ROAM: A Decision Support System for Software-Defined Farms

Shiang-Wan Chin
Cornell University
Ithaca, New York, USA
sc2983@cornell.edu

Gloire Rubambiza
Cornell University
Ithaca, New York, USA
gloire@cs.cornell.edu

Yifan Zhao
Cornell University
Ithaca, New York, USA
yz348@cornell.edu

Keyvan Malek
University of Illinois
Urbana-Champaign
Champaign, Illinois, USA
k1malek@illinois.edu

Hakim Weatherspoon
Cornell University
Ithaca, New York, USA
hweather@cs.cornell.edu

ABSTRACT

The growing disparity between food supply and demand requires innovative Digital Agriculture (DA) systems to increase farm sustainability and profitability. However, current systems suffer from problems of complexity. To increase farm efficiency and understand the tradeoffs of these new DA innovations we developed ROAM, which is a decision-support system developed to find a Pareto optimal architectural design to build DA systems. Based on data from five live deployments at Cornell University, each DA design can be analyzed based on user defined metrics and evaluated under uncertain farming environments with ROAM. Paired with this, we develop a web interface that allows users to define personalized decision spaces and to visualize decision tradeoffs. To help validate ROAM, it was deployed to a commercial farm where the user was recommended a method to increase farm efficiency. ROAM allows users to quickly make key decisions in designing their DA systems to increase farm profitability.

Keywords: Digital agriculture, Decision making under deep uncertainty, Systems optimization, Systems engineering, Internet of Things, Sustainability

1 INTRODUCTION

The 2018 Global Agricultural Productivity (GAP) index highlights a growing disparity between food supply and demand, for both developed and developing countries [50]. Conservative estimates predict that agricultural production will need to increase by 25-70% above current levels to meet the demand expected by 2050. As a result, the world is likely to face a large-scale food security crisis [50]. A major challenge to increasing food production is farm efficiency which is challenged by limited rural infrastructure [52].

Digital Agriculture (DA), which is the use of data-driven techniques to increase farm productivity and sustainability, is thought of as a method of addressing the crisis [8]. Research into data-driven agriculture is growing. It envisions a future in which on-farm data collection, processing, and transmission are ubiquitous[22]. Several start-up companies are developing applications for data-driven farms [24], while major agribusiness firms are developing data collection and processing systems [24].

According to Douthwaite et al., DA innovations are complex and require involving farm stakeholders to understand their goals and constraints to successfully deploy [52]. First, current DA solutions are often fragile due to non-interoperable hardware and software

[42]. Second, DA solutions often take a generalized approach that is not suitable for the myriad of farmers, each of whom has unique demands and constraints which require personalized solutions; e.g. a specialty grape farm can focus on achieving a specific taste profile while a row crop corn farm can focus on optimizing yield [8]. These challenges often lead to low understanding, slow adoption, and high costs in implementing DA systems [52].

In this paper, we present the Realtime Optimization and Management System (ROAM), which helps identifies a Pareto optimal set of tradeoffs that helps farmers identify a desired point within the tradeoffs space. Based on several years of experience deploying DA systems in several research farms associated with Cornell University, we have determined which data and decision points should be accounted for, and designed a user-friendly platform for farmers to define the unique goals and constraints for their particular farm. ROAM determines a Pareto front of optimal DA system architectures a farmer can choose between, usually eliminating the vast majority of potential architectures. Thus, ROAM advances the state of the art in deploying DA systems. It performs up-front analysis necessary to deploy DA systems and eliminates major barriers to the diffusion of DA techniques into real-world farms and increasing farm efficiency.

The design of ROAM is based on formalizing a method to evaluate a DA architecture by encoding user generated evaluation metrics and uncertainties to assess each architectural decision into a ROAM Configuration File. An architectural decision is the choice between different components of the DA system such as between a soil moisture or light sensor. Then, the ROAM Configuration File is used to create nodes or objects that represent unique architectural configurations of a DA system. The architectural representation is a subset of architectural decisions made to create a DA system. The nodes are then passed into an optimization function to uncover the one architectural representation most suitable to a user's need. To abstract away the complexity of the ROAM implementation a front-end user interface is designed and used to allow for easy entry of key features of the user's farm, constraints, and uncertainties. This frontend creates the ROAM Configuration File used for ROAM evaluation. In addition, as output, the frontend displays an interactive 3-D data visualizations of the farmers potential DA system tradespace, which is then used to allow for better understanding of the recommendations of the system. The entire process from beginning to end, from encoding the ROAM Configuration File to the end step of visualization of the analysis is modularized to allow for swapping in and out interchangeable software. For example, different types of optimization models can be used in the ROAM.

To validate the generalizability of ROAM, it was used by Cheng Xin Garden LLC, a commercial California-based viticulture farm. As part of the process, ROAM considered different decisions to create a DA system based on their needs through in-depth user interviews. ROAM identified 324 architectural decisions and narrowed that down to one based on many factors such as climate change and location of the farm. The identified optimal architecture increases Cheng Xin Garden's farm efficiency while accounting for constraints and uncertainties. To summarize our work the research contributions are the following:

(1) Experience developing and deploying several different DA systems
(2) Recommendations for a Pareto optimal DA system deployment
(3) Design and implementation of ROAM
(4) A commercial farm deployment using and validating ROAM's utility

The rest of this paper will be structured as follows. Digital agriculture systems that motivate the development of ROAM will be outlined in Section 2. Section 3 discusses how the ROAM software is built. The tradespace model formulation is described in Section 4. Section 5 delves into how user inputs are compiled into a configuration file and optimization function are applied. Optimization libraries and concepts are used for deeper analysis in Section 6. Section 7 outlines the user interface for users to input farm data. A commercial farm deployment of ROAM is described in Section 8. We conclude with a discussion of our results in Section 9 and summarize our findings and work in Section 10.

2 NETWORK-ENABLED FARM

Digital Agriculture (DA) is the use of data to improve farm decision making that can lead to increased environmental sustainability and farm profitability [36]. DA is composed of sensing, storing, computing, and actuating technologies that leverage on-farm data [44]. Gathering massive amounts of sensor data requires a robust network, but this is a challenge as farms in rural areas often have limited or no on-farm networking or Internet access. A Network-Enabled Farm (NEF) addresses these issues by using new technologies or old technologies repurposed to provide networking capabilities in the middle of a farm such as, 4G LTE, Long Range Radio (LoRa), and unlicensed TV White Spaces (TVWS) [5]. A Software-Defined Farm (SDF) leverages a NEF to sense, transmit, and analyze farm data to produce actionable insights for farm stakeholders, as described in Seamless Visions, Seamful Realities: Anticipating Rural Infrastructural Fragility in Early Design of Digital Agriculture [42]. The NEF provides the networking infrastructure for the SDF to enable data-driven DA to optimize farm management.

The SDF is a modular abstraction of software and hardware technologies that is designed to fit the various needs of farmers. The software abstraction is split into 3 modules: Sensing, Computing, and Actuating. The Sensing module abstracts away sensors that allows different hardware sensors to be connected through software. The Computing module allows for different analytics algorithms to be run to support decision making. The Actuating module performs some type of action such as releasing irrigation valves. These modules can connect manufacturer agnostic hardware devices such as computers located at the farmhouse, field sensors, and water valves. With both the software and hardware connected, farmers can visualize aggregate data from normally incompatible farming systems on a web application interface [52]. To gain operational insights, farmers can run analytics on their data to make farm decisions. Lastly, an SDF enables the creation of digital twins of the physical farming system to automate farm processes such as precision irrigation.

The SDF interfaces for the Sensing, Computing, and Actuating modules are well defined and static, but the implementation of the modules change to fit the need of the SDF user needs. For instance, different sensors such as soil moisture, light and/or wind can be used for the Sensing module. Different analytics implementations such as machine learning disease detection, irrigation scheduling, and/or cow health monitoring can be run in the Computing module. Lastly, the Actuating module can be in the form of an email alert, turning on irrigation value, or controlling greenhouse internal temperatures. Note that the modules can be hosted by different cloud providers such as Microsoft Azure, Google Compute Platform (GCP), or Amazon Web Service (AWS), and/or run in the farm house at the “edge” of the cloud.

We have experience implementing and deploying several SDF instances, including an apple orchard, corn and cannabis greenhouse, dairy cow farm, and a vineyard [41]. These instances of SDF deployments utilized research farms associated with Cornell University and were implemented over a span of three years. These deployments highlight both the flexibility of the SDF concept, as well as the importance of tailoring each deployment to fit the needs of each individual farm. The SDF instances use cutting-edge networking technologies such as TV White-space, LoRa, and sensors such as in situ plant water sensors [41] (See Figure 1). Figure 1 shows a data-driven irrigation graphic of how the SDF connects the Sensing Module through a sensor (1), sensor box (2), and subedge or edge computation device (3) to the Computing Module through a cloud software service (4) to the Actuating Module with a raspberry pi (5) [39] and actuation function (6).

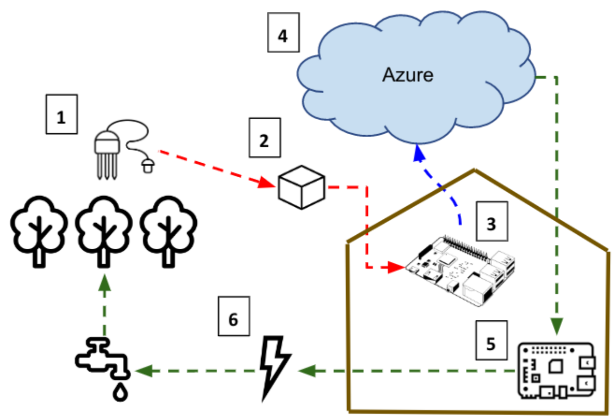

Figure 1: SDF Decision Space

One constant throughout our experience implementing these various SDF deployments was the difficulty in balancing complex, multifaceted tradeoffs between cost, risk, and performance. Here, cost refers to the monetary cost of hardware, software, installation, and maintenance needed to deploy and maintain a SDF. Risk refers to the potential for interruption of sensor devices and networking. Performance is an aggregate metric that combines anticipated yield increase with anticipated water, electricity, and labor cost savings. Drawing from our three years of experience analyzing these tradeoffs, we present the design and implementation of the Tradespace Exploration System (ROAM), a tool and computational method to assist in optimizing cost, risk, and performance of an SDF. Furthermore, the ROAM incorporates user input and uncertainties such as climate change in a farming environment. In the following section, we describe ROAM in more detail.

3 SOFTWARE DESCRIPTION

ROAM is an open-source software. It includes a client-side browser-based interactive application and a server-side back-end service. ROAM is designed and developed in a back-end and front-end setup due to the need for computational resources and data storage in the back-end, as well as the need for a user-friendly interface to lower technology barriers to our various stakeholders. The server-side back-end is developed with Python as the core programming language and hosts most functionalities, including optimization, analytics, and data storage. We selected the Python Flask framework to develop the client-side web application with Javascript as a core programming language. Both the back-end service and the front-end application integrates functionalities from multiple external libraries and custom modules.

The system consists of 4 main modules: the Decision, Rhodium, Uncertainty, and Graphical User Interface (GUI) modules as seen in Figure 2. The Decision module defines and maintains the tradespace architecture from the Decision Configuration File and it hosts the Tradespace Enumeration and Optimization algorithms. The Uncertainty module defines the uncertainty variables and models uncertain farming environments using real-time data. The Rhodium module hosts functions responsible for extension and orchestration of the integrated third-party Many-Objective Robust Decision Making (MORDM) libraries and provides key analysis of the tradespace. The GUI hosts the front-end interface and handles user data acquisition and visualization. The external libraries are selected from popular and regularly maintained open-source communities. A summary of these systems and libraries is provided in Table 1.

Library	**Language**	**Usage**
Rhodium	Python	MORDM
j3	Python	Visualization
oapackage	Python	Optimization
plotly	JavaScript	Visualization
d3	JavaScript	UI, data acquisition, visualization

Table 1: Tools and Libraries

4 TRADESPACE MODEL

To model and evaluate SDF designs, we draw from the study of systems architecture for developing configurable complex systems and evaluating how well they satisfy stakeholder needs [45]. To

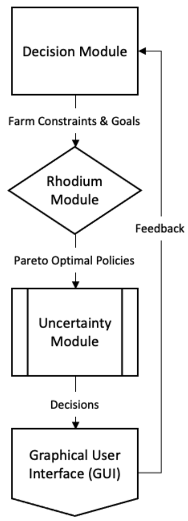

Figure 2: System Modules

decompose a complex system, we formulate a systems architecting optimization problem that represents a complex architecture as a set of decisions using an encoding scheme. Generally, optimization problems that result from decisions in systems architecture are combinatorial. To treat programmed decisions analytically we segment the decisions into six canonical decisions classes using real-world problems: standard form, assigning, partitioning, permuting, downselecting, and connecting [45]. These patterns are interlinked and have some overlap, so we can think of the six classes as combinations of standard form and down-selecting decisions.

The standard form (SF) decisions are decisions in which a user can only select one option from a set of alternatives. When making multiple SF decisions, the number of possible combinations of decisions is given by

$$\prod_{i=1}^{N} m_i \tag{1}$$

where m_i is the number of alternatives for an i decision and N is the number of decisions to be made [45]. In contrast, downselecting (DS) decisions are where a user can choose more than one alternative. The number of possible choices is given by

$$2^N \tag{2}$$

where N is the number of alternatives. The next step of creating the tradespace model is to create decisions to define the architecture space and subsequently to create metrics to evaluate the architectures. As emphasized, the SDF needs to focus both on pragmatic deployments of software and hardware components, so in any decision space we need to consider multiple types of decisions. Table 2 is an example of a set of decisions, their descriptions and importance, and the canonical class used to create and evaluate a SDF.

4.1 Problem Formulation

Once the tradespace has been constructed, defining metrics is needed for the evaluation of each architecture [53]. We conducted a stakeholder analysis by interviewing 11 farmers in California, Washington, and New York. We identified 3 metrics (cost, performance, and risk) as those most important when evaluating new technology investments. The farmers we interviewed expressed sensitivity to decisions that affected these metrics and through our analysis we understood variations across different architectures using principles in system architecture [45]. Based on decisions defined by a user of the system, value functions need to be created that evaluate each decision based on metrics for each architecture in the tradespace as will be shown in subsection 5.1 [45]. A value function, as described by Crawley, can be seen as a "transfer function" where the input is a system architecture and the output is an evaluation of the given architecture. Given the complexity of a real system, metrics need to be backed up via extensive testing, simulations, and fine tuning in future iterations.

The metric formulations and their subsequent values were based on data from journal publications [19][51], 11 farmer interviews, and experience with five SDF deployments described in section 2. Examples of the data include the real yield increment each year, the production each year, the price of the devices, and the cost of each component; as well as subsequent maintenance costs. The

#	Decision Name	Why it is important	Importance	Justification
1	Product Information	The type of Product Information to be collected is an important decision that will also impact scalability. Animals will likely require a higher-frequency monitoring as opposed to plants.	Very High	This is a downselecting decision as we are able to decide for multiple alternatives from the initial set. Decisions range from resources that require the lowest-frequency monitoring to animals requiring the highest-frequency monitoring.
2	IoT Devices	IoT devices are a crucial decisions that must be weighed between cost and functionality. The devices that are too costly will not be feasible for farmers to implement, while those that aren't functional will not be able to collect robust enough data.	High	This is a standard form decision because we think that a system with more than one manufacturer would not be scalable enough to accommodate a host of users.
3	User Interface	The type of user interface is an important component which can affect performance and user's attraction. The different user interfaces can provide different functions and it is where the customer can directly interact with our system, so we think it's a high priority.	Medium	We can provide multiple types of user interfaces for our users at the same time, such as message, website, and application. These options are not exclusive to each other.
4	Systems Architecture	The possibility of scaling is important for our system as different system architecture might rule out a growing user base in the future. Similarly, scalable architectures are likely to require more initial effort to set up the system and will only pay off with a large user base.	Medium	This decision is SF since it is formulated as picking one range from a set of options.
5	Data Type	The type of Data Storage is an important decision as it determines the security measures we intend to implement. A blockchain-based data storage would be the most secure decision which will impose constraints on the scalability of the possible user base.	Low	This is a downselection decision as we could think of a hybrid system that uses a cloud-based database and a blockchain backend in concert with each other. A CSV based backend would have the smallest amount of dependencies but would likely lack scalability and performance.
6	Data Collection	One important process in our system is Data Collection from the user side. There are multiple ways we can do them, each method can strongly affect our system architecture and performance. For example, if we choose manual input, then we need to consider a model for human labors. The options are flexible since the method of collecting data does not block our system performance.	Low	Since our system has multiple components for data collecting, such as measuring temperature, track product information. Some of them can be automatic, while some of them have to be manual. We can have manual, automatic, or semi-automatic.
7	Data Storage	The data storage size is used to limit our capacity for storing our product information, user account information, and some intermediate data. The scale of our storage size determines our project scale and server stress.	Low	We consider this decision as SF since the options are exclusive with each other, we can only choose one from them.
8	Notification System	This is a process that is crucial for the functionality of the system. In order for the stakeholders in the network to receive value, they must be able to interface with the system.	Low	We can see this as a down selecting decision as a subset of alternatives would be possible such as Email and real-time display simultaneously.
9	File Exchange Type	File exchange types that are streamlined will allow the system will run more efficiently. If they are not, then the processing time will increase.	Very Low	This is a standard form decision as a system with more than one file format would be very fragile with respect to ensuring data consistency.
10	Machine Learning	Machine learning model allows us to do prediction on yield, risk, weather, etc.	High	This is a standard form decision since it takes too long to do prediction; at present we can try only one option.

Table 2: Description of canonical decisions and their importance for the architecture

experience refers to the five SDF deployments that were deployed at Cornell University mentioned in section 2.

4.2 Formulation of Metrics

To formulate the Pareto optimal investment operating policy for a given farmer we create a function composed of three metrics. From work done by Cohon and Marks, and Reed we can define our multi-objective problem with a vector, $F(d)$, as demonstrated by the following equation [13][25].

$$F(d) = (F_{cost}, F_{risk}, F_{perf}) \tag{3}$$

$$\forall d \in \Omega \tag{4}$$

subject to user defined:

$$F_{cost}, F_{risk}, F_{perf} \tag{5}$$

Here d is a vector of decision variables in the tradespace Ω . These decisions can be expressed as real numbers utilizing value functions. Each $F(d)$ operating policy is evaluated based on its cost, risk, and performance which can be constrained by user input. For example, in the SDF referred to in section 2 because we want the system to be low cost, we can constrain cost to be less than or equal to $2000 and it would be denoted as $F_{cost} \leq 2000$. In terms of optimization, the performance metric is maximized while the cost and risk metrics are minimized. Each metric will be explained in the following sections.

4.2.1 Cost Metric. The first goal of the system is to minimize cost as denoted by the equation:

$$F_{cost} = H + M + S + I \tag{6}$$

The cost metric includes the cost of Hardware (H), subsequent Farm Maintenance Cost (M), Software (S), and Installation (I). The hardware and installation costs are vital to minimize the total costs of implementing an SDF. Farmers typically have a limited upfront budget for investments and face many costly decisions in investing in new technologies [49]. For example, the cost of sensors may make deploying full sensor networks prohibitively expensive in this context [28]. Thus, if the sensors are too expensive, they will not be implemented on farms where capital and cash reserves are a constraint. On the other hand, if the sensors are very cheap, the system may display low performance and have a high risks of malfunction when used over time. As a result, cheap sensors that need constant repair would increase the maintenance cost, resulting in large labor costs for the farmer. We factor in the time needed to calibrate sensors, fix devices, clean equipment, and change batteries based on experience from deployments of sensors onto a farm [26]. If the costs to keep the systems running outweigh the benefit of optimizing the farm, it will be ineffective at helping farmers. Lastly, software costs are increasingly important as corporations pivot to Software as a Service (SaaS) models where cost per computation is the norm. As a result, for larger farms with an abundance of sensors, computation costs and software services will be much more expensive. It is also important to note that the type of farm, region, and climate also influence which sensors and decisions are the most suitable. For example, a soil moisture sensor is less suitable in environments where temperature regularly drops below freezing point and the ground freezes. It is important to note that efforts were made to create a holistic cost metrics, but in complex living systems such as a farm there are many unforeseen costs.

4.2.2 Risk Metric. The second system goal is to minimize risk,

$$F_{risk} = S + N \tag{7}$$

This equation quantifies the interruption risk of the Sensor Devices (S) and Networking (N) of an SDF design. In a deployed SDF, there are two reasons why data from sensors might be incorrect or missing. First, the sensor hardware itself can malfunction due to climate, environmental, or implementation factors. These malfunctions can lead to both gaps in data collection and incorrect data collection, both of which can lead to inaccurate decision support and potentially necessitate costly repairs. These risks are captured by S in the above formula. On the other hand, if the network is unreliable, even if the sensors are collecting data properly, it cannot be transmitted to edge and cloud computers. This risk is captured by N in the above formula.

Understanding S and N are important for the quality of insights the SDF can generate. As a result, if there is a great deal of interruption risk, it can be linked to a bad quality SDF architecture. In ROAM, we define interruption risk as the probability of failure in the S to send and N to transmit data packets. While ROAM can use default quantities for these risks determined through averaging the risks experienced by farmers in our user interview studies, we allow farmers to instead provide their own quantities based on their personal evaluation based on the local conditions at their farm. As systems become ever more complex with many dependencies the risk metric will be all the more important.

4.2.3 Performance Metric. The third system goal is to maximize performance

$$F_{perf} = Y + W + E + L \tag{8}$$

The equation above represents the utility of the system's service to users, a metric directly tied to creating value for users. The performance metric is developed as a combination of Yield Increase (Y), Water Cost Savings (W), Electricity Cost Saving (E), and Labor Cost Savings (L), representing four ways in which an SDF deployment can add value for farmers. One of the primary ways in which an SDF can improve farms is by generating insights that allow farmers to grow more high-quality crops per acre of farmland. For example, the SDF can identify underperforming parts of the field and suggest how to improve them. In addition, an SDF can improve water costs by suggesting optimal watering amounts based on sensor data such as soil moisture levels [17]. SDFs also have different electricity costs depending on the specific technologies used; for example, solar power may be cheaper than disposable batteries in the long run. Finally, SDFs can remove the need for human labor in some cases. For example, one of the farmers we interviewed during our user research described needing to hire a worker to walk the field everyday to measure soil moisture in every hectare of the farm, labor which would not be necessary in an SDF with a sensor network to measure soil moisture. Performance was often thought about as the most important metric for our farmers in evaluating new technology investments.

4.3 Uncertainties

Once we establish the metrics and value functions for evaluating architectures in the tradespace, we must define the uncertainties and their effects on the various architectures within the tradespace. With the goal to improve farmers' competitiveness and extract insights from farming for decision-making, the system must be evaluated under the deeply uncertain farming environment reflecting reality. More formally, an uncertainty in the tradespace model characterizes the behavior of an uncertain factor affecting a farm as a variable [25]. The reason for having these uncertainties is to capture the attributes and metrics of architecture in multiple instances of the uncertain environment, which provides a more realistic evaluation of the architecture and aids the decision-making process section 6. This section focuses on the uncertainty variables constructed in ROAM. In contrast, the relationship between each uncertainty and metrics of each decision will vary depending on different tradespace configurations, which is showcased in section 8.

Climate Complexity. The farm climate is a complex nonlinear system, where different levels of short-term climate complexity may affect the performance of the farm. Climate Complexity (CC) can lead to risks of sensor malfunction and suboptimal performance of hardware devices as they operate while exposed to outdoor farming environments. For example, solar power sources can face risks of interruption in extreme weather events such as large storms. Utilizing information theory techniques, the CC uncertainty variable aims to represent an approximate proxy to analyze and predict the level of regional short-term climate variability in a given farm area. CC uncertainty is modeled using an entropy-based measurement that is referred to as SampEn. It provides a nonlinear approach for analyzing and predicting the entropy or complexity of climatic time series [47]. It is a probability measure that quantifies the likelihood that sequences of consecutive data match one another within a tolerance r and remain similar when the length of the sequences is increased by one sample. In this way, we quantify the regularity and the unpredictability of fluctuations in weather to factor into our model. In order to calculate individual farm level SampEn we use data from the Global Climate Models (GCMs) dataset [2]. The data is then processed based on the algorithm introduced in the paper Approximate Entropy and Sample Entropy: A Comprehensive Tutorial [15]. According to the SampEn calculations of climate complexity of regional meteorological data found by Shuangcheng in his paper Measurement of Climate Complexity, he found from using random climate data that SampEn approached 0 and with fully homogeneous data that it approached 3 [47]. As a result we use the SampEn range from 0 to 3 with a uniform distribution to model climate complexity as shown in table 3.

Rainfall. Rainfall has been directly linked to impacting yield of agricultural products [23]. According to Hunho, it is seen that increased rainfall leads to a longer growing season and higher yields which in turn becomes higher profits for the farmer. On the other hand, in this study published in the journal Global Change Biology, rainfall was detrimental to certain crop yield [34]. In the study corn yields were reduced by as much as 34 percent during years with excessive rainfall [34]. It was estimated that between 1989 and 2016, intense rain events caused $10 billion in agricultural loss [34].

The effects of climate change has a large impact on rainfall [23]. It was cited as a reason for the increased and unpredictable rainfall [23]. Rainfall is highly regional, so climate change is a great cause of concern for rainfall in the future as farmers will need to plan for excessive or shortages in rainfall which will affect the profitability of the farm. To model rainfall we utilize a normal distribution of historical annual precipitation and calculate the mean and standard deviation for the region of the farm area being studied. To anticipate how precipitation affects the performance of the farm, we built linear regression models that correlate historical precipitation measurements with historical crop yield to represent the effect of precipitation on crop yield. To set the range we use the empirical rule which states 99.7 percent of values lie above and below three Standard Deviations (SD) of the mean [35].

Uncertainty Variable	Notation	Lower Bound	Upper Bound
Climate Complexity	C	0	3
Rainfall	R	0	3*Expected Rainfall (user input)

Table 3: Uncertainties Problem Formulation

5 TRADESPACE OBJECT

The Tradespace Object is produced by the Decision Module and represents an instance of the specific farm setup defined by the Configuration File, including basic setup information such as number of decisions, price level, and a network representation of the tradespace. The initialization of the Tradespace object is invoked by the 'Generate Tradespace' function from the User Interface, which must be execute prior to any other action.

5.1 Tradespace Configuration File

The Tradespace Configuration File (TCF) is a JavaScript Object Notation (JSON) file that describes a set of architectural decisions in a digital farm system (e.g. farm sensors, data storage method, plant watering physiological model, etc.). Each decision item describes the decision type, decision weight (importance), and a range of implementation options (alternatives) with detailed attributes and measurement information. The TCF defines the basic elements and structure of the tradespace, and the Decision Module processes the extracted data into a Tradespace Object for downstream analytics.

5.2 Network Structure

Building on the TCF, the Tradespace Network (TSN) data representation consists of three layers of data manipulations: decision pool, policy pool, and tradespace nodes. The decision pool is a list structure data set generated from the TCF by the function 'make_decision_pool'. Each element in decision pool is a dictionary that stores information of a decision and a list of alternatives instantiated as decision objects. The decision pool represents all the decisions available in the tradespace. The policy pool is a list structure dataset generated by the enumeration function (subsection 5.3), where each element in the policy pool is a list of decision objects. The policy pool represents all possible policies that can be formed and validated by the information and rules defined in the TCF. The tradespace nodes is a list structure dataset generated by mapping the 'make_node' function to each element of the policy pool, where the 'make_node' function calculates metadata for each policy in the policy pool and produce a node object. Each node

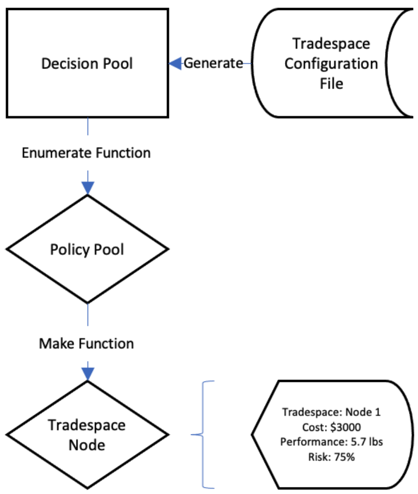

Figure 3: The layers of the Tradespace Network (TSN) structure is represented here.

object contains information describing the corresponding policy, including the metrics, length of the policy, and a link to the policy object in policy pool.

5.3 Enumeration and Optimization

In the initial stage of the tradespace exploration, an enumeration of all possible solution states without uncertainty in the tradespace is generated using the TCF (subsection 5.1) inputted by the user. Then, the Pareto front of the generated tradespace is calculated through an exhaustive search of the enumeration. This frontier represents a set of equally optimal SDF architectural decisions (policies) for the user-defined farm environment, without considering the effect of any uncertainty variables. The Pareto front at this stage presents a basic understanding of the differences and tradeoffs between various policies, which are further analyzed in the following stages with the added effects of uncertainties in the farming environment.

Due to the large size (often sized in millions) of the tradespace and its deterministic nature when calculating without uncertainty, we created the Enumeration and Optimization Algorithm in the Decision module. This algorithm is used to generate all possible valid combinations of architectural decisions (policy) and then calculate the Pareto fronts in the tradespace, without considering the influence of uncertainties. A policy is a set of alternatives defined by the TCF. A policy is valid only if its composition satisfies the rules defined in section 4. Under these rules it can usually consist of a variable number of alternatives, but these alternatives should be selected from a fixed number of decisions and in ways regulated by the type of each decision. The ‘enumerateTS’ function generates a set of all possible policies that exists in the tradespace by first calculating a combination/permutation of the alternatives under each decision depending on the decision type, denoted as ‘comb’ and then calculating combinations of all comb. After a set of all possible policies in the tradespace is found, each policy in this set is then processed into a node, a data structure defined in subsection 5.2. Here, a node represents an instance of a possible policy, with a collection of metadata (performance, cost, risk metrics, and policy length) used for downstream calculations and visualizations.

After the ‘enumerateTS’ function completes, the ‘calcPareto’ function can be invoked to calculate the tradespace Pareto front without considering an uncertain environment. In this case, a Pareto front represents a Pareto optimal set of policies calculated by multi-objective optimization based on the performance, cost, and risk metrics detailed in subsection 4.3, which means every policy in this set is equivalently optimal and no metric can be enhanced by any one alternative decision without compromising at least one other metric. The calculation of the Pareto front is implemented by extending the functions from the Orthogonal Array package. The set of policies found by ‘enumerateTS’, processed and represented as an array of nodes, is then passed into the extended data structures and functions to calculate the Pareto front based on the objectives to maximize performance and minimize cost and risk. The implementation of this algorithm can be found in our code repository with the file titled ‘tradespace_explore.py.’ Note that with the custom enumeration and optimization functions and the decision data structure described above, the resulting Pareto front

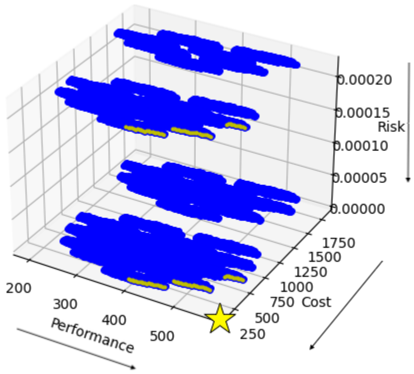

Figure 4: The tradespace enumeration optimizes toward the yellow star (i.e. the ideal state) and displays policies as blue points with those that are Pareto optimal as yellow.

can be traced to each ‘policy’ and ‘decision’ endpoints. This traceability is important in the integration of Rhodium models as well as in extending new functionalities and visualization features in the future.

Using an example TCF from section 2 with an input of 10 decisions and 29 corresponding alternatives (5 standard form decisions with 3 alternatives each, 1 standard form decision with 2 alternatives, and 4 down selecting decisions with 8 alternatives each), the enumeration algorithm yields a result of 1,166,886 policies. Using the Enumeration and Optimization Algorithm can determine a Pareto optimal set of size 142 optimal values and corresponding 284 optimal policies, by searching for policies that increase performance, decrease cost, and decrease risk. It is important to note here that various policies can result in the same optimal values. The graph in Figure 4 visualizes the process of optimizing toward an ideal point as depicted by a gold star.

6 MORDM USING RHODIUM

In creating ROAM, we leverage functionalities provided by the open-source Python library Rhodium to accomplish Many Objective Robust Decision Making (MORDM) for our system, especially in exploring and analyzing the system’s performance in an uncertain environment (subsection 4.3) [20][3]. Robust Decision Making (RDM) is an analytic framework developed by Robert Lempert and his collaborators at RAND Corporation that helps identify potential robust strategies for a particular problem, characterize the vulnerabilities of such strategies, and evaluate trade-offs among them [25]. MORDM is an extension of RDM to account for problems with multiple competing performance objectives, enabling the exploration of performance tradeoffs with respect to robustness [13]. We use the Multi-Objective Evolutionary Algorithm (MOEA) provided by Rhodium to optimize the Pareto set of ‘policies’ calculated in the Decision Module (subsection 5.3) under a representative or average instance of the uncertain environment (State-Of-World, or SOW). Each representative instance is taken by examining a distribution and utilizing the average. The Pareto efficient policies are further explored using the uncertainty analysis functions provided by Rhodium. Finally, the sensitivity analysis provided by SALib python library [13] is used to analyze and categorize the effect of different uncertain elements in the farming environment.

6.1 Optimization

The optimization function in Rhodium Module is a MOEA that utilizes the NSGA-II algorithm provided by the Rhodium library [3]. This function is used to find the Pareto optimal set of ‘policies’ based on the performance, cost, and risk metrics ((subsection 4.2)). It is important to note that the optimization function in the Rhodium Module differs from the ‘calcPareto’ function in Decision Module mentioned in subsection 5.3 in that it takes the uncertainty parameters into account in order to prioritize policies that are robust. These two optimization methods serve different purposes in exploring the Tradespace. The ’calcPareto’ function in the Decision Module enumerates all possible policies solely based on the static decision configurations defined by the Tradespace Configuration File, which finds the initial optimal set of policies on paper based on prior knowledge about the decisions. The optimization function in the Rhodium Module iteratively adjusts the controlled parameters or combination of decisions as discussed in subsection 5.2 while searching for the optimal set of policies under the mean SOW. This function then finds the set of policies most optimal in this specific state of uncertainty model.

The optimal set of policies found by the optimization function in the Rhodium Module is a subset of the set of policies found earlier in the workflow by the 'calcPareto' function, as the former function uses outputs of the latter function as inputs. We choose to have the optimization function in the Rhodium Module only search through a subset, because the 'calcPareto' function helps remove less optimal policies from further examination during later steps. By limiting the scope of input to the optimization function in the Rhodium module, the amount of computation is substantially reduced and the user experience is enhanced through a shorter response time.

6.2 Scenario Discovery

The scenario discovery function, imported from the Rhodium library, is used to explore and analyze the influence from uncertainties in the Pareto optimal set of 'policies' that are found by the optimize function in the Rhodium module [3]. First, a set of uncertainty variables are defined using the parameters and distributions on the Uncertainty model (subsection 4.3). Then, a Rhodium internal function, 'sample_lhs,' is called to generate a standard 1000 SOWs through a Latin Hypercube Sample – a technique used to reflect the true underlying distribution on the uncertain parameters [3]. Each SOW consists of a combination of uncertainty variables and represents an instance of the uncertain environment. Then, the policy evaluation function is executed to evaluate each 'policy' in the Pareto optimal set on the 1000 SOWs. The results produced from scenario discovery can be used to visualize and explore different characteristics of various Pareto policies, such that policies demonstrate tradeoffs in metrics when evaluating against uncertainties. The analysis of these tradeoffs can provide us with insights into how different system architectures may be a better fit for certain scenarios (e.g. excessive rainfall) that causes a policy to fail and be vulnerable. These tradeoffs will be further explored and conclusions can be drawn through sensitivity analysis.

6.3 Sensitivity Analysis

The Rhodium library's internal implementation extended from Python's SALib is used to perform global and regional Sensitivity Analysis (SA) on modeled uncertainties which are performed to prioritize the factors (parameters) most significantly affecting the output and fix those that are not [3]. This functionality is enabled by the browser-end interface that will be described in section 7; here users can specify a 'metric' and 'policy' of their interest to investigate, then the SA function performs global SA using commonly used methods. First the Method of Morris is used to analyze which decisions are most influential to the output metrics and the effect of uncertainty variables in isolation [10]. Second, the Sobol method is used to calculate second-order and total-order indices for capturing the interactional effects between uncertainties [48]. The function can also perform one-at-a-time (OAT) or regional SA to explore each parameter in detail. In OAT SA, we fix all parameters at their default value except one [43]. For this one parameter, we then sample across its entire range and observe how the metric of interest changes.

6.4 Rhodium Implementation

The Rhodium package is used to help calculate the optimal policy of the system under uncertainty. The first step is to define the farm uncertainties and how they affect the architectural policy with the function "farm_approach". For example, with greater rainfall, yield may increase and watering costs may decrease. Importantly, we use the function "setupModel" to allow for user input through the web interface of what the average uncertainty value will be for their farm. Once these uncertainty parameters are set, we use the "optimizeModel" function to run 10,000 function evaluation calls of NSGAII to calculate the optimal policy in the uncertain state of the world. The output is the set of optimal policies and there associated policy name, subset of decisions, cost, performance, and risk.

7 USER INTERFACE: APPLICATION STRUCTURE AND FEATURES

The user interface is a dynamically created web-based interface using a Python Flask framework with HTML, CSS, and JavaScript. The User Interface includes 5 sections: About section, which provides an overview of the application and its functionalities; Parameter section, which takes in user input parameters to modify the Trade Space Model; Function section, which users can use to invoke different actions; Log section, which records the user action sequence; and Output section, which displays a series of results, data, and visualizations based on user actions.

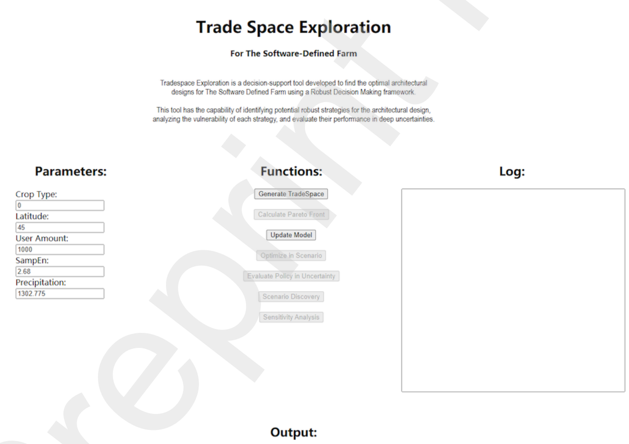

Figure 5: User Interface at its initial state, with no action invoked and no results generated

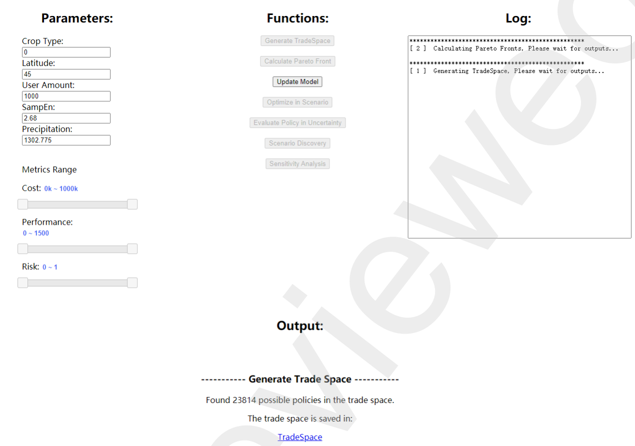

Figure 6: Shows the User Interface after the Generate Tradespace function runs

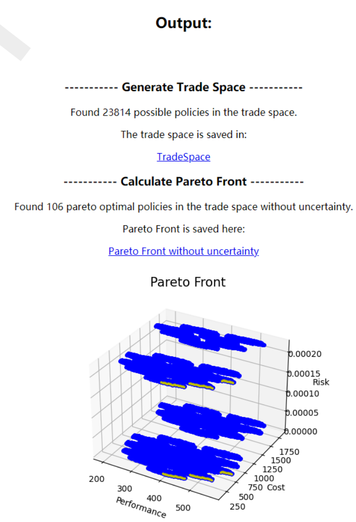

Figure 7: Workflow shows the Tradespace Exploration Tool workflow for web-end users

The Output section displays a brief summary of the results after execution of each function and provides the ability to download the results in a comma-separated values (CSV) format. This export gives users the ability to perform their own analysis. The Output section plots interactive 3-D visualizations, with performance, cost, risk metrics on each axis. These visualizations can be inspected through user actions, including drag, zoom in/out, click, and selection. The "Optimal Policies in Uncertain SOWs" visualization resulted from running "Evaluate Policy." In addition this visualization uses color, brightness, and size in the 3-D scatter plot to illustrate policy group and magnitudes of uncertainty.

The interface is designed with simplicity in mind. Ideally it allows can be used by any type of user from farmer to researcher. In addition, the graphs are used to visualize complex trade offs between different design decisions and user constraints that can be dynamically updated. Lastly, this design allows users the flexibility to pursue further modeling of the data.

8 EXAMPLE USE CASE

In this section, we provide an example use case of ROAM using a configuration developed by a farm owner client for his viticulture farm, Cheng Xin Garden LLC (CXG), located in Bakersfield, California. The capabilities and workflow of ROAM will be demonstrated through this use case.

8.1 Stakeholder Analysis

Due to limitations in California's water supply caused by frequent droughts and forest fires, CXG was seeking to increase their farm efficiency. Their wine grapes use a significant amount of water and often need a very precise amount. For example, the amount of water used was highly correlated to the taste profile of the wine produced. As a result, precise levels of water irrigation are needed for water savings and to achieve the optimal grape taste.

To test our software CXG farms served as an ideal use case scenario for our system, where the decision maker of the farm hopes to improve the performance of their farming practices, but is constrained by the lack of knowledge on available technologies or the ability to envision the results from adopting a SDF. By helping the farm owners translate their insights about their farms as well as their requirements into a TCF, we can use ROAM to provide crucial information and suggestions to support their decision making.

After weeks of interviews with relevant stakeholders, we holistically understood the current situation, needs, and challenges of CXG's farming practices and created a configuration for their viticulture farm, which is a 120-acre farm area growing wine grapes. By conducting analysis of the farm's environment, management, labor, and technology use, we learned that one of the major challenges they face on the farm was water management, similar to that of many Californian vineyards. Due to the hot desert-like climate with frequent droughts in Bakersfield as well as the need for irrigation for grape-growing, water usage was the largest factor in the operational cost, and precision irrigation is closely associated with yield quality. Through this process, the farmer shared his data that he has been collecting for over 6 years. Hence, CXG's decision space was constructed with an emphasis on improving the farm's production performance through optimizing water usage, labor size, and cost. A set of decision alternatives are selected for each decision based on the availability and compatibility of the technologies as well as specific needs addressed by the farm owner. So we understand how each decision and alternative affects the farmer and the different interaction effects, for example in the case of CXG a manual water tensiometer saves them 20% of water usage and their cost of water is \$100 per day in California, which can vary from \$50-\$200 per day [4]. The resulting decision space is then translated into a Configuration File format and inputted into ROAM for further decision support.

8.2 Generate Tradespace

The first step of using ROAM was to identify all of CXG's decision points to create the TCF that represents the needs and constraints of their farm environment. The TCF was created from a JSON skeleton provided by ROAM, which consists of a list of decision structures as detailed in section 5. Table 4 shows the various decision points we identified and encoded in the TCF.

#	Decision Name	Description	Alternative 1	Alternative 2	Alternative 3	Class	Importance
1	Water Tensiometer	The methods to collect water stress data	Manual Sampling	Glass	Digital Sensor	SF	1
2	Environment Humidity Sensor	The methods to collect humidity and temperature	Manual Sampling	Digital Sensor	N/A	SF	1
2	Microcontroller	The devices put in the agriculture field	FarmBeats	CR6 datalogger	Arduino	SF	.75
4	Data Storage	The type of storage for product information and user data	Raspberry pi	Cloud	N/A	SF	1
6	Plant watering Physiological model	The model for prediction or applications for analytics of water stress	Model Predictive Control (machine learning model)	On/off control (closed loop)	Scheduling (open loop)	SF	1
8	Irrigation Controller	How to water the plants	B-Hyve Smart Hose Watering Timer	Rachio	Raspberry pi	SF	1

Table 4: Configuration for Cheng Xin Garden LLC

After the configuration was imported, Generate Tradespace initiates the Tradespace Exploration workflow by generating the Tradespace Network (subsection 5.2) and enumerating all possible policies that can be constructed based on the given configuration. In the unconstrained architecture space, there are 6 SF decisions with 3 alternatives each and 2 SF decisions with 2 alternatives. 324 possible decisions are found in this tradespace, and users are provided with an option to download the tradespace enumeration in a CSV format, as shown in Figure 8.

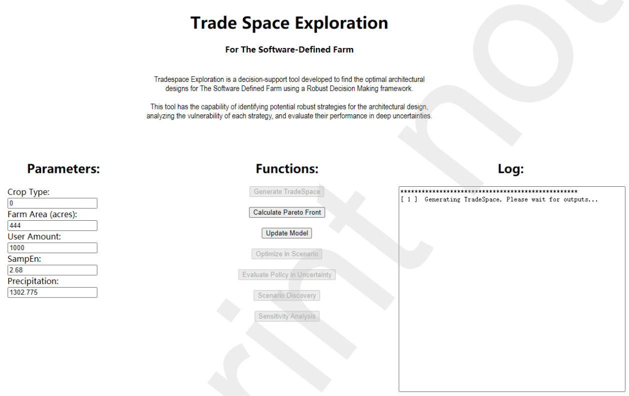

Figure 8: Generate Tradespace

Calculate Pareto Front computes the Pareto optimal set of policies in the architectural space without considering uncertain factors in the farming environment. Using the optimization algorithm detailed in subsection 5.3, 18 Pareto optimal policies are found in the tradespace for Cheng Xin Garden's configuration and can be exported in a CSV format. These 18 policies are a significantly smaller set to proceed with for further analysis in MORDN where we mine for the Pareto optimal set of policies to analyze decision tradeoffs. In Figure 9, the entire tradespace is visualized on a three-dimensional plot, where each axis represents one of the metrics, and the optimal set of policies is highlighted to display their relation to the tradespace. In Figure 10, an interactive visualization of the Pareto front allows users to inspect the plot from different perspectives and select policies to display further details.

8.3 Analysis in MORDM

Many-Object Robust Decision Making (MORDM) was then used to decipher the policies' performance, cost, and risk of CXG under simulated uncertain environments. To initiate analysis using MORDM, CXG inputted additional constraints and specifications on the tradespace. This allowed CXG to narrow the scope of analysis by specifying the key parameters of their farm region, including the type of crop grown, the area of the farm, the estimated climate complexity, and the average rainfall level. CXG was then also able to specify a set of constraints on the metrics to define the ideal tradeoffs for their farm. Finally, a metric range is used to classify the policies of interest and identify the key uncertainties for later

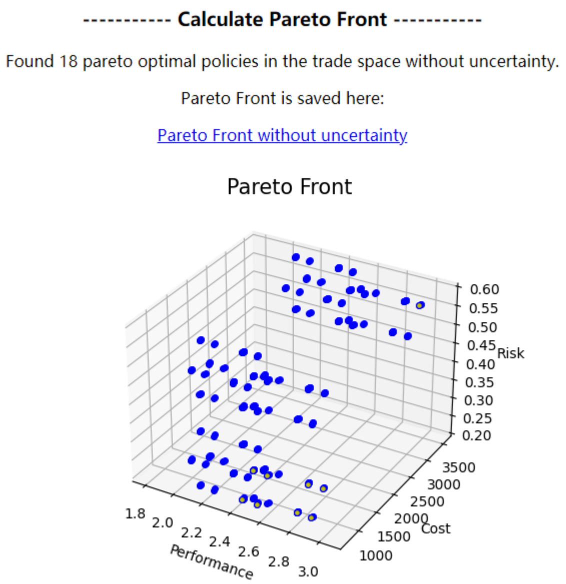

Figure 9: Visualization of the Pareto front and enumeration of trade space

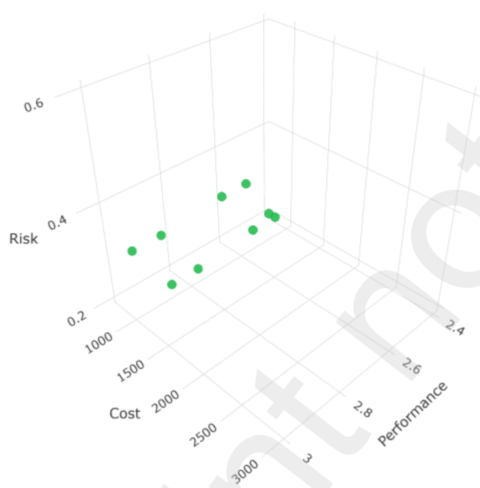

Figure 10: Visualization of the Pareto front

analysis. In Figure 11, the parameters are set to represent the attributes of CXG and metric ranges are set to the farm owner's ideal tradeoff.

With the defined parameters, the software used a many-objective optimization algorithm to calculate the optimal set of policies among the tradespace Pareto front, at the mean state-of-the-world

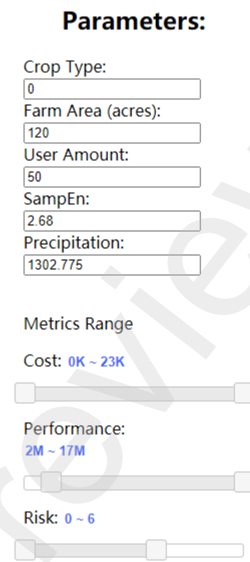

Figure 11: Parameter inputs for the MORDM model

described by the parameters in Figure 11. The new set of policies found by the algorithm are the optimal policies after accounting for the effects of uncertainties, modeled through stakeholder analysis and research detailed in subsection 4.3. As shown in Figure 12, there are 9 optimal policies found among the tradespace Pareto front of 18 policies, which demonstrates a significant reduction of the range of policies that we needed to examine.

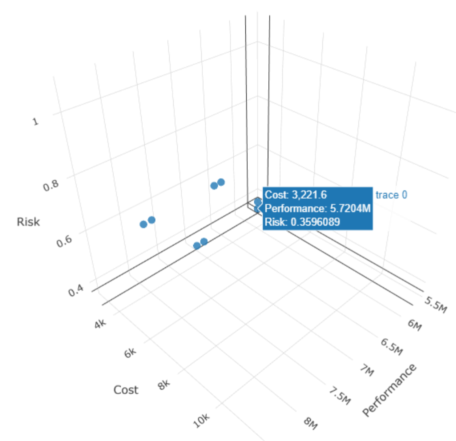

Figure 12: Optimal set of policies under mean state-of-world

With the optimal set of policies on the mean state-of-the-world identified, the software performs analysis of each Pareto policy through Scenario Discovery under more robust uncertainties. A set of 1000 states-of-the-world are generated based on the distribution defined for every uncertainty variable. The set of optimal policies are evaluated in the set of 1000 SOWs to reflect each policy's characteristics and vulnerabilities under uncertainty. Consider that each policy consists of various numbers of different decision alternatives, making each policy uniquely exist in the trade space. As uncertainties may affect decisions differently, policies with similar metrics in appearance may demonstrate distinct characteristics under uncertainty. The key objective of Scenario Discovery is to illustrate such distinction among the equivalently optimal set of policies to support further decision making.

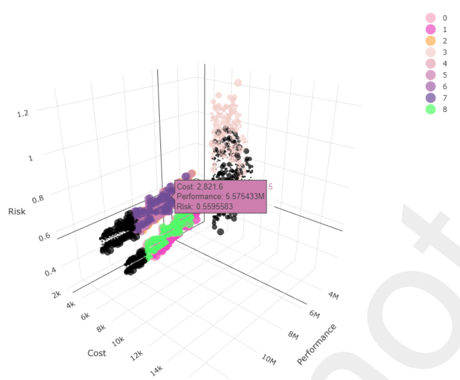

Figure 13: Visualization of optimal decisions in Scenario Discovery

8.4 Results

Based on the results generated from the Tradespace Exploration, we are able to zoom in onto 9 policies out of 324 possible policies. According to the farm owner of CXG, he placed more weight on improvements of performance than the cost of the system in the tradeoff between performance and cost, and he has a relatively high tolerance for risk on his farm (Figure 11). The decision maker's preference lead us to only consider Policy 5 and Policy 6. These policies demonstrate a similar tradeoff between risk and cost, but with a small difference in their performances, as shown in Figure 14. Then, with the information provided by Scenario Discovery, we learn that Policy 5 is more likely to perform within the decision maker's preferences than Policy 6, as shown when comparing Figure 15 and Figure 16. Figure 16 shows points that fall outside of the accepted system performance as black. Such a difference is likely caused by the difference between policies' sensitivity to the labor cost and area of the farm. Since both of these factors are likely to vary during the operation of the farm, the difference in how the two policies perform under the uncertain environment are important to the evaluation. Hence, we recommended Policy 5 as the system setup for CXG under their reported circumstances. These ideas and results were conveyed to the farm owner who hopes to implement our recommendations in the future.

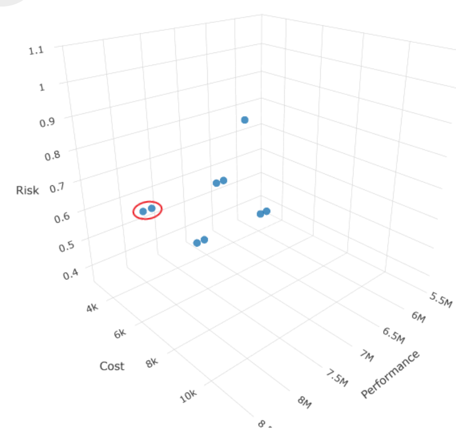

Figure 14: Policy 5 (right) and Policy 6 (left)

9 DISCUSSION

With ROAM, farmers can understand what a Pareto optimal set of choices for a farm of interest might be. The idea of creating a DA system is daunting due to the number of choices that must be made. In section 8 the farm owner had over 324 policies to consider. ROAM simplified the process and allowed the user to understand the tradeoffs when examining design decisions and to filter choices based

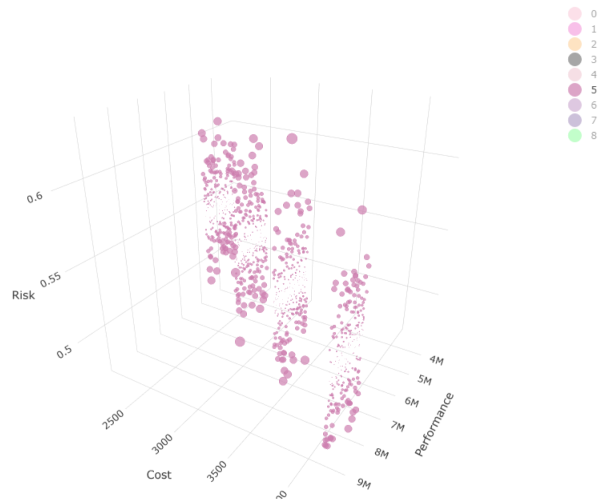

Figure 15: Policy 5 in Scenario Discovery

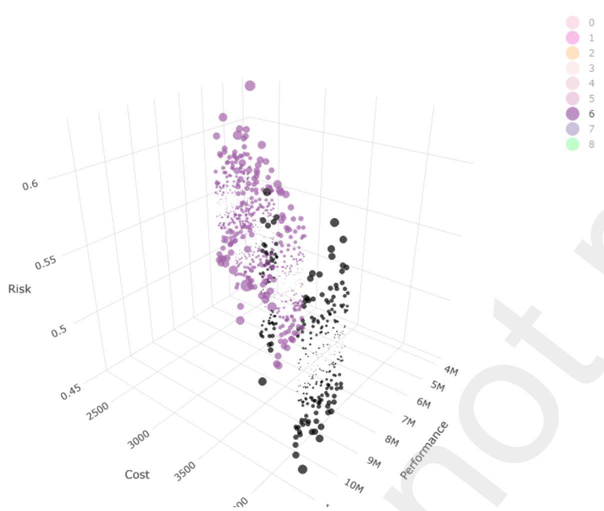

Figure 16: Policy 6 in Scenario Discovery

on their needs. ROAM presents Pareto optimal farm architectures based on performance, cost, and risk and climatic uncertainties. Further, ROAM is extensible, as the code is written in an object-oriented manner, and allows interchanging new parameters and analytical optimization models. ROAM users can conceptualize what a data-driven farm management system might look like based on their specific goals and farming environment.

To allow for ease of use for our target users, farm-owners, scientific researchers, industry professionals, and decision makers, we developed the browser-end interface to host the workflow of ROAM. Users use ROAM to generate interactive visualizations for communication and demonstrations with colleagues. Farm-owners and farm stakeholders specifically utilize the configuration file and input parameter features to customize and explore the decision space for their farms. ROAM's current implementation optimizes for cost, performance, and risk. For additional goals, an extension on the software and further data analysis must be implemented.

10 CONCLUSION

We presented the Realtime Optimization and Management System (ROAM). It is designed to identify the Pareto optimal set of tradeoffs for a Digital Agriculture (DA) based farm, where DA is seen as an approach to address the Global Agricultural Productivity (GAP) shortfall [50]. Specifically, DA enables data driven farm management, which requires on farm networking. A Software-Defined Farm (SDF) uses new networking on a farm to enable DA. Based on deploying five SDFs, 11 farmer interviews, and testing on a farm in California, ROAM is able to present Pareto optimal SDF architectures for a given farm area of interest. ROAM presents general recommendations as to how to best implement a SDF based off of data inputted by the user and climatic data.

11 MISCELLANEOUS

Software

Description: The Tradespace Exploration is a decision-support tool developed to find the optimal architectural design for the Software-Defined Farm using a Robust Decision Making framework. It identifies potential robust strategies for architectural design, analyzes each strategy's vulnerability, and evaluates their attributes under deeply uncertain farming environments. Paired with a browser-based application, it hosts the trade space exploration functionalities and interfaces for user interactions and data visualizations.

Software name: ROAM

Developers: Yifan Zhao, Shiang Chin

Language: Python 3.6+

Supported systems: Microsoft Windows, GNU/Linux, macOS

Licence: GNU General Public Licence v3

Source code: https://github.com/ShiangC/Cornell_SDF

Acknowledgments

This work is supported by the US National Science Foundation (NSF) under grants #1955125, NSF STC under grant #2019674, a Microsoft Investigator Fellowship, and the Cornell Institute for Digital Agriculture (CIDA). Shiang-Wan Chin is supported by the Foundation for Food and Agriculture Research (FFAR) and a Microsoft Cornell Summer Research Fellowship. The authors would like to thank Ranveer Chandra, Sachille Atapattu, Daniel Amir, Patrick Kastner, and Xinpi Du and the anonymous reviewers for their constructive feedback and insightful discussions during the manuscript's preparation. The authors appreciate the support from Microsoft partners, including Tusher Chakraborty, Elizabeth Bruce, and Stacey Wood. All opinions expressed are those of the researchers, not the NSF, Microsoft, or CIDA.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

[1] Thomas A. 2002. Seasonal and spatial variation of evapotranspiration in the mountains of Southwest China. *Journal of Mountain Science* (2002), 20(4):385–393.

[2] J.T. Abatzoglou. 2011. Development of gridded surface meteorological data for ecological applications and modelling. *International Journal of Climatology* (2011).

[3] David H. Patrick R. Antonia H., David G. 2020. Rhodium: Python Library for Many-Objective Robust Decision Making and Exploratory Modeling. *Journal of Open Research Software* (2020).

[4] Aquaoso. 2021. California Agricultural Water Price by Water District. (2021).

[5] Moscibroda T. Murty R. Bahl P., Chandra R. and M. Welsh. 2009. White Space Networking with Wi-Fi like Connectivity. *SIGCOMM 2009* (2009).

[6] World Bank. 2018. International Labor Organization. *ILOSTAT Database* (2018).

[7] S. Banks. 2002. Tools and techniques for developing policies for complex and uncertain systems. *Proceedings of the National Academy of Sciences of the United States of America* (2002), 3:7263–7266.

[8] Antle J. Basso, B. 2020. Digital agriculture to design sustainable agricultural systems. *Nature Sustainability* (2020).

[9] Lempert R. Bryant, B. 2010. Thinking inside the box: A participatory, computer-assisted approach to scenario discovery. *Technological Forecasting and Social Change* (2010), 77(1):34–49.

[10] Cariboni J. Saltelli A. Campolongo, F. 2007. An effective screening design for sensitivity analysis of large models. *Environmental Modelling Software* (2007).

[11] C. Chamberlin. 1890. The method of multiple working hypotheses. *Science* (1890), 15(366):92–96.

[12] C. Clímaco. 2004. A critical reflection on optimal decision. *European Journal of Operational Research* (2004), 153(2):506–516.

[13] Marks D. Cohon, J. 1975. A review and evaluation of multiobjective programming techniques. *Water Resources Research 11* (1975), 208–220.

[14] Pratap A. Agarwal S. Meyarivan T. Deb, K. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* (2002), 6(2):182–197.

[15] Marshak A. Delgado-Bonal, A. 2019. Approximate Entropy and Sample Entropy: A Comprehensive Tutorial. *Entropy* (2019).

[16] Marshak A. Delgado-Bonal, A. 2019. Approximate Entropy and Sample Entropy: A Comprehensive Tutorial. *Entropy* (2019).

[17] Jaskulska I. Jaskulska D. Szczepanczyk M. Galezewski, L. 2021. Analysis of the need for soil moisture, salinity and temperature sensing in agriculture: a case study in Poland. *Scientific Reports* (2021).

[18] Tanabut C. Gertphol S., Pariyanuj C. 2018. Predictive models for Lettuce quality from Internet of Things-based hydroponic farm. *2018 22nd International Computer Science and Engineering Conference* (2018).

[19] P. Gralla. 2018. Precision Agriculture Yields Higher Profits, Lower Risks. *Enterprise.nxt* (2018).

[20] Herman J. Reed P.M. Keller K Hadka, D. 2015. An Open Source Framework for Many-Objective Robust Decision Making. *Environmental Modelling & Software* (2015), 74:114–129.

[21] Reed P. Herman, J. 2015. How Should Robustness Be Defined for Water Systems Planning under Change? *Journal of Water Resources Planning and Management* (2015), 141(10).

[22] MacDonald J. Hoppe R. 2016. America's Diverse Family Farms. *Economic Information Bulletin* (2016), 16.

[23] Ngaira J. Ogindo H. Masayi N. Hunho, J. 2012. The changing rainfall pattern and the associated impacts on subsistence agriculture in Laikipia East District, Kenya. *Journal of Geography and Regional Planning* (2012).

[24] Schipanski M. Atwood L. Mortensen D. Hunter M., Smith R. 2017. Agriculture in 2050: Recalibrating Targets for Sustainable Intensification. *Management Science* (2017), 386–391.

[25] S. Nataraj P. M. Reed Kasprzyk, J. R. and R. J. Lempert. 2013. Many objective robust decision making for complex environmental systems undergoing change. *Environmental Modelling & Software* (2013), 42:55–71.

[26] Thessler S. Koskiaho J. Hannukkala A. Huitu H. Huttula T. Havento J. Jarvenpaa M. Kotamaki, N. 2009. Wireless in-situ Sensor Network for Agriculture and Water Monitoring on a River Basin Scale in Southern Finland: Evaluation from a Data User's Perspective. *Sensors* (2009), 9(4), 2862–2883.

[27] Pruyt E. Kwakkel, H. 2013. Exploratory modeling and analysis, an approach for model-based foresight under deep uncertainty. *Technology Forecasting Social Change* (2013), 80(3):419–431.

[28] Sasloglou K. Goh H. Wu T. Stephen B. Gilroy M. Tachtatzis C. Glover I. Michie C. Andonovic I. Kwong, K. 2009. Adaptation of wireless sensor network for farming industries. *2009 Sixth International Conference on Networked Sensing Systems (INSS)* (2009).

[29] Griffin P. Moorman J. Lake D., Richman J. 2002. Sample entropy analysis of neonatal heart rate variability. *American Journal of Physiology-Regulatory Integrative and Comparative Physiology* (2002), 283: R789–R797.

[30] Kirshen H. Vogel M. Lane, E. 1999. Indicators of impacts of global climate change on U.S. water resources. *Journal of Water Resource Planning and Management* (1999), 4(194):194–204.

[31] Cosgel M. Langlois, N. 1993. Frank Knight on risk, uncertainty, and the firm: A new interpretation. *Economic Inquiry* (1993), 32(3):456–465.

[32] Groves D. Popper S. Bankes S. Lempert, R. 2006. A General, Analytic Method for Generat- ing Robust Strategies and Narrative Scenarios. *Management Science* (2006), 52(4):514–528.

[33] J. Lempert. 2002. A new decision sciences for complex systems. *Proceedings of the National Academy of Sciences of the United States of America* (2002), 99(3):7309–7313.

[34] Guan K. Schnitkey G. DeLucia E. Peng B. Li, Y. 2019. Excessive rainfall leads to maize yield loss of a comparable magnitude to extreme drought in the United States. *Global Change Biology* (2019).

[35] Serlin R.C. Marasculio, L.A. 1988. Statistical methods for the social and behavioral sciences. *American Psychological Association* (1988).

[36] Gerber J. Johnston M. Ray D. Ramankutty N. Foley J. Mueller, N. 2012. Closing yield gaps through nutrient and water management. *Nature* (2012), 254–257.

[37] O'Hare G. O'Grady M. 2017. Modeling the smart farm. *Information Processing in Agriculture* (2017), 4(3):179–187.

[38] V. Pareto. 1896. Cours d'economie politique professe a l'Universite de Lausanne. *Travaux de Sciences Sociales* (1896).

[39] Raspberry Pi. 2022. Raspberry Pi 3 Model B. (2022).

[40] Reed P. Giuliani M. Castelletti A. Quinn, J. 2019. What Is Controlling Our Control Rules? Opening the Black Box of Multireservoir Operating Policies Using Time-Varying Sensitivity Analysis. *Water Resources Research* (2019), 55(7), 5962–5984.

[41] Chin S. Atapattu S. Rehman M. Jose M. Weatherspoon H. Rubambiza, G. 2022. Comosum: Design Experiences with An Extensible Hybrid Cloud Architecture for Digital Agriculture. *TBD* (2022).

[42] Sengers P. Weatherspoon H. Rubambiza, G. 2021. Paradoxes in Producing the Future of Farm Work: Anticipating Social Impact through the Lens of Early Adopters. *CHI* (2021).

[43] A. Saltelli. 2008. Global sensitivity Analysis. *The Primer* (2008).

[44] Lampietti J. Elabed G. Schroeder, K. 2021. What's Cooking : Digital Transformation of the Agrifood System. *Agriculture and Food Series, World Bank* (2021). https://openknowledge.worldbank.org/handle/10986/35216

[45] Cameron B. Crawley E. Selva, D. 2016. Patterns in System Architecture Decisions. *Systems Engineering* (2016), 19(6): 477–497.

[46] National Agricultural Statistics Service. 2017. Census of Agriculture. *United States Department of Agriculture* (2017).

[47] Qiaofu Z. Shaohong W. Erfu D Shuangcheng, L. 2006. Measurement of climate complexity using sample entropy. International Journal of Climatology. *International Journal of Climatology* (2006), 2131–2139.

[48] I.M. Sobol. 2001. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation* (2001).

[49] Rutten C.J. Oude Lansink A.G.J.M. Hogeveen H. Steeneveld, W. 2017. Why not investing in sensors is logical for dairy farmers. *Paper presented at 8th European Conference on Precision Livestock Farming* (2017).

[50] Zeiglerm M. Steensland, A. 2018. Global Agricultural Productivity Report. *Global Harvest Initiative* (2018), 1–2.

[51] B. Tekinderdogan. 2020. Systems Architecture Design Pattern Catalog for Developing Digital Twins. *Sensors* (2020), 20(18), 5103.

[52] J.S. Wallace. 2000. Increasing agricultural water use efficiency to meet future food production. *Agriculture, Ecosystems Environment* (2000).

[53] Singh R. Reed P. Keller K. Ward, V. 2015. Confronting tipping points: can multi-objective evolutionary algorithms discover pollution control tradeoffs given environmental thresholds? *Environmental Modeling Software* (2015), 73:27–43.

[54] Reed M. Simpson T. Woodruff, J. 2015. Many objective visual analytics: Rethinking the design of complex engineered systems. *Ecology and Society* (2015), 20(3):12.