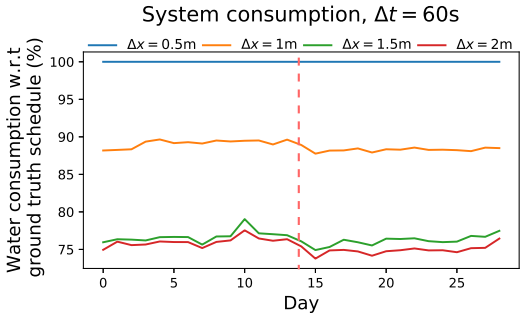
(a) Time required to optimize schedule as the model's temporal granularity is varied



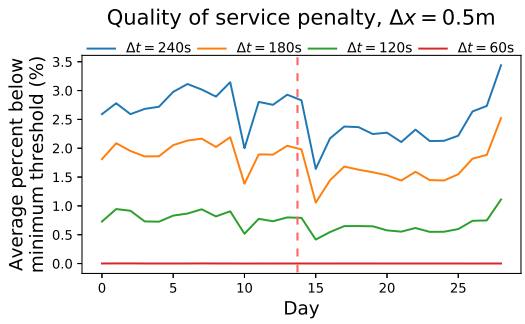
(b) Time required to optimize schedule as the model's spatial granularity is varied



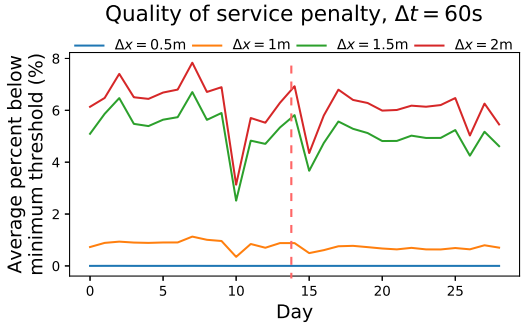
(c) Simulated system water consumption as temporal discretization is varied



(d) Simulated system water consumption as spatial discretization is varied



(e) Degradation of quality of service as temporal discretization is varied



(f) Degradation of quality of service as spatial discretization is varied

Figure 4.10: Discretization effects across first (left of red line) and second (right of red line) deployments. Lower is better for all metrics.

the discretization. The resulting schedule could then be run using the maximum spatial-temporal granularity model available as ground truth, to see the effect of using a lower granularity model when solving the optimization problem and corresponding solution schedule.

To consider the effect granularity has on performance, optimization is run for each day of the deployment using several temporal and spatial discretization levels, and the run-time is recorded as shown in Figures 4.10a and 4.10b, respectively. When varying the spatial discretization, we fix the temporal discretization to a very fine level, and likewise when varying the temporal discretization we fix spatial discretization to a very fine level. As we expect, a very fine discretization level in both time and space results in the highest run-time of just a couple of minutes, with reduced granularity allowing the problem to be solved much faster, as a rougher discretization reduces the number of optimization variables and constraints that must be evaluated by the optimizer. In addition, we consider the run-time of the CVX framework used in our deployments in comparison to the Julia framework; we find that at the standard spatial and temporal discretization level used in our deployments, CVX takes around 40 minutes to optimize the schedules. In comparison, we found Julia is able to solve the same optimization problem in just under 1 second, a performance improvement of about 2400x. Using the Julia optimization framework, we found that we could solve optimization problems with significantly more constraints and optimization variables much faster, but as the problems grow very large, the limiting factor eventually becomes system memory on the machine running the optimization. For instance, considering optimization of our schedules with spatial granularity of .1m and temporal granularity of 15 seconds, the number of optimization variables exceeds 28M, with a similar number of equality and inequality constraints. While this LP can be solved within tens of minutes using Julia, reasonable as optimization

takes place only once per day, the Julia process solving this optimization reports that around 12G of system memory is required.

It is clear that using a model with a coarse granularity has performance benefits, but we also want to see how this sacrifice will affect model accuracy. The best way to determine this effect would be vary the temporal/spatial granularities and re-run the experiments. However, as each experiment takes weeks, this would severely limit our ability to thoroughly search the full range of discretization intervals. Instead, for each day of experimentation, we solve the optimization at varying temporal and spatial granularities, and then run the resulting schedule through our model at the highest granularity to act as a simulated *ground-truth model*. By comparing the computed schedule to one computed with the ground-truth model we can see the effect it has on system efficiency, and by finding on average how much each simulated location falls beneath our desired moisture level at the end of irrigation we can see its effect on quality of service.

The change in irrigation water consumption as the temporal discretization is varied can be seen in Figure 4.10c, and the change in quality of service can be seen in Figure 4.10e. We can see that as the temporal granularity is made more coarse, the resulting schedules consume slightly more water but on average, we see that the quality of service slightly degrades as well. As the causes of water movement in the space occur at different timescales, the timing of water application to the soil can have a strong effect on the final distribution of soil moisture. This is an effect described in Section 4.3.1, where variations in environmental characteristics can make it necessary to utilize more advanced irrigation schedules, for instance through intermittent actuation. However, as the model used in optimization becomes more and more temporally coarse, it becomes more difficult to accurately anticipate the timing requirements of irrigation, and can result in a decreased

quality of service, despite using slightly more water. As neither increased water consumption or degraded quality of service is desired, it is clear that the finer the temporal granularity used to compute schedules, the better.

In comparison, varying the spatial granularity has a more profound effect; the simulated change in system efficiency can be seen in Figure 4.10d, and quality of service can be seen in Figure 4.10f. Reduction in spatial granularity removes spatial locations from the moisture movement model, meaning their moisture requirements are no longer considered when optimizing schedules. As shown in Figure 4.10f, this results in a significantly decreased quality of service, with locations across the space missing their required moisture levels by an average of almost 10%. However, by underestimating water needs in the irrigation system, the system also requires less water to reach the perceived (although incorrect) moisture requirements. As Figure 4.10d shows, this leads to significant water savings in the system, with as much as 25% less water consumed with the roughest spatial granularity. A sacrifice in the model's spatial granularity causes a clear tradeoff between quality of service and system water consumption. It is difficult to estimate the long-term effects of this degraded quality of service on the health of the plant, and it is possible that very hardy grass species may survive, potentially resulting in water savings in the system. In general, however, maintaining the high quality of service is absolutely necessary in maintaining plant health, and this is achieved by using a model with the finest spatial granularity.

4.4 Case Study: Live Deployment

In many applications, it is possible to compare a newly-developed model to other accepted models. However, as an all-inclusive model for our application does not exist, we compare to reality by evaluating the performance of an irrigation system

Figure 4.11: Deployment side-view

using the control modifications we have proposed.

We chose to deploy two systems side-by-side once a suitable location is found. Ideally, the two systems will cover similar soil and turf, face the same direction so that sun exposure will be equivalent, and have completely independent irrigation to avoid cross-contamination. In addition, to ensure all sprinkler coverage is the same, the same water source is used to power both systems, and actuation is provided to both sides with the installation of our DICTUM nodes. The only difference between the systems are the control schedules sent to each side.

In the beginning of our project, we intended to use an existing irrigation system to perform our deployment. In looking for a suitable location, we came to realize that the granularity of irrigation control on our University's campus was less than ideal. Locations spanning more than ten thousand square feet across heterogeneous terrain were actuated by a single control valve. As such, it would not be feasible to show the benefits of higher-granularity actuation using the existing system, so we began planning a custom irrigation system.

4.4.1 Seeking a Suitable Location

With the help of University personnel, a suitable location was found on a stepped hillside (see Figure 4.11) far away from the nearest foot-traffic. The hill, rising about 9 feet over a distance of 70 feet, acted as an elevated surround for a University soccer field, the hill stretching more than 200 feet. To take advantage of this topography, it was decided to place two irrigation systems side by side, each spanning a 70'x70' area along the hillside. Between the two systems were 5 feet of unirrigated space, to prevent any spray from one system to enter the other side. As a whole, the two irrigation systems spanned an area of 70'x145', approximately 10,150ft^2.

4.4.2 System Development and Deployment

The underground irrigation system used by groundskeepers to maintain the hillside we deployed on was plumbed with high-flow PVC water lines leading to each of their sprinklers. As it would be risky to tap into this permanent infrastructure to test our temporary system, we used lower-flow quick-couplers, fitted with standard hose spigots. We discovered that the nearest quick-coupler could provide enough water flow to activate one of our side-by-side irrigation systems, but not both simultaneously. Throughout our case study, irrigation of the two systems is performed one immediately after the other to avoid this issue.

To choose the type and number of sprinklers for our side-by-side deployment, we consulted sprinkler manufacturer specifications; a general rule of thumb for system planning is that the coverage of one sprinkler should reach 75-100% of the distance to the next closest sprinkler, to avoid uncovered areas on the diagonal. To cover one of the two proposed areas covering 70'×70' with these recommendations, we would require four sprinklers in a 2×2 grid, each with a reach exceeding

52 feet, or 9 sprinklers in a 3×3 grid, each with a reach exceeding 26 feet. Sprinklers that can exceed 52 feet are generally of the "rotor" variety, pointing in only one direction and rotating slowly to cover the region. The flow required by these larger rotors is typically more than our quick-coupler water source could provide. It was found the the low-flow MP-Rotator 3000 sprinkler [hunc], a new rotor that is known to be remarkably efficient, could reach up to 30', 9 of which can be easily powered by our quick-coupler water source. As the MP-Rotator is quickly replacing older sprinkler technologies at our University for their slow-application and minimum runoff, we decided they would be the best choice for our deployment.

As the deployment was meant to be temporary, we designed it such that it could be removed in a reasonable amount of time. Although a commercialized version of DICTUM would be installed in the ground, our temporary system was placed on the surface for ease of access to our prototype. The nodes themselves were placed on the ground just next to the sprinklers, while the solenoid providing actuation was fixed to the sprinkler riser. By placing our system on the surface, we are able to avoid damage to the lawn that installing into the ground would cause.

Lastly, our deployment included a central basestation fitted with power, a small Sheeva Plug computer [she], an elevated 802.15.4 mote to receive data from the sensing nodes, and a 4G hotspot, to allow us to communicate with the wireless sensor network from a remote location. Although our basestation was overbuilt to facilitate ease of debugging and close monitoring of the prototype system, a commercialized system may have only the mote to interface with the sensor network, and an interface to any external service (local, cloud, etc).

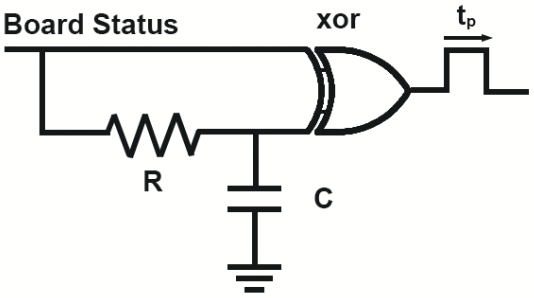


Figure 4.12: Example safety mechanism for latching solenoids

4.4.3 Node Development

In our first deployment, our device was fitted with an ordinary solenoid designed to regulate the flow of high-pressure water. This solenoid was found to work very reliably, and as it defaults to an off state when power is cut, it is also a safe choice. However, its weakness was in its high power requirement, as it must pull a constant 350mA to hold the valve open. Later discussed in Section 4.5.4, this dominates the power profile of the device and caused the node lifetime in our first deployment to fall under 1 week. Additionally, as the power supply in our first hardware version was unregulated, the collected sensor data would begin to sag at the end of battery life as the sensor's supply voltage would drop. We mitigated these weaknesses during our first deployment with frequent battery changes to prevent this sensor sag from occurring.

These problems are addressed in the hardware version used for our second deployment, where we chose to use a latching solenoid for sprinkler actuation to extend system lifetime. Whereas a normal solenoid requires constant power to stay in an open position, a latching solenoid requires only a 50ms pulse of either positive or negative voltage to switch between an open or closed state. One advantage to a non-latching solenoid is the inherent safety; a loss of power

in the control board will automatically deactivate water flow. To provide this safety in a latching solenoid, a simple circuit can be built into the control board that will close the solenoid if the board fails, as shown in Figure 4.12. A pin is maintained on the microcontroller that changes from 0 to 1 in event of failure (or from 1 to 0, a NOT gate can be used to invert it). Once this pin is active it immediately brings the top pin of an XOR logic gate to 1, allowing current to flow through while the capacitor on the bottom pin charges. Once the capacitor charges, the second pin of the XOR logic gate activates, terminating the current. This pulse, generated while the capacitor charges, is for a period $t_p \propto CR$, where C and R are the capacitance and resistance values used in the circuit. This pulse will deactivate the solenoid, preventing additional water consumption after node failure. With this configuration, the control node will benefit from the extremely low-power operation of a latching solenoid without sacrificing safety.

The main challenge in the development of the DICTUM node was the organization of multiple input/output connections from the mote. To address this, we manufactured custom printed circuit boards that would organize the connections from the tmote sky. This interface board, shown with the rest of our prototype device in Figure 4.13, has connections for battery power, sensor (right) and solenoid (left). The different voltages required by the mote and solenoid were provided by voltage regulators built into the board. As the chosen latching solenoid requires the board to produce a positive (opening) and negative (closing) voltage for operation, the board was equipped with an h-bridge. Commonly used in robotics to drive a motor forwards and backwards from a single dc power source, an h-bridge is designed for applications like ours that require bi-directional current.

The other key feature of the DICTUM node is the ability to measure the volumetric water content in the surrounding soil. We opted to purchase research-

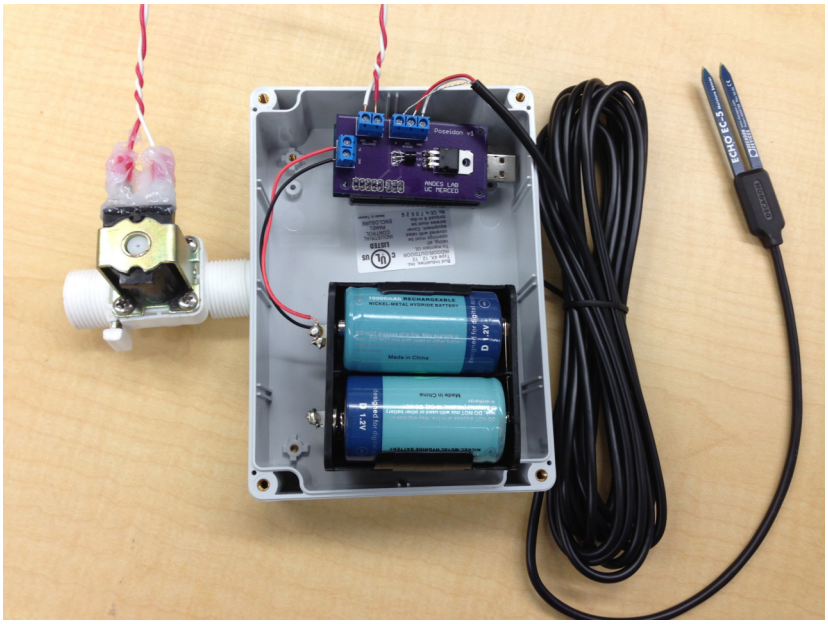


Figure 4.13: Sensing/actuation (DICTUM) node

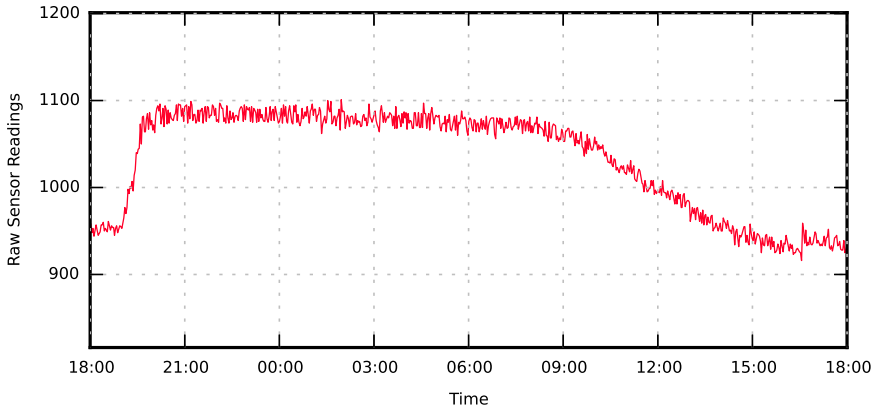


Figure 4.14: Daily soil moisture cycle

quality Decagon EC-5 [dec] sensors, with a reported accuracy of ±3%. The Decagon sensors cost roughly $110, but data fidelity in our model verification was paramount. Raw sensor readings collected over a period of one day with a high sampling frequency can be seen in Figure 4.14. The sensors report the dielectric constant of the soil, an electrical property highly dependent on the volumetric water content. A linear calibration function provided by the sensor manufacturer is used to convert the raw readings to volumetric water content. An attentive reader will notice the sharp increase in the late afternoon due to irrigation, stable readings throughout the night, with sensor fall beginning as the sun comes up at 7:45am. This decrease steepens as the sun rises and faces the deployment directly after 10am. In a sensor-dense environment, a more economical alternative to the EC-5 may be chosen, such as the 35$ Watermark sensor [SB92], which can be calibrated to ±5% accuracy.

4.4.4 System Comparison

In this study, we compare the operation of the DICTUM system against two baseline control systems. The first baseline, evaluated across 2 weeks of fine-grained data collection, employs a trial-and-error control strategy used widely in practice (including our campus). In this technique, a greenskeeper will monitor an irrigation system for days or weeks; if an excess of runoff provides evidence of over-watering, or if brown patches provide evidence of under-watering, they will adjust the system accordingly. This irrigation scheduling, often remaining unchanged through entire seasons, leads to a misuse of water as it does not account for changing weather or soil requirements. We emulate this strategy by matching exactly the amount of water coverage as would be provided by the greenskeepers of our campus.

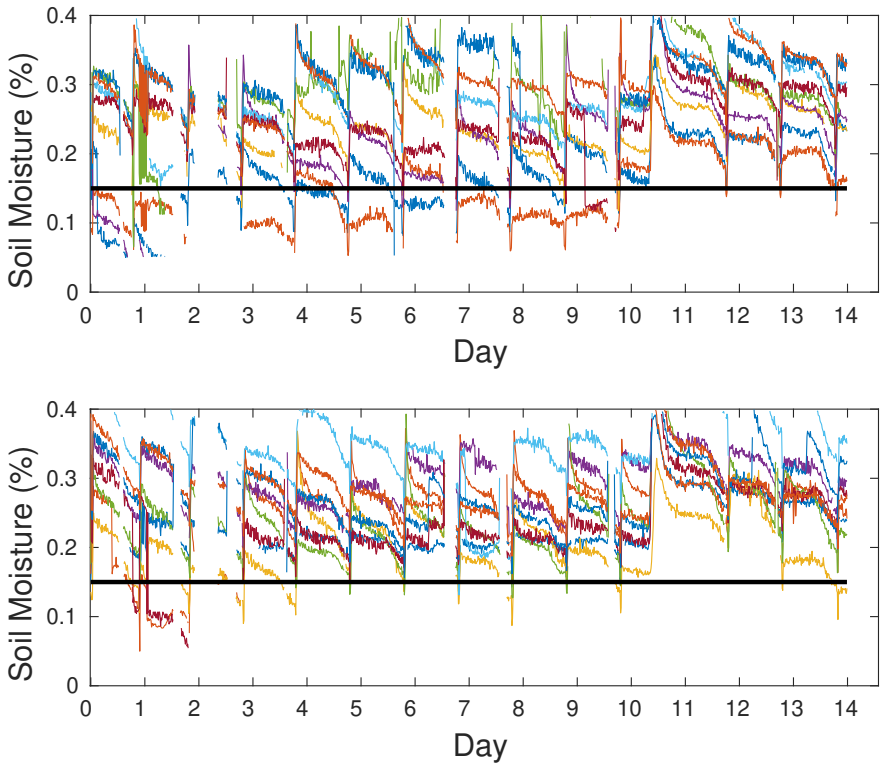


Figure 4.15: Deployment sensor trends for all Campus (top) and DICTUM (bottom) systems as collected by the installed DICTUM nodes

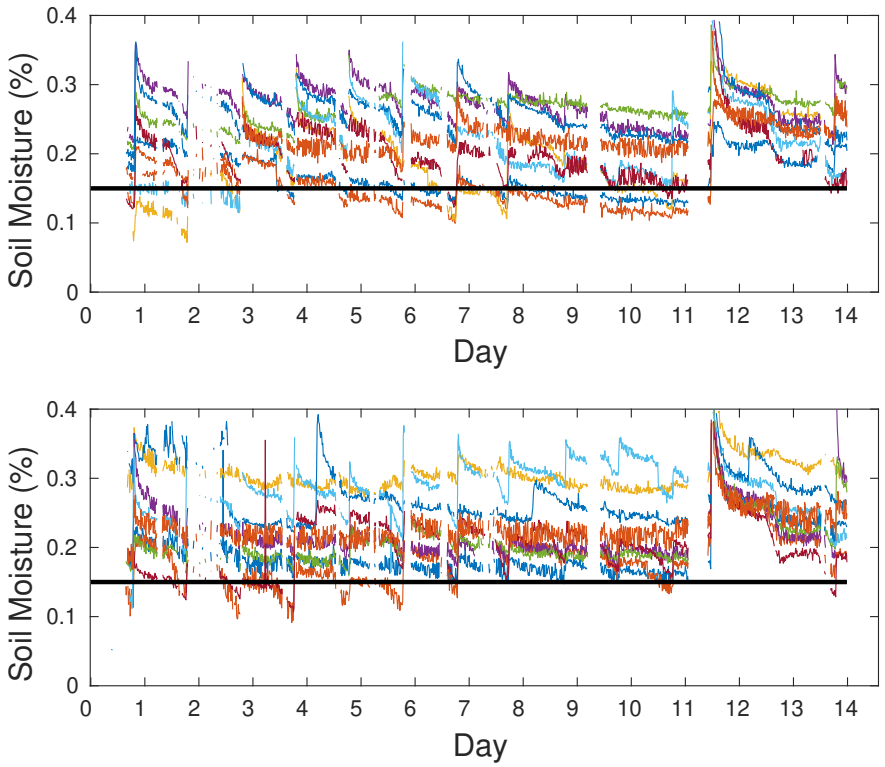


Figure 4.16: Deployment sensor trends for all Evapotranspiration (top) and DICTUM (bottom) systems as collected by the installed DICTUM nodes

The second baseline control strategy, evaluated in a 2-week deployment, employs a state-of-the-art evapotranspiration (ET) control strategy. As described in Section 2.1.1, these systems use weather forecasting to estimate the amount of water lost by the soil due to evaporation and plant transpiration. Irrigation controllers that use ET technology typically irrigate every 1-3 days, replacing water lost over that period. To emulate an ET system, we query a local weather station that computes hourly ET loss, and compute the previous day's water losses. With our sprinklers' surface coverage rate, we create daily valve schedules to do exact replacement of these losses.

4.5 Experimental Results

Through 4 weeks of fine-grained data collection, we ran two irrigation systems side-by-side on identical patches of turf, periodically collecting soil moisture data from each. In the first deployment, our campus' control strategy was tested and for the second deployment, state-of-the-art evapotranspiration (ET) control was used. In both deployments, these systems were compared against schedules computed by our model-based optimization, actuated using our custom-made independent actuation/sensing platform (DICTUM) nodes. The goal of these case-studies was to determine if a system could be made that reduced the amount of water used, while maintaining a satisfactory level of moisture in the soil for the turf to remain healthy.

4.5.1 Quality of Service

The primary objective of an irrigation system is to maintain plant health. A very efficient system that is unable to meet this objective will be replaced with a less-

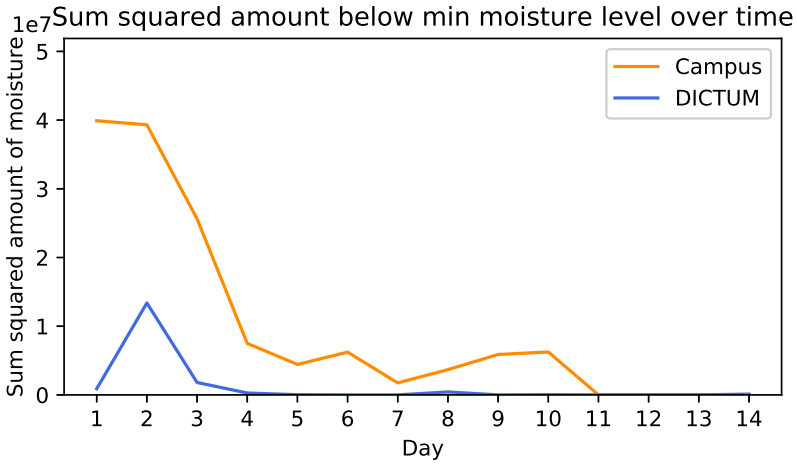


Figure 4.17: Daily quality of service vs Campus strategy (lower is better)

efficient system that provides satisfactory water coverage to the turf. To remain healthy, turf needs proper soil nutrients in the appropriate amounts, adequate solar irradiation to power biological processes, and an adequate level of moisture in the soil near the plant roots. Although the irrigation system has no control over solar exposure or soil nutrients, we have direct control of soil moisture through actuation of the sprinklers. In plant physiology, the level of soil moisture at which plants can no longer extract water from the soil is known as the permanent wilting point (θ_{pwp}) [MD17, usd, Kir04]. Long-term exposure to soil with moisture below this level will cause the turf to wilt and die, so we aim to minimize the amount of time spent beneath this threshold as it will ensure a better opportunity for the plant to thrive.

The permanent wilting point (θ_{pwp}) for loamy soils like that found in our deployment is typically between 10-15% [pwp], so we assume the worst case and assign θ_{pwp} to be 15%. We expect that if DICTUM were to distribute moisture in a smarter way by targeting areas that would otherwise receive inadequate water, the DICTUM sensor readings will spend less time underneath the θ_{pwp} thresh-

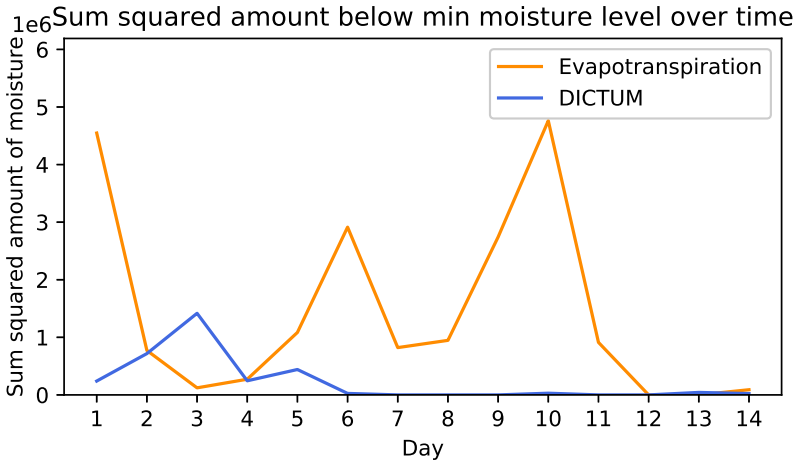


Figure 4.18: Daily quality of service vs evapotranspiration strategy (lower is better)

old than the compared baseline. The sensor data collected by the distributed DICTUM nodes from the collection period and the minimum healthy saturation θ_{pwp}=.15 (black line) are shown for our first deployment against the campus control strategy and our second deployment against the ET control strategy in Figures 4.15 and 4.16, respectively.

We can see that in both deployments, sensors in the DICTUM system spend significantly less time beneath the minimum moisture threshold than the compared baseline. Interestingly, we can see that seasonal rain was experienced once during each deployment, on day 10 in Figure 4.15 and on day 11 in Figure 4.16. In both cases, this resulted in a steep increase in all moisture sensor values which continued into the following two days. In our first deployment, the Campus and DICTUM control strategies spend a total of 56.9 hours and 16.3 hours beneath θ_{pwp}, respectively, indicating a 3.5x improvement in quality of service. Similarly, in our second deployment, the Evapotranspiration system spends a combined 68.1 hours beneath the θ_{pwp} across the entire deployment, over 4x more than the

16.7 hours experienced by the DICTUM system.

Interestingly, we can see in both side-by-side deployments that there were occasional periods of data loss, on days 2, 6, 7, and 9 in Figure 4.15 and days 2 and 11 in Figure 4.16. Our system included a basestation Sheeva Plug [she] computer that acted as a bridge between the public internet and our wireless sensor network, with data forwarded over a usb connection to an attached telosb. Occasionally, this USB connection would fail at night, resulting in corrupted data transmission between the two devices; we noted that this particularly occurred when the basestation was exposed to cold ambient temperatures at night, but further investigation is required to determine the exact cause of this failure. Important to note is that as the data connection would re-establish itself when conditions improved and a snapshot of data is only required as initial conditions to optimization once per day, these lapses in data had no effect on the operation of the DICTUM system.

Although the amount of time spent beneath the minimum moisture threshold gives a good indication of quality of service, it signifies that all time spent beneath the threshold is equal. In reality, the further under the minimum moisture threshold, the worse the health of the plant will become. To take this into account, we consider the sum squared amount of time spent beneath θ_{pwp}. In this way, the system will be punished more the further below the minimum moisture threshold. The sum squared time beneath θ_{pwp} can be seen across our first deployment in Figure 4.17, where the campus control strategy is significantly worse than the DICTUM system until day 11, when the seasonal rain pushes the soil moisture levels into the acceptable range. In total, the campus control strategy is 8.17x worse than the DICTUM system in this metric. Likewise, we evaluate our second deployment as shown in Figure 4.18. On day 3, we can see

that the DICTUM system quality of service dips temporarily, but on all other days provides significant improvement in comparison to the evapotranspiration system. We can also see on day 12 than the rain on the previous day pushes the soil moisture levels to acceptable levels across the space. Across this deployment, the evapotranspiration system is 6.28x worse than the DICTUM system with respect to this metric. Although these analyses shows that DICTUM is not perfect, it also demonstrates that more precise watering strategies can provide a significantly improved quality of service, despite using less water, as we will see in the following section.

4.5.2 Water Consumption Analysis

Throughout the deployment, the total on-time for each sprinkler was recorded in both systems. With knowledge of each sprinkler's distribution angle and the regulated water pressure, we can accurately estimate the amount of water consumed under any given schedule.

In comparison to the campus irrigation system, DICTUM is consistently more efficient as shown in Figure 4.19, due to the campus irrigation system using a fixed schedule. We can see, however, that as our campus irrigation system is equipped with a rain detection sensor, the 11th day of irrigation is completely disabled in response to that day's precipitation. Irrigation is also disabled here in the DICTUM system, but this is in response to significantly higher soil moisture levels removing the need for irrigation. We can see that whereas the campus strategy continues providing full irrigation on the following days, the DICTUM system is able to significantly reduce the amount of irrigation in response to continued elevation in soil moisture levels for the last 3 days of system deployment.

A side-by-side comparison can also be seen for the deployment against the

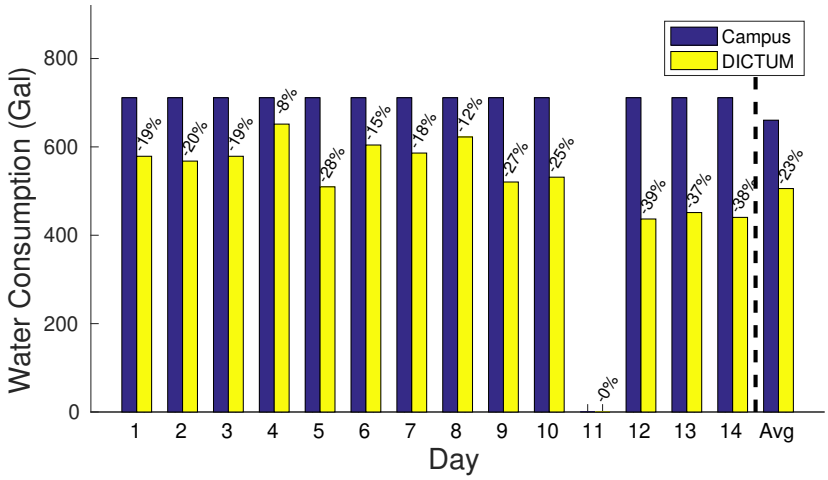


Figure 4.19: Daily water consumption of Campus and DICTUM systems

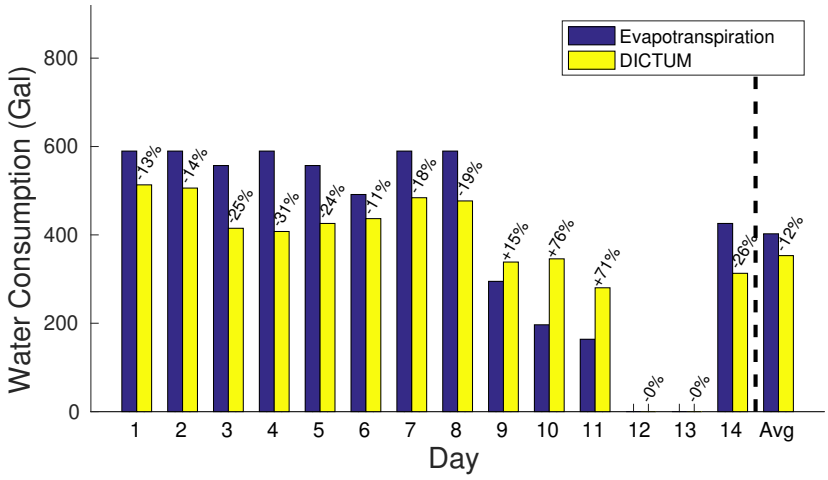


Figure 4.20: Daily water consumption of Evapotranspiration and DICTUM systems

evapotranpiration controller in Figure 4.20. The variation of water consumption of the Evapotranspiration system indicates changes in local weather. Low water consumption of both systems on days 9, 10, and 11 is due to cooler local weather, where the ET side consumed significantly less water. However, we can see in Figures 4.16 and 4.18 that these days of reduced water consumption also result in the worst days of quality of service, meaning the ET controller was too aggressive in its water savings on these days. Interestingly, a similar effect can be seen on day 6; the ET controller reduces water use in response to weather conditions, but at the cost of a reduction in quality of service as shown in Figure 4.18. Rain on the 12th day caused both systems to cease irrigation until two days later, when the soil became dry enough to require it.

Across the two deployments, the DICTUM system consumed 12.3% less water than the evapotranspiration strategy, and 23.4% less in comparison to our campus' control strategy. Schedules created by the DICTUM system were shown to consume as much as 651 gallons and as little as 280 gallons, a 371 gallon variation in response to the state of soil moisture. The reason our system is able to save water while providing a higher quality of service to the space is due to our ability to pinpoint regions within the irrigated space with varying moisture requirements. Using our model-driven approach, we can optimize actuation to send more water to areas that would otherwise receive insufficient moisture, while sending less water to the areas that would otherwise be over-watered.

Sensor readings through rain during our first deployment can be seen in Figure 4.21. To respond to such weather events, the weather stations used for evapotranspiration monitoring are generally equipped with a precipitation sensor. These sensors deactivate the irrigation system when rain is first experienced [raia], or when a measured amount of rain has fallen. In the case that the rain is suffi-

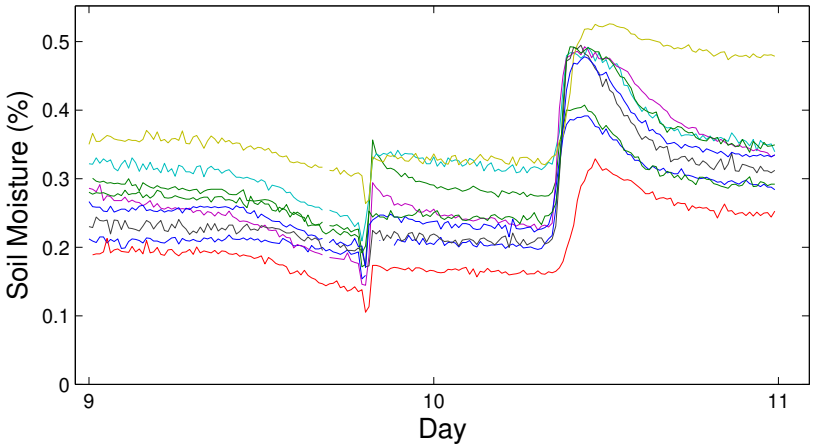
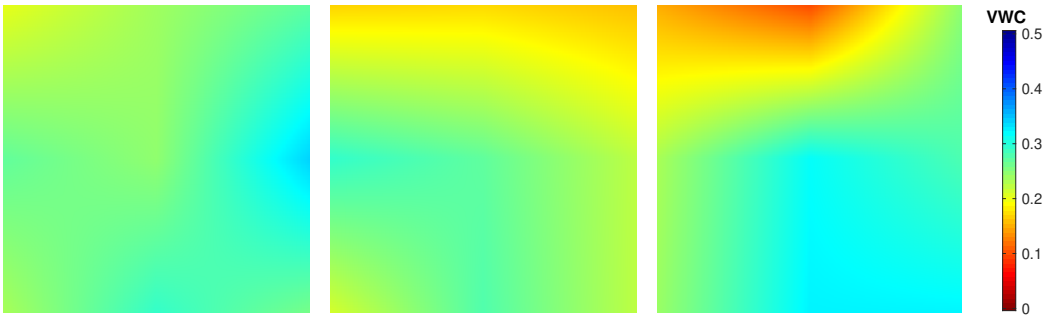


Figure 4.21: DICTUM sensor readings through rain



(a) DICTUM (b) Evapotranspiration (c) Campus

Figure 4.22: Average moisture coverage of compared systems

cient to irrigate the turf, this is satisfactory, but if the rain is very light, DICTUM might choose to provide additional irrigation.

4.5.3 Moisture Uniformity

An interesting pattern recognized throughout the two deployments was the emergence of moisture uniformity. The optimization performed by DICTUM will create schedules that compensate for any areas of high or low moisture levels using soil moisture and topographical information. However, due to lack of moisture information and the physical limitations of the irrigation system (inability to actuate individual sprinklers), the ET and our campus' systems are unable to correct for uneven moisture. These artifacts can be seen in Figure 4.22, which shows a spatially-interpolated view of the average moisture level of the three considered systems across our two deployments.

In this figure, the top of the image aligns with the uphill region depicted in Figure 4.8. We can see that as the irrigated water tends to flow downhill, the Campus and ET systems end up with inadequate soil moisture at the top of the hill, and in the Campus system we can see extra water building up at the bottom of the hill. In comparison, the topographical effects are accounted for in the schedule creation of the DICTUM system, as more uniform moisture can be seen across the space with exception of a small spot on the right hand side where a bit of extra water can be seen to accumulate. Although variation in seasonal weather patterns between these two separate deployments makes a true side-by-side comparison difficult to make, these results demonstrate DICTUM's ability to produce schedules that correct for heterogeneities to provide homogeneous water coverage by taking advantage of distributed actuation.

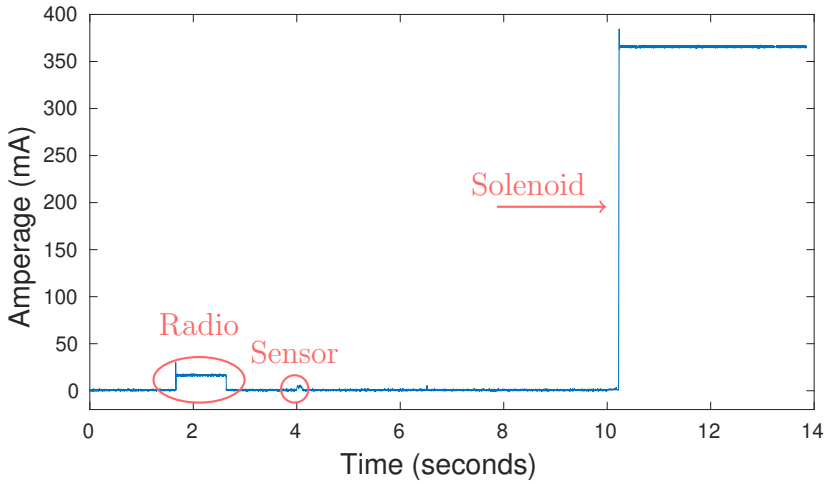


Figure 4.23: Example energy trace of the sensing/actuation node with non-latching solenoid as used in the first deployment. Note that the high power consumption associated with solenoid activation will continue until the sprinkler is deactivated, potentially as long as an hour into the future.

4.5.4 Energy Consumption Analysis

From a wireless sensor network standpoint, the ability of a system to operate for a long period of time without user intervention is fundamental. Irrigation control devices are no different, especially if they are meant to be buried in the ground. The first version of the devices used a non-latching solenoid, which requires constant power to allow water to flow at irrigation-time, dominating the power profile of the device as shown in Figure 4.23. As a solenoid can be active for as much as an hour each day for irrigation, we found that during our first deployment we had to constantly change batteries, once or twice per week. Our second version greatly improved with the use of a latching solenoid, which required only a short pulse of power to throw the valve between the off and on positions. This allows the device to be truly low-power, as this on-off cycle will only occur a few times per day in the worst case, with a clearly improved power

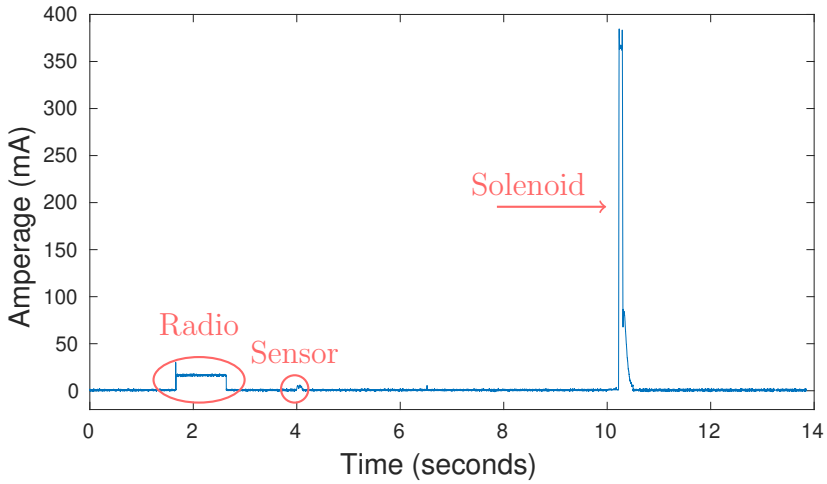


Figure 4.24: Example energy trace of the sensing/actuation node with latching solenoid

profile as shown in Figure 4.24.

For additional energy savings, the radio in each node is duty-cycled, activating for only a 10 second period every 10 minutes. It was computed that using this duty-cycle, the 4 D cell batteries providing power to our nodes could run for over 2 years without requiring change. However, we note that collected data is only required in our processing pipeline just before irrigation each day as initial conditions to optimization. In practice, a much more energy-efficient solution would be to continue sampling the onboard sensors but to store all this data on the mote's flash storage, and send the entire dataset in batch just before irrigation. As these data samples are small, a day's data can be easily sent within a minute of radio on-time each day, allowing our prototype irrigation system to run uninterrupted in excess of 14 years, while still performing its daily irrigation and data collection. The power effect of these peripheral devices on our latest prototype can be seen in Figure 4.24, where radio, solenoid, and sensor power consumption is shown over background cpu usage.

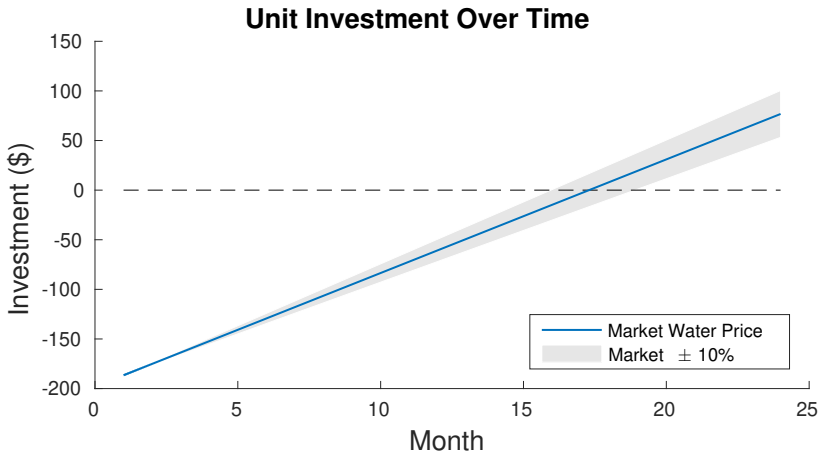


Figure 4.25: Return of Investment timeline with varying water pricing

4.6 Return on Investment Analysis

Installing a system such as ours has many health, political, and environmental benefits that are difficult to quantify, but the economic benefits are crucial when deciding whether or not to install such a system. In particular we must consider the return on investment, or the time it takes a system to save enough money to cover the cost of installation and usage. To calculate the return on investment, we take into account the initial cost of the replacement system and the monetary savings expected from the increased efficiency of the replacement system.

Here we consider the cost to develop a single DICTUM node in bulk for return on investment analysis. The primary components can be readily found; the sensor, solenoid, batteries, and waterproof enclosure are all possible to purchase from other manufacturers. In our prototype the communication module used was a tmote sky [tmo]. However, as our application requires very specific circuitry to provide power to the various modules, commercialization of the DICTUM node would involve the manufacture of a stripped-down communication module, with the inclusion of the additional components described in Section 4.4.3. The pricing

of a barebones tmote sky replacement, as well as the other various DICTUM components, can be found in Table 4.3.

To evaluate the expected return of investment, we compute the cost of the system and calculated the net investment as time progresses. The factor that most influences system pay-back is price of water. As this value is constantly changing, we perform the analysis using the current price for our campus, $5.60 per thousand gallons, and incorporate a deviation of 10% to account for changing water market prices. Although the unit itself carries a high initial cost, return of investment can be expected to occur in 16-18 months, as shown in Figure 4.25. Considering the immense political pressure for irrigation water to be more expensive [calb, cala], it is a good bet that the savings from an irrigation system such as ours will be on the rise in the near future.

It is difficult to directly extend the savings seen in our prototype to all irrigated space on a University's campus due to the heterogeneity of the installed system architectures. However, with slight modifications, the independent actuation control platform can be easily extended to control sprinklers of any type, delivering site-specific actuation for small-large scale systems. For example, on a campus such as ours, the majority of irrigated spaces use rotor sprinklers. As the rotors use substantially more water, independent actuation could provide an even greater positive environmental and financial impact, to be investigated in future work.

4.7 Limitations and Future Work

To simplify the placement of irrigation infrastructure, sprinklers are almost always installed in a grid pattern. However, it might be possible to use the developed

Table 4.3: Sprinkler Node Manufacture Cost

Component	Price
Mote	$37.57
Moisture Sensor	$110
Batteries	$4
Solenoid	$15
Waterproof Enclosure	$10
Manufacture & Assembly	$10
	$186.57

model to optimize sprinkler locations to compensate for natural topological characteristics for a system that has not yet been installed. In future work, we hope to evaluate further potential for water savings by also finding optimal sprinkler positions.

In optimization, we wish for our schedules to guide soil moisture into healthy levels, i.e. above θ_{pwp}. Although we use a value of θ_{pwp} as physically measured and reported in plant physiology literature, in practice it may be better to tune this value by hand. Although θ_{pwp} may have been correct in the original model described in Section 4.2, it will not be correct in the model we optimize due to the approximations we introduce in Section 4.2.4. The method of tuning this parameter and the effect of this choice is left for future work.

To correct for PDE model error, it is possible to perform intermediate schedule optimization using fresher soil moisture data in the middle of the irrigation period. However, with the CVX optimization library used during our experiments, the 40 minute optimization time would be substantially disruptive in the middle of irrigation, as it would cause a long pause for computation in irriga-

tion that the optimization did not find necessary. Furthermore, due to system pressure limitations, groundskeepers often assign a maximum irrigation time to allow different areas to be scheduled in series, in our system chosen to be 3 hours. However, with the more recent port to the Julia programming language, subsequent optimizations can be performed in around 1 second, making this a very reasonable feature for future implementations of the DICTUM system. By reducing the persistent model error through this re-correction, the DICTUM system will be able to more effectively meet its primary goal of maintaining adequate moisture levels, while doing its best to minimize water consumption. Depending on the nature of the model error, it is possible that this re-correction may cause a penalty to the system efficiency in order to further improve quality of service, but as the DICTUM system already has substantially improved quality of service, we expect any penalty to be slight. As further experiments would be required to quantify these potential improvements by adding this feature, we leave it to future work.

As initial conditions of soil moisture are collected at only a finite number of sensing locations and upsampled by interpolation to the spatial granularity of the PDE grid, it is possible that conditions between the sensed locations may deviate from those expected by the model. Unfortunately this can be caused by several anomalous conditions such as differing soil depth, soil type, faulty or incorrectly-installed sprinklers, which will cause model disagreement to the true conditions in the space and may be difficult to notice and correct in the model. However, there are several emerging aerial imaging technologies [CS01, pho] that allow the plant health to be monitored at a significantly higher spatial resolution than our distribution of soil moisture sensors. While they do not allow the soil moisture to be directly measured, necessary for short-term control decisions during irrigation, they will allow the control system to identify regions of the space that have

unhealthy plants. If this plant health data shows regions between sensors that are experiencing poor plant health, it can be used to expose a local anomaly, which can then be incorporated into the PDE model for improved control, or trigger an alert for local groundskeepers to search for a physical system fault. This will allow our system to provide quality guarantees at all locations throughout the space.

Due to the computational requirements of optimizing the schedule, the model only considers moisture movement that occurs *during* irrigation, and as such does not include long-term effects such as leeching and weather which cause the fluid to be lost throughout the rest of the day. This is limiting, as it forces us to assume that all locations in the space lose water in the same way, which may not be true in practice. Future work is necessary to consider these losses across the full 24-hour cycle and include this information when finding irrigation schedules to ensure satisfactory water levels at all times.

4.8 Conclusions

In this chapter, we seek to improve the efficiency of turf irrigation systems by analyzing heterogeneous water needs across a span of turf. To this end, we develop a computationally-light moisture movement model to be used as constraints in an optimization problem, which is then used to produce optimal valve scheduling within an irrigation system to minimize water consumption while maintaining healthy levels of moisture everywhere. To test its effectiveness we produce the DICTUM sprinkler node, with the ability to actuate, sense local soil conditions, and communicate wirelessly with sister nodes in the network. Through two separate deployments spanning a total of 4 weeks, we find that the DICTUM system can reduce system water consumption by 23.4% over our campus' control strat-

egy, and by 12.3% over a state-of-the-art evapotranspiration system. Despite this reduced water usage, DICTUM was also found to reduce turf exposure to unhealthy levels of moisture by a factor of 3.5 over the campus' control, and a factor of 4.08 over the evapotranspiration control. The DICTUM system is expected to return its investment in 16-18 months based on water savings alone.

CHAPTER 5

OPTICS: OPTimizing Irrigation Control at Scale

Chapters 3 and 4 discuss the development of the distributed sensor/actuator node that allows the irrigation system to be actuated at significantly finer granularity than standard irrigation systems. In order to optimize schedules for the distributed actuation system, Chapter 4 introduces a PDE model that predicts the movement of water through the irrigated space subject to surface topography, soil type and depth, and other parameters, and found that the optimized schedules result in both improved efficiency and quality of service. However, due to the cost and difficulty of sampling soil type, soil depth, direct solar irradiance, and other key factors across an irrigated space, in practice an installer can only take a small set of collected values, and make assumptions this it is representative of the conditions across the space, whereas in practice these factors tend to be very heterogeneous. In addition to requiring this manual model generation, the DICTUM control framework offers no model correction over time and does not incorporate future weather prediction, potentially causing reduced accuracy in prediction. Furthermore, the size and complexity of the model and optimization problem resulted in it requiring simplification via linearization and spatio-temporal discretization to make it tractable. Even with these simplifications that sacrifice model accuracy, significant processing was still required and

could only guarantee satisfactory moisture levels for the immediate hours after irrigation, not the full 24-hour cycle.

As all other control strategies are designed for the centrally-located water valve that is industry standard, they are unable to control in such a way that all locations in the space are adequately irrigated, while also minimizing water consumption. In response to these system limitations, we introduce OPTICS, a control system that tailors itself to the space in a data-driven way that requires no human intervention. This allows us to not only deploy the system with ease and minimal configuration, but perhaps more importantly, it allows us to *learn and adapt* to the dynamic local conditions experienced in the field. OPTICS does not require cumbersome measurements of soil type, topography, direct solar irradiance, but rather adapts to the conditions measured from the moisture data, even if the conditions are heterogeneous accross the field.

In this chapter, OPTICS uses this alternate approach to solve this very complex problem. We argue that a model adaptively trained from data will react to unforeseen conditions better than a system using a mechanistic model with approximated parameters and fixed assumptions. The contributions of this chapter are as follows: (1) As no manual input is required of the installer, OPTICS is a truly plug-and-play system, avoiding costly expertise to determine environmental characteristics (e.g. soil characteristics, topography, solar exposure) which can be difficult or infeasible to measure accurately at scale; (2) Constant model re-training with fresh data allows OPTICS to automatically adapt to unforeseen/changing environmental conditions and seasonal variations, and weather forecasting allows OPTICS control to react to future weather conditions; (3) To improve system scalability, OPTICS decouples short- and long-term models, allowing the latter to become spatially independent; (4) As OPTICS closes the loop,

the lightweight learning model reduces the computational complexity, allowing us to compute optimal schedules in a timely manner without significant computational resources while maintaining overall accuracy. In 4 weeks of deployment during the summer and fall, we demonstrate that in addition to these significant improvements to ease-of-use and scalability, OPTICS reduces water consumption against all baselines, while simultaneously improving quality of irrigation.

5.1 System Overview

Our system takes advantage of a distributed irrigation control system, with sensing/actuation nodes installed beneath each sprinkler. Each node is equipped with a wireless sensing mote [tmo] providing minor computational capability and wireless communications, a volumetric water content (VWC) sensor to sense local conditions, and a solenoid, allowing the opening and closing of water to the sprinkler on command. These devices form a mesh network, and are accessible through a border router, a special node physically connected to a nearby internet-accessible computer. Sprinkler schedules are sent outbound along this link, and real-time data from the sensing nodes are sent inbound along this link, allowing us to automate the system with any strategy we like.

The goal of our irrigation system is to keep the turf healthy and in order to do so, a number of requirements must be satisfied. Adequate solar exposure must be provided, the soil must contain the correct types of nutrients in appropriate amounts, and an adequate amount of moisture must be provided to the soil to be absorbed by the plant roots. Although our irrigation system has no control over solar exposure and soil nutrient composition, it has direct control on the application of water onto the surface of the soil. In plant physiology, two primary thresholds of volumetric water content (VWC) will determine the happiness of

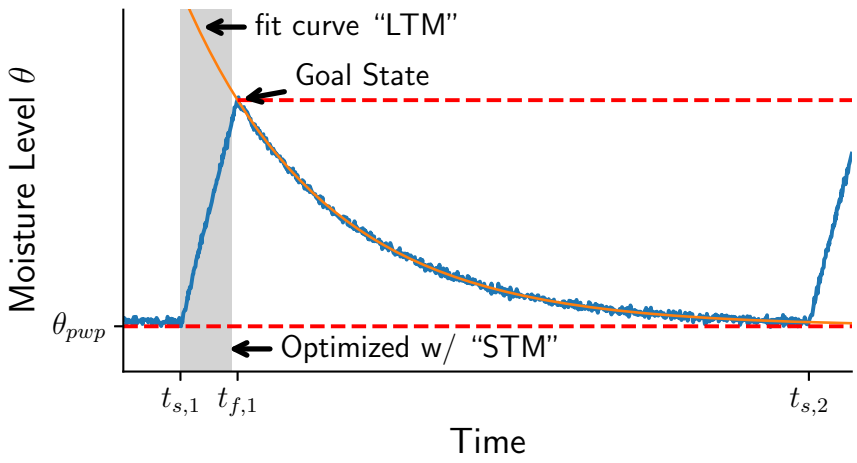


Figure 5.1: Sample fluid curve across the 24-hour cycle

the plant. The first represents the level where the plant will no longer be able to extract water from the soil, and is known as the Permanent Wilting Point, or θ_{pwp} [MD17, usd, Kir04]; if a plant spends an extended period of time in soil beneath this moisture level, it will begin to wilt and die. The second threshold more conservatively describes the level of moisture below which the plant *starts* feeling stressed, and is a fraction p of the total available water in the soil, where p is tailored to the particular crop type in question [APR98]. While the more conservative no-stress threshold is certainly more applicable in the installation of a more decorative turfgrass, we chose the more restrictive θ_{pwp} moisture threshold in this work, as it will better demonstrate the OPTICS system's ability to maintain even the most challenging moisture levels. While this threshold can be easily modified in the system depending on needs, the goal of the OPTICS system is the same under any of them; to minimize the amount of time the system spends beneath the chosen minimum threshold, as verified in our system with the installation of VWC sensors to constantly monitor soil moisture levels. By doing so, we will be minimizing the plant's exposure to unhealthy moisture conditions,

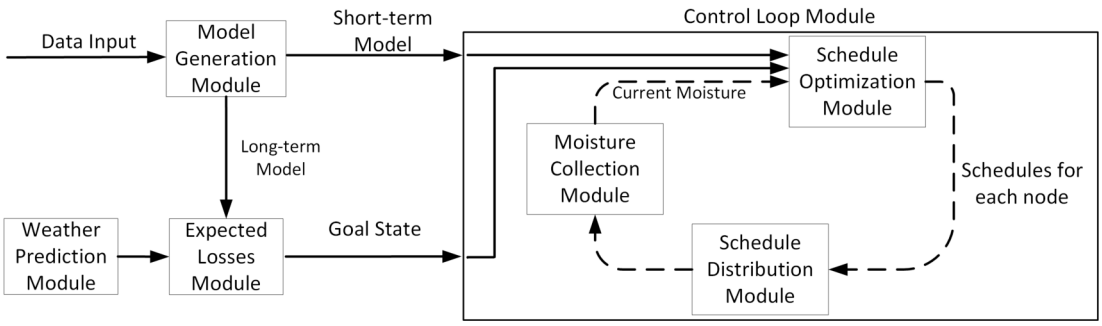


Figure 5.2: OPTICS System Architecture

giving it the best possible chance to thrive.

Figure 5.1 shows an example 24-hour irrigation cycle. After day 1's irrigation ends at $t_{f,1}$, the water begins discharging from the soil subject to the effects of diffusion, leaching, and weather effects, which occur very slowly relative to effects during irrigation. After hours of these losses, the soil VWC will reach its minimum value just before the start of irrigation at day 2, $t_{s,2}$. The effects of diffusion, leaching, and weather are not homogeneous; differences in environmental factors such as soil type, soil depth, and solar irradiance change the rate at which moisture is lost across the space. Over time, we use the historical loss trends to build a *long-term model* (LTM) that characterizes the way water is lost *between* irrigation cycles for each individual sensing node. With knowledge of the desired minimum moisture threshold θ_{pwp} we wish to maintain in the soil and our *long-term model* describing losses that are expected to occur, we can work backwards to intelligently choose a *Goal State* of moisture that we wish to reach by the end of irrigation. By reaching this *Goal State*, we ensure we will stay above minimum VWC θ_{pwp} across the full 24-hour cycle without wasting water.

Between day 1's start of irrigation $t_{s,1}$ and finish of irrigation $t_{f,1}$ as shown in Figure 5.1, the water applied to the space gradually increases VWC within the soil. $t_{f,1}$, the time that irrigation ends is not known beforehand, but occurs when

VWC has reached the pre-computed *Goal State* for this sensor node. During this irrigation time, we utilize a data-driven *short-term model* (STM) describing how the actuation of one or more sprinklers affects the moisture in the soil at all sensor locations, and solve an optimization problem to find the most efficient irrigation schedule to reach the *Goal State* at each sensor location, allowing us to save water by utilizing sprinkler overlap, soil runoff, and schedule intermittency to our advantage. The de-coupling of these two models allows this technique to scale to control very large irrigation systems. The two are intertwined, as one produces the *Goal State* used by the other. In the short-term, the model and optimization takes into account the effects of all nodes' sprinkler coverage jointed spatially and temporally, but once irrigation ends, the models describing losses across the field become spatially independent.

Figure 5.2 shows the data processing required to achieve these goals at irrigation-time (daily irrigation at dusk, by request of campus groundskeepers). First occurs model generation. The freshest data from the irrigated space is used to build the long- and short-term models for use in loss prediction and irrigation schedule optimization. The short-term model describes the direct in-flow of moisture as sprinkler moisture lands above the sensor, and takes into account sprinkler overlap and water runoff effects. The long-term model shows how the moisture tends to move across the full 24-hour cycle due to soil transport effects such as diffusion and leaching [Chi69], and a separate weather prediction module predicts future weather trends in the form of reference evapotranspiration [JBA90].

Next, using the long-term model, the *Expected Losses* module computes a *Goal State* for each node in the space as shown in Equation 5.1. This state is computed by taking the minimum acceptable VWC and adding the node's expected moisture losses between irrigation cycles. As later explained in Sec-

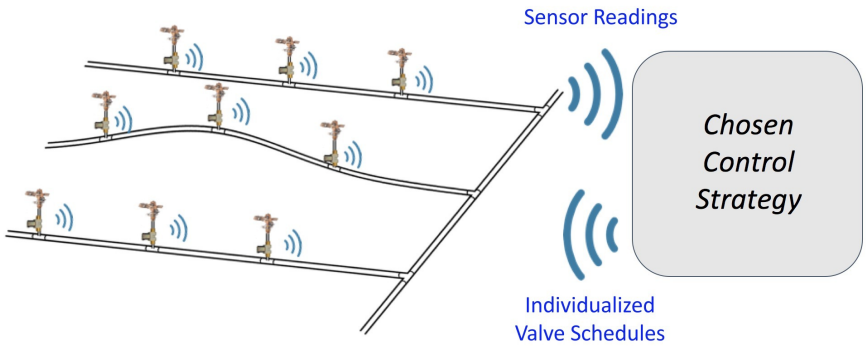


Figure 5.3: Data communication across irrigation system with distributed actuation

tion 5.2.2, future weather predictions are decoupled from the long-term model, so forecasted evapotranspiration losses are also added at this stage, computed as described in Section 5.2.1. In this way, the minimum moisture, experienced right before the next irrigation will begin, should be at or just above the minimum VWC threshold. This *Goal State* is later used by the optimization module.

$$Goal\ State = \theta_{\text{pwp}} + Expected\ Losses + ET_{\text{forecasted}} \tag{5.1}$$

Once these initial conditions are defined for irrigation, the control loop is entered, which will fetch the freshest data snapshot from all nodes across the space. This data is used as the initial moisture conditions for an optimization problem that computes the optimal actuation sequence for each individual node in the space, such that each node's *Goal State* VWC will be reached by the end of the irrigation period. If our short-term model used in optimization is accurate, we can perform this control loop only once at the beginning of irrigation. However, for safety and for accuracy, it is advantageous to occasionally re-run this control loop during the irrigation period using the most current VWC data across the space to help correct for drift caused by errors in the *short-term model*. This prevents schedules from over- or under-watering due to stale starting conditions,

and will allow the optimizer to re-search for more efficient schedules from the freshest starting point.

In our deployment, the model generation and optimization takes place on a computationally-weak Raspberry Pi [rasa] that is collocated with our irrigation system. Attached to the Raspberry Pi via USB is a TMote Sky [tmo], through which schedules can be sent and data can be received. Once optimization is complete, the schedules produced by the optimizer are sent over USB and through the wireless sensor network to their respective sensing/actuation node. The schedule is run, and if desired, a fresher data snapshot will be obtained and optimization will occur again.

To maintain a fresh snapshot of the soil moisture content at all times, the distributed sensor/actuator devices are constantly polling their attached sensors. During irrigation time, when fresh data is crucial, the sampling frequency is set to be equal to the *Control Timestep* of the actuator, chosen in our system to be 1 minute. When irrigation is complete, the sampling period is reduced to once every 10 minutes to maintain a long battery life. Sampling across the full 24-hour cycle is useful for us to evaluate the control strategies we are testing, but in a commercial system it may only be necessary to collect data during irrigation, consuming less sensing and radio transmission energy and extending system lifetime. When sensing is occurring at a 10-minute timestep, we later post-process by linear interpolation to ensure we have a consistent sampling period throughout the entire day. This data is routed from the sensing nodes across the wireless sensor network to the basestation as shown in Figure 5.3 for use in modeling, visualization, etc. Due to the relatively small distances of our test irrigation system, only a single-hop wireless network was required to communicate from sensing nodes to the basestation, as later discussed in Sections 5.5.4 and

5.8.

Control on the distributed sensor/actuator devices also operates at the same *Control Timestep*, chosen in our system to be 1 minute. Individualized valve schedules are routed from the basestation across the wireless sensor network as shown in Figure 5.3, each containing a control timestep and a binary sequence of actuations for a particular sensor/actuator node. Once received, the sensing/actuation node will then route a pulse of either forward or reverse current to enable or disable the connected solenoid, following the provided schedule. If a schedule is received while another is already running, the first is cancelled and overwritten by the fresher one.

5.2 System Modeling

The purpose of the control system is to decide how much moisture must be applied to the surface by the sprinklers. Whereas standard irrigation controllers use weather-only or rule-of-thumb techniques to make this decision, we hope to leverage the rich, spatially-distributed data collected by our sensing/actuation nodes to make this process more efficient. Towards this goal, we generate data-driven models that help us understand how moisture tends to move in the short- and long-term.

The water used for irrigation moves through the space subject to many factors, which all tend to occur within two different time horizons. In the short term, *during* irrigation, factors such as sprinkler distribution and water runoff on the surface of the soil cause water movement that will occur for seconds or minutes, movement that tends to comes to an end when irrigation is completed or soon thereafter. Once this moisture infiltrates fully into the soil, much slower effects

continue to take place in the long term *between* irrigation periods. For instance, depending on the type of soil, leaching of water beyond the root zone and diffusion can occur on the order of $10^{-2} - 10^{-3}$ cm/s [Bea72], and will continue to move for hours or days.

Previous work [WWB16] combines all of these moisture movement factors together in one large mathematical model based on first principles. However, the size of the resulting models have performance repercussions, and the lack of model correction could potentially cause it to deviate from reality. In this chapter, we solve these problems by modeling water movement using a lighter data-driven approach based on machine learning techniques. The implicit advantage of this approach is that it is based on moisture data measured locally, being able to cope with heterogeneous conditions like soil type, topography, and solar exposures that vary across the field. To reduce the computational complexity of the model-based optimization problem to minimize water consumption subject to quality constraints, we chose to use a two-model approach, each of which represent one of the distinct time horizons. The long-term model, which learns how water tends to be lost between irrigation cycles, is used to compute a *Goal State*, the required moisture level that must be reached at each location in the field so that moisture levels will not be depleted below our minimum moisture threshold before the next irrigation period. The short-term model is used by the *Schedule Optimization Module* to determine the best schedules that will take advantage of runoff, overlapping sprinkler coverage, and other short-term effects to compute schedules that bring moisture to the *Goal State* while consuming minimal water. With a cleverly-chosen *Goal State*, we will maintain adequate moisture levels across the full 24-hour cycle, while only requiring optimization during the irrigation period, allowing optimization to be run on very computationally-weak machines.

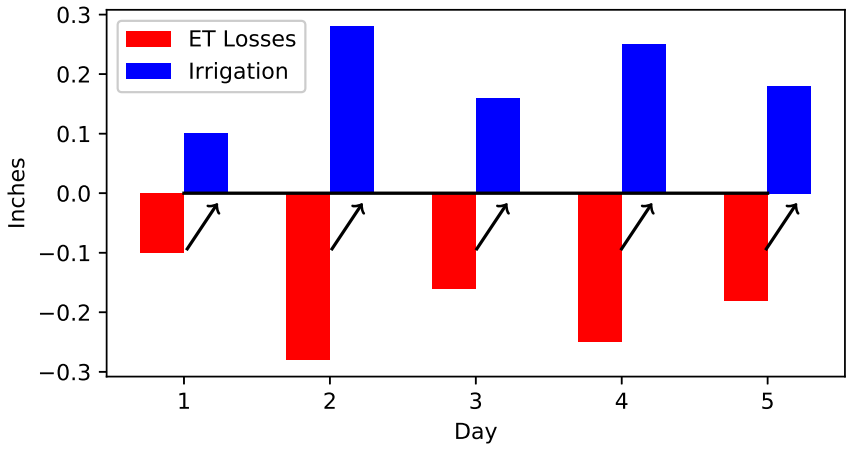


Figure 5.4: Water inputs and outputs to reactive irrigation system

5.2.1 Weather Forecasting

A large portion of losses on the 24-hour cycle is expected to be caused by the combined environmental effects of evaporation and plant transpiration, referred to together as evapotranspiration [JBA90] (ET). The industry-standard ET estimation uses four environmental factors, temperature, humidity, wind, and solar irradiance to estimate ET losses, or the water that has been removed from the soil due to weather conditions.

Evapotranspiration-based irrigation controllers are becoming industry standard, and the control strategy follows intuition. Weather stations in the region take periodic measurements of the four environmental factors, and make a calculation of evapotranspiration (ET) losses. The ET irrigation controller communicates with the weather station for an estimate of total losses that have occurred since the last irrigation. With knowledge of the sprinklers' water application rates, these losses are then used to decide the irrigation schedule to replace the losses that have occurred. However, this technique has a fundamental limitation, as the control can only react to past weather losses, and cannot prepare

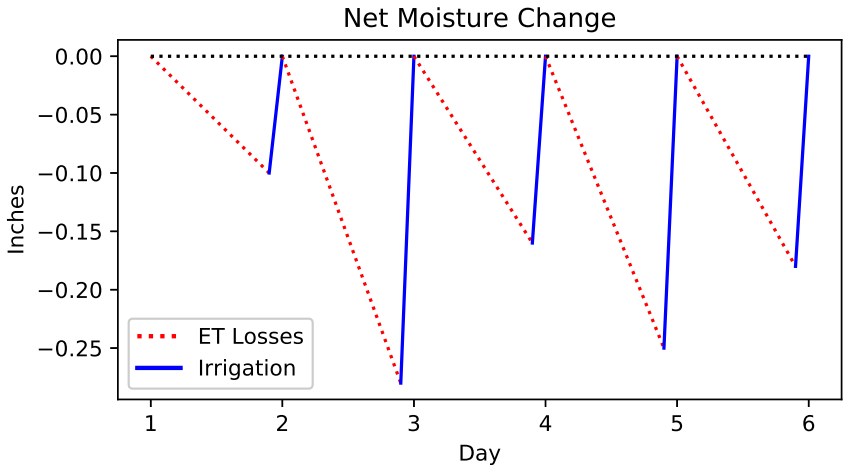


Figure 5.5: Net change in soil moisture content under reactive system

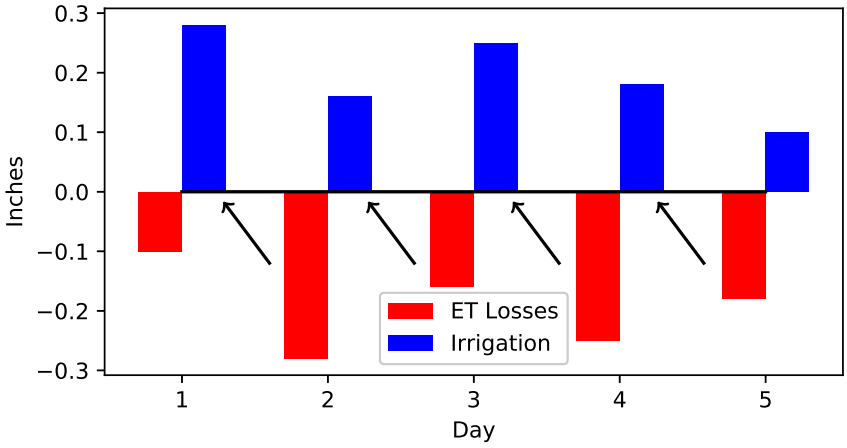


Figure 5.6: Water inputs and outputs to predictive irrigation system

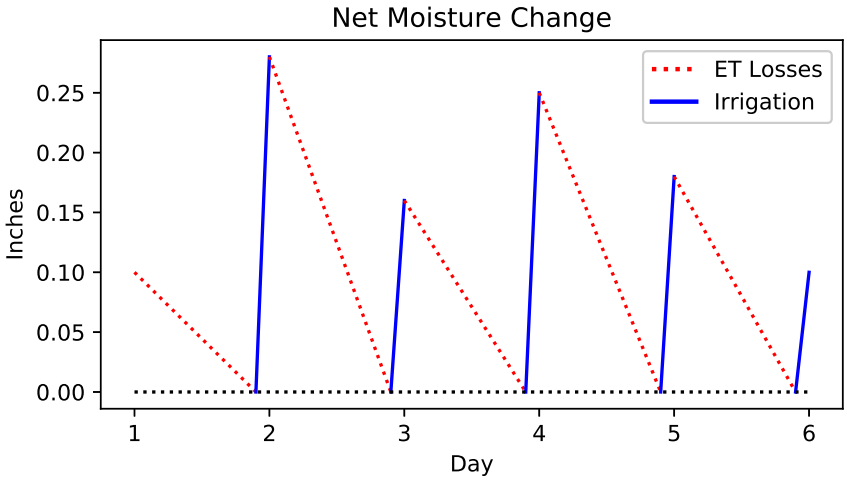


Figure 5.7: Net change in soil moisture content under predictive system

for future weather losses. Figure 5.4 shows example water inputs and outputs in an irrigation system with a strategy that reacts to the previous day's weather trends, where irrigation each day is chosen to match the losses of the previous day. Here, the water applied and lost is displayed in inches of water, the standard unit used to describe irrigation and precipitation inputs, and evapotranspiration losses. This unit describes a column of water on the surface, and when describing irrigation amount is a function of the sprinkler's distribution pattern and on-time. The resulting net change in water can be seen in Figure 5.5, where it can be seen that performing a direct water replacement results in the soil in general receiving a net loss in water content due to the system's inability to prepare for future losses. In practice, to ensure the plant receives enough water, commercial ET irrigation controllers provide a direct replacement of past losses *plus* a safety margin of extra water to guide the soil moisture towards adequate levels over time. This inexact method of overwatering by design leads to unnecessary water waste or reduced quality of service.

In contrast, if we have the ability to *predict* the losses before they occur, we

can pre-irrigate to ensure net moisture change is always positive or zero, without requiring an arbitrary safety band of extra irrigation. The loss/irrigation inputs and outputs of such a system can be seen in Figure 5.6, and the resulting net moisture change in the soil can be seen in Figure 5.7. Each day, the amount of irrigation is chosen based on the next day's *predicted* losses rather than past losses, which allows us to allocate the water ahead of time to prepare for or take advantage of weather that is expected to occur. In addition to avoiding unhealthy moisture levels in the soil, prediction will allow the system to save water if rain is forecasted in the near future, whereas a reactive system will continue to irrigate as usual. If more rain occurs than was predicted, the OPTICS system will sense this through soil conditions and prevent wasteful irrigation from being applied to the space. If less rain occurs than was predicted, the system will recuperate these losses after-the-fact in the same way a standard ET controller would.

Although *prediction* of future ET losses will allow an irrigation system to provide better quality of control and system efficiency, this data is not available in practice. However, we found that 15 years of historical data trends [cim] of the 4 ET factors (temperature, humidity, wind, and solar irradiance) were available at hourly intervals at our location, as well as the daily ET losses that occurred on those days. Additionally, we consider the fact that in most locations, it is possible to find *forecasted* hourly weather trends (for at least temperature) for at least 24 hours into the future. To use the data we have available to provide future ET prediction, we trained a k-nearest-neighbors regression model to accept as input the hourly vectors of all (or a subset, depending on forecast availability at the system's location) of the 4 ET factors, and as output the resulting ET losses for the entire day. For instance, in our locale we found that the Wunderground API [wun] has hourly forecasting of both temperature and humidity. If the deployed system is not located near a weather station, the National Oceanic

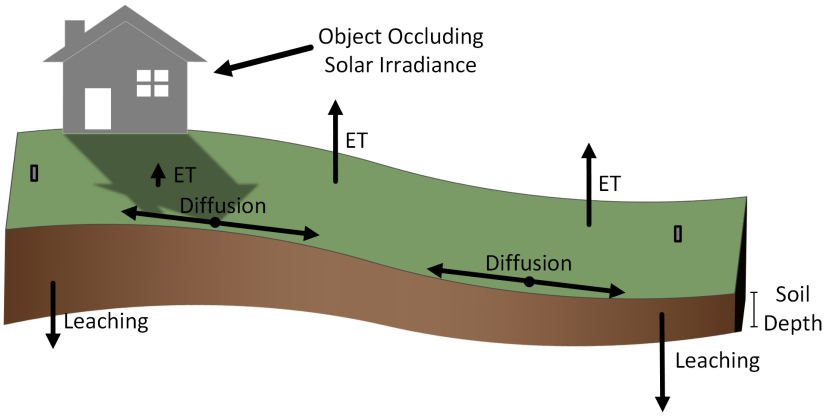


Figure 5.8: Sources of water movement *between* daily irrigation (Long-term)

and Atmospheric Administration (NOAA) provides temperature, humidity, wind speed and cloud cover prediction at hourly intervals for any point in the United States [noa]. Using our 15 years of historical data with approximately 5475 data samples, we step through each day and create a 48-element feature vector composed of 24 hourly temperature readings and 24 hourly humidity readings, and as output the cumulative ET losses from this 24-hour period as a single value. We train our KNN on these features and outputs with a k-value of 5, so in prediction the regressor will use the 5 historical days with the most similar hourly weather trends to choose the evapotranspiration value that is likely to occur. Finally, each day we query Wunderground to get *tomorrow's* forecasted hourly temperature and humidity readings, which are then passed to the KNN regressor to predict the ET losses that are expected to occur tomorrow based on those trends, which is then used by the OPTICS controller to optimize irrigation schedules. Evaluation of the accuracy of this technique can be found in Section 5.6.1.

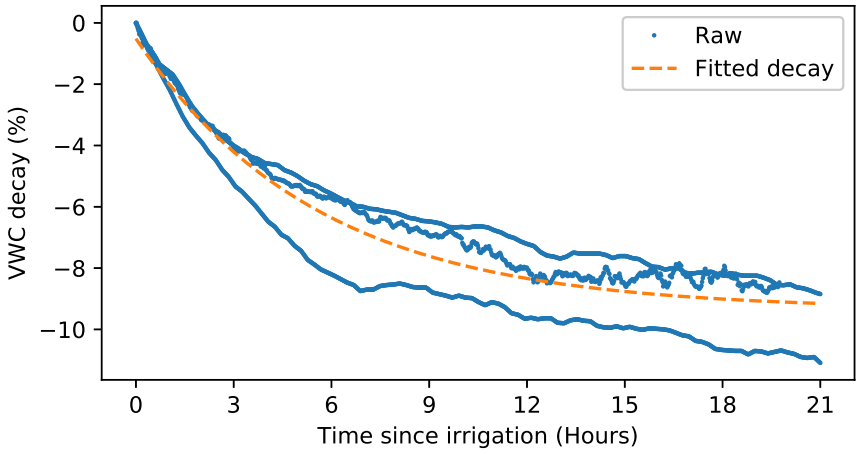


Figure 5.9: Sample moisture decay fit between irrigation

5.2.2 Long-term Model

To provide the highest quality of control, we wish to maintain at least a minimum level of moisture in the soil, θ_{pwp}, as discussed in Section 5.1. Although industry-standard irrigation systems choose a target on-time for the irrigation system, our rich coverage of soil moisture sensors allow us to instead set a *Goal State* of soil moisture. This *Goal State* is unique for each node in the space, acting as a target volumetric water content (VWC) for the *Schedule Optimization Module* to reach as shown in Figure 5.1. In order to choose a useful *Goal State*, we must consider both future predicted losses due to weather patterns (evapotranspiration) as discussed in Section 5.2.1 and environmental loss data trends from previous data for each individual node.

Once irrigation is complete and the irrigated moisture has infiltrated into the soil, it begins to move more slowly through the processes of diffusion, leaching, and weather patterns as shown in Figure 5.8. Diffusion is the tendency of moisture in the soil to move from areas of higher to lower concentration due to differences in hydrostatic pressure caused by intermolecular forces in the soil, and leaching is

the tendency of water to eventually move beyond the root zone of the plant due to the force of gravity. Movement under either of these forces tends to be very slow, as hydraulic conductivity of water through certain soils can be on the order of $10^{-2} - 10^{-3}$ cm/s [Bea72]. Losses due to evapotranspiration (ET) occur slowly as well, as they are primarily caused by solar radiation and high air temperature which occur during daylight hours, or windy conditions.

Figure 5.9 shows how moisture losses occur at one selected sensor site across three consecutive days. This example curve is unique at this location due to its specific soil characteristics. In this figure, $t = 0$ is time-aligned to the end of irrigation where moisture losses start to occur, and $t = 21$ hours corresponds to the beginning of irrigation on the following day. By tracking these losses using our deployed VWC sensors, we found that the discharge of water from the soil medium tends to occur as an exponential decay when irrigation ends as shown in the figure. In our system, we use these historical loss trends to fit an exponential decay curve as our "LTM", or Long-term model, as shown in Figure 5.9. In addition, for each day $d \in \{1, \ldots, D\}$, we record the measured evapotranspiration that occurred, ET_d. With this fitted model, the expected loss of this node is computed by taking the difference of the curve at $t = 0$ and at $t = t_{s,d+1} - t_{f,d}$, the expected delay between irrigation of today and tomorrow, e.g. 21 hours. This computed loss is finally offset by the average of the daily evapotranspiration levels measured during the training period.

$$\textit{Expected Losses} = (\text{LTM}(0) - \text{LTM}(t_{s,d+1} - t_{f,d}) - \frac{1}{D}\sum_{d=1}^{D} ET_d \tag{5.2}$$

We subtract the weather to ensure that the weather experienced when the training trends were recorded do not impact the expected losses for future irrigation. Before use as the *Goal State* for optimization, the *predicted* ET for the following day will be re-added as shown in Equation 5.1. We decouple these

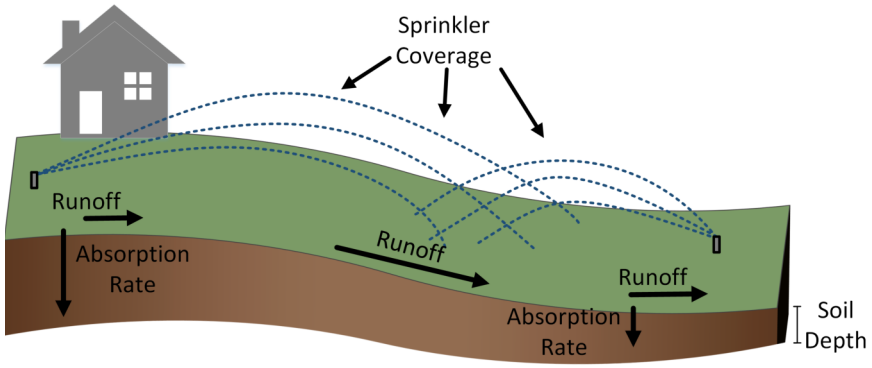


Figure 5.10: Sources of water movement *during* irrigation (Short-term)

weather effects to prevent learning past weather trends into our model. In a climate where the weather changes very little from day-to-day, it may be reasonable to assume that past weather trends will continue into the future, but to allow OPTICS to be more generalizable and reactive we choose to utilize local weather forecasting separately.

When the system is first turned on, the model has no understanding of expected losses. It is not until after the first day of irrigation, in our case a fixed schedule to train the short-term model as discussed in Section 5.2.3, that the long-term model can be trained. In our experiments, the model is retrained each day before irrigation using the loss data from all of the preceding days. That is to say, on day d, we re-train the model using $d-1$ days of data. This worked well in our experiment, but in a very long-term installation, re-training the model on just the most recent N days of data may allow the system to be more responsive to changing seasonal conditions. We have not investigated the optimal choice of N, and we leave this for future work.

5.2.3 Short-term Model

As our system was designed to require no manual configuration from the installer, at install-time the OPTICS system does not understand which sprinklers and sensors are within range of each other. For later use in optimization in Section 5.3, we must understand how the actuation of the sprinklers in the system will influence the moisture in the soil as shown in Figure 5.1. For this task, we employ a *Short-Term Model*, which captures the moisture movement effects *during* irrigation. An example irrigated space is shown in Figure 5.10; differences in sprinkler overlap affect the amount of water that lands on the surface of the soil, surface topography affects how quickly runoff will occur, and heterogeneous soil composition and depth will affect the rate of infiltration. As it is difficult and error-prone to manually measure these effects, we wish to learn them in an automated, data-driven way.

We choose to use a linear regressor to model these effects using the variables enumerated in Table 5.1. As input, we provide the current VWC at each of the K sensor locations as vector $\mathbf{s}_t$, and the current binary actuation of each of the K sprinklers as vector $\mathbf{f}_t$. The output of the linear model is the predicted VWC for each of the K sensor locations at some time Δt in the future, as vector $\mathbf{s}_{t+\Delta t}$. In practice, as this model will be later used to compute optimal irrigation schedules, the length of Δt is chosen to be the same as the control actuation period, 1 minute in our experiments. The linear function $\mathbf{g}$ defines the following relationship:

$$\mathbf{g}(\mathbf{s}_t, \mathbf{f}_t) = \mathbf{s}_{t+\Delta t} \tag{5.3}$$

Ideally, we would train our model by running all possible combinations of sprinkler actuations and measuring the sensor response. However, as this would require an immense amount of time and potentially wasteful irrigation actuation

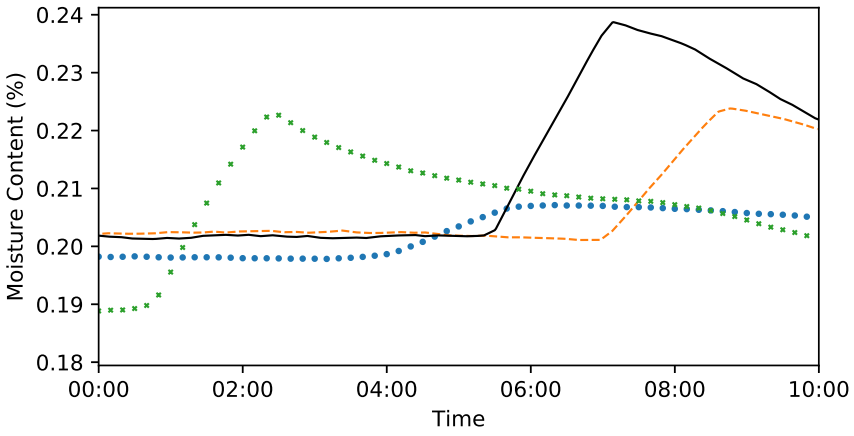


Figure 5.11: Sensor data from selected nodes during Short-term model training period

Table 5.1: Short-Term Model Variables

Variable	Description
$t \in \{0, \ldots, T\}$	Temporal index
K	Number of sensing/actuating nodes in system
$\mathbf{s}_t \in \mathbb{R}^K$	Vector of moisture levels at time t, size K
$\mathbf{f}_t \in \{0, 1\}^K$	Vector of binary sprinkler actuation at time t, size K

to achieve, we instead chose to actuate each sprinkler one-by-one for model training, which can be done much more quickly. While the resulting model will have some error in moisture estimation when many sprinklers are active at once, the OPTICS system will continuously re-train this model as more complex schedules are used in irrigation, improving prediction over time.

In this way, each sprinkler is triggered one-by-one for a fixed time (1 hour in our case) and the resulting soil moisture is measured at all locations. Moisture levels during our training cycle can be seen for 4 selected devices in Figure 5.11 with rises caused by the activation of the nearest sprinkler, and although each sprinkler is active for the same amount of time, it can be seen that the amounts of increase are different for each device due to spatial heterogeneity of runoff, sprinkler coverage, and soil characteristics. Furthermore, as sprinkler overlap would cause multiple sensors to rise simultaneously as a single sprinkler is active, it can be seen that this irrigation system has minimal sprinkler overlap, as only one sensor rises at a time. In a system with more sprinkler overlap, the methods of data collection and processing would be identical.

The data from our deployment training period is parsed into ∼850 training pairs of $[\mathbf{s}_t, \mathbf{f}_t]$ vectors as inputs, and $[\mathbf{s}_{t+\Delta t}]$ vectors as outputs, and the regressor is trained. As sensor data is inevitably noisy it is important to choose a Δt that is not too big, which would result in increased predictive error or too small, where noise dominates the state signal. Our choice of one minute seemed to work well in practice, and further evaluation of this choice can be found in Section 5.6.2.

Table 5.2: Optimization Variables

Variable	Description
$t \in \{0, \ldots, T\}$	Temporal index
$k \in \{1, \ldots, K\}$	Sprinkler location index
$\mathbf{s}_t \in \mathbb{R}^K$	Vector of moisture levels at time t, size K
$\mathbf{f}_t \in \{0, 1\}^K$	Vector of binary sprinkler actuation at time t, size K
$s_{kt} \in \mathbb{R}$	Volumetric water content (VWC) of location k at time t
$f_{kt} \in \{0, 1\}$	Sprinkler k actuation at time t
$c_k \in \mathbb{R}$	Consumption of sprinkler k (Constant, known beforehand)
$\theta_k \in \mathbb{R}$	Measured VWC of sensor k (Constant, known beforehand)

5.3 Optimization Over the Schedule

We wish to use our models as described in Section 5.2 to compute irrigation schedules that will allow us to reach our goal volumetric water content (VWC) on each sensor/actuator node while minimizing system water consumption. With a goal state, $\theta_{\text{goal},k}$ and recently measured VWC θ_k for each node index $k \in \{1, \ldots, K\}$, we construct the following optimization problem using optimization variables as defined in Table 5.2.

$$\min_{\{f_{kt}, s_{kt}\}_{k=1,t=0}^{K,T}} \sum_{k=1}^{K} \sum_{t=0}^{T} c_k f_{kt} \quad \text{s.t.} \tag{5.4a}$$

$$0 \leq f_{kt} \leq 1 \qquad k = 1, \ldots, K \quad t = 0, \ldots, T \tag{5.4b}$$

$$s_{kt} \geq \theta_{\text{pwp}} \qquad k = 1, \ldots, K \quad t = 0, \ldots, T-1 \tag{5.4c}$$

$$s_{kT} \geq \theta_{\text{goal},k} \qquad k = 1, \ldots, K \tag{5.4d}$$

$$s_{k0} = \theta_k \qquad k = 1, \ldots, K \tag{5.4e}$$

$$\mathbf{s}_t = \mathbf{g}(\mathbf{s}_{t-1}, \mathbf{f}_{t-1}) \qquad t = 1, \ldots, T \tag{5.4f}$$

We define f_{kt} to be the binary actuation of sprinkler k at discrete time index $t \in \{0, \ldots, T\}$, relaxed to $[0, 1]$ in the optimization. The objective function in Eq. 5.4a is the sum of f_{kt} for all $k \in \{1, \ldots, K\}$, $t \in \{0, \ldots, T\}$, weighted by the water consumption rate of each sprinkler k, c_k, a function of the sprinkler's angle of coverage as defined in the sprinkler datasheet. With pressure-regulated sprinklers, this weighted sum represents the total system water consumption under schedule $\mathbf{F}$, which we are trying to minimize. The VWC at each discrete sensor location k at temporal index t is defined as s_{kt}. This moisture level is constrained to remain above the minimum acceptable moisture threshold, θ_{pwp} at all times in Eq. 5.4c, and above the goal state at final time index $t = T$ in Eq. 5.4d. Eq. 5.4e assigns this state variable the most recently measured sensor value θ_k at starting time index $t = 0$. Changes in moisture level as a result of sprinkler actuation is modeled by the linear function $\mathbf{g}$ in Eq. 5.4f, representing our *Short-term Model* as defined in Section 5.2.

As sprinkler valves can physically be either on or off, sprinkler actuation f_{kt} is a binary variable in practice. This makes the defined problem an integer linear program (ILP), which is NP-Complete. We have found that solving this problem with reasonable values of K and T can take as long as several minutes on our computationally-weak basestation. As this optimization may be required to run as often as once per control timestep, chosen in our system to be 1 minute, we find an approximated solution more quickly by treating f_{kt} as a real number within $[0, 1]$ as shown in Eq. 5.4b, and then rounding the computed optimal value to the closest binary value. This simplification makes the resulting problem a linear program (LP), which is much simpler to solve in practice. Evaluation of this choice can be found in Section 5.6.3.

We found that with our irrigation system architecture and environmental

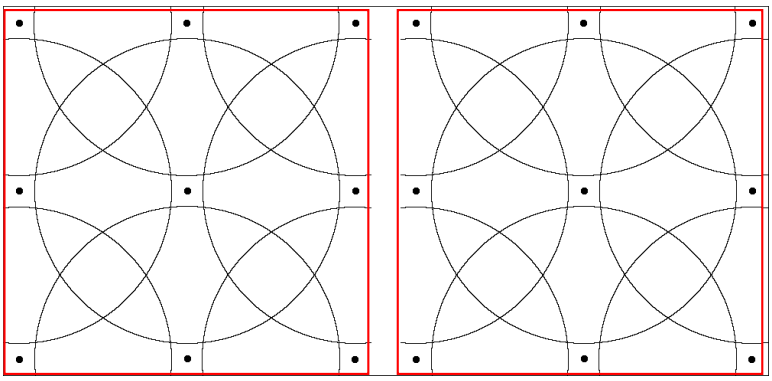


Figure 5.12: Side-by-side experimental layout

characteristics, our irrigated space generally requires 30-60 minutes of irrigation to be sufficiently watered. To give our optimization time to find schedules that are as efficient as possible, we give the optimizer an irrigation window of 2 hours, well above the required time of a simple schedule. With a control timestep of 1 minute, this 2 hour irrigation window is converted to $T = 120$. By allowing a larger irrigation window, the optimizer may find schedules that are more efficient, but due to University water pressure limitations, irrigation in different regions of the campus must operate within slots, making very large time windows impractical.

We chose to use the Julia programming language [BEK14] as an interface to the GNU Linear Programming Kit (GLPK) [glp] solver. We chose these tools for their ease of use and sufficient performance. Our linear program has $2K(T+1)$ variables and $4K(T+1)$ constraints. In our deployment, solving for a schedule with a selected setup of $K = 9$ and $T = 120$ takes less than a second on our computationally-weak basestation.

5.4 Case Study: Live Deployment

To perform a fair comparison of our OPTICS system against a baseline, we installed two identical irrigation systems that allow us to run two control strategies side-by-side. As shown in Figure 5.12, the two systems were oriented next to each other with a gap purposefully left in-between to prevent cross-contamination between the two systems. An alternate technique to ensure fair comparison between two control strategies would be to periodically alternate the strategies between the two irrigation systems to ensure results aren't skewed by environmental variations. However, this would require us to wait for soil moisture to settle between each alternation, causing all experimentation to take significantly longer. Instead, we chose this side-by-side arrangement, so that each system would have as homogeneous solar exposure, slope characteristic, and soil characteristics as possible.

5.4.1 Environmental Description

The two irrigation systems were installed side-by-side, and were designed to be identical in hardware, sprinkler coverage, etc. Each irrigation system measured 60'x60', with sprinklers arranged in a 3x3 grid, each 30' from the next. The sprinklers chosen were MP Rotators by Hunter Industries [hunc], which are currently considered state-of-the-art in sprinkler technology. In these devices, the water pressure is focused through many rotating nozzles on the sprinkler head which allow much greater range at lower water flow rates, applying water more efficiently than their rotor counterparts. The MP Rotators can be adjusted to a range of 15'-30', making them ideally suited for our system.

The deployment area was located on a sloped area that drops 3' from the

Figure 5.13: Sensing and actuation node

highest to the lowest point. During installation of the sprinkler system, we noted that the topsoil was a "Clay Loam" type, approximately 10" deep. Beneath this layer sat a thick clay layer that went beyond 3' deep. The grass growing in the installation area is a natural field grass and not the type found on a sports field or in university landscaping, but the goal of irrigation is identical; providing a satisfactory amount of moisture to sustain healthy turf.

5.4.2 Hardware Description

The hardware used in our experiments stemmed from the design introduced in [WWB16], with battery life and system safety as primary concerns. Control was provided by a latching solenoid, which requires a 50ms pulse of power in either the positive or negative direction to open or close the valve. The sensor chosen was the Decagon EC-5 [dec], commonly used in research for its high accuracy of ±3% and power consumption of just 10mA for 10ms to take each sample. Although the EC-5 outputs a raw voltage, Decagon provides a linear

function that maps this voltage to Volumetric Water Content (VWC) for our use. Each sensor is inserted into the soil at the expected depth of the root zone for the turf on site, within the first 6 inches of soil in the case of our experimental system. A standalone board was developed to sit on the General Purpose Input/Output (GPIO) pins of the Tmote Sky, whose purpose is to route power from the attached 4xAA battery source to the peripheral devices. With these peripherals installed, our devices could communicate with each other, monitor the local soil moisture conditions, and control the flow of water to the attached sprinkler. These primary hardware components in their waterproof case can be seen in Figure 5.13.

Collocated with the irrigation system was a basestation module, consisting of a Raspberry pi with a Tmote Sky attached via USB, and a wifi hotspot to allow the system to be accessible remotely. The data processing pipeline as described in Section 5.1 and the optimization as described in Section 5.3 was all run on this device. Schedules computed by the *Schedule Optimization Module* were forwarded over USB to the Tmote Sky for wireless distribution, and incoming data was pushed by the Raspberry Pi to an off-site database for remote monitoring and analysis. Later explained in Section 5.5, the only equipment failure that occurred in our deployment was a failed USB connection between the Tmote and the Raspberry Pi, likely due to environmental factors. However, firmware running on the sensing/actuation devices is designed to handle such a failure by automatically returning to the default "Off" state, preventing significant water loss.

5.4.3 Baseline Strategies

To allow the two side-by-side irrigation systems to operate independently, all sprinklers are installed with a sensing/actuation node. In this way, the only dif-

ference between the two systems are the schedules sent to the sensing/actuation nodes. In this chapter, we compare the OPTICS system to two baseline systems, an Evapotranspiration control strategy, the current industry leader in system efficiency, and DICTUM, the current state-of-the-art in academia. As the University irrigation systems operate on a daily schedule, all baseline systems and OPTICS was configured to irrigate daily as well. The OPTICS system would operate identically on a different irrigation cycle with no reconfiguration necessary.

5.4.3.1 Evapotranspiration

Many weather stations are available to the public that provide ET estimates based on measurements of the other weather factors. To mimic an ET controller, we query a local weather station for the previous day's ET losses, provided in units of surface water height. With this data, we use our sprinkler's surface application rate to compute how many minutes the system must activate to replace the previous day's losses. In a commercial evapotranspiration controller, this amount is then the amount irrigated, *plus* a safety margin of water. However, despite contacting two of the largest providers of ET controllers, Hunter [huna] and Rain Bird [raib], we were unable to find the safety margin they use in practice, so we assumed NO safety margin. This means two things - it means a commercial system using a safety margin may provide better quality of service than what we see in our deployment, but at the cost of increased water consumption. Please note that the OPTICS system is able to achieve both goals, water savings *and* improved quality of service.

5.4.3.2 DICTUM

The DICTUM control framework requires that the installer pre-defines key irrigation and field characteristics before use. The irrigation system characteristics including coverage of sprinklers, application rates, angles, and positions are defined as described in Section 5.4.1 to match the physical deployment. Likewise, the topography was modeled to reflect the 3' elevation drop of the field, and the estimated soil type was chosen as observed to be a "Clay Loam" of depth 10", sitting atop a deep clay layer. To allow fair comparison to the OPTICS system, the DICTUM optimization was defined with a 2-hour irrigation evening window, to match the campus' irrigation scheduling policy to avoid over-use of the system pressure.

5.4.4 End-to-end Performance Metrics

As previously discussed in Sections 5.1, 5.2.2, and 5.3, we have chosen a minimum acceptable moisture threshold for all systems that will ensure the system is maintaining satisfactory water levels to keep the plant healthy. As maintaining plant health is the primary goal of an irrigation system, our primary metric for irrigation quality is the system's ability to maintain soil moisture levels above the chosen threshold at all times across the space. By doing so, we are guaranteeing that the plant has sufficient moisture to be healthy. In this paper, we call this the quality of service provided by the irrigation system.

Although we must maintain moisture above a minimum threshold, it is also detrimental to over-water the space. In addition to the environmental and financial impact, an over-abundance of water in the soil can, over time, lead to the rotting of the plant roots, discoloration of the plant (aesthetic penalty), and in extreme cases excess irrigation has been linked to the leaching of fertilizer chemicals

into human drinking water supplies. As each sprinkler uses a pressure-regulated water supply and we directly control the times at which each sprinkler is active, we can monitor the amount of water consumed by both systems at all times to determine the efficiency of each system. Thus another metric that is relevant is the water consumption, which we would like to minimize subject to the quality of service constraints.

Finally, an aesthetic side-effect of uneven moisture distribution is the appearance of "hotspots" where not enough water is received and oversaturated regions where standing water remains on the surface. These can be identified by a difference in color, and detract from the appearance of the space. Although they can take a long time to develop, with our sensing/actuation platform, we can investigate the long-term moisture trends under our comparative control strategies; a more even moisture distribution prevents these localized effects from happening, and uniformity can be monitored through the deployment's sensor data.

5.5 Experimental Results

In this section, we discuss the experimental results of OPTICS when compared to both the ET and the DICTUM systems side-by-side. These tests are performed under the same conditions as explained in Section 5.4.1, analyzing the quality of service, water consumption, and moisture uniformity metrics discussed in Section 5.4.4.

5.5.1 Quality of Service

Irrigation systems are installed to maintain health in the planted turf. However, these systems often fall short of their quality goals. As such, a potential replace-

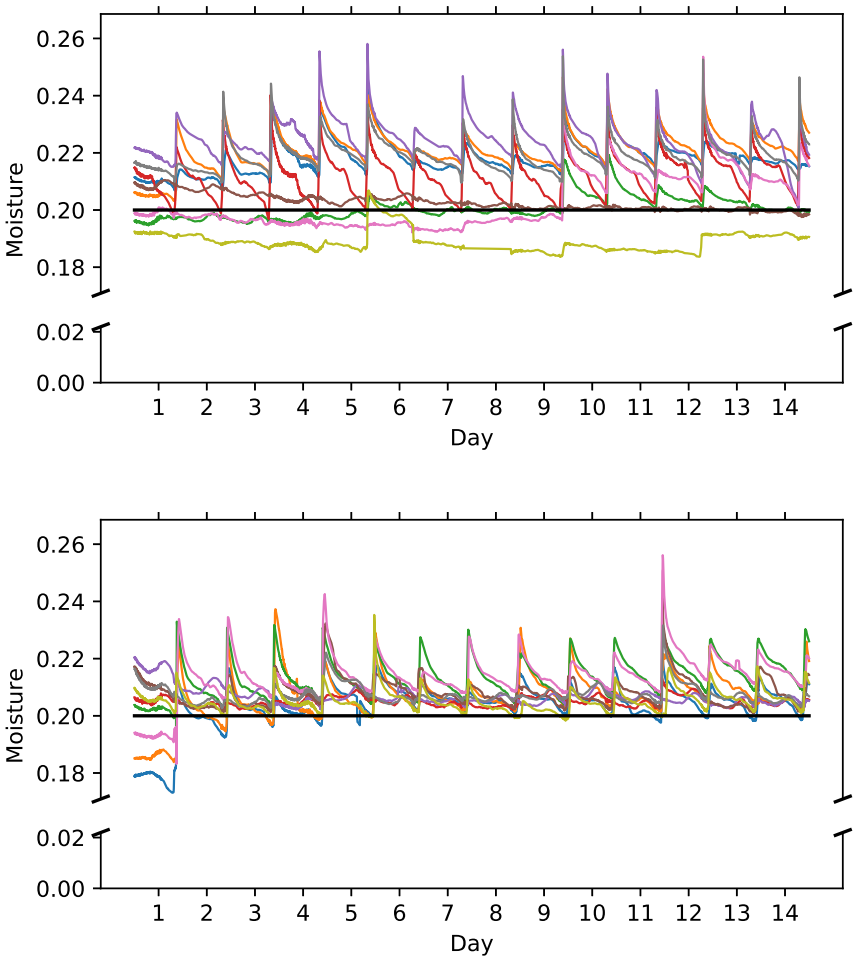


Figure 5.14: Collected VWC data across deployment for all Evapotranspiration (top) and OPTICS (bottom) nodes

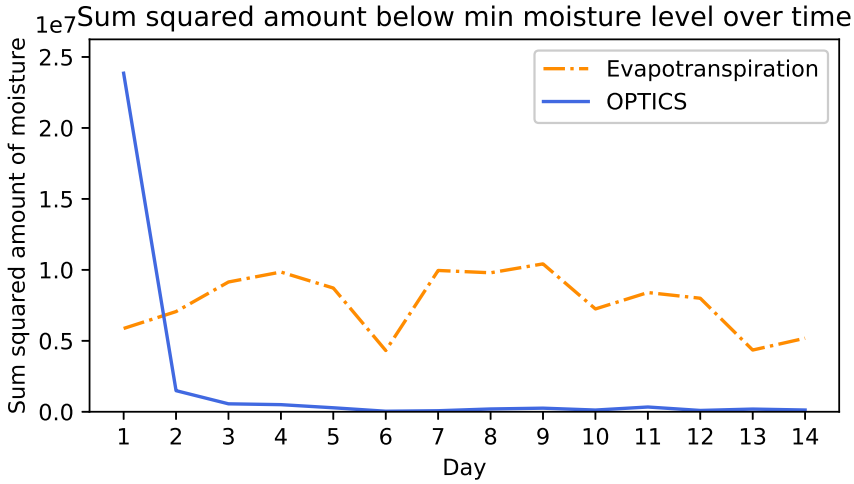


Figure 5.15: ET vs OPTICS quality of service results (lower is better)

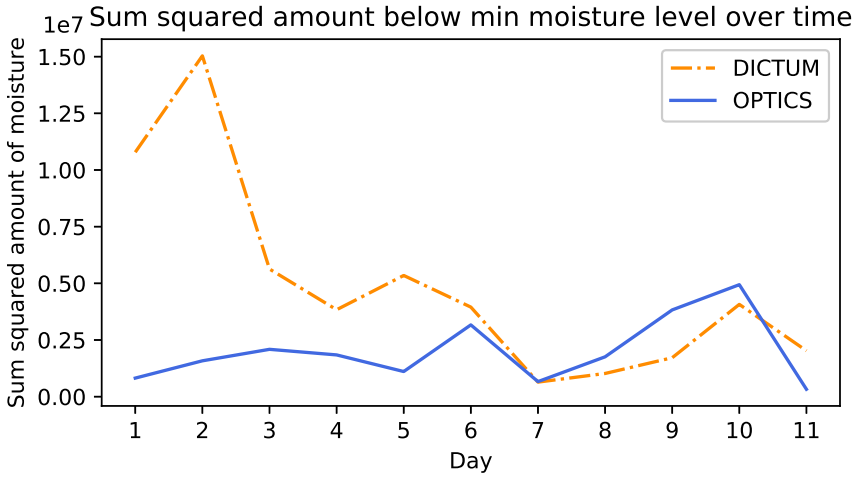


Figure 5.16: DICTUM vs OPTICS quality of service results (lower is better)

ment system must either maintain or improve the quality of service. Figure 5.14 shows the raw moisture data for each sprinkler in the field for both ET and OPTICS. The black horizontal line in the center shows the permanent wilting point (pwp), representing the minimum moisture level we with to always maintain. We can see that the ET system spends more time and in some cases it is well below that mininum threshold for many of the moisture sensors. In order to quantify how much below the minimum moisture threshold (pwp) each system spends over time, we plot the sum of the squared amount below the minimum moisture level over time in Figures 5.15 and 5.16 for the experiments comparing OPTICS with ET and DICTUM respectively. We used the squared amount to emphasize that the larger the amount below the pwp, the worst the quality of service provided.

In our deployment of OPTICS against the evapotranspiration control strategy in Figure 5.14, we see that at least two nodes in the ET system spend significant time below the minimum moisture threshold we wish to maintain, causing the ET system to provide reduced quality of service across the entire deployment as shown in Figure 5.15. In addition, looking closely, we can see that there are several nodes in the ET system that are significantly *above* our minimum moisture threshold we wish to maintain as well. This emphasizes the limitations of ET and the core of our work. The irrigated regions don't receive moisture the same way, and without learning these eccentricities, a system will often provide unsatisfactory quality of service and/or consume more water than necessary, as in our ET baseline system. This is a common problem in irrigation systems, as uniform irrigation across the field, without understanding local variations will result in moistures that can vary wildly. In order to make this uniformly-irrigating system provide perfect quality of service, we would have to irrigate *all* sprinklers enough to raise the driest area above our minimum moisture threshold. This would be a significant waste of water, as the rest of the space would be severely

over-watered.

It should be noted that before the OPTICS system was turned on, the ET control strategy was used to irrigate both irrigation systems. This can be seen in day 1 in Figure 5.14, where several sensors have moisture levels below our threshold. However, in the first 4 days, the OPTICS system learns these increased needs and applies tailored moisture to raise them above the threshold, resulting in a steady improvement in quality of service as shown in Figure 5.15. In the comparison against DICTUM in Figure 5.16, we can see that several days are spent with a poorer quality of service in comparison to OPTICS. On days 1 and 4 in particular, it's clear that DICTUM's model believes it can afford to reduce water consumption, causing decreased quality of service on the following days. Likewise, although the DICTUM system attempts to send increased water on day 2, the mismatch between the model and the physical deployment does not send enough to reach this goal.

Overall, OPTICS improved irrigation quality of service by an average factor of 4.04 in comparison to the ET system, and if we ignore the first day, by a factor of 24.7. When compared to DICTUM, OPTICS improved irrigation quality of service by an average factor of 2.47. In all, our system provides significant improvements with respect to quality of service than the other irrigation systems.

5.5.2 Water Consumption

When a new irrigation control system is considered, a primary concern is the efficiency of the proposed system. The system's ability to return its investment based on increased efficiency will often dictate the acceptance of the technology. In addition, the environmental benefits of reduced freshwater consumption are clear and help promote system adoption.

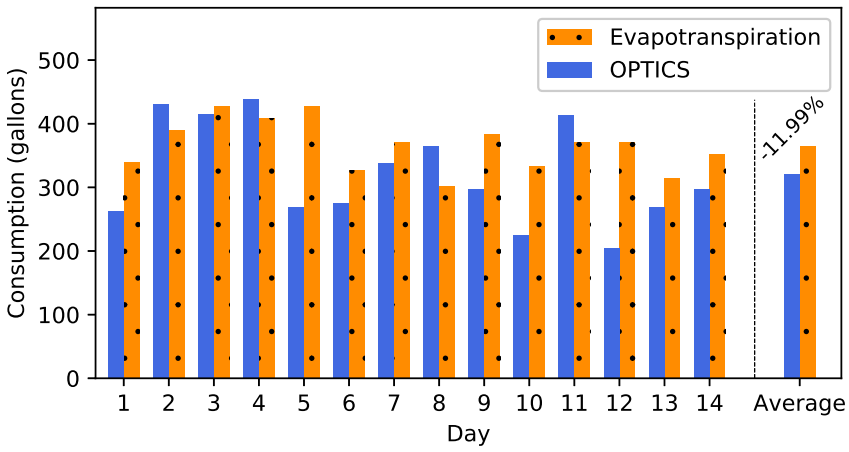


Figure 5.17: Consumption of ET vs OPTICS

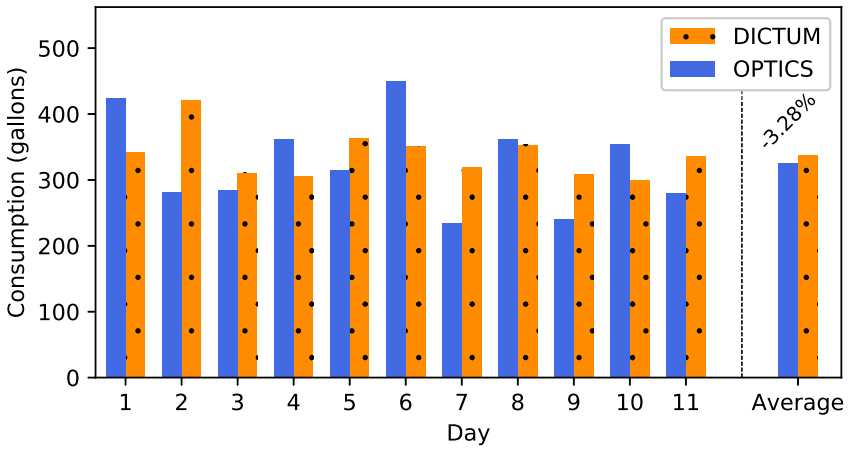


Figure 5.18: Consumption of DICTUM vs OPTICS

In our experimental setup, the water source for each sprinkler is pressure-regulated to the industry standard, 40psi. A pressure-regulated sprinkler head distributing water at a known angle uses a clearly-defined amount of water per unit time, as described in the sprinkler documentation. By tracking exactly when each sprinkler is actuated by the system, we can determine very accurately how much water has been consumed. In this way, we compute the daily system consumptions for the ET vs OPTICS and DICTUM vs OPTICS, as shown in Figures 5.17 and 5.18 respectively.

As discussed in Section 5.5.1, the experimental system started with moisture levels significantly beneath our desired threshold. For this reason, the first 5 days of OPTICS control in our first deployment had steadily improving quality of service as it learned its models and raised moisture to satisfactory levels, but on days 2 and 4 this resulted in slightly higher water consumption than the ET system. Day 8 saw slightly higher consumption than the ET system as well, but these were some of the ET system's worst days in terms of quality of service as shown in Figure 5.15.

Day 11 of our first deployment saw increased water consumption as well, caused by a hardware malfunction. A command telling 4 of the 9 OPTICS nodes to "Stop irrigation" was lost due to a faulty USB connection to the basestation mote, causing unintentional irrigation that was not corrected until failsafes in the node's firmware automatically disabled irrigation. This caused the OPTICS system to consumed more water than intended, as shown on day 11 in Figure 5.14. We can also see OPTICS's ability to recover from such mistakes on days 5 and 12, where significantly less water is required due to the residual moisture from the day before.

The water consumption of OPTICS when compared to DICTUM as seen in

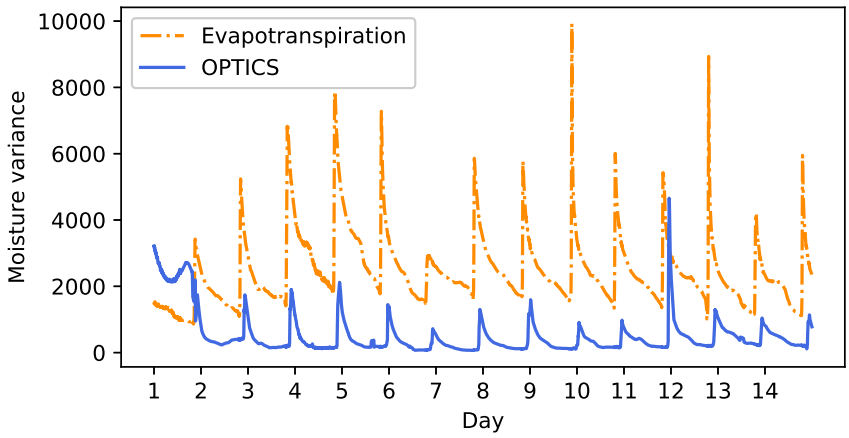


Figure 5.19: Spatial moisture variance over time, deployment 1 (lower is better)

Figure 5.18 was much closer, with some days using slightly less and others slightly more water. However, on days 1 and 4, it is particularly evident that although DICTUM saves significant water, the quality of service suffers immensely.

Across the two deployments, the OPTICS system reduced water consumption by an average of 11.99% compared to the ET system, and 3.28% compared to the DICTUM system.

5.5.3 Moisture Uniformity

Moisture uniformity is not considered a primary goal, but is a nice side-effect of good irrigation control. If an irrigation system provides water coverage that is not uniform, drier turf can turn brown and crispy, while wetter turf can turn a different shade of green and become muddy, both affecting aesthetics and usability of the space. With our learning model, a core assumption is that as losses occur at different rates at different spatial locations, we are required to apply different amounts of water across the space. However, if we apply the appropriate amount of moisture across the space, the water in all sensing locations will settle towards

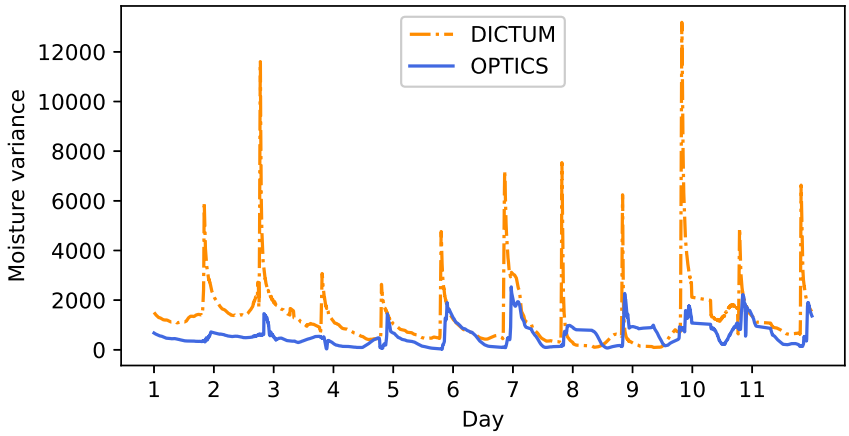
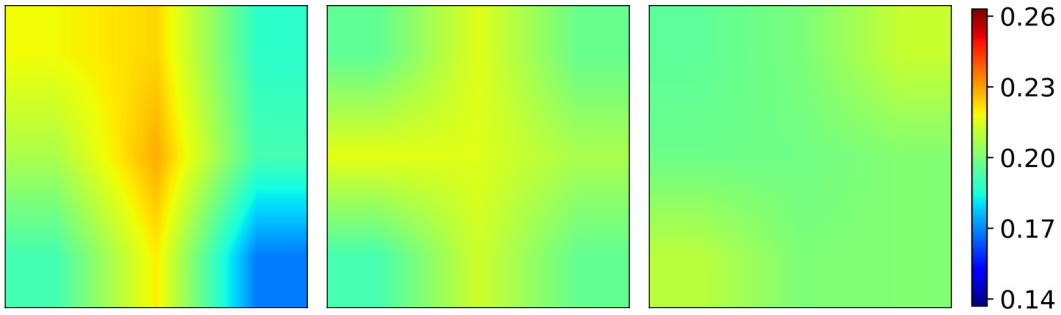


Figure 5.20: Spatial moisture variance over time, deployment 2 (lower is better)



(a) ET (b) DICTUM (c) OPTICS

Figure 5.21: Average VWC (%) of compared systems across deployments

a uniform distribution as the moisture levels approach the uniform minimum moisture *just before* irrigation.

Throughout our deployments, we periodically recorded the soil moisture at all sensor locations in both side-by-side irrigation systems. To determine the moisture uniformity, we step temporally through these data traces and compute at each timestep the variance in soil moisture readings for both irrigation systems. This moving variance of soil moisture across the irrigation system can be seen for deployment 1 in Figure 5.19 and for deployment 2 in Figure 5.20, where lower values indicate a more uniform soil moisture level across the deployment. Each day during watering, the variance for all systems is seen to spike, as different areas will not receive water at the same rates. Then, as the soil moisture diffuses across the space, the variance will decrease to a daily minimum before the next irrigation occurs. It is clear in both deployments that the OPTICS system is able to maintain a much lower moisture variance in general, indicating OPTICS's ability to learn the heterogeneous water needs across the space and apply water in response. In the first deployment, the variance is seen to be higher on days 1 and 11, but these are due to OPTICS's poor initial soil moisture conditions and the hardware failure that resulted in unintentional excess water, as described in detail in Section 5.5.2. Additionally, the two figures demonstrate that the DICTUM system is able to maintain a tighter band of irrigation than the ET controller, providing comparably low variance at times later in the second deployment.

Finally, we consider each individual sensing location and compute its average volumetric water content (VWC) across the weeks of deployment and visualize them as a heatmap in Figure 5.21 for each of the three compared systems. For ease of visualization, a bilinear interpolation is used to produce the figure (a more complex interpolation such as cubic may create artifacts *between* sampling

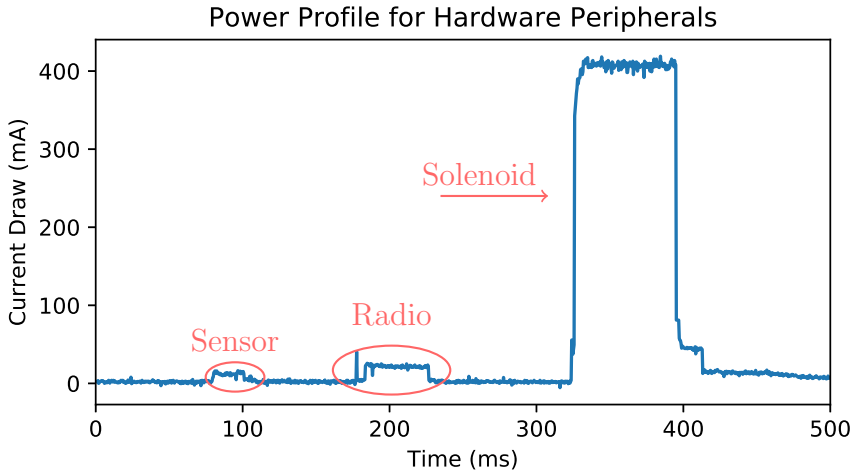


Figure 5.22: Energy profile of the sensor node

points). We see that the ET control strategy results in the least uniform coverage, with a heavily over-irrigated area around the center of the field, and an under-irrigated area close to the lower right corner of the field. In contrast, both DICTUM and OPTICS strategies result in a more uniform moisture distribution, with DICTUM being slightly above the minimum in a cross-like pattern, and OPTICS being slightly above the minimum in the upper right and lower left corners.

5.5.4 Energy Consumption

In our devices, the three peripherals that consume significant energy are the sensor, solenoid, and the radio, as visualized in Figure 5.22. Each sensor sample consumes 10mA of power for 10ms, and each flip of the latching solenoid requires 400-450mA of power for 50ms. In our system, to ensure we don't cut power too early, we add a safety band of 50% on the timing on both of these devices, powering each for 15ms and 75ms for the sensor and solenoid, respectively. The tmote sky [tmo] radio consumes 23mA max when in transmitting mode. With clever

use of these peripherals, this system can achieve a substantial system lifetime using our current power source of 4xAAs [l91].

In our 25 day deployment, we found that on average DICTUM flips the solenoid of a node 12 times per day. Assuming we sample our sensor 1 time per minute, flip the solenoid 12 times per day, and utilize a constant 1% radio duty cycle ratio across the full 24 hours for communication using techniques such as Low Power Listening [PHC04], even with our conservative timing of the peripherals, our system lifetime is estimated to be 1.7 years, with 95% of this energy consumed by the radio. However, as fresh data is only required less than 3 hours per day during irrigation, by simply leaving the radio off the other 21 hours each day while storing sampled data in memory to be sent at the beginning of the irrigation cycle, this lifetime is easily extended to ∼11.9 years. We note that the DICTUM system achieved similar system lifetime, but required a 4 D-cell power supply to do so, whereas the OPTICS system uses only a 4 AA-cell power supply. This reduced system consumption is due to more efficient voltage regulation hardware on our power supply board, and minor improvements in firmware solenoid control routines, preventing the solenoid from receiving power to enable/disable the valve when it is already in the desired state.

Due to the relatively small size of our irrigation system, radio communications between sensor nodes and basestation were observed to only require 1 or 2 hops using a simple, application-specific routing strategy. In a larger deployment, we would need to either provide more basestations to keep the number of hops from each node to a basestation small, or employ a routing protocol such as ORPL[DLV13], CTP[FGJ06], or ORW[GLS14] designed for low-power communication between sensor nodes and the basestation at large scale. Routing across a very large irrigation system will surely lead to an increase in energy consumption

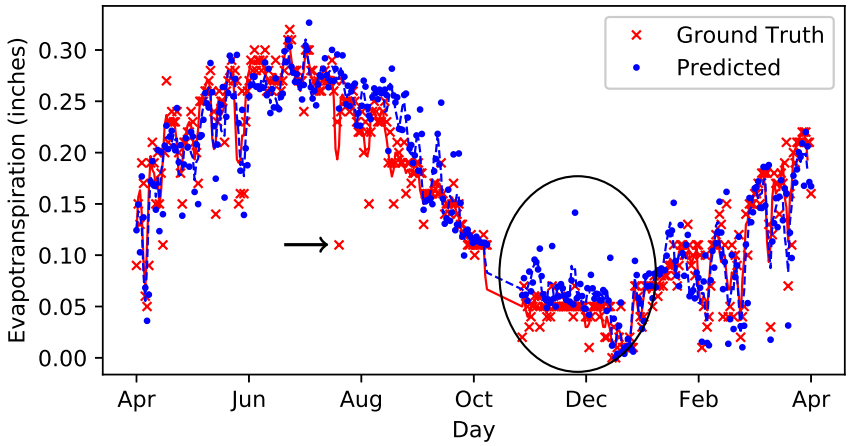


Figure 5.23: ET prediction of KNN regressor using historical temperature and humidity trends as features. Days that were particularly challenging to predict are emphasized with an arrow and an oval.

across the network. Fortunately, our application's low data rate and lax timing requirements will allow the system to de-activate the radio for the majority of the day, and duty-cycle the radio aggressively during irrigation when data communication between sensing nodes and basestations is required. In Section 5.8, we discuss techniques that will nearly completely eliminate data communication requirements between the sensing nodes and the basestations.

5.6 Performance Analysis and Evaluation

5.6.1 Weather prediction evaluation

Our weather prediction method uses hourly readings of at least one of the 4 ET indicators, temperature, humidity, wind, and solar irradiance to predict the resulting ET losses due to these weather conditions. When OPTICS is used to control a live irrigation system, it uses forecasted hourly readings of these ET

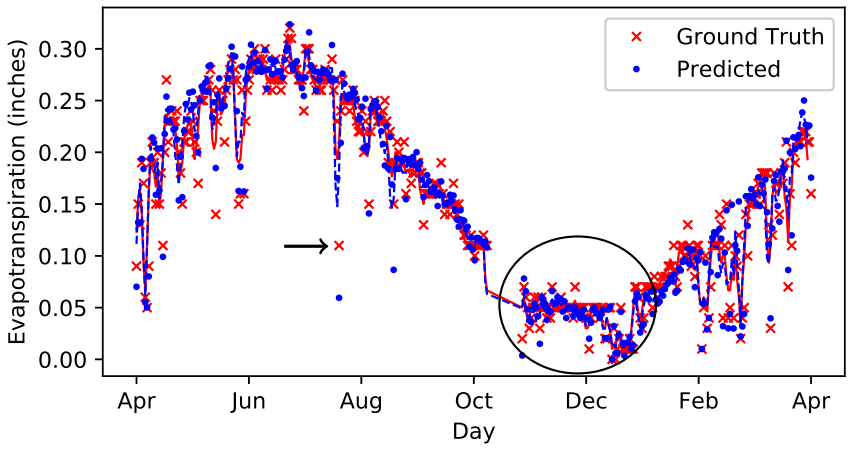


Figure 5.24: ET prediction of KNN regressor with all four features available. Days that were particularly challenging to predict are emphasized with an arrow and an oval.

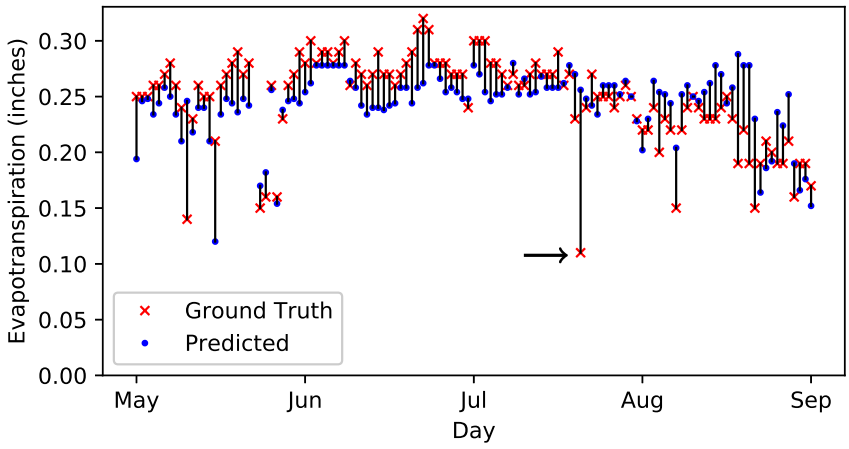


Figure 5.25: ET prediction of KNN regressor using *forecasted* temperature and humidity features. Days that were particularly challenging to predict are emphasized with an arrow.

indicators as input to our KNN regressor model, which then predicts the next day's ET losses. The use of these forecasted ET indicators incorporates some error into our final prediction that is outside our control, as these forecasts are provided by local weather stations or government sources that are imperfect. Before determining the end-to-end predictive accuracy of our solution, we first wish to remove the forecasting error that is out of our control to determine the accuracy of our KNN predictor on its own. To do so, we refer to historical data trends of the hourly ET indicators and the associated daily ET losses *of the same day* to determine the accuracy of this mapping between indicators and resulting ET. By doing so, we remove the error associated with the error forecasting the hourly trends, and focus on best-case predictive power.

Figure 5.23 shows the daily ET as predicted using the *same day's* hourly vectors of temperature and humidity, as well as the ground truth ET that occurred and a moving average of each. We can see that in general, the predicted ET follows the trend of ground-truth ET, but on one day in mid-July (emphasized with an arrow) our system predicts a "normal" ET level when in fact an abnormally-low ET value occurs. Additionally, on several days in early December (emphasized with an oval) we can see that our system predicts more abnormal ET when in reality the weather trends were closer to the previously-seen weather trends. In contrast, Figure 5.24 demonstrates the ET prediction *IF* all four ET indicators were available. We can see that our predicted ET levels are much closer to the ground truth on the days in mid July and early December where regression using only temperature and humidity did not provide enough information in Figure 5.23. This indicates that days in Figure 5.23 with particularly high error are caused by heavy influence due to solar and wind factors on these days. In practice, the fit quality using temperature and humidity as features is more indicative of the real-world accuracy of this prediction technique as these

Data Used	RMSE	NRMSE	Cumulative pred. ET	Rel. error
All four indicators	.0171	.0813	30.05”	+0.9%
Temp/humidity	.0325	.1547	30.64”	+2.9%
Temp/humidity (Forecasted)	.0348	.1655	29.29”	-1.6%

Table 5.3: Prediction across 120-days of collection

features are the most likely to be available using forecasting. Table 5.3 shows the root mean square error (RMSE) of prediction with these feature subsets, as well as the RMSE normalized by the magnitude of the individual ground truth values (NRMSE). We can see that availability of all four features results in half the error as prediction using just temperature and humidity.

Next, to better understand the practical accuracy, we re-introduce the forecasting of hourly trends by using the *forecasted* temperature and humidity from the day before as input to our KNN regressor using 120 days of collected forecasting data around our period of deployment. Figure 5.25 shows the daily ground truth ET, along with the predicted ET using forecasted hourly trends, along with the associated error of each prediction. Here, we can see clearly that our challenging day in mid July (emphasized with an arrow) had the most error as it also did *without* forecasting in Figure 5.23, as neither of these predictions had wind and solar exposure information available as features. Whereas prediction using these ground-truth features resulted in a root mean square error of .0325, we find that using forecasted features increases this error metric to .0348. Put another way, forecasting our hourly data trends increases our daily prediction error by just 7% in comparison to performing regression using ground-truth trends.

The cumulative distribution function (CDF) of the individual daily errors in forecasted ET prediction across our 120 days of data collection can be seen in Figure 5.26. Here, we see that about 50% of our predictions are within 10% of the

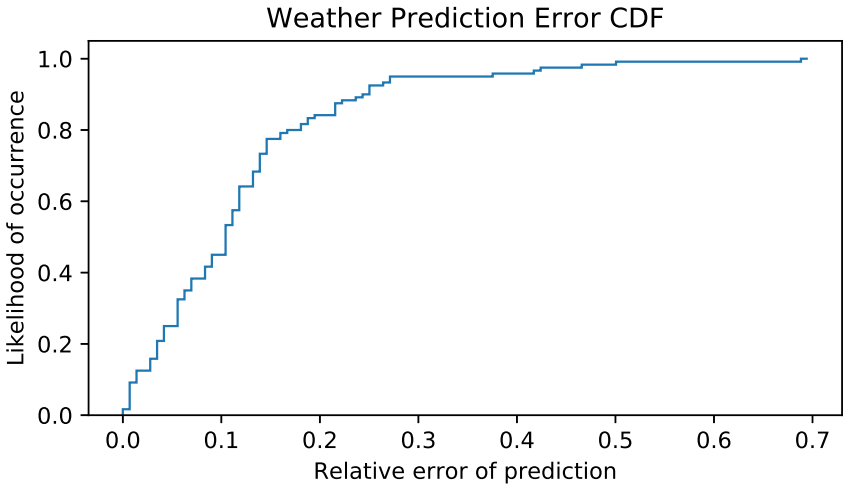


Figure 5.26: CDF of weather prediction errors

ground truth ET, and 90% of predictions are within 25%. To get a sense of the overall performance, we consider that across these 120 days, ground truth data shows that a total of 29.78 inches of ET losses occurred. Table 5.3 shows that the cumulative losses of our predictive methods with and without prediction using various features all fall within 3% of error across this period, which we believe is more than adequate for our irrigation controller.

5.6.2 Short-term model analysis

One of the high-level system parameters that affects our modeling is the choice of Δt, determining how far in the future our model must predict and the temporal granularity of our control routines. The correct way to determine the *optimal* choice for this parameter would be to repeatedly vary Δt, launch a full irrigation system deployment with modeling and control using the new choice of Δt and analyze the primary metrics of success (water consumption and quality of service). However, due to the substantial effort required to repeatedly perform full

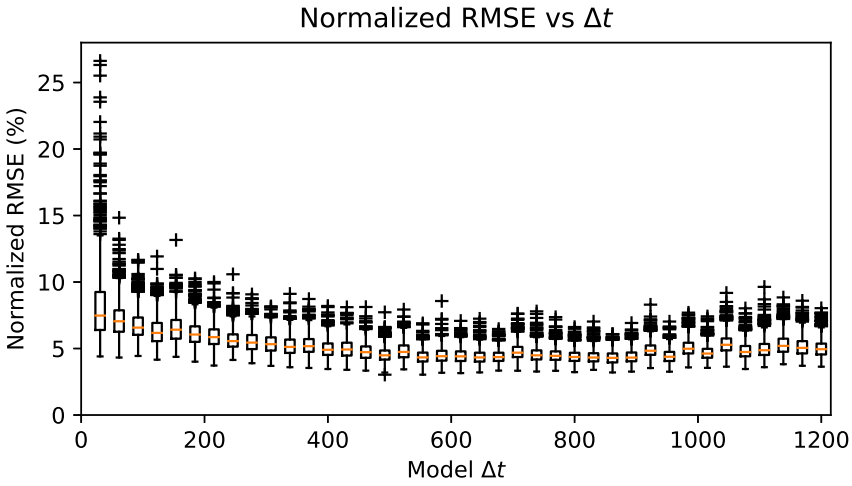


Figure 5.27: Box and whisker plot of errors as model Δt is varied. Crosses denote outlier values.

experimentation makes this process impractical. Instead, we consider the effect of this parameter choice just on the accuracy of our short-term model's predictive accuracy.

Across our irrigation system deployment, we store detailed records of the schedules used and the sensor trends at all node locations over time. To determine how model accuracy responds to changes in model Δt, we take these deployment data traces and step through temporally at changing values of Δt, building training pairs of $[\mathbf{s}_t, \mathbf{f}_t]$ vectors as inputs, and $[\mathbf{s}_{t+\Delta t}]$ vectors as outputs, as described in detail in Section 5.2.3. We then train our linear regressor on a random 90/10 train/test split of the dataset and note the model prediction accuracy on the test set. As the random training/testing data split will change the prediction accuracy, this is repeated many times for each value of Δt, with the error distribution box plot as shown in Figure 5.27, and the mean error shown separately in Figure 5.28. The error shown in these figures is the root mean squared error across the full dataset, averaged across all datapoints and across

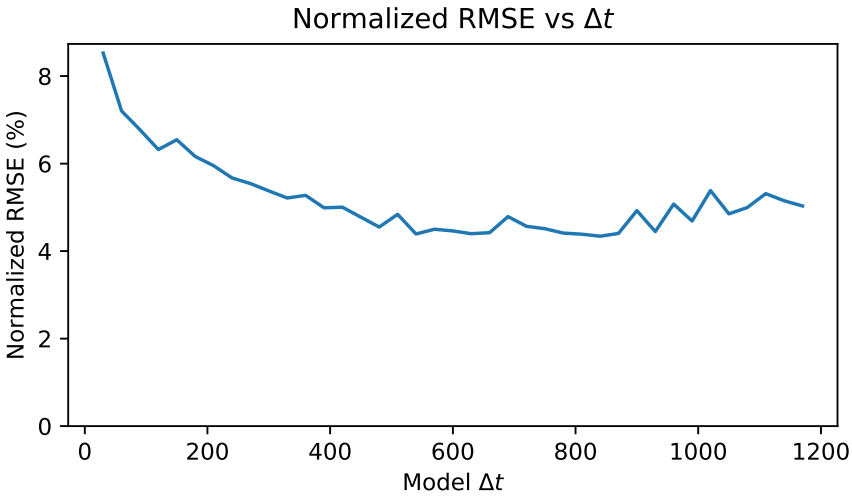


Figure 5.28: Mean error as model Δt is varied

the K sprinklers. The change in moisture that occurs in Δt units of time is very small relative to the full range of moisture values that can occur, so we then normalize this absolute error to the mean moisture change that occurred in Δt units of time. In this way, we show the relative error between the ground truth moisture change and the predicted value for each individual sensor prediction Δt seconds in the future.

In these plots we can see that when Δt is very small, the average prediction error quickly increases as noise becomes a dominant factor in prediction. In our experiments, our finest temporal granularity in soil moisture data is 30 seconds, and using $\Delta t = 30s$ to build our linear model, we see a maximum outlier error on the order of 25% in Figure 5.27. In addition, we see in Figure 5.28 that at $\Delta t > 800s$, our error begins to increase again, with the lowest error experienced around $600s < \Delta t < 800s$ at around 4.5% mean relative error. It is important to note that this range of optimal Δt is specific to the environmental conditions that exist in *our* irrigation system, and will be different if the system is installed

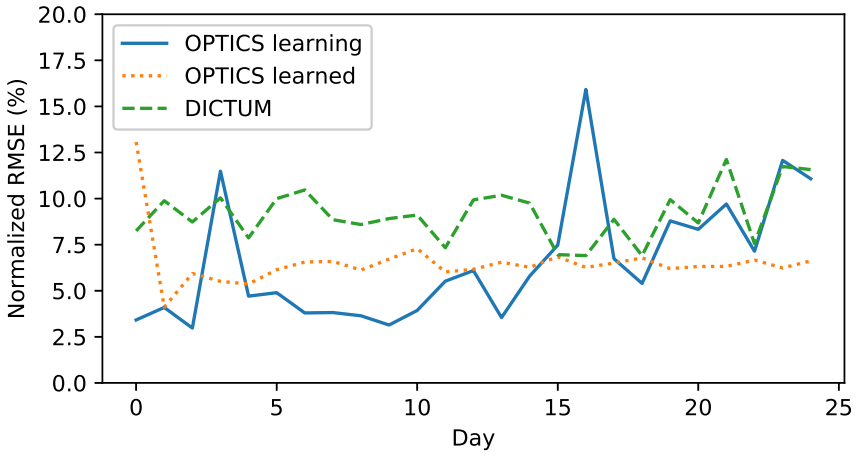


Figure 5.29: Daily error of DICTUM, Learning (Re-learns each day w/new data), and Learned (Trained once with all data)

elsewhere, due to differences in sprinkler type, soil parameters, topography, etc. At the $\Delta t = 60s$ we chose to use in our experimentation, we can see that the average error can be expected to be around 50% higher than at the optimal Δt value. This sub-optimal parameter was selected for our experiments as it would allow us finer granularity of control, and as we frequently re-correct our moisture conditions during optimization using the most recently collected moisture data, we ensure that our model does not cause significant drift in the system.

In addition, to evaluate the OPTICS model, we consider the single-step predictive error as it changed across our 25-day dataset from our live deployment in Figure 5.29. Here we consider the error of three different models: the first is the PDE model used in the DICTUM system, as it was configured for our experimentation in our deployment. The second is the OPTICS system, employing an *on-line model improvement*; this model is first trained using our training day of data as described in Section 5.2.3, and then retrained on each day D using the previous $D - 1$ days of data. This technique simulates a model that has been

freshly-installed on a new irrigation system, and must learn its model adaptively day-by-day. The third model is a pre-trained OPTICS model, where all data from our pre-deployment *AND* our deployment were used to train a model beforehand that then does not change. This technique simulates a model that has been trained over time on our irrigation system, and no longer requires updating. As in the previous sections, the error shown is the average single-step predictive error of a single sensor node, normalized to the average change in moisture experienced in Δt seconds, with $\Delta t = 60s$ as it was in our deployment.

As we can see in Figure 5.29, the OPTICS model that is constantly being retrained has more variation in errors, with consistently lower error on days 4-14 and significantly higher error on days 3 and 16 where error spikes occur. These periods of high error are caused when the model experiences moisture conditions or valve scheduling that it has not experienced before in the data used for training. We can see that these increases are temporary, and that once these conditions are included into the model, the error decreases again quickly in the following days. In all, we see that some days this model can have as little as 3% error, with temporary spikes as high as 16.5%. In comparison, the DICTUM PDE model tends to be more stable in its predictive error as the moisture predictions are not data-driven, but made based on deterministic models that can still operate reasonably when faced with unseen conditions. The accuracy of this model across our deployments tends to lie between a minimum error of 7% on day 16 to a maximum of 12.5% on day 21. Finally, we can see that the OPTICS model that is pre-trained with all data is significantly more stable in its predictions, with the exception of unusually-high error in the very first day. This technique shows a minimum of about 4% error and a maximum peak of 12.5% error on the first day, but provides an average predictive error of just 6.5% across the deployments. In general, we find that the OPTICS model with on-line updating

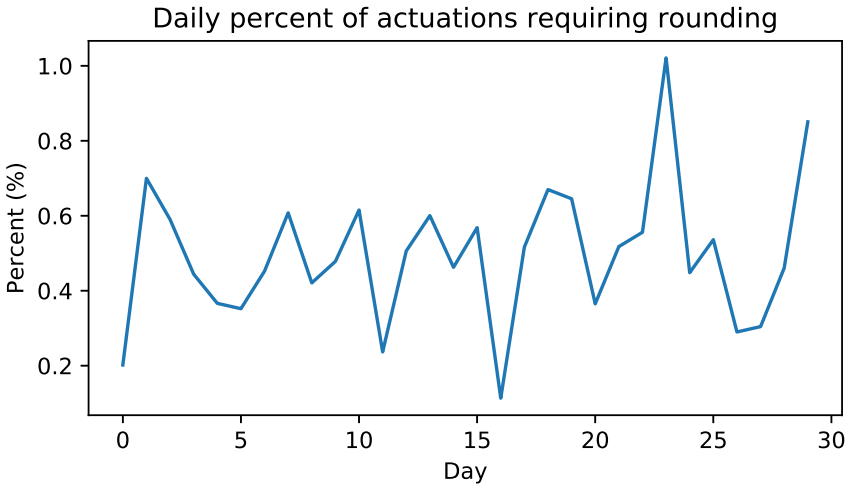


Figure 5.30: Percent of actuations requiring rounding across the two deployments

reduced predictive error by 26.0% compared to the DICTUM PDE model, and the pre-trained OPTICS model reduced predictive error by 30.4% compared to the DICTUM PDE model across our deployment dataset. The improved predictive accuracy of OPTICS stems from its ability to learn the heterogeneous factors of water movement that are assumed to be homogeneous in DICTUM's PDE model. The slight average accuracy improvement of the pre-trained model is expected, as the more extensive dataset it had been trained with included the atypical conditions that were difficult for the model that was doing on-line updates.

5.6.3 Effect of optimization simplification

As discussed in Section 5.3, we simplify an integer linear program (ILP) by relaxing the integer valve constraints to be continuous on $[0, 1]$, and then rounding the resulting values to be binary. Although this makes the optimization a linear program (LP) that is easier to solve, it can potentially cause the resulting schedule to deviate from the one chosen by the optimizer. To determine this effect, we

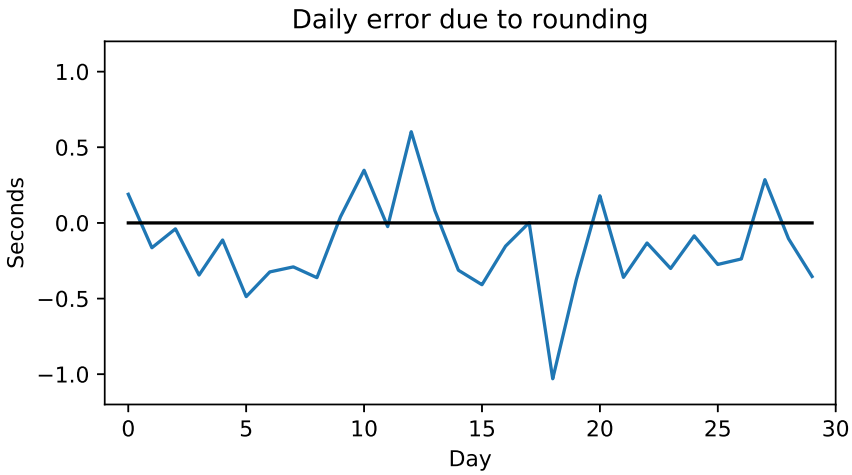


Figure 5.31: Daily error in irrigation schedules due to rounding of LP solution

consider the individual valve actuations produced by the optimizer across our 4 weeks of deployment. As shown in Figure 5.30, we find that the resulting schedules tend to lie on the 0/1 integer boundary, with fewer than 1% of actuations requiring rounding. Figure 5.31 shows how the amount of irrigation deviates each day due to this rounding, and we can see that the worst day (day 17) has a resulting schedule deviation of just over 1 second, negligible when the total irrigation is on the order of an hour.

5.6.4 Performance and scalability of OPTICS

In the deployment of the OPTICS system, we chose to re-run the optimization using the most up-to-date sensor readings every control timestep, chosen to be 1 minute. This has the benefit of preventing drift due to errors in the short-term model by constantly re-correcting, but imposes an upper bound on optimization time. To determine the effect this limitation has on the scalability of the system, we train a short-term model using spatially-repeating trends of deployment

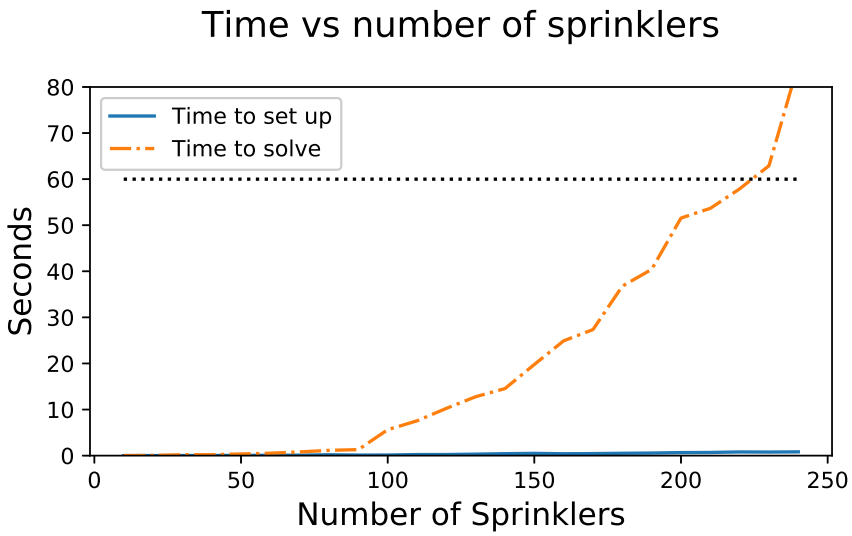


Figure 5.32: Time required to optimize schedules

data to simulate an irrigation system that has an arbitrarily large number of sprinklers (K), and track the time required to optimize schedules. This can be seen broken into two discrete time intervals in Figure 5.32, where set-up time is the time required of our chosen optimization tool to set up the constraints, and solve-time is the time to solve the LP. Also shown as a dotted line is the 60s maximum available time to solve optimization. We can see that as the number of sprinklers in the system increases, the amount of time required to optimize schedules increases as well, until solve-time exceeds our 60-second bound around $K = 225$ sprinklers.

Appendix A.3 derives how to compute the spatial coverage area of an irrigation system with K sprinklers of radius R arranged in a square grid in the three possible irrigation system architecture arrangements. As shown in Figure 5.32, enforcing our 1 minute maximum solve-time, the OPTICS system can provide optimal schedules for an irrigation system of $K \approx 225$ sprinklers. Estimating using the third case of Equation A.9 which applies to our coverage conditions

here, we find that for sprinklers such as those used in our deployment with radius $R = 30$ft and assuming a conservative, dense sprinkler deployment where each sprinkler can reach all the way to the next closest sprinkler in the grid, the OPTICS system can cover ≈ 5.2 acres of land using a single short-term model. With sprinklers used more commonly in sports fields with radius $R = 60$ft, this coverage increases to ≈ 20.7 acres of land with a single model. Although the largest irrigated sports fields, 18-hole golf courses, cover an average of 100 acres [gcs], these irrigated spaces tend to be naturally partitioned with golf cart paths that would prevent these areas from being continuous anyways. Furthermore, due to the very slow speed of moisture movement through the soil, in practice moisture has a relatively limited region of movement. Both of these characteristics of the space allow us to break such a large irrigated space into smaller, more manageable models of water movement. By doing so, we facilitate superior performance and scalability, as further discussed in Section 5.8.

In practice, although the OPTICS system is *able* to optimize schedules for an irrigation system with hundreds of sprinklers, the *training* of a short-term model at this scale becomes challenging. In our deployment, we simply stepped through each sprinkler for a fixed period of irrigation of 1 hour, but scaling this technique to a system with K sprinklers would take K hours. As discussed in Section 5.8, however, in practice each sprinkler's area of effect is limited by the sprinkler radius and water movement speed across the surface and through the sub-surface. In the most densely-configured lawn irrigation systems, the actuation of one sprinkler may affect the soil moisture at the next closest node in the grid, but not the next. Due to this limitation of movement, we can break an irrigation system's sprinkler grid into blocks of 3x3 sprinklers. Then during training we can actuate, for instance, the sprinkler in the $(1, 1)$ position of each 3x3 block simultaneously across the entire space, and any sensing node will have at most 1 active sprinkler

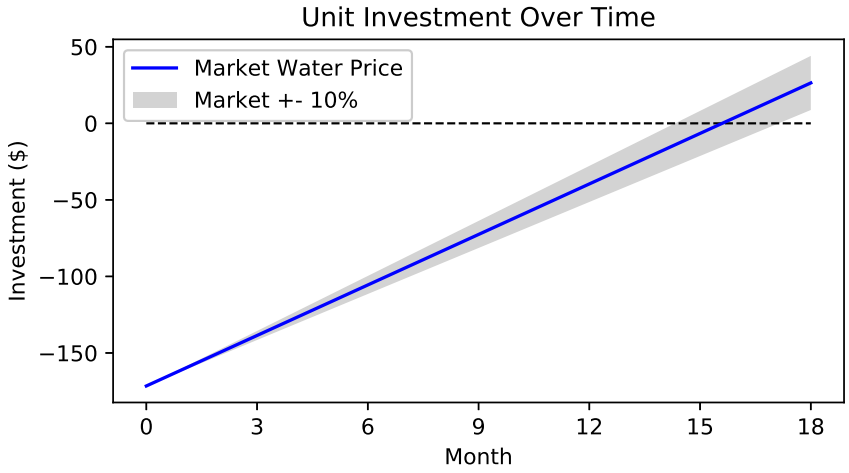


Figure 5.33: Return on investment curve

within range. At the end of the hour, we can activate the $(1, 2)$ position in each 3x3 block, etc. until we have activated all 9 sprinklers in each block. In this way, we can irrigate an arbitrarily large grid of sprinklers in at most 9 hours without cross-contamination between active sprinklers.

5.7 Return on Investment Analysis

There are social and political motivations behind a project like this, but a primary consideration before the purchase and installation of a replacement control system is the return on investment, or the time it takes a system to save enough money to cover the cost of installation and usage. To calculate the return on investment, we must take into account the initial cost of the replacement system and the monetary savings expected from the increased efficiency of the replacement system. In Table 5.4, we list the cost of production for one of our irrigation control devices. Other than screwing these devices under each sprinkler head, the original infrastructure does not need to be modified in any way.

Financial savings stem from the system's ability to save water. In Figure 5.33,

Table 5.4: Sprinkler Node Manufacture Cost

Component	Price
Mote	$37.57
Moisture Sensor	$110
Batteries	$4
Solenoid	$15
Waterproof Enclosure	$10
Manufacture & Assembly	$10
	$186.57

we show how OPTICS will return its own investment based on water savings alone. As the return on investment will differ depending on the sprinkler heads used, we consider OPTICS's installation on a representative University sprinkler head with 11.99% water efficiency improvement compared to industry best, and the fact that our University pays $5.60 per thousand gallons for irrigation. Additionally shown in Figure 5.33 is a ±10% band in water pricing, to take variation of water pricing into account. In this way, each unit device is estimated to return its investment in 14-17 months. OPTICS's improved efficiency and reduced need for a powerful computation machine makes it even more financially attractive than DICTUM, especially due to DICTUM's potential additional setup costs in parameter measurement that we are not including here.

5.8 Limitations and Future Work

As shown in Section 5.7, the unit cost of our control devices is dominated by our soil moisture sensor, which was chosen in our experiments due to its very high accuracy (±3%[dec]). As the size of the system scales up, the initial cost of the

system may become impractically high. To scale to very large systems, then, we must consider the use of significantly cheaper soil moisture sensors at the price of slightly less accurate measurements. For instance, previous work [SB92] has found that the Watermark line of soil moisture sensors can be calibrated to have an accuracy of ±5%, and costs only 35$. By using these sensors instead, the unit cost of our sensor/actuator devices would be reduced by 40%, with a similar reduction in return on investment and increased adoptability.

As OPTICS is designed for turf irrigation, it is unlikely to provide benefit in shrubbery or tree irrigation, where much simpler drip irrigation systems can be used. In addition, some very different turf species may require varying minimum moisture levels to maintain health. Therefore, in turf irrigation systems where the system covers *multiple* turf species such as a golf course irrigation system, minimal configuration will be required beforehand to tell OPTICS where each grass type is located. For instance, "Sprinklers 1-90 irrigate turf species A, Sprinklers 91-100 irrigate turf species B", so that differing minimum moisture constraints can be spatially assigned based on the species. Assigning heterogenous minimum moisture constraints will not require any modification to our processing pipeline.

As discussed in Section 5.2.2, we chose to fit our Long-term moisture losses as an exponential decay function and re-train our model each day using all previous data. However, the system may be more reactive to changing conditions by only taking the latest N days of data in retraining or another weighted updating strategy to put emphasis on the freshest data, representing the latest environmental conditions. Furthermore, in some irrigation systems where the sprinklers do not provide a uniform coverage onto the ground, this method may be sensitive to delayed effects in moisture movement, and may benefit from another modeling strategy that does not require the losses to occur precisely between the end and

the start of irrigation, but rather follows the min/max moisture trends across the daily cycle.

5.9 Conclusions

In this chapter, we seek to improve the efficiency of turf irrigation systems by designing, implementing and evaluating OPTICS, a data-driven control strategy that automatically adapts to local conditions and weather patterns, requiring virtually no human input in both setup and maintenance. Our system reduces the water consumption by an average of 12.0% against the industry best and 3.3% against state-of-the-art research. Despite this reduced water use, OPTICS was found to reduce turf exposure to unhealthy levels of moisture by a factor of 4.0x and 2.5x with respect to the two systems mentioned above. The OPTICS system is expected to return its investment in 14-17 months based on water savings alone.

CHAPTER 6

WISDOM: Watering Intelligently at Scale with Distributed Optimization and Modeling

Although the modeling techniques introduced in the previous chapter were found to substantially improve the quality of service provided to the plant and the irrigation system efficiency, the proposed systems require a centralized controller to process all data and make control decisions. This introduces significant computational bottlenecks in optimization due to the growth of the global models used at scale, a single point of failure that can cripple the system, potentially causing massive water waste as experienced in OPTICS, and increases the overall energy consumption of the sensor network as all data must be forwarded across the network for processing. With the additional requirement of periodic battery changes in these systems, it is unlikely these systems could be adopted for control of irrigation systems at scale.

In this chapter, we recognize that nearly all of the aforementioned system limitations are due to centralized architectures, prompting the development of WISDOM, a *distributed system* for the control of truly large-scale irrigation systems whose primary intellectual contributions are as follows:

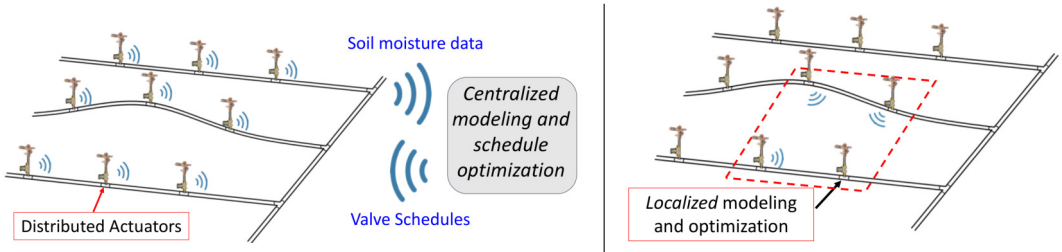
(1) Evaluating the mechanics of water movement in the space, we modify existing modeling and optimization techniques for use in *localized* neighborhoods, and introduce a processing pipeline for *distributed* irrigation decision making on

the edge devices themselves. By operating within local neighborhoods, we remove the point of failure, computational bottleneck, and energy penalties caused by centralized control, making the system truly scalable.

(2) As no existing system architecture provides the low-power operation of the traditional embedded device *and* the processing capabilities required for model training and optimization on the edge devices, we provide a low-power embedded device with a relatively high-performance co-processor to meet these requirements. With power duty-cycling by the low-power "master" device, the co-processor *vastly* improves storage, memory, and processing power, while incurring a relatively small energy penalty during operation. As far as we know, WISDOM is the first system to utilize a co-processor architecture for distributed computation in data-driven modeling and control.

(3) Finally, recognizing that our devices are collocated with sprinklers throughout the space, we take advantage of the available fluid energy flowing through them with small water turbines installed underground that allow us to harvest energy for battery recharge during irrigation. We show that even with the additional energy requirements of our co-processor, the energy harvesting capabilities allow the WISDOM system to operate indefinitely with true energy independence. In the wireless sensor network domain, WISDOM is the first to utilize sprinkler water source for energy harvesting.

Across 4 weeks of deployment and extensive analysis in simulation, we demonstrate that the WISDOM system is capable of achieving *all* of the quality of service and efficiency benefits of existing systems while maintaining true energy independence while controlling irrigation systems at *any* scale.



(a) Centralized architecture

(b) Distributed architecture

Figure 6.1: Comparison of OPTICS (left) and WISDOM (right) system architectures

6.1 System Overview

WISDOM is a *distributed control system* for lawn irrigation at scale, made up of a wireless network of sensor/actuator devices that sit below each sprinkler in the irrigation system. Each device is managed by a low-power master device in the form of an SAM R21 [sam] SoC, which can perform simple computational tasks, speak wirelessly with sister nodes in the space with its 802.15.4 compliant radio, and maintain device peripherals. These peripherals include a soil moisture sensor to monitor the local soil moisture status near the device, a solenoid allowing the master device to control the flow of water to the sprinkler independently of the rest of the network, and an onboard co-processor (Raspberry Pi Zero [rasa]), providing significant storage and computational capabilities directly on the edge devices. With the inclusion of the co-processor, the end devices have the ability to make intelligent decisions for irrigation *without* the need for cross-network data communication for centralized decision-making. To ameliorate energy costs associated with co-processor operation, each device is equipped with its own energy harvesting capability using a water turbine that activates when the attached sprinkler is activated, recharging the batteries. In sum, each device in the system has the ability to monitor local conditions, communicate with the network to

make sprinkler actuation decisions, and put those actuations into effect.

For a plant to remain healthy, three conditions must be met. There must be adequate solar exposure, the soil must have the correct nutrients in appropriate amounts, and sufficient water in the soil must be available near the roots for uptake. Although an irrigation system has no control over solar exposure and soil nutrient availability, its primary purpose is to distribute water where it is required to maintain adequate moisture levels at all locations in the space. In plant physiology literature, the level of soil moisture below which the plant can no longer extract fluid through its roots is known as the Permanent Wilting Point (θ_{pwp}), and as such serves as the lower bound of acceptable moisture levels in our irrigation system. While more conservative moisture thresholds exist such as the crop fraction for no stress [APR98] that may be more appropriate for system installation on decorative turfgrass, we choose the more restrictive θ_{pwp} as it serves as the most restrictive and difficult goal for the irrigation system, and thus helps demonstrate the WISDOM system's ability to meet even the most strict requirements. Although the θ_{pwp} threshold may be constant across the space, Chapters 4 and 5 have demonstrated across several irrigation system deployments that uniform water application will not result in uniform soil water levels, due to differences in the way water will move in different areas. To learn these local variations, we learn models of water movement over time, trained by data traces collected at each location, to be used to compute optimal irrigation customized to the space.

As shown in Figure 6.1a, each device in the OPTICS system transmits its data to a centralized controller for processing. The controller trains a *global* water movement model, which is then used to optimize valve schedules. These schedules are then re-distributed to the devices for actuation. With a large ir-

rigation system, the global water movement model makes schedule optimization time too slow as later discussed in Section 6.5, increasing network traffic causes forwarding nodes to consume more energy, and any lapse in communication with the controller can cripple the system.

Although we could break a very large centralized system into smaller centralized systems to help overcome the computational limitation, such a solution does nothing to improve the other fundamental limitations, single point of failure and cross-network data communication. For these reasons, we design the WISDOM system to be *fully distributed* as shown in Figure 6.1b to overcome all of these issues. As the WISDOM system runs, its installed devices periodically measure and log the state of moisture in the space, as well as the on/off status of the solenoids. Each device shares this information with all devices in its *local neighborhood*, which in practice is defined to include all nearby devices that are within radio range. Over time, using these collected data traces, each device learns *localized models* that describe the effect of actuation by all nodes within the neighborhood on the soil moisture distribution across the neighborhood. Likewise each device learns how its own local soil moisture will degrade over time after irrigation has ended, in a two-model technique inspired by the OPTICS system. These models, trained by the co-processor, are then used by the co-processor in optimization to find schedules that minimize system water consumption while maintaining adequate moisture levels everywhere in the system, our two primary goals. By performing these tasks in a distributed way, we enable a system that can robustly scale to control an irrigation system of any size, without sacrificing system efficiency or quality of service, as later discussed in Sections 6.4 and 6.5.

To maintain a long system lifetime even during seasons when energy harvesting is unavailable, we must be cautious with our energy budget to prevent

unnecessarily burning stored power with our co-processor when it isn't required. Each day, all master devices in the network wake up and distribute all buffered data to the surrounding nodes within the local neighborhood. Once all data is received, the master device will use the current moisture distribution in the neighborhood to make a decision whether or not irrigation is necessary on this day, returning to low power if it is not necessary. If irrigation is likely to be required, the co-processor is activated and all recently-collected data from the neighborhood is transmitted by wire to the co-processor to be integrated into the data-driven models, and an on-board optimization problem is initiated that computes valve scheduling within the local neighborhood to guide the moisture levels in the soil towards adequate levels. Adequate levels are defined as the daily *Goal State* of moisture as defined in Chapter 5, the soil moisture level to be achieved by the end of irrigation that considers the minimum moisture level θ_{pwp} we wish to maintain at all times, the local losses we have learned to expect from historical data, and the forecasted weather. As internet access is required to retrieve weather forecasts, weather prediction is the *single* parameter that must be provided by a central source.

$$\textit{Goal State} = \theta_{\text{pwp}} + \textit{Expected Losses} + ET_{\text{forecasted}} \tag{6.1}$$

Once optimization is complete, the schedules are sent from the co-processor to the low-power master device to be actuated by the attached solenoid. As irrigation continues, this optimization may be performed several times using the freshest data from the neighborhood to correct for any deviation due to error in the data-driven models used. When the daily operations are complete, the master devices deactivates the co-processor and returns the device to a low-power state until the next scheduled wakeup.

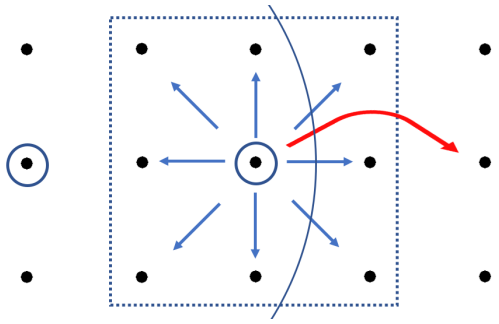


Figure 6.2: Minimum local neighborhood definition (square dotted box) for modeling and control

6.2 System Design

6.2.1 System Modeling

In an irrigation system, each sprinkler has a relatively limited area of fluid influence on the soil. Standard sprinkler installations use sprinklers that reach 75%-100% of the distance to the nearest sprinkler in the grid, to ensure that water can also reach the diagonal region between the elements of the grid installation. Once the fluid is distributed onto the surface of the soil, the fluid will flow through the grass layer due to diffusion or runoff effects, if the system is installed on sloped land. The fluid will flow until it is absorbed by the soil, after which it will move very slowly, on the order of $10^{-2} - 10^{-3}$ cm/s [Bea72]. Due to fluid's very slow movement through the space, the global modeling proposed in Chapter 5 is computationally very wasteful, as it tracks each sprinkler's effect on the soil moisture at *every* location throughout the space. In a very large system, a sprinkler will have no effect on soil moisture at a *vast* majority of locations, so modeling these interactions are very wasteful and causes optimization to take more time. To improve on this technique and allow distributed operation of the system, each device in the space instead maintains information of only its

surrounding neighborhood of embedded devices. In practice this neighborhood includes all devices within radio range; as radio range can be on the order of 100m and sprinklers are at most 30m apart, each node will maintain information about its second or even third neighbor, with a neighborhood of tens of devices. However, in this work we posit that even a minimum neighborhood including only the nearest neighbors will provide sufficient information for the distributed system to have equivalent performance to the centralized system using global modeling, while making the system truly scalable and improving robustness of operation. To test this hypothesis, the minimum neighborhood containing only direct neighbors as shown as the square box in Figure 6.2 is used in both our live deployment and evaluation in simulation.

To train the fluid movement model later used to optimize schedules, we collect a training dataset in which each sprinkler is actuated one at a time. As the sprinkler runs, all devices in the neighborhood record their sensor data along with the neighborhood actuations. This data is then shared across all N_K devices in the neighborhood and used to train a linear regressor that takes as input current moisture levels at all N_K locations as vector $\mathbf{s}_t$ and current sprinkler actuations at all N_K locations as vector $\mathbf{f}_t$, with values as defined in Table 6.1. This regressor then provides as output the predicted moisture levels at all N_K locations at Δt seconds in the future as vector $\mathbf{s}_{t+\Delta t}$, where Δt is the control timestep, 60 seconds in our system. The linear function $\mathbf{g}$ is defined as follows:

$$\mathbf{g}(\mathbf{s}_t, \mathbf{f}_t) = \mathbf{s}_{t+\Delta t} \tag{6.2}$$

The model, trained on local historical data within the node's local neighborhood, can then be used to predict future moisture changes as a function of sprinkler actuation.

Variable	Description
t	Temporal index $\in \{0, \ldots, T\}$
N_K	Number of nodes in local neighborhood
$\mathbf{s}_t$	Vector of moisture levels at time t, size N_K
$\mathbf{f}_t$	Vector of sprinkler actuation at time t, size N_K

Table 6.1: Model Variables

Variable	Description
t	Temporal index $\in \{0, \ldots, T\}$
k	Sprinkler location index $\in \{1, \ldots, N_K\}$
$\mathbf{s}_t$	Vector of moisture levels at time t, size N_K
$\mathbf{f}_t$	Vector of binary sprinkler actuation at time t, size N_K
$f_{k,t}$	Sprinkler k actuation at time t, $\in \{0, 1\}$
$s_{k,t}$	Volumetric water content (VWC) of location k at time t
c_k	Flow rate of sprinkler k (Constant, known beforehand)
θ_k	Measured VWC of sensor k (Constant, known beforehand)

Table 6.2: Optimization Variables

6.2.2 Optimization Over the Schedule

Each day during irrigation time, each active irrigation device in the system will use its locally-gathered information to find the best sprinkler scheduling to satisfy the needs of the space; in using local models only in a distributed fashion, we are solving relatively small optimization problems in parallel making the system truly scalable. Using our model of neighborhood fluid movement during irrigation as described in Section 6.2.1 and the goal state of moisture defined in Section 6.1, we set up an optimization problem that will minimize water consumption in the system subject to quality of service constraints. We define the binary sprinkler actuation as $f_{k,t}$ for each sprinkler in local neighborhood $k \in \{1, \ldots, N_K\}$ and time index $t \in \{0, \ldots, T\}$, soil moisture values as $s_{k,t}$ for each sprinkler in local neighborhood $k \in \{1, \ldots, N_K\}$ and time index $t \in \{0, \ldots, T\}$, and other optimization variables as shown in Table 6.2. The optimization problem is defined as follows:

$$\min_{\{f_{k,t}, s_{k,t}\}_{k=1,t=0}^{N_K,T}} \sum_{k=1}^{N_K} \sum_{t=0}^{T} c_k f_{k,t} \quad \text{s.t.} \tag{6.3a}$$

$$0 \leq f_{k,t} \leq 1 \qquad k = 1, \ldots, N_K \quad t = 0, \ldots, T \tag{6.3b}$$

$$s_{k,t} \geq \theta_{\text{pwp}} \qquad k = 1, \ldots, N_K \quad t = 0, \ldots, T-1 \tag{6.3c}$$

$$s_{k,T} \geq \theta_{\text{goal},k} \qquad k = 1, \ldots, N_K \tag{6.3d}$$

$$s_{k,t=0} = \theta_k \qquad k = 1, \ldots, N_K \tag{6.3e}$$

$$\mathbf{s}_t = \mathbf{g}(\mathbf{s}_{t-1}, \mathbf{f}_{t-1}) \qquad t = 1, \ldots, T \tag{6.3f}$$

As we wish to minimize the amount of water consumed by the irrigation system, the objective function is defined by the sum of all $f_{k,t}$, weighted by the water flow rate of each sprinkler c_k, which represents the total water consumed under

schedule $\mathbf{F}$. The quality of service is constrained in two discrete time horizons; fluid levels *during* irrigation are constrained to remain above satisfactory levels, but fluid levels are held in sufficient levels *between* irrigation cycles by constraining that each device reach its *Goal State* of moisture by the end of irrigation, which offsets for predicted losses to maintain adequate soil moisture at all times. The soil moisture state at time $t = 0$ is defined to be the most recently measured state of moisture by the WISDOM system sensors in the neighborhood, defined as θ_k, and future moisture levels will evolve with respect to sprinkler actuations as defined by our linear function trained in Section 6.2.1. As the sprinkler actuation values $f_{k,t}$ are binary in the physical irrigation system, the defined problem is an Integer Linear Program (ILP), making the problem NP-Complete. Solving this original problem can take as long as several minutes, which can interfere with our control cycle, as we may want to optimize schedules as often as our control timestep, defined to be 1 minute. To instead find an approximate solution, we allow the $f_{k,t}$ values to be real within [0,1] and then round the resulting actuations to be binary, a simplification that has been found in previous chapters to have minimal effect on the resulting schedules while making the optimization a Linear Program (LP), which is much easier to solve.

As written, this LP has $2N_K(T + 1)$ variables and $4N_K(T + 1)$ constraints. Although solving this optimization results in schedules for each device within our local neighborhood, the device will only extract the schedule optimized for itself to be actuated by the attached solenoid. By considering the initial conditions and solving for optimal scheduling on our neighbors, we ensure our local schedules take into account the likely actions of our neighbors to prepare for or take advantage of any shared conditions such as sprinkler overlap or runoff effects that affect neighbors. We solve this problem using the Julia programming language [BEK14] to interface with the GNU Linear Programming Kit (GLPK) [glp] solver, and

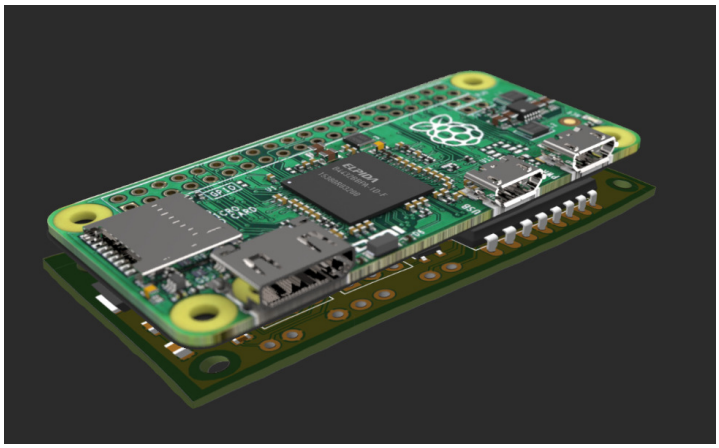


Figure 6.3: Prototype WISDOM device

later discuss the performance and scalability in Section 6.5.

6.2.3 Node design

In order to handle the computation required of the modeling and optimization of Sections 6.2.1 and 6.2.2, we need significant improvements in system performance. The simplest LP solvers such as Simplex [NM65] have a a storage complexity for the problem matrix that is O(mn), where m is the number of constraints and n is the number of optimization variables. Assuming a small neighborhood $N_K = 9$ and 2 hours of irrigation $T = 120$, even our small LP introduced in Section 6.2.2 requires a matrix with 9M entries to store the problem, not feasible on embedded devices with 32-48KB of system memory, let alone space to perform matrix operations. Training our linear model from Section 6.2.1 using ordinary least squares requires the entire dataset be able to fit into system memory, and although gradient descent has significantly less memory requirements, the data points must still be stored and retrieved, and both techniques require significant processing power. In addition to system memory limitations, the standard embedded devices run

at relatively slow processor speeds on the order of 48MHz and typically do not have a hardware floating point unit, requiring floating point to be approximated using integer math which is much slower. Finally, these devices have very little support for standard libraries, likely requiring us to port standard libraries ourselves. Rather than trying to reinvent the wheel, we decide to export these tasks to a separate co-processor that is capable of handling these tasks with ease.

To make this happen, we design our own system management chip to meet the requirements of our application, as shown in Figure 6.3. The low-power devices designed for use in embedded systems are unbeatable in their very low power consumption for long system lifetime on a limited energy budget. For this reason, we use the SAM R21 [sam] SoC, which combines the powerful cortex m0+ microcontroller with the ATRF233 radio, as our system's low-power master device. While insufficient to perform our heavy computation, its 802.15.4 wireless capabilities and basic computational performance makes it perfect for performing regular background tasks such as data syncing and peripheral management while maintaining a low power profile. Its 256KB of onboard flash storage allows it to buffer data in the short-term until it can be transferred to the co-processor for processing. The master device is wired to our latching solenoid via an H-Bridge, which allows us to send of pulse with positive and negative voltage to open or close the valve according to any schedule we choose. The master device is also wired directly to an attached Decagon EC-5 sensor [dec], allowing the device to monitor the local soil moisture conditions. Finally, the device is equipped with a header that allows it to be directly connected to the co-processor, a Raspberry Pi Zero [piz], a $5 device that provides the flexibility of storage and computation that far exceeds the low-power master device. The Zero has a single core 1GHz Arm processor, 512MB of system memory, and microsd external storage. Furthermore, the Zero runs a flavor of Debian Linux with significant support for languages,

platforms, and standard tools. The co-processor and master device are connected via a bi-directional UART communication channel, allowing data and commands to be sent between the two devices as required. Finally, the master device has direct control over the power source to the co-processor, allowing the master device to completely disable the co-processor's power to prevent unnecessary power draw when the co-processor is not required.

Use of our co-processor significantly increases peak power consumption. To put its requirements into perspective, analysis later performed in Section 6.2.5 finds that our co-processor consumes an average of 165mA power to complete its daily tasks, a draw about 12× higher than our master device with the radio in a transmitting state at full power (13.8mA), its most energy-hungry activity. To ameliorate this increased power draw, we recognize that as each device is collocated with a periodically-active water source, there is substantial fluid energy that can be harvested. In researching potential harvesting equipment, we found that small water turbines that are capable of providing a steady 5 volt output at up to 2 amps are available off the shelf. Furthermore, we found that charge circuits such as the MCP73831[cha] are readily available that can charge a LiPo battery at a rate up to 500mA. With the use of a 1Ah LiPo battery, this means we can fully charge the battery from empty with 2 hours of irrigation. While 2 hours of irrigation is not typical in a single day of irrigation, we will later show in Section 6.2.5 that by being cautious with our energy budget, this harvested energy is more than sufficient to keep our system operating indefinitely.

6.2.4 Daily system operation

Each day, the low-power master devices in each WISDOM node will wake up at a time scheduled by the installer or system administrator, likely to be either early

morning or evening. Once awake, the following actions are performed:

(1) At wake-time, each master device goes into a low-power listening state [PHC04] while the neighborhood wakes up. During this time, the basestation will use the DRIP [TC05] dissemination protocol to distribute the single value that requires internet from a centralized source, future weather prediction. This value is disseminated during network wake-up, giving it plenty of time to cross the largest networks without disrupting the processing pipeline.

(2) Once a device receives a status ping from all listed neighbors indicating the neighborhood is awake, it will turn the radio into an all-on state and broadcast all data single-hop from the previous day to their neighbors. In its current configuration, the WISDOM system samples its sensor at a maximum granularity of 1 minute during irrigation, as this data is used to perform re-training of the data-driven models, and reduces the sampling rate to 10 minutes for the rest of the day. Considering that the maximum irrigation period is set as 3 hours, the maximum period allowed with our campus water pressure requirements and regulation, each device will have at most 306 data samples stored as 16-bit unsigned integers, or 612 bytes each day. We additionally package the node's locally-actuated schedules for model training; 3 hours of irrigation has at most 180 binary actuations at a minute granularity, which can be encoded 8-per-byte in just 23 bytes. With a radio packet payload of 100 bytes, supported by our hardware and software platforms, this daily data is sent to all neighbors in at most 7 data packets.

(3) Once all data is received by the master device on each node, it will be buffered in onboard flash storage. The master device will sample the most current long-term model and use the weather forecast we have received to compute the *Goal State* of moisture at its location, the amount of moisture we must reach by the end of irrigation to maintain adequate moisture across the full 24-hour cycle.

If the most recent moisture level measured by the sensor tells us that our current moisture is already higher than our *Goal State* plus a safety margin that we define to be 5% volumetric water content to account for any water losses that may occur *during* irrigation, then the master device decides that irrigation is not required today, and will put the device back into sleep mode, to wake up on the following day. The ability for the master device to terminate the processing pipeline is crucial for long system lifetime during the winter months, as there will be no energy harvesting to recuperate the energy losses incurred by the co-processor.

(4) If the master device decides that irrigation is required, the co-processor will be powered on by flipping a transistor activating the co-processor's power source. Once the co-processor boots, a wired serial communication channel allows the master device to transfer all buffered soil moisture data and actuations from the neighborhood and weather forecasts to the co-processor. Automatic processes running on the co-processor receive these new data streams to re-build the long-term and short-term models as discussed in Section 6.2.1, and then the models and current data state are used to optimize schedules for all nodes in the neighborhood. In this way, we will use all knowledge from our local neighborhood to find optimal valve scheduling for *our* device, while respecting the likely actions of our neighbors as well. The optimized valve schedule for our device is then communicated back to the master device along with the updated long-term model for use in Step 3 above, and the schedule is applied using the attached solenoid. If further irrigation is required today, the co-processor will remain active to perform any additional optimization, as later discussed in Section 6.2.5.

To use as little energy consumption as possible, our co-processor has a discrete set of tasks to perform to minimize on-time. These functions are enumerated in Figure 6.4 along with their effect on the co-processor's power consumpion. This

current data is measured by tracking the voltage drop across a high accuracy shunt resistor with an oscilloscope, and post-processing with a moving average filter to reduce signal noise. Once power is provided by the master device, the co-processor boots its "Raspbian" [rasb] operating system as shown in the figure as (a). Once the operating system finishes its boot cycle, a local Julia HTTP server is launched, which will handle all requests for schedule optimization until the co-processor is powered down. The Julia initialization period is shown in the figure at (b), taking approximately 25 seconds. During this initialization time, all buffered data is transferred from the master device to the co-processor via a wired connection and used to retrain the data driven models, operations that are complete in seconds and do not seem to impact power consumption considerably.

Once Julia initialization is complete, a python process is launched that gathers the freshest models and soil moisture readings from the neighborhood and connects to the Julia server to initialize schedule optimization. Figure 6.4 shows that schedule optimization, enumerated with (c), takes just over a minute to complete. When finished, the optimized schedules are transfered back to the master device to be actuated by the attached solenoid. At point (d) when the optimization has completed, the co-processor will do one of two things; if the optimized schedules show that no further irrigation is required on this day, the co-processor will perform a graceful shutdown and the master device will cut off its power entirely to return to low-power operation, and schedule wakeup for the following day. If further irrigation is required and optimization is to be repeated, the co-processor will return to an idle power level as shown in the figure, and the Julia server will remain running to be re-used, amortizing its startup costs. In our experience, although the initialization of the Julia server and first optimization take on the order of 100 seconds, subsequent optimization problems are solved in a fraction of a second. The power profile of subsequent optimization can be seen

as a small blip in energy at (e) in the figure, 60 seconds after the first problem is solved.

If the optimized schedule activates the attached sprinkler, the water turbine will immediately begin charging the LiPo battery to recuperate the energy consumed in steps (a)-(d), along with all communication, sensing, and actuation costs incurred by the master device since the last battery recharge. As it is difficult to monitor energy harvested by a turbine in our live system deployment, we perform a table-top test of the turbine's power output, attached to a standard hose spigot providing approximately 40psi of water pressure. The power output of the turbine, nearly 700mA at 5v, is then plotted as the green dashed line in Figure 6.4. As our battery charge circuit [cha] features a maximum charge rate of 500mA, the thin blue line in the figure shows the effective system charge rate. Although any power harvested above this rate is not used, the charge rate far surpasses the consumption of our device, even at peak power load. In a long irrigation cycle, the power consumption across repeated optimizations will be similar to that shown between points (d) and (e) consuming around 100mA for the majority of on-time, allowing a rapid battery recharge during irrigation.

6.2.5 Co-processor energy profile

In terms of energy, the worst-case scenario is for soil moisture values at irrigation time to be just above the *goal state* so that optimization will still occur to be cautious, but the optimized schedules do not require the attached sprinkler to be activated. This is possible if the small amount of water required can be received by the nearest sprinklers, or if the learned daily losses are small enough, such that no irrigation is required to stay above the goal state. In this case, the energy costs to start the co-processor and optimize schedules will not be recovered as water

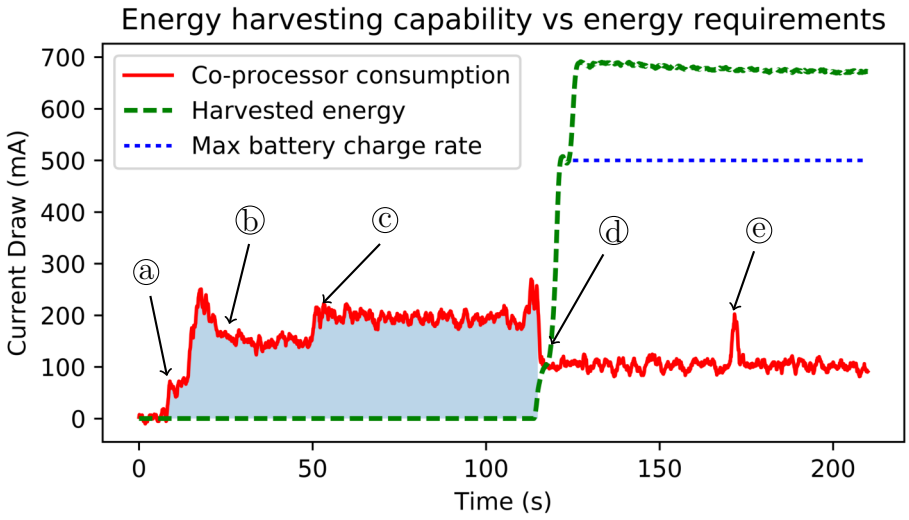


Figure 6.4: Node energy profile across (a) boot, (b) Julia initialization, (c) first optimization solve, (d) completion and return to idle, and (e) subsequent optimization

will not be flowing through the turbine. The energy that would be unrecovered in this worst-case is shaded in blue in Figure 6.4, which we compute to be an average current draw of 165mA at 5v for 110 seconds, a total power draw of 25.2mWh per day. As our 1Ah, 3.7v LiPo battery has a capacity of 3700mWh, our system can handle this worst-case scenario *each day* for nearly 5 straight months before the battery will be drained. In practice, this occurrence would be next to impossible, as even a very minor change in weather (temperature, humidity, or precipitation) can slightly raise soil moisture adequately such that the master device can safely skip co-processor activation for optimization, or slightly lower soil moisture such that irrigation is required, allowing the battery to be recharged. To guarantee energy neutrality, WISDOM is configured to run a short irrigation cycle (10 minutes) to force a battery recharge if the battery is detected to be critically low. This would come at the expense of some water

waste, but such a condition is expected to never occur.

6.3 Case Studies

6.3.1 Live Deployment

To test the WISDOM control strategy against existing systems, we launch two side-by-side irrigation systems on our campus, covering a total of just under 10,000 ft^2, purpose-built for irrigation system analysis. As closely identical as possible, the two irrigation systems each cover an area of 60' by 60' along a gently-sloping hillside that drops approximately 3 feet from the highest to lowest end, and are each installed with a 3x3 grid of Hunter Pro-Spray [hund] sprinklers, providing irrigation coverage to the area. By constructing these systems to be as identical as possible, it allows us to compare two irrigation control strategies simultaneously, such that they are both experiencing weather conditions, runoff effects, sprinkler coverage, and soil conditions as similar as possible to allow a true side-by-side comparison of their operation. In this way, we compare the WISDOM system to the state-of-the-art evapotranspiration control strategy for 2 weeks, and to the OPTICS control strategy for 2 weeks.

As two systems run side-by-side, we monitor the soil moisture levels across the space, as well as the schedules that are being run by the actuators. Using this information, we compute and compare several metrics that indicate how well the two systems are performing relative to one another. The first primary metric is the quality of service provided to the plant, as indicated by the amount of time the sensed locations spend in unhealthy levels of soil moisture. The second is the amount of water consumed by the irrigation system, which can be easily computed by considering the sprinkler flow rates as noted by the manufacturer and the

amount of on-time each sprinkler spends, as indicated by the schedules that are run. Finally, a secondary metric we consider is the spatial uniformity of moisture across the irrigation system; a highly uniform moisture coverage will prevent the occurrence of "hotspots", where local areas of high or low moisture can cause discoloration of the plant or the growth of fungus in the soil. By minimizing water consumption, minimizing exposure to inadequate moisture levels in the soil, and maximizing the spatial uniformity of soil moisture, we will be providing the best irrigation possible.

6.3.2 Simulation

Although it is best to compare irrigation control strategies side-by-side in a live deployment, our test deployment gives no indication of how well a control strategy will perform on edge-case systems, such as those located on steep hillsides, with unusual sprinkler system architectures, or at very large scale. Due to the configuration of our local models to maintain information about the nearest devices, any environment or irrigation system architecture in which water is able to move *beyond* the reach of this local neighborhood will be unknown to our system, making these edge-case systems more challenging for the WISDOM system to control. In such systems, WISDOM may find schedules that are less efficient as those produced using *all* information contained in a global model, such as in the OPTICS system. Specifically, to confirm that the localized techniques of the WISDOM system do not break down due to these effects, we wish to compare WISDOM to the globally-modeled OPTICS system in conditions with challenging soil conditions, steep surface slope, and sprinkler installations with unusual amounts of overlap, as these have the potential of causing water to move beyond a device's local neighborhood in the WISDOM system.

To allow us the flexibility to test the effect of these changing environmental and architectural features in an irrigated space, we develop an hybrid water movement model using the nonlinear PDE model developed in Chapter 4 to model water movement during irrigation actuation, and the long-term loss models developed in Chapter 5 to model fluid loss between irrigation cycles. In doing so, we can track water movement during irrigation subject to surface runoff, sprinkler overlap, diffusion, and infiltration into the soil at any desired level of spatial and temporal resolution. Then, once water enters the soil where it moves very slowly, we model fluid flow as spatially-independent loss models, represented by exponential decay functions. These loss functions are randomly generated at each measuring location throughout the space to be similar in magnitude to those experienced in reality, and then random noise is added to losses on each day of simulation, providing losses that are noisy but consistent at each location, similar to what we expect in reality. This hybrid simulator allows us to construct sample irrigation systems from scratch under a wide variety of conditions and closely monitor how the water is moving through it with respect schedules chosen by any control strategy. Not only does a simulator's use allow us to compare WISDOM's operation to another system across many environments, but it also allows us to quantify the benefits of the WISDOM system at significantly larger scale than our test irrigation system allows. A sample simulation of the WISDOM system across a 14-day period can be seen in Figure 6.5, where the control strategy can be seen to react to initially dry soil conditions across the first 4 days to surpass a minimum level of moisture at all locations in the space.

In our side-by-side comparison with OPTICS in simulation, we consider three primary metrics. The first is the water consumption of the irrigation system, as this will be directly related to the optimized models used by the two control strategies, and the second is the quality of service provided, as computed by the

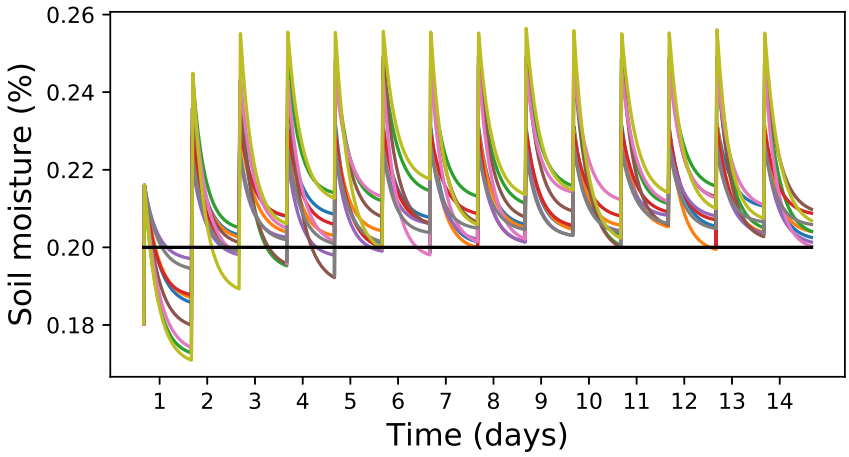


Figure 6.5: Example 14-day simulation of WISDOM control

amount of time our sensed points remain below θ_{pwp}, in unhealthy levels of moisture. If the local models contain sufficient information about the environment, we expect these metrics to be identical between the WISDOM and the OPTICS systems, with any deviation indicating that the local models of WISDOM lack information present in the global model of OPTICS. The third metric of comparison is the time required by the two systems to optimize their schedules as the scale of the irrigation system is increased.

6.4 Experimental Results

To test the operation of the WISDOM system, we perform two side-by-side system deployments for two weeks each against the Evapotranspiration and OPTICS control strategies. Although it is not necessary for the operation of the WISDOM system at scale, to monitor the system we had all experimental devices forward a copy of its operational data (soil moisture and schedules) to a centralized location for visualization and analysis. The soil moisture data collected during the first

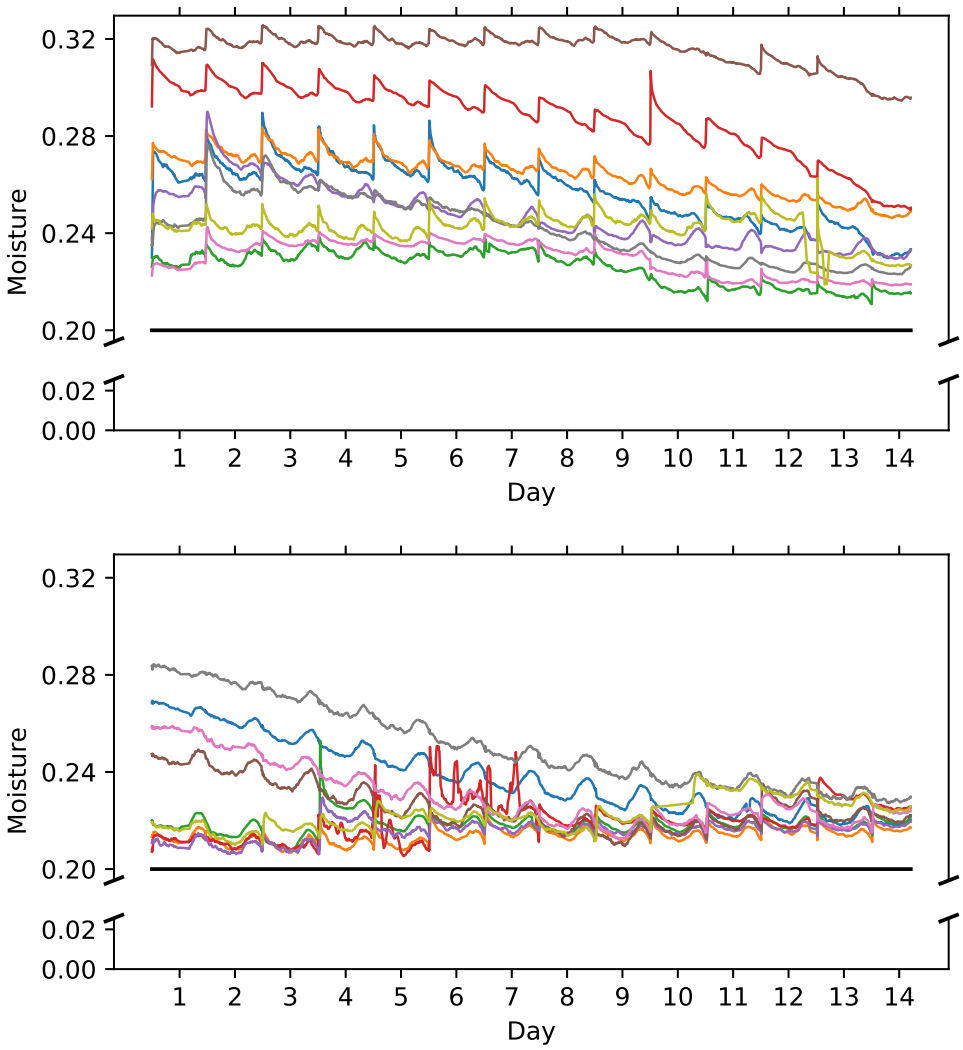


Figure 6.6: Collected VWC data across deployment for all Evapotranspiration (left) and WISDOM (right) nodes

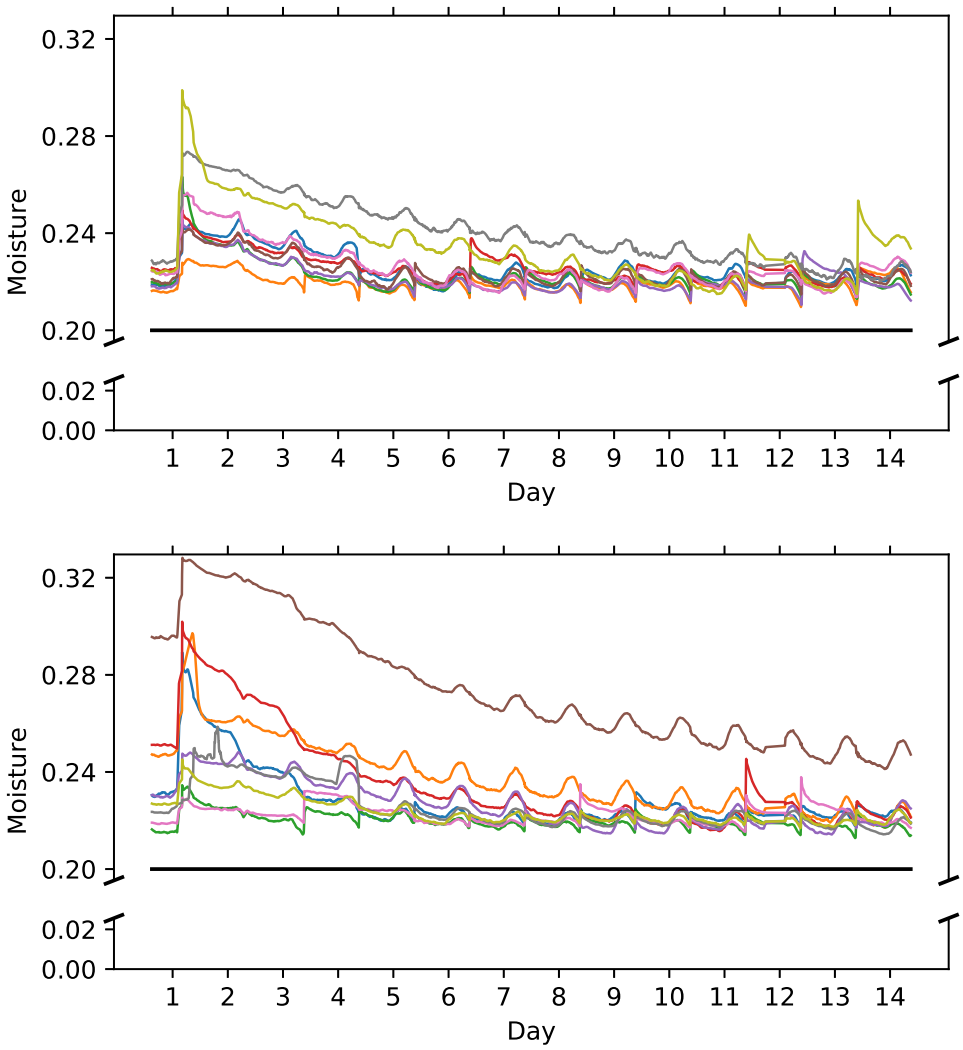


Figure 6.7: Collected VWC data across deployment for all OPTICS (left) and WISDOM (right) nodes

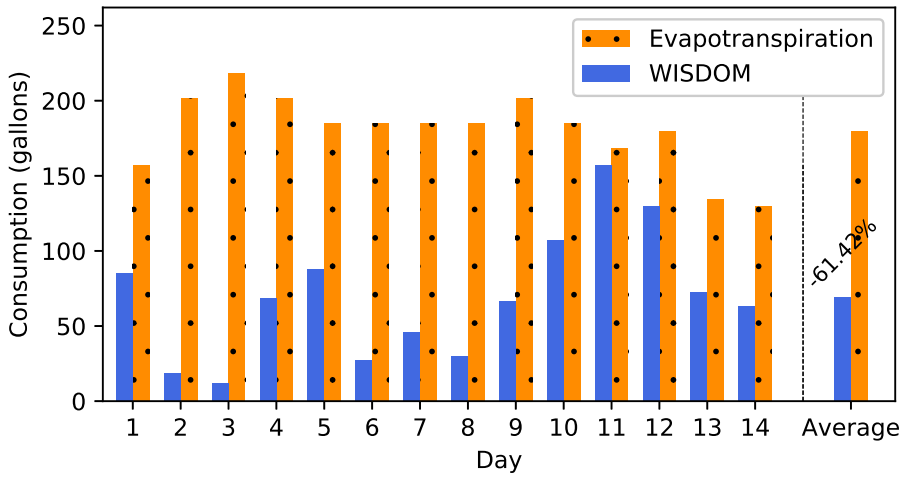


Figure 6.8: Water consumption of ET vs WISDOM

deployment comparing Evapotranspiration and WISDOM systems can be seen in Figure 6.6, and the soil moisture data collected during the second deployment comparing OPTICS and WISDOM systems can be seen in Figure 6.7. Also shown in these figures is the horizontal line indicating the soil moisture level θ_{pwp}, which acts as the lower limit the system should allow at all times. As previous results presented in Chapter 5 saw the data-driven modeling technique occasionally drop below the threshold due to model error, in this system we included a 5% safety band on the value of θ_{pwp}; in other words, although we want to maintain a 20% volumetric water content (VWC) as shown, both data-driven systems are actually trying to maintain a 21% VWC due to this safety band.

Each day on the ET controlled system, we can see that irrigation causes a sharp increase in soil moisture at all measuring locationsas the irrigated fluid reaches the system. On both the WISDOM and OPTICS systems, which are utilizing sensed water distribution to optimize schedules, we can see this spike in moisture levels only on the sensor values that are reaching near our minimum allowed moisture level, as the data-driven systems feel that without irrigation

moisture levels will drop into dangerously low levels. Interestingly, in the WISDOM and OPTICS systems, locations that are well above this level and require no irrigation still see a steady rise and fall in sensor values each day. This is an effect of the metric measured by our EC-5 sensors [dec]; the soil moisture rise is due to increasing soil temperature, which has influence on the dielectric permittivity [CGM12] which is being measured. With more expensive sensors that also measure temperature, these deviations could be corrected for, but as our installed sensors do not have this feature, we experience a daily soil moisture cycle even when no irrigation is applied. Another interesting artifact was the occurrence of rain on day 1 of the second deployment, leading to a great increase in soil moisture levels. It can be seen that one region within the WISDOM system retained a very large amount of water, with water levels staying high for the remainder of the experiment preventing the requirement of irrigation in that area for the entire deployment. Between days 4 and 7 in the first deployment we can see one sensor exhibiting strange behavior. An animal had chewed a sensor cable, and on these days irrigation was entering the damaged area and causing strange sensor values until day 7, when the problem was diagnosed and corrected.

The primary goal of an irrigation system is to maintain healthy levels of moisture everywhere, and previous chapters have defined the quality of service metric as the amount of time the system spends in unhealthy levels of moisture. In our experiments, shown in Figures 6.6 and 6.7, it is clear that all systems tested maintained a healthy level of moisture (above our minimum level θ_{pwp}) at all times. However, there is a clear tradeoff between quality of service provided and system efficiency. While the evapotranspiration control strategy was able to maintain healthy levels at all time, partially due to the very high initial moisture conditions shown on day 1, we can see in Figure 6.8 that the ET system consumed a very high amount of water in comparison to the WISDOM system for the entire

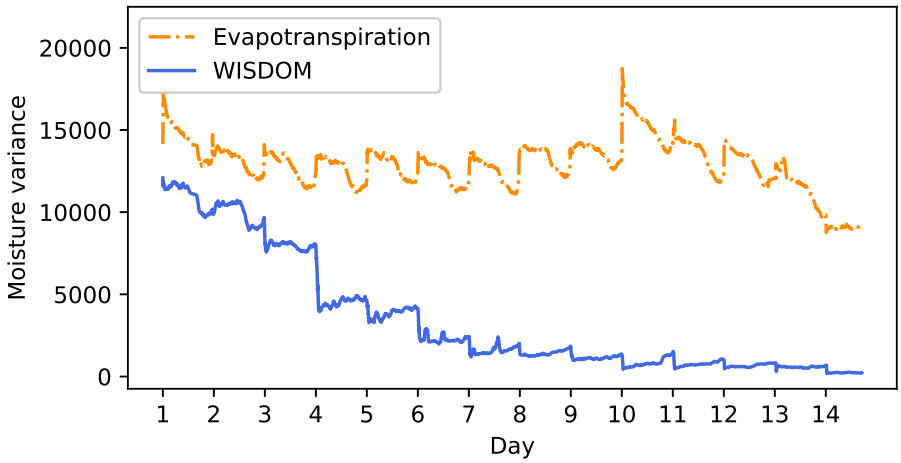


Figure 6.9: Moving variance of ET vs WISDOM

experiment. Whereas the ET system responds only to recent weather conditions, the WISDOM system also has knowledge of the changing soil moisture levels, irrigating only when more water is required. For this reason, we can see over the data in Figure 6.6 that the WISDOM system is able to avoid irrigating areas with sufficient moisture already, saving an average of 61.4% of water. Looking at the last 4 days of the first deployment, as all of the WISDOM devices enter the moisture region where irrigation is required, we find that the system is saving an average of 32.9%, which is more indicative of the potential savings of WISDOM in steady state.

As shown in Figure 6.7, the WISDOM system has considerably higher soil moisture values after the rain on the first day, preventing one device from requiring irrigation for the entire deployment. Figure 6.10 shows that in the first several days the OPTICS system consumes considerably more water than WISDOM, with WISDOM saving an average of 46% of water across the two weeks of deployment. However, we can see that as the deployment progresses and WISDOM nodes drop into the range requiring irrigation, the consumption of the two

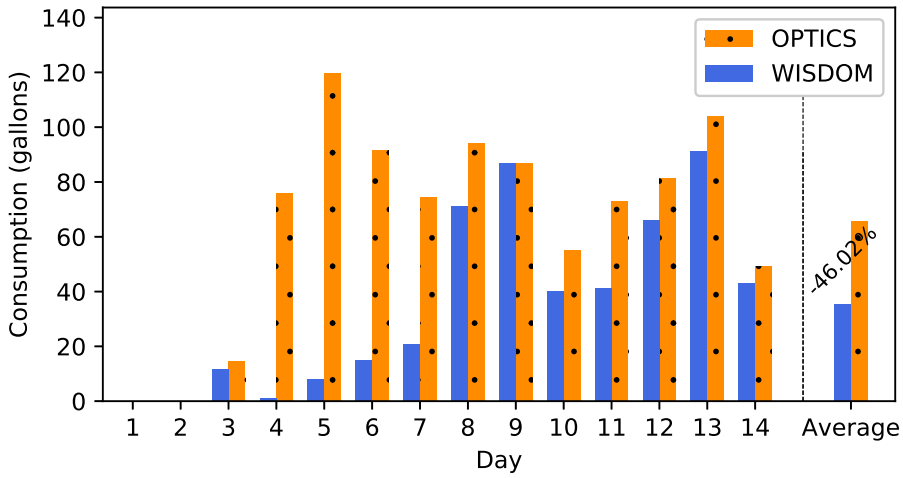


Figure 6.10: Water consumption of OPTICS vs WISDOM

systems become more and more similar, until the last 7 days of deployment find an average savings of 19.7%. Although the WISDOM systems found savings in this deployment, we are confident that this is a result of the initial conditions of the experiment, with WISDOM nodes having higher initial moisture levels, and expect that a longer deployment would find these two systems approaching equivalent performance in efficiency once all the devices fall into a steady state. Their equivalence is expected as they are both maintaining data-driven modeling techniques, and the sprinkler system architecture and environment do not facilitate enough water movement to make the local models a limiting factor, as later verified in simulation in Section 6.5.

Finally, as a metric of moisture uniformity, we plot the spatial variance of moisture in all systems as a function of time in Figures 6.9 and 6.11. Even though the ET controller is applying water uniformly onto the surface, differences in environment cause the water to settle in a non-uniform way, as seen by a large spread of moisture levels in Figure 6.6 and high levels of variance in Figure 6.9. In comparison, as the data-driven OPTICS and WISDOM systems learn the needs

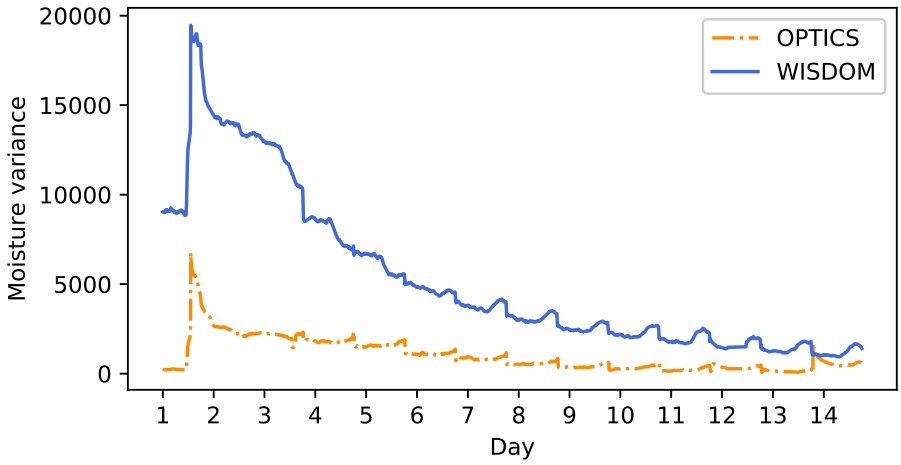


Figure 6.11: Moving variance of OPTICS vs WISDOM

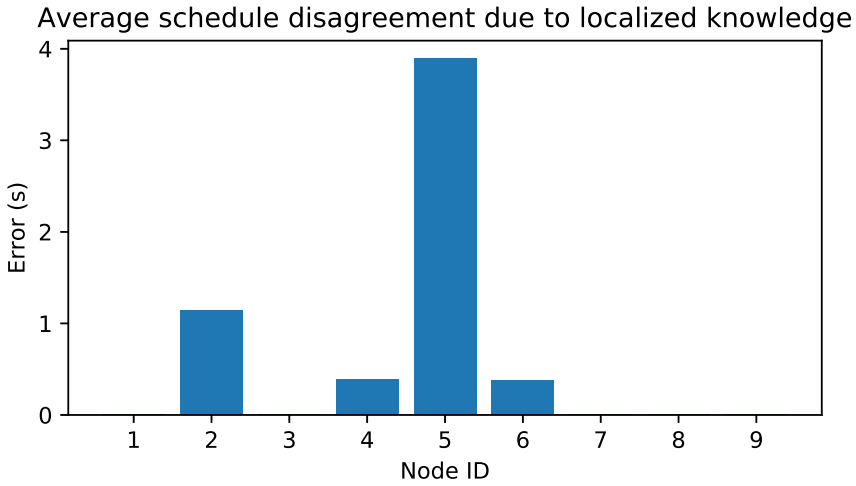


Figure 6.12: Average schedule error caused by local models

of the space over time, the variance can be seen to quickly decrease to very low values across the system deployment, indicating that the system is learning the needs of the space and distributing water exactly as required.

6.4.1 Local model disagreement

When local optimization occurs on each device, it computes the optimal valve scheduling for itself *and* all other devices within its local neighborhood. On the boundary of the local neighborhood, however, the local moisture movement information may be missing boundary effects that will influence moisture levels at nodes on the edge of our neighborhood, resulting in a disagreement between the edge node's schedule we've computed and the schedule that edge node computed for itself. To determine the magnitude of this error due to edge effects, we compare the optimized schedule of each node *for itself* to the optimized schedule by the *node's neighbors* for the node, and collect the average error made by the neighbors. If the error is very high, we will know that schedules are highly dependent on the influence of neighbors' irrigation in our test irrigation system, as this could indicate that localized models will introduce significant error in the schedules computed locally. On each day, we compare the on-time of the optimized schedule as computed on a node to those computed by the node's neighbors, and report the average error in sprinkler on-time for each node's neighbors in Figure 6.12. In this plot, the bar for node 5, for instance, is the average error in sprinkler on-time of all of node 5's neighbors in estimating node 5's schedule.

We see in the figure that at worst, the average schedule deviation is less than 5 seconds, indicating that the localized models make very good estimates of neighboring actions. Interestingly, the resulting errors agree with intuition. The neighborhoods of nodes 2, 5, and 4 as shown in Figure 6.13 will contain the

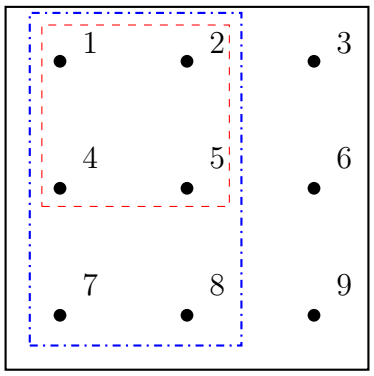


Figure 6.13: Scope of local models within system topology

entire neighborhood of node 1 shown as the red dotted box, so their predictions will have zero error, and similar for all corner nodes 3, 7. and 9. Neighbors of edge nodes 2, 4, 6, and 8 will be missing at most two nodes; for instance nodes 1 and 2 will be missing node 4's neighbors 7 and 8, so these edge nodes will have some error. Central node 5 should have the most error; if node 1 estimates node 5's schedule, it is missing nodes 3, 6, 9, 7, and 8, resulting in high error. These expectations match the results of Figure 6.12 with the exception of node 8, which shows zero error in its neighbors' predictions. The cause for this is the fact that node 8 is the node shown in Figure 6.7 that remains well above irrigation levels. For this reason, regardless of neighbor influence, the optimal schedule is to disable the sprinkler, which all neighbors correctly predict.

6.5 Results in simulation

To challenge the local models used in the WISDOM system, we use our simulator introduced in Section 6.3.2 in 4 environmental and architectural scenarios. Each experiment tracked water movement across a space of 65m $\times$ 65m, just over 1 acre, irrigated by a 7 $\times$ 7 grid of sprinklers, and run for a continuous period of 100

days. The first was a typical environment across flat land with a typical "loam" soil commonly found in standard irrigated spaces, with each sprinkler radius 75% of the inter-sprinkler distance. The second introduces a soil with a higher clay content, which reduces the water infiltration rate into the soil, increasing the amount of fluid diffusion that occurs on the surface. The third uses the same soil type as the second experiment, but sits on a hillside with a 15° angle, which will lead to a higher amount of runoff, particularly with the lower infiltration rate of the clay soil. A 15° slope was chosen, as it is the maximum slope supported by most ride-on lawnmowers, making it the steepest slope likely on a large expanse of lawn installation. Finally, to ensure an example where the fluid influence of a sprinkler reachs beyond the neighborhood of the WISDOM local models, we configure a system architecture similar to the first experiment, but with each sprinkler reaching 2.5× the inter-sprinkler distance, which can be visualized as the arching circle centered on the leftmost node in Figure 6.2. We *strongly* note that standard irrigation system installation practices use sprinklers that reach no more than 1× the inter-sprinkler distance, and that this is purely an exercise to find the breaking point of the localized modeling techniques; such a system would be prohibitively costly, and would likely consume significantly more water than the source could provide. The results of these four independent experiments are shown in Figure 6.14, with the positive y axis showing the over-all system consumption across the period, and the negative y axis showing the total amount of time spent below our desired moisture threshold at our sensing points in the space.

We find that across the 100 days of simulation for the first experiment (bar 1 in the figure), both control strategies consume 520,000 gallons of water, spending a total of 103,000 minutes beneath the desired level of moisture. Averaged across the 49 nodes in the system, each node spends 35 hours beneath this level, but

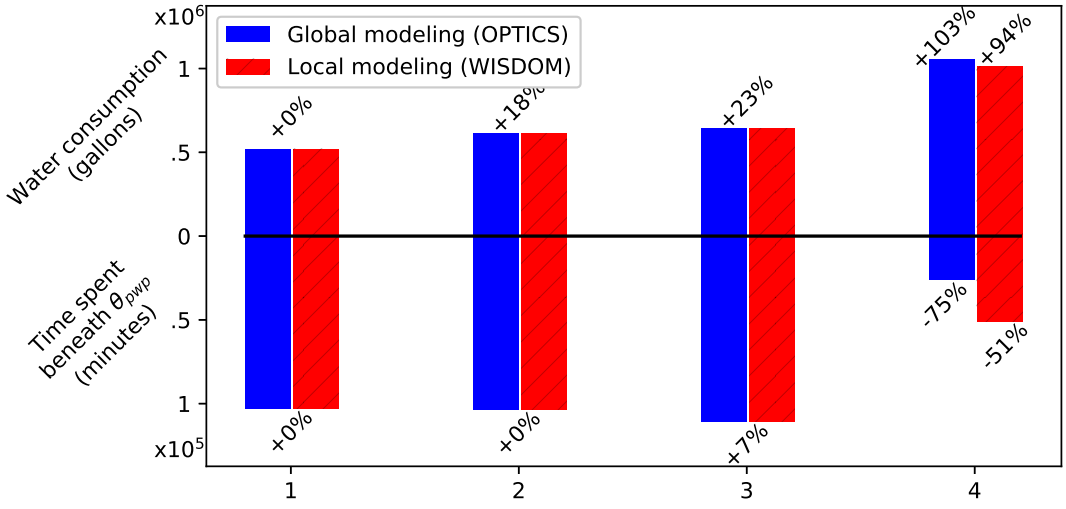


Figure 6.14: Resulting system consumption and quality of service penalty for 4 environmental setups: (1) Flat land (2) Flat land with clay soil (3) 15° slope with clay soil (4) Installed sprinklers reaching more than twice the inter-sprinkler distance

we can see in Figure 6.5 that this time stems from the first days after system installation, when each node is learning its loss models. In the second experiment (bar 2), we find that the installation of soil with a higher clay content increases the system consumption by 18% across the experiment. This increased consumption is caused by a higher loss to the boundaries of the space, as the fluid will sit on the surface for a longer period where it has the tendency to leave the space more quickly than if it were moving through the soil. The third experiment (bar 3) found an increased water consumption of 23% in comparison to experiment 1, and 7% increase in exposure to unhealthy levels of moisture. The additional consumption is intuitive, as the slope will cause an additional amount of water to flow out of the space on the boundaries over time, exacerbated by the clay soil preventing infiltration of the fluid into the soil. The increased penalty to quality of service is caused during the first days of experimentation, when the uphill region in the space will lie in unhealthy levels of moisture during the first few days to a greater extent, due to the slope carrying fluid away.

Finally, the fourth experiment shows a difference in result between the OPTICS and WISDOM systems with a sprinkler layout that reaches beyond the neighborhood of the local modeling technique. This sprinkler coverage allows the system to apply a significant amount of water on the first days to substantially reduce the quality of service penalty that would otherwise occur, while this increased coverage effectively doubles the water usage in comparison to the system in experiment 1. However, we see that the schedules optimized with the global model are able to find schedules that are roughly twice as good as the locally-computed ones with respect to the quality of service metric, but also led to an increase in system water consumption of 9%. The takeaway of these simulated results is that the control systems using global and local models tend to be identical due to the limited rate of water movement in these irrigated spaces.

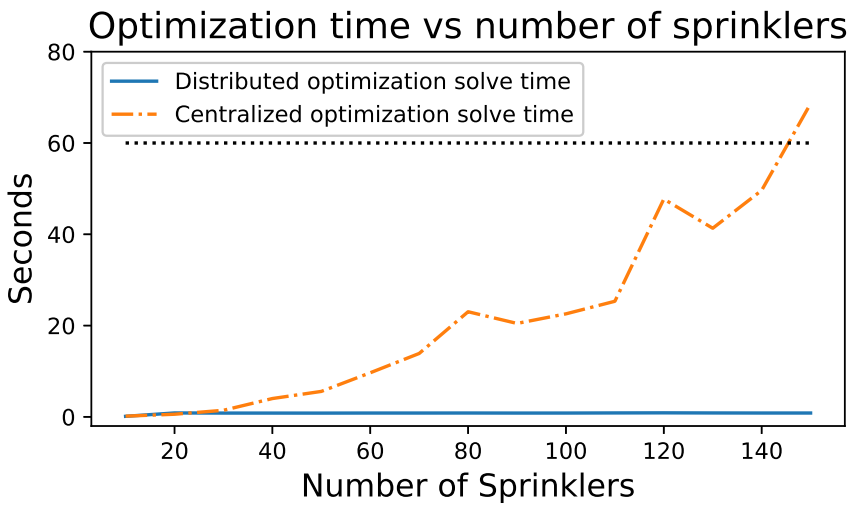


Figure 6.15: Time required to optimize schedules using centralized (OPTICS) and distributed (WISDOM) systems

While it is possible for schedules optimized using local models to find inferior solutions to schedules optimized with global ones, it will not occur in even very challenging scenarios (experiments 1-3), but require unrealistic irrigation system architectures to occur (experiment 4).

In addition, to quantify the performance of the OPTICS and WISDOM systems side-by-side, we use our simulator to train local and global models on irrigation systems of increasing sizes, and optimize schedules from a fixed initial moisture state to a fixed goal moisture state, both interpolated to the size of the irrigated space. The goal is to make the optimization problems as identical as possible, with the only variable being the size of the irrigation system. The resulting optimization time required of the two systems as a function of system size can be seen in Figure 6.15. The WISDOM system is solving K optimization problems, each using a model of a relatively small local neighborhood, in a distributed way. While this makes the computation required of the WISDOM

system linear on K, it is solved in parallel across K devices, making the optimization solve time constant at all system scales. In contrast, the OPTICS system is solving a centralized optimization problem using a single global model that grows quadratically on K, as shown in the figure. We find that with a system with more than 145 sprinklers, the centralized control strategy is unable to solve the optimization within our control timestep. In practice, as irrigation systems found in the largest sports fields can contain anywhere from 500-5000 sprinklers [gol], the scalability of the centralized system is inadequate. While we could increase the control timestep to allow a larger optimization problem to be solved, the quadratic growth of the optimization solve-time will quickly overcome this time cap, and an increased control timestep would cause less efficient schedules to be found by making the temporal granularity of the model more coarse. Regardless, no such sacrifice is required with the distributed system of the WISDOM system, allowing it to operate at any scale.

6.6 Return on investment

To ensure system adoptability, we must be sure it is affordable as a system replacement. We consider the unit cost of each device as shown in Table 6.3, where we list unit prices of our components but estimate manufacture and assembly at the scale of 1000 units based on a quote from PCBWay [pcb]. We note that even with the addition of the co-processor and energy harvesting, removing unnecessary hardware through custom design increases the unit cost by just 1.3% in comparison to the prototype proposed in Chapter 5. On our campus, water for irrigation is billed at a price of $5.60 per 1000 gallons of water. Using the water savings from the last 4 days of our first deployment against the ET control strategy, we extrapolate a 32.9% water savings onto the larger-scale irrigation

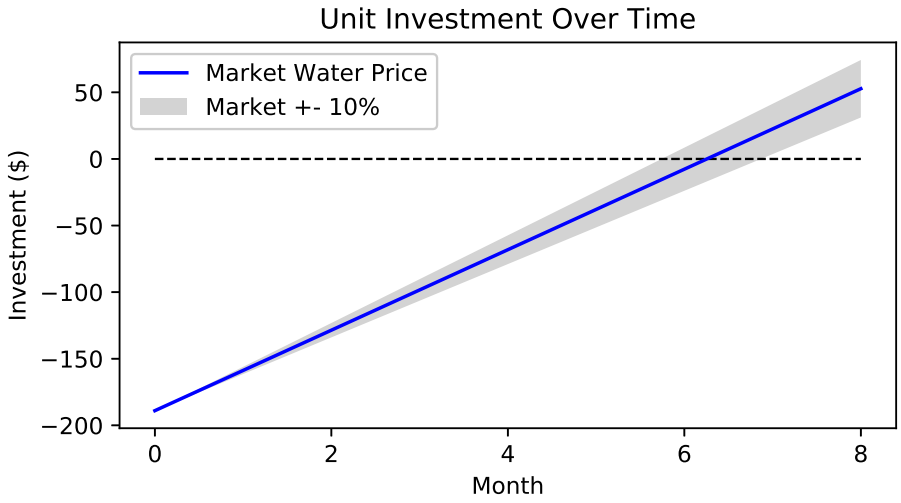


Figure 6.16: Device return on investment over time

systems used to irrigate our sports fields at our university, which are equipped with Hunter I-40 rotor sprinkler heads. Figure 6.16 shows the evolution of the investment over time, and we find that based on water savings alone, the WISDOM devices will pay for themselves in approximately 5-7 months, considering market variation of 10%. We do note, however, that Chapter 5 found that optimized scheduling using data-driven modeling saved a more conservative 12%, due to seasonal and environmental differences in the experiments run. Even in such an environment where these more conservative savings occur, our system is still estimated to return its investment in 15-18 months.

6.7 Limitations and Future Work

The accuracy of the localized modeling techniques used in this section are sensitive to the size of the local neighborhoods. Based on our results in our case study and simulation, neighborhoods do not need to be large in general system

Component	Price
Mote (including manufacture)	$25.00
Raspberry pi zero	$5.00
Battery and charge circuit	$21.00
Harvesting turbine	$12.00
Sealed enclosure	$13.00
Decagon EC-5 sensor	$99.00
Echotech latching solenoid	$14.00
	$189.00

Table 6.3: Sprinkler Node Manufacture Cost

operation. However, it is possible that there exist edge cases where very large neighborhoods are required to capture sufficient information to maintain the same model accuracy of the globalized technique. In these cases, rather than choosing a neighborhood size as large as possible and risking optimization exceeding the available computation time, it may be useful to understand the relationship between model accuracy and neighborhood size, such that an optimal value of the neighborhood size can be selected for the system's operation. In a particularly special case, if a neighborhood size of one (no neighbor data required) results in minimal impact to model accuracy, it would allow the device to make decisions completely independently, removing data communication entirely. We leave the evaluation of these tradeoffs to future work.

In this chapter, we have introduced a method of using locally-collected data to train local models of water movement. One key reason of partitioning the problem is to reduce the time required in optimization using the global model. However, an alternate approach would be to train the models of water movement using all data in a centralized way, but then using the structure of the non-zero

elements in the global linear model to prune it for use in localized optimization within neighborhoods. While the model would need to be trained centrally, the distributed linear program would have the performance improvements of a localized model, and as the model was trained with all data, it would not be sensitive to edge conditions of neighborhood boundaries. We leave a comparison of this third technique to future work.

The distributed techniques introduced in this chapter address limitations of energy and congestion with cross-network communication and global computation. However, in some long and thin irrigation system geometries such as small golf courses, it may be possible that the primary limitation is the physical distance between each device and the central controller. In such conditions, an alternate communication technology such as LoRa [lor] may be more appropriate, as it would enable each device to communicate over vastly larger distances than the devices we have used in our study. We do not believe this would be helpful in every case, as larger communication distances will greatly increase the number of links in the network, exacerbating the issue of network congestion. We also note that this technique will do nothing to help the single point of failure as it will still rely on a centrally-located controller, but we leave the analysis of this technology to future work.

6.8 Conclusions

Although the work of the previous chapters have realized substantial efficiency improvements with the use of distributed actuation and model-based schedule optimization, the proposed centralized processing architectures introduce single points of failure, computational bottlenecks in data processing, and increase network energy usage for cross-network data forwarding, which all prevent the

technologies from being adoptable for control at large scale. In this chapter we introduce WISDOM, a *distributed system* for control of irrigation systems at any scale that utilizes energy harvesting to allow a perpetual system lifetime. Across 4 weeks of live system deployment, we find that up to 32.9% water savings is possible in comparison to industry-standard controllers, and demonstrate through extensive comparison in simulation that the proposed distributed system can provide *all* of the efficiency and quality of service benefits of the centralized one, while allowing energy independence and the robust control of irrigation systems of *any* size.

CHAPTER 7

Conclusions

Across this body of work, the state-of-the-art of lawn irrigation has been moved forward with several key contributions. Through the use of wireless sensor networking, we have developed a system allowing each sprinkler in the space to be actuated independently, enabling location-specific irrigation for substantially improved efficiency and quality of service to the turf. Additionally, through the development of a co-processor architecture, these distributed devices are able to perform *local* computation, avoiding the need for energy-expensive cross-network data transfer and a single point of failure at the centrally-located controller. Recognizing energy budgeting as a primary concern in a distributed system at scale, we introduce energy harvesting to the system as well as processing pipelines that use the computational resources sparingly, guaranteeing energy-neutral operation for a perpetual system lifetime. Due to the increased complexity of an irrigation system with distributed actuation, we propose and implement data-driven modeling strategies of fluid movement through the environment, which are then used as constraints in optimization to find valve schedules that minimize water consumption while maintaining suitable moisture levels across the space at all times. In this way, the system will learn and adapt to changes in the local environment through the sensed moisture conditions across the space and choose the best possible schedules for the distributed actuators, allowing completely autonomous operation with no required human input for installation or adjustment. Finally,

these innovations are tested against competitive baseline strategies in real-life deployments and simulation and are found that the introduced techniques can lead to significant reductions in system water consumption (12-32.9%) allowing a short return on investment of the devices while also *significantly* improving the quality of service provided to the turf. In doing so, we have demonstrated that turf irrigation systems are an ideal candidate for the robust and efficient control of turf irrigation systems at any scale.

APPENDIX A

Appendices

A.1 Grass as Porous Media

Darcy's law [Dar56] is applicable to systems where the drag due to the multiple obstructions is the dominant fluid force. To determine when flow through turf may be modeled as flow through a porous medium, we estimate and compare the drag, viscous force, and inertial force per unit volume. Denoting a typical flow velocity scale as U, a typical grass blade size as a, viscosity as μ, density as ρ, the thickness of the liquid layer as h, and the porosity as ϕ, we estimate the drag per volume, based on the low Reynolds number drag of an elongated obstruction of length h and lateral size a, as

$$D \sim \frac{4\pi\mu U h(1-\phi)}{ha^2} = \frac{4\pi\mu U(1-\phi)}{a^2}. \tag{A.1}$$

Within the liquid, the viscous force per volume is estimated from the viscous term in the Navier-Stokes equations, as $F_v \sim \mu U/h^2$. Finally, the inertial force per volume is estimated, also from the Navier-Stokes equations, as $F_i \sim \rho U^2/h$.

As typical values for water and grass blades, we use $\rho = 1\text{g/cm}^3$, $\mu = 0.01\text{g/cm s}$, $a = 0.1\text{cm}$, $h = 1\text{cm}$, and $\phi = 0.9$, and find $\frac{F_v}{D} \sim \frac{a^2}{4\pi h^2}\frac{1}{1-\phi} \sim 1/100$ showing that the drag is dominant over viscous forces for any velocity of the flow. The ratio of inertial forces to drag is $\frac{F_i}{D} \sim \frac{a^2\rho U}{4\pi\mu h}\frac{1}{1-\phi} \sim U$. The applicability of our model

is therefore limited to systems where the water flows at velocities $U < 1\text{cm/s}$, which corresponds to most intermittent irrigation regimes.

A.2 Discretized Model Formulation

The following 6 equations hold at each spatiotemporal cell (i, j, t) and represent the discretized, linearized flow motion of Equations (4.6)–(4.9) over the variables u and v (velocity of water in soil and surface, respectively, both horizontal components x and y for each), h (surface fluid height) and theta (volumetric moisture content). They also represent the equality constraints in our optimization problem.

$$\begin{aligned}\frac{h_{i,j,t+1}-h_{i,j,t}}{\Delta t} = &-\frac{1}{2\Delta x}\Big((\hat{h}v_0^x + h_0\hat{v}^x)_{i+1,j,t} - (\hat{h}v_0^x + h_0\hat{v}^x)_{i-1,j,t} \\ &+(\hat{h}v_0^y + h_0\hat{v}^y)_{i,j+1,t} - (\hat{h}v_0^y + h_0\hat{v}^y)_{i,j-1,t}\Big) + F_s \\ &-\eta\Big(h_0K(\theta_0) + h_0K'(\theta_0)\hat{\theta} + \hat{h}K(\theta_0) + \hat{h}K'(\theta_0)\hat{\theta}\Big) \qquad \text{(A.2)} \\ \frac{\theta_{i,j,t+1}-\theta_{i,j,t}}{\Delta t} = &-\frac{1}{2\Delta x}\Big((\hat{\theta}u_0^x + \theta_0\hat{u}^x)_{i+1,j,t} - (\hat{\theta}u_0^x + \theta_0\hat{u}^x)_{i-1,j,t} \\ &+(\hat{\theta}u_0^y + \theta_0\hat{u}^y)_{i,j+1,t} - (\hat{\theta}u_0^y + \theta_0\hat{u}^y)_{i,j-1,t}\Big) \\ &+\zeta\Big(h_0K(\theta_0) + h_0K'(\theta_0)\hat{\theta} + \hat{h}K(\theta_0) + \hat{h}K'(\theta_0)\hat{\theta}\Big) \qquad \text{(A.3)}\end{aligned}$$

$$
\begin{aligned}
\hat{u}^x_{i,j,t} = &- \tfrac{K(\theta_{0i,j,t})}{2\Delta x}(\hat{h}_{i+1,j,t} - \hat{h}_{i-1,j,t} + h_{0i+1,j,t} - h_{0i-1,j,t}) \\
&- K'(\theta_{0i,j,t})\hat{\theta}_{i,j,t}(h_{0i+1,j,t} - h_{0i-1,j,t}) + \tfrac{K(\theta_{i,j,t})\vec{\tau}^x}{\rho g} \\
&- \tfrac{\varphi(\theta_{0i,j,t})}{2\Delta x}(\theta_{0i+1,j,t} - \theta_{0i-1,j,t} + \hat{\theta}_{i+1,j,t} - \hat{\theta}_{i-1,j,t}) \\
&- \varphi'(\theta_{0i,j,t})(\theta_{0i+1,j,t} - \theta_{0i-1,j,t})\hat{\theta}_{i,j,t} - u^x_{0\,i,j,t} \qquad \text{(A.4)} \\
\hat{u}^y_{i,j,t} = &- \tfrac{K(\theta_{0i,j,t})}{2\Delta x}(\hat{h}_{i,j+1,t} - \hat{h}_{i,j-1,t} + h_{0i,j+1,t} - h_{0i,j-1,t}) \\
&- K'(\theta_{0i,j,t})\hat{\theta}_{i,j,t}(h_{0i,j+1,t} - h_{0i,j-1,t}) + \tfrac{K(\theta_{i,j,t})\vec{\tau}^y}{\rho g} \\
&- \tfrac{\varphi(\theta_{0i,j,t})}{2\Delta x}(\theta_{0i,j+1,t} - \theta_{0i,j-1,t} + \hat{\theta}_{i,j+1,t} - \hat{\theta}_{i,j-1,t}) \\
&- \varphi'(\theta_{0i,j,t})(\theta_{0i,j+1,t} - \theta_{0i,j-1,t})\hat{\theta}_{i,j,t} - u^y_{0\,i,j,t} \qquad \text{(A.5)} \\
\hat{v}^x_{i,j,t} = &- \alpha_h\Big(\tfrac{h_{i+1,j,t} - h_{i-1,j,t}}{2\Delta x}\Big) + \tfrac{\kappa_g}{\eta}\vec{\tau}^x - v^x_{0\,i,j,t} \qquad \text{(A.6)} \\
\hat{v}^y_{i,j,t} = &- \alpha_h\Big(\tfrac{h_{i,j+1,t} - h_{i,j-1,t}}{2\Delta x}\Big) + \tfrac{\kappa_g}{\eta}\vec{\tau}^y - v^y_{0\,i,j,t} \qquad \text{(A.7)}
\end{aligned}
$$

A.3 Spatial system coverage as a function of number of sprinklers

To evaluate the scalability of our control strategy, it is useful to know how the spatial coverage of the irrigation system will change as we increase the number of sprinklers in the system. Here we consider this relationship on a theoretical irrigation system in a grid of K sprinklers arranged in a square of $k \times k$ sprinklers. Each sprinkler installed has a radius R and will be installed a distance d to the next closest sprinkler in the grid. [Wei03] demonstrates that the area of overlap between two circles of equal radius R with distance d between them as shown in Figure A.1 can be written as:

$$A_{\text{intersection}} = \begin{cases} 2R^2\cos^{-1}\Big(\dfrac{d}{2R}\Big) - \dfrac{d}{2}\sqrt{4R^2 - d^2}, & \text{if } d \leq 2R \\ 0, & \text{if } d > 2R \end{cases} \tag{A.8}$$

Using this identity, we derive irrigation system coverage in the three possible irrigation system configurations to be:

$$\text{coverage} = \begin{cases} K\pi R^2, & \text{if } \dfrac{R}{d} < \dfrac{1}{2} \\ 4\Big((2k-1)\dfrac{\pi R^2}{4} - (k-1)\dfrac{A_{\text{intersection}}}{2}\Big) + (k-1)^2\Big(\pi R^2 - 4\dfrac{A_{\text{intersection}}}{2}\Big), & \text{if } \dfrac{1}{2} \leq \dfrac{R}{d} \leq \dfrac{\sqrt{2}}{2} \\ 4\Big((2k-1)\dfrac{\pi R^2}{4} - (k-1)\dfrac{A_{\text{intersection}}}{2}\Big) + (k-1)^2 d^2, & \text{if } \dfrac{R}{d} > \dfrac{\sqrt{2}}{2} \end{cases} \tag{A.9}$$

Case 1: $\dfrac{R}{d} < \dfrac{1}{2}$

In the first possible system architecture, none of the sprinklers in the irrigation system will overlap each other as shown in Figure A.2. In this simple case, each of the K sprinklers will have a spatial coverage of πR^2, with a total irrigation

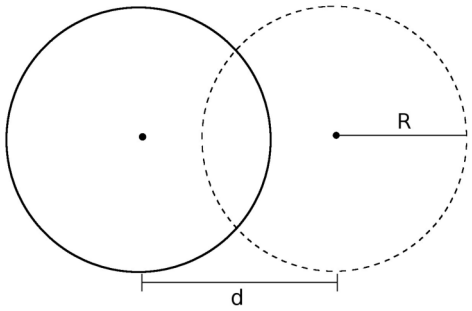


Figure A.1: Overlapping of circles

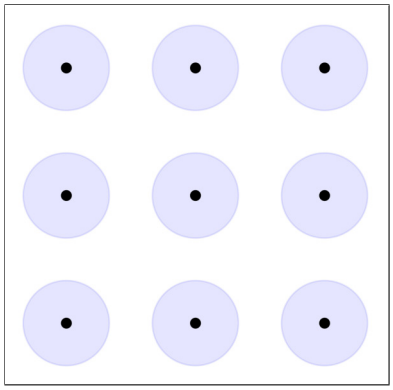
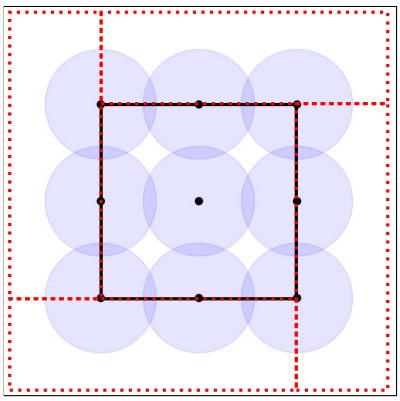
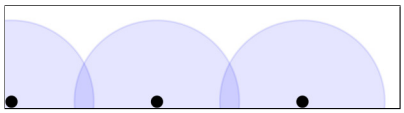


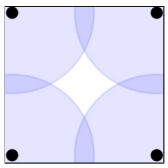
Figure A.2: Case 1: No sprinkler overlap is present in the system



(a) Total system coverage



(b) Repeats 4 times on grid borders



(c) "Cell" repeats $(k-1)^2$ times within grid

Figure A.3: Case 2: Sprinkler overlap is present, but is not sufficient to cover the diagonals grid distance. This results in incomplete coverage within the grid of sprinklers

system coverage of $K\pi R^2$.

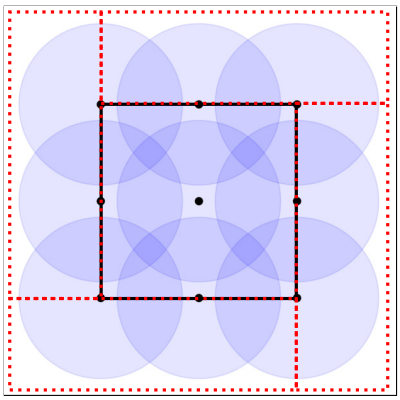
Case 2: $\frac{1}{2} \leq \frac{R}{d} \leq \frac{\sqrt{2}}{2}$

In this system architecture, the sprinkler can sufficiently reach the next nearest sprinkler in the grid, but can not reach the diagonal sprinklers in the grid, as shown in Figure A.3. For this reason, there are areas that will not receive direct irrigation in the center of the system. To compute the coverage of the system, we break the space into two areas. The first area lies *outside* the grid, as shown by four symmetrical regions surrounded by red dotted lines in Figure A.3a. The second is the area *within* the sprinkler grid, shown in Figure A.3a surrounded by a solid black line. Together, these two areas constitute the full coverage of the irrigation system.

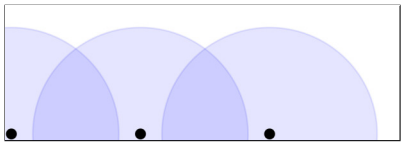
To compute the coverage of the irrigation system *outside* the grid, we consider one of the 4 outer boundary areas as shown in Figure A.3b. As shown, the outer area is composed of $2k-1$ quarter-circles of coverage, for a coverage of $\frac{\pi R^2}{4}(2k-1)$. To remove the darker overlapping coverage appearing in Figure A.3b, we compute the area of each of the $k-1$ overlapping areas, written as $A_{\text{intersection}}$ using Eq. A.8, noting that only half of each sprinkler's overlap will fall in this border region with the other half falling into the interior region. In this way, each of the four border areas will have coverage:

$$(2k-1)\frac{\pi R^2}{4} - (k-1)\frac{A_{\text{intersection}}}{2} \tag{A.10}$$

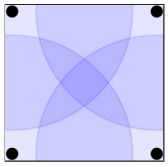
To compute the sprinkler coverage *within* the sprinkler grid, we break the grid into $(k-1)^2$ cells as shown in Figure A.3c. Within each grid cell, each of the four sprinklers will provide a quarter-circle of coverage for a total area of πR^2.



(a) Total system coverage



(b) Repeats 4 times on grid borders



(c) "Cell" repeats $(k-1)^2$ times within grid

Figure A.4: Case 3: Sprinkler overlap is sufficient to provide direct irrigation to all regions within the irrigation system sprinkler grid

We will then remove the 4 areas of overlapping coverage that appear darker in the figure written as $A_{\text{intersection}}$, again using Eq. A.8, noting that half of each overlapping region will fall into the grid cell we are considering. Then, each grid cell will have a coverage of:

$$\pi R^2 - 4\frac{A_{\text{intersection}}}{2} \tag{A.11}$$

In total, then, the irrigation system as shown in Figure A.3a with 4 symmetrical strips of coverage along the external boundary and $(k-1)^2$ internal grid cells will cover an area of:

$$4\Big((2k-1)\frac{\pi R^2}{4} - (k-1)\frac{A_{\text{intersection}}}{2}\Big) + (k-1)^2\Big(\pi R^2 - 4\frac{A_{\text{intersection}}}{2}\Big) \tag{A.12}$$

Case 3: $\frac{R}{d} > \frac{\sqrt{2}}{2}$

Finally, in the system architecture that is found in all standard irrigation system, all regions within the grid of sprinklers can receive direct sprinkler coverage, as shown in Figure A.4. Similarly to Case 2, above, we break the irrigation system as shown in Figure A.4a into 4 symmetrical boundary regions as shown surrounded by red dotted lines and the region within the grid surrounded with a solid black line.

Computing the coverage of each of the four surrounding boundary regions is identical to the process introduced in Case 2, above. Considering the coverage of the $2k-1$ quarter-circles and removing the overlap between the sprinklers, each boundary strip as shown in Figure A.4b has a total coverage area of:

$$(2k-1)\frac{\pi R^2}{4} - (k-1)\frac{A_{\text{intersection}}}{2} \tag{A.13}$$

To compute the sprinkler coverage *within* the grid of sprinklers, we break the system into $(k-1)^2$ grid cells, one of which is shown in Figure A.4c. As the sprinkler radius R is sufficient in this architecture for complete coverage within each grid cell, the coverage within each cell is simply d^2, where d represents the distance between the sprinklers as shown in Figure A.1. Then, the total coverage of the irrigation system with 4 symmetrical boundary regions and $(k-1)^2$ interior grid cells of coverage can be written as:

$$4\Big((2k-1)\frac{\pi R^2}{4} - (k-1)\frac{A_{\text{intersection}}}{2}\Big) + (k-1)^2 d^2 \tag{A.14}$$

References

[AF96] Steven F. Ashby and Robert D. Falgout. "A Parallel Multigrid Preconditioned Conjugate Gradient Algorithm for Groundwater Flow Simulations." *Nuclear Science and Engineering*, **124**(1):145–159, 1996.

[APN94] Aftab Hussain Azhar, BJC Perera, and Ghulam Nabi. "A Simple Soil Moisture Simulation Model to Address Irrigation Water Management Issues." *Water SA*, 1994.

[APR98] Richard G Allen, Luis S Pereira, Dirk Raes, Martin Smith, et al. "Crop evapotranspiration-Guidelines for computing crop water requirements-FAO Irrigation and drainage paper 56." *FAO, Rome*, **300**(9):D05109, 1998.

[ARV15] Karim C Abbaspour, Elham Rouholahnejad, Saeed Vaghefi, Raghavan Srinivasan, Hong Yang, and Bjørn Kløve. "A continental-scale hydrology and water quality model for Europe: Calibration and uncertainty of a high-resolution large-scale SWAT model." *Journal of Hydrology*, **524**:733–752, 2015.

[Bea72] J. Bear. *Dynamics of Fluids in Porous Media*. Dover Civil and Mechanical Engineering Series. Dover, 1972.

[BEK14] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. "Julia: A Fresh Approach to Numerical Computing." *CoRR*, 2014.

[BL94] AA Bloem and MC Laker. "Criteria for adaptation of the design and management of centre-pivot irrigation systems to the infiltrability of soils." *Water SA*, 1994.

[BL16] Sourav Bhattacharya and Nicholas D. Lane. "Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables." In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, SenSys '16, pp. 176–189, New York, NY, USA, 2016. ACM.

[BMG11] Getnet D Betrie, Yasir A Mohamed, Ann van Griensven, and Raghavan Srinivasan. "Sediment management modelling in the Blue Nile Basin using SWAT model." *Hydrology and Earth System Sciences*, **15**(3):807, 2011.

[cala] "Farming Irrigation Price Increases." http://www.bloomberg.com/news/articles/2014-07-24/california-water-prices-soar-for-farmers-as-drought-grows.

[calb] "Price of Water is Too Low." http://www.nytimes.com/2014/10/15/business/economy/the-price-of-water-is-too-low.html.

[CDM08] Bernard Cardenas-Lailhacar, Michael D Dukes, and Grady L Miller. "Sensor-based automation of irrigation on bermudagrass, during wet weather conditions." *Journal of Irrigation and Drainage Engineering*, **134**(2):120–128, 2008.

[cer] "Aerial Imagery - Spectral Imaging." http://www.ceresimaging.net.

[CGM12] André Chanzy, Jean-Claude Gaudu, and Olivier Marloie. "Correcting the temperature influence on soil capacitance sensors using diurnal temperature and water content cycles." *Sensors*, **12**(7):9773–9790, 2012.

[cha] "Adjustable LiPo Charger." https://www.sparkfun.com/products/14380.

[Chi69] Ernest Carr Childs. *An introduction to the physical basis of soil water phenomena*. A Wiley Intercience Publication John Wiley And Sons Ltd,; London; New York; Sydney; Toronto, 1969.

[cim] "CIMIS Weather Data." http://www.cimis.water.ca.gov/.

[CLL07] Jiabing Cai, Yu Liu, Tingwu Lei, and Luis Santos Pereira. "Estimating reference evapotranspiration with the FAO PenmanMonteith equation using daily weather forecast messages." *Agricultural and Forest Meteorology*, **145**(1):22–35, 2007.

[com] "Comsol Subsurface Flow Module." http://www.comsol.com/subsurface-flow-module.

[Cos06] Antonio Costa. "Permeability-porosity relationship: A reexamination of the Kozeny-Carman equation based on a fractal pore-space geometry assumption." *Geophysical Research Letters*, **33**(2):n/a–n/a, 2006.

[CS01] Laury Chaerle and Dominique Van Der Straeten. "Seeing is believing: imaging techniques to monitor plant health." *Biochimica et Biophysica Acta (BBA) - Gene Structure and Expression*, **1519**(3):153 – 166, 2001.

[Dar56] Henry Darcy. "Les fontaines publiques de la ville de Dijon. Paris: Dalmont." 1856.

[dec] "Decagon Devices." http://www.decagon.com/products/soils/.

[DLV13] Simon Duquennoy, Olaf Landsiedel, and Thiemo Voigt. "Let the Tree Bloom: Scalable Opportunistic Routing with ORPL." In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pp. 2:1–2:14, New York, NY, USA, 2013. ACM.

[DMD13] NA Dobbs, KW Migliaccio, MD Dukes, KT Morgan, and YC Li. "Interactive irrigation tool for simulating smart irrigation technologies in lawn turf." *Journal of Irrigation and Drainage Engineering*, 2013.

[edi] "Intel Edison Development Platform." https://software.intel.com/en-us/iot/hardware/discontinued.

[epaa] "Tune Your Irrigation System to Save Water and Money." https://www.epa.gov/sites/production/files/2017-03/documents/ws-watering-can-be-efficient.pdf.

[epab] "US EPA Watering Tips." https://www.epa.gov/watersense/watering-tips.

[epac] "US Outdoor Water Use." www3.epa.gov/watersense/pubs/outdoor.html.

[epad] "Water Efficiency Management Guide, Landscaping and Irrigation." https://www.epa.gov/sites/production/files/2017-12/documents/ws-commercialbuildings-waterscore-irrigation-landscape-guide.pdf.

[FGJ06] Rodrigo Fonseca, Omprakash Gnawali, Sukun Kim Kyle Jamieson, Philip Levis, and Alec Woo. "The Collection Tree Protocol (CTP)." Technical report, TinyOS Enhancement Proposals, TinyOS 2.0 Network Protocol Working Group, August 2006.

[fre] "Freshwater Crisis." http://environment.nationalgeographic.com/environment/freshwater/freshwater-crisis/.

[gal] "Galaxy S10 Intelligence." https://www.samsung.com/global/galaxy/galaxy-s10/intelligence/.

[GBG03] B Grizzetti, F Bouraoui, K Granlund, S Rekolainen, and G Bidoglio. "Modelling diffuse emission and retention of nutrients in the Vantaanjoki watershed (Finland) using the SWAT model." *Ecological Modelling*, **169**(1):25–38, 2003.

[GBY08] Michael Grant, Stephen Boyd, and Yinyu Ye. "CVX: Matlab software for disciplined convex programming.", 2008.

[gcs] "Golf Course Superintendents Association of America Environmental Report." https://www.gcsaa.org/uploadedfiles/Environment/Environmental-Profile/Property-Profile/Golf-Course-Environmental-Profile–Property-Report.pdf.

[Gen80] M Th Van Genuchten. "A closed-form equation for predicting the hydraulic conductivity of unsaturated soils." *Soil science society of America journal*, **44**(5):892–898, 1980.

[GHN03] Quentin Gallas, Ryan Holman, Toshikazu Nishida, Bruce Carroll, Mark Sheplak, and Louis Cattafesta. "Lumped element modeling of piezoelectric-driven synthetic jet actuators." *AIAA journal*, **41**(2):240–247, 2003.

[glp] "GNU Linear Programming Kit." https://www.gnu.org/software/glpk/.

[GLR14] Petko Georgiev, Nicholas D Lane, Kiran K Rachuri, and Cecilia Mascolo. "Dsp. ear: Leveraging co-processor support for continuous audio sensing on smartphones." In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pp. 295–309. ACM, 2014.

[GLS14] Euhanna Ghadimi, Olaf Landsiedel, Pablo Soldati, Simon Duquennoy, and Mikael Johansson. "Opportunistic Routing in Low Duty-Cycle Wireless Sensor Networks." *ACM Trans. Sen. Netw.*, **10**(4):67:1–67:39, June 2014.

[gol] "Golf Course Irrigation Systems - Water Efficiency." http://www.liaquifercommission.com/images/Golf_Course_Irrigation-_Systems_march_april_2015.pdf.

[HC02] Jason L Hill and David E Culler. "Mica: A wireless platform for deeply embedded networks." *IEEE micro*, **22**(6):12–24, 2002.

[HL12] Thomas Harter and Jay R. Lund. "Addressing Nitrate in California's Drinking Water." Technical report, University of California, Davis, 2012.

[huna] "Hunter: Advanced Evapotranspiration Weather Control." https://www.hunterindustries.com/en-metric/irrigation-product/sensors/et-system.

[hunb] "Hunter Industries ET Sensor." https://www.hunterindustries.com/irrigation-product/sensors/et-sensor.

[hunc] "Hunter MP Rotator." hunterindustries.com/irrigation-product/nozzles/ mp-rotator.

[hund] "Hunter Pro-spray sprinklers." https://www.hunterindustries.com/irrigation-product/spray-bodies/pro-sprayr.

[hyd] "Hydrus Liquid Simulator." http://www.pc-progress.com.

[iPh] "A12 Bionic." https://www.apple.com/iphone-xs/a12-bionic/.

[JAC16] Z. Jia, M. Alaziz, X. Chi, R. E. Howard, Y. Zhang, P. Zhang, W. Trappe, A. Sivasubramaniam, and N. An. "HB-Phone: A Bed-Mounted Geophone-Based Heartbeat Monitoring System." In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 1–12, April 2016.

[JBA90] Marvin Eli Jensen, Robert D Burman, and Rick G Allen. "Evapotranspiration and irrigation water requirements." ASCE, 1990.

[JW01] Jim E. Jones and Carol S. Woodward. "NewtonKrylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems." *Advances in Water Resources*, **24**(7):763 – 774, 2001.

[Kal03] Eugenia Kalnay. *Atmospheric modeling, data assimilation, and predictability*. Cambridge university press, 2003.

[KDB17] J.G. Kroes, J.C. van Dam, R.P. Bartholomeus, P. Groenendijk, M. Heinen, R.F.A. Hendriks, H.M. Mulder, I. Supit, and P.E.V. van Walsum. "SWAP version 4; Theory description and user manual." Technical report, Wageningen, Wageningen Environmental Research, 2017.

[Kir04] M.B. Kirkham. *Principles of Soil and Plant Water Relations*. Elsevier Science, 2004.

[KM06] Stefan J. Kollet and Reed M. Maxwell. "Integrated surfacegroundwater flow modeling: A free-surface overland flow boundary condition in a parallel groundwater flow model." *Advances in Water Resources*, **29**(7):945 – 958, 2006.

[l91] "Energizer Ultimate Lithium." http://data.energizer.com/pdfs/l91.pdf.

[LKS10] Jongdeog Lee, Krasimira Kapitanova, and Sang H Son. "The price of security in wireless sensor networks." *Computer Networks*, **54**(17):2967–2978, 2010.

[lor] "LoRa Alliance." https://lora-alliance.org/.

[LTL15] Yufeng Luo, Seydou Traore, Xinwei Lyu, Weiguang Wang, Ying Wang, Yongyu Xie, Xiyun Jiao, and Guy Fipps. "Medium range daily reference evapotranspiration forecasting by using ANN and public weather forecasts." *Water resources management*, **29**(10):3863–3876, 2015.

[Max13] Reed M. Maxwell. "A terrain-following grid transform and preconditioner for parallel, large-scale, integrated hydrologic modeling." *Advances in Water Resources*, **53**:109 – 117, 2013.

[MD17] S.B. McCreath and R. Delgoda. *Pharmacognosy: Fundamentals, Applications and Strategies*. Elsevier Science, 2017.

[MPT17] M. Moazzami, D. E. Phillips, R. Tan, and G. Xing. "ORBIT: A Platform for Smartphone-Based Data-Intensive Sensing Applications." *IEEE Transactions on Mobile Computing*, **16**(3):801–815, March 2017.

[MTT93] Miguel A Marino, John C Tracy, and S Alireza Taghavi. "Forecasting of reference crop evapotranspiration." *Agricultural water management*, **24**(3):163–187, 1993.

[nas] "Looking for Lawns." http://earthobservatory.nasa.gov/Features/Lawn/printall.php.

[NM65] John A Nelder and Roger Mead. "A simplex method for function minimization." *The computer journal*, **7**(4):308–313, 1965.

[noa] "NOAA Maps and Models." https://www.weather.gov/akq/mapsmodels.

[Oli97] M. J. Oliver. "Evapotranspiration forecasting irrigation control system.", December 1997. US Patent 5,696,671.

[pcb] "PCBWay Manufacturing and Assembly." https://www.pcbway.com/.

[PHC04] Joseph Polastre, Jason Hill, and David Culler. "Versatile low power media access for wireless sensor networks." In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 95–107. ACM, 2004.

[pho] "Photonics - Plant health imaging filters." https://www.photonics.com/Product.aspx?PRID=60994.

[piz] "Raspberry Pi Zero." https://www.raspberrypi.org/products/raspberry-pi-zero/.

[POT10] Jorge Portilla, Andrés Otero, Eduardo de la Torre, Teresa Riesgo, Oliver Stecklina, Steffen Peter, and Peter Langendörfer. "Adaptable security in wireless sensor networks by using reconfigurable ECC hardware coprocessors." *International Journal of Distributed Sensor Networks*, **6**(1):740823, 2010.

[PSL13] Enrique Playán, Raquel Salvador, Cristina López, Sergio Lecina, Farida Dechmi, and Nery Zapata. "Solid-set sprinkler irrigation controllers driven by simulation models: Opportunities and bottlenecks." *Journal of Irrigation and Drainage Engineering*, **140**(1):04013001, 2013.

[pwp] "Permanent Wilting Point." http://nrcca.cals.cornell.edu/soil/CA2/CA0212.1-3.php.

[raia] "Hunter Rain-Clik Rain Detection." hunterindustries.com/irrigation-product/sensors/rain-clikr.

[raib] "Rain Bird Evapotranspiration Manager." https://www.rainbird.com/documents/turf/bro_ETManager.pdf.

[raic] "Rain Bird RSD Series Rain Shut Off." http://www.rainbird.com/products/rsd-series-rain-shut.

[rasa] "Raspberry Pi." https://www.raspberrypi.org/.

[rasb] "Raspbian Operating System." https://www.raspbian.org/.

[RGV04] E Rodriguez, F Giacomelli, and A Vazquez. "Permeability-porosity relationship in RTM for different fiberglass and natural reinforcements." *Journal of composite materials*, **38**(3):259–268, 2004.

[Ric31] L. A. Richards. "Capillary Conduction of Liquids Through Porous Mediums." *Physics*, **1**:318–333, November 1931.

[RME02] Rolf H Reichle, Dennis B McLaughlin, and Dara Entekhabi. "Hydrologic data assimilation with the ensemble Kalman filter." *Monthly Weather Review*, **130**(1):103–114, 2002.

[sam] "Microcontrollers and Processors - SAM R21." https://www.microchip.com/wwwproducts/en/ATSAMR21G18A.

[SB92] EgbertJ. A. Spaans and JohnM. Baker. "Calibration of Watermark soil moisture sensors for soil matric potential and temperature." *Plant and Soil*, **143**(2):213–217, 1992.

[she] "Sheeva Plug DevKit." https://www.globalscaletechnologies.com/t-sheevaplugs.aspx.

[swa] "Soil and Water Assessment Tool Theoretical Documentation." https://swat.tamu.edu/media/1292/SWAT2005theory.pdf.

[TC05] Gilman Tolle and David Culler. "Design of an application-cooperative management system for wireless sensor networks." In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pp. 121–132. IEEE, 2005.

[tmo] "Tmote Sky." http://www.snm.ethz.ch/Projects/TmoteSky.

[ugm] "UGMO Irrigation." http://www.ugmo.com/.

[usd] "Soil Quality Indicators." https://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/-nrcs142p2_053288.pdf.

[wea] "Weathermatic weather sensors." https://www.weathermatic.com/products/weather-sensors/.

[Wei03] Eric W. Weisstein. "Circle-Circle Intersection." http://mathworld.wolfram.com/Circle-CircleIntersection.html, 2003.

[WJS93] KH Wesseling, JRC Jansen, JJ Settels, JJ Schreuder, et al. "Computation of aortic flow from pressure in humans using a nonlinear, three-element model." *Journal of applied physiology*, **74**:2566–2566, 1993.

[WSC13] Yi Wang, Hailong Shi, and Li Cui. "EasiSec: a SoC security coprocessor based on fingerprint–based key management for WSN." *International Journal of Sensor Networks*, **13**(2):85–93, 2013.

[wun] "Wunderground - Weather Forecast and Reports." https://www.wunderground.com/.

[WWB16] D. A. Winkler, R. Wang, F. Blanchette, M. Carreira-Perpinan, and A. E. Cerpa. "MAGIC: Model-Based Actuation for Ground Irrigation Control." In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April 2016.

[YZS18] Shuochao Yao, Yiran Zhao, Huajie Shao, ShengZhong Liu, Dongxin Liu, Lu Su, and Tarek Abdelzaher. "FastDeepIoT: Towards Understanding and Optimizing Neural Network Execution Time on Mobile and Embedded Devices." In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, pp. 278–291, New York, NY, USA, 2018. ACM.

[YZZ17] Shuochao Yao, Yiran Zhao, Aston Zhang, Lu Su, and Tarek Abdelzaher. "DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework." In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, pp. 4:1–4:14, New York, NY, USA, 2017. ACM.