Team Purple: Final Report

John Pothier Nate Harada Sam Zeckendorf

May 7, 2014

Abstract

We address the difficulty of indoor gardening, and its effects on urban and low-income areas. We propose a solution based on hydroponics, which leverages machine learning to grow healthy plants automatically without user expertise. We present a plant growth optimization algorithm based on collaboration between independent systems and continuous user feedback. We discuss our system architecture at a high level and our working prototype in detail. We present a mathematical model of plant growth for English Ivy, and discuss experimental results evaluating the efficacy of the optimization algorithm applied to this model. We identify the strengths and weaknesses of our approach, and propose avenues for future work.

1 Introduction

Gardening has found its place as an international pastime. With its own cable television channel, hundreds of shows, and over 65,000 gardening books available on Amazon, it is clear that gardening is viewed as a skill hobby, similar to playing the piano or woodworking. This viewpoint appeals to the millions of amateur gardeners who are constantly seeking to improving their skills, but what about the millions more who cannot dedicate the time or space to creating a successful garden?

Clearly individual skill is only one part of a successful garden, and for the estimated 250 million Americans who live in or near a city, space for plants is a premium few can afford. Hydroponics offers a solution for urban growers, but at the expense of a heavy domain knowledge requirement and an even higher penalty of failure. Hydroponic systems tend to be expensive, clunky, and require an intimate understanding of their systems in order to be effective for most plants. Most plants will die within a day when a soilless systems fails, and the more complex the system the less likely the user will be able to make repairs. In this paper we introduce *Seed*, a self contained, learning hydroponic system that we hope can solve these problems.

An individual *Seed* system represents one of many on a central server. This server collects plant health data from all surrounding systems and applies an online gradient descent algorithm in an attempt to optimize the plant health values for the entire population. The server then provides new growth parameters to each system, and the process repeats each time new data is provided by the user. We strive to provide a simple interface for both plant health ratings as well as monitoring for the user, while still maintaining powerful algorithms "under the hood". We have developed a prototype hardware system, as well as

a scalable, cloud based web framework which would support additional parameters and algorithms. We describe each component in detail, as well as provide experimental results for the Seed system based on several different hypotheses.

2 Theory

2.1 Motivation

Gardening requires a non-negligible amount of commitment from the average person. Watering must be scheduled daily for outdoor gardening, and weeding and transplanting must take place on a regular basis. Even for a homeowner it is easy to forget to water or weed with the hustle and bustle of modern life. For the city dweller who lacks outdoor garden space, the situation becomes even more difficult. Indoor gardening solutions generally tend towards hydroponics, as the lack of natural light and fertilizers make soil-based growing difficult to control. These hydroponics solutions require even more commitment from the grower — there is less margin for error and the gardener requires a fair amount of knowledge to maintain a system.

The option of growing one's own fruits and vegetables has become more attractive in the past decades. The increasingly geographically segregated economy and environmental concern associated with shipping food has generated considerable pushback against non-locally sourced food. Unfortunately, locally sourced food tends to be more expensive and be less available than mass produced produce. A hydroponic solution can lower prices for luxury and staple items, as well as allow for non-seasonal items.

In socioeconomically depressed urban areas, fresh fruits and vegetables fetch even more of a premium. These geographic areas are referred to as "food deserts", and occur when there is inadequate access to quality food, especially produce. Food deserts have been linked to diet related health problems in affected areas, especially for children and young adults who may not have the motivation or means to travel to a quality supermarket. Hydroponic systems such as Seed could offer these areas a way to easily grow their own produce, and if implemented in enough households could potentially provide an oasis of quality, healthy food to residents.

There has already been some work in self-maintained gardening systems [**?**] but in general the idea of networked optimization has not been explored. Additionally there has been work to develop algorithms capable of recognizing and rating plant's health quantiatively [**?**] that would be useful as a metric in our system.

2.2 High Level Design

At a high level, the *Seed* solution is viewed as a convex optimization problem. Each plant on the network represents one datapoint, with the user providing an oracle that is called once a day for each datum. The *Seed* hardware acts as an interface to mediate between the algorithm itself and the user. We assume that for any plant, there is an optimal set of parameters that will maximize a "health metric", or an objective function which measures the health of the plant. The basic procedure to grow plants is as follows:

1. Startup Phase
    (a) Transplant young plants to rockwool filled pots found in the *Seed* system.
    (b) Mix a nutrient solution using any hydroponic nutrient solution.
    (c) Add new systems to the *Seed* server.
2. Run Phase
    (a) Upload plant health data based on one of many possible plant health metrics.
    (b) Server runs an iteration of learning algorithm (gradient descent)
    (c) Each system will query server for parameters on a set interval

The hardware itself is based on an Arduino Uno microcontroller, while the software running the algorithm is a cloud-based Django server. The exact specifications for each component of *Seed* will be discussed later in detail.

2.3 Clustered Approximate Gradient Descent

Our learning algorithm is a distributed variant of gradient descent. In contrast to typical gradient descent, where you sample the objective function one point at a time and calculate the exact gradient of the function, here we sample at multiple points and approximate the gradient.

2.3.1 Problem Setup

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex cost function. At time t, the learner chooses N points (*feature vectors*) $\mathbf{x}_{i,t} \in \mathbb{R}^n$, and an oracle returns the values $f(\mathbf{x}_{1,t}), ..., f(\mathbf{x}_{N,t})$. That is, at each time step, the learner is allowed to sample f at N arbitrary points. The goal is to find the optimal point $\mathbf{x}^*$ such that $f(\mathbf{x}^*)$ is the global minimum of f.

2.3.2 Algorithm

The idea of the algorithm is to cluster $\mathbf{x}_{i,t}$ into k clusters, such that the points in each cluster are close enough together to closely approximate the gradient of f. Using the resulting collection of gradients of f, we can intelligently choose $\mathbf{x}_{i,t+1}$ such that the next collection of samples $f(\mathbf{x}_{i,t+1})$ are smaller than the current samples $f(\mathbf{x}_{i,t})$. Thus, eventually the sample collection will converge on the minimum of f.

Algorithm 1 Clustered Approximate Gradient Descent (CAGD)

input: N,M,α,k
set: $L \leftarrow 0$
Randomly choose initial sample points $\mathbf{x}_{i,0} \forall i \in \{1, ..., N\}$
for $t = 1, 2, 3, ..., M$ **do**
  Receive samples $f(\mathbf{x}_{1,t-1}), ..., f(\mathbf{x}_{N,t-1})$
  Set $S \leftarrow \{(\mathbf{x}_{i,t-1}[1], ..., \mathbf{x}_{i,t-1}[n], f(\mathbf{x}_{i,t-1})) \mid i = 1, ..., N\}$
  Cluster S into k clusters, $c_1, ..., c_k$
  for $\kappa = 1, 2, ..., k$ **do**
    Compute linear regression on $\{(\mathbf{x}, f(\mathbf{x})) \mid (\mathbf{x}, f(\mathbf{x})) \in c_\kappa\}$ to produce the coefficient vector $\mathbf{g}_\kappa$
    if $||\mathbf{g}_\kappa|| > L$ **then**
      $L \leftarrow ||\mathbf{g}_\kappa||$
    end if
  end for
  Set $\eta \leftarrow \frac{\alpha}{\sqrt{LM}}$
  for $\kappa = 1, 2, ..., k$ **do**
    Set $C_\kappa \leftarrow \{i \mid (\mathbf{x}_{i,t-1}, f(\mathbf{x}_{i,t-1})) \in c_\kappa\}$
    for $i \in C_\kappa$ **do**
      $\mathbf{x}_{i,t} \leftarrow \mathbf{x}_{i,t-1} - \eta \mathbf{g}_\kappa[i]\mathbf{1}$
    end for
  end for
end for

At every iteration, we first cluster the previous sample points, then for each cluster we compute the linear least squares fit to the locus of feature vectors and function samples. The vector $\mathbf{g}_\kappa$ contains the coefficients output by the linear regression that correspond to the non-constant terms of the best-fit hyperplane $\hat{f}_\kappa(\mathbf{x}) = c + \langle \mathbf{g}_\kappa, \mathbf{x} \rangle$–that is, $\mathbf{g}_\kappa$ is the gradient of the hyperplane, which approximates the gradient of f inside the cluster c_κ. After computing $\mathbf{g}_\kappa$, we perform a gradient descent step on all of the points in the cluster c_κ. In this sense, each step of CAGD is k simultaneous gradient descent problems. In the end we hope each $\mathbf{x}_{i,t}$ will converge to the same point $\mathbf{x}^*$. However, to find $\mathbf{x}^*$ it is only required that one of the clusters arrives at a stationary point, where $\mathbf{g}_\kappa \approx \mathbf{0}$. This will imply that cluster κ has "found" $\mathbf{x}^*$, and $\mathbf{x}^* \approx \bar{\mathbf{c}}_\kappa$, where $\bar{\mathbf{c}}_\kappa$ is the centroid of $\{\mathbf{x} \mid (\mathbf{x}, f(\mathbf{x})) \in c_\kappa\}$.

3 Project Design

3.1 Overview

The Seed system functions atop an abstraction of three subsystems: A centralized server, individual controllable units, and the functional network of every Seed system in tandem. The end-user provides feedback to the system between individual units and the server through a web front-end or an iPhone application, by rating the health of the individual units and providing raw data.The relationship between these subsystems can be observed in Figure 1.

The centralized server back-end is composed of three separate subsystems as well: An HTTP server designed to handle and serve requests to plants and

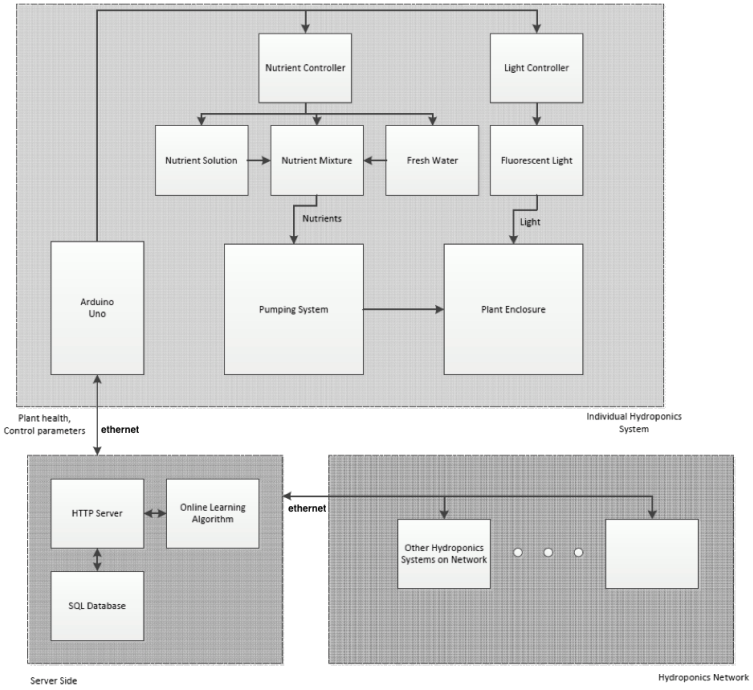

Figure 1: Systems Overview

users, a PostgreSQL database used to store plant and user information, and the learning algorithm, designed to process database information and provide the HTTP server meaningful control states for individual plant systems. The HTTP server is written in Django, a web framework in the Python programming language that provided the core RESTful[1] framework for requests and responses. This framework makes use of the MVC[2], a powerful abstraction that lets a controller (the Seed Django application) provide unique views of data models depending on the source or type of request. Users make requests to the server for plant information, which is provided in views visually through charts and graphs. Plant units make automated requests to the server to update control parameters; the server responds with a control state view, the binary operation of nutrient-rich water flow and light system, following a computed duty cycle. The PostgreSQL database is used to store the data models, which capture current and past information about plants, as well as basic data about their owners. Using this database, the Django application can serve views of processed data, computed by the learning algorithm. The server queries the algorithm for updated parameters, which are in turn delivered to respective plant units.

Seed control systems are electromechanical schemes that control water and nutrient flow, as well as full-spectrum light delivery, to individual plants. These systems are composed of a web-enabled microcontroller, water and air pumps,

[1]Representational State Transfer: web-architecture constraints within hypermedia systems.
[2]Model View Controller: Common server side architecture for serving data.

and an electrically controlled light. The microcontroller, in this case an Arduino UNO, queries the server for the most recent control states in binary strings:$\{\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3\ldots\}$ where each bit is an On/Off state for nutrients, light, or any future parameters. These control states are actualized through the use of mechanical relays, enabling and disabling AC power to pumps and lighting systems. The circuit can be observed in Figure 2.

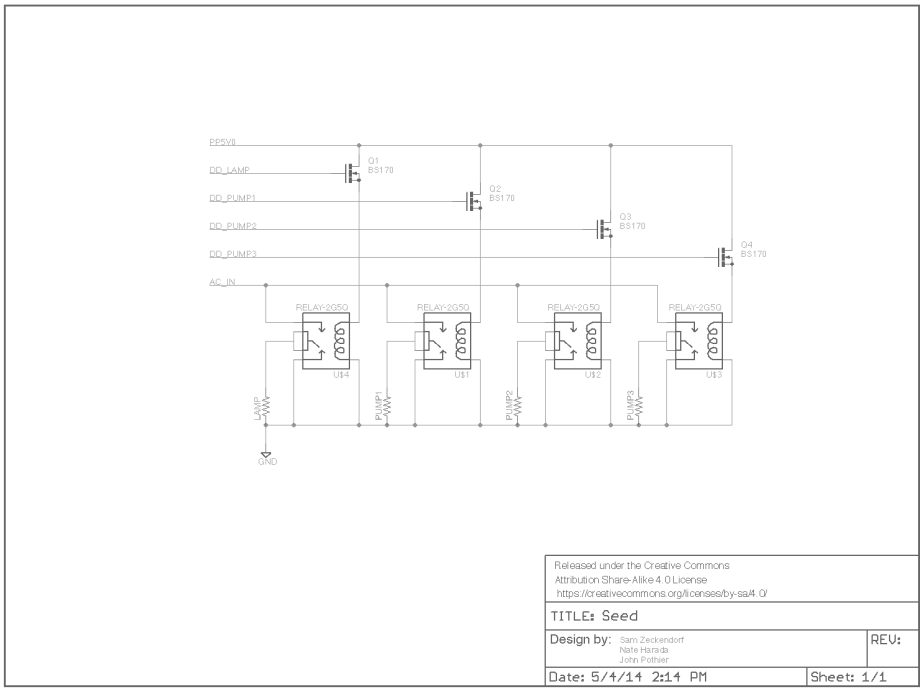

Figure 2: Controls Schematic

The cluster of networked systems acts as a separate subsystem, providing feedback to the server that relates to a given individual plant through proximity in parameters – while this is currently wired into performance, information such as temperature, geography, and humidity could provide alternate methods of identifying similarity between units. The behavior of plants at large hypothetically allow for the server's algorithm to create statistically significant data for control states; as a layer of abstraction, the networked systems represent the "Big Data" behind Seed's learning aspect.

3.2 Prototype Design

Our prototype consists of one physical structure, which contains a plant enclosure and an eletronics enclosure. See Fig. 3.

The plant enclosure contains four English Ivy plants, each in its own pot with rockwool. Each plant has its own water pump, which sits in a bucket full of nutrient solution and pumps nutrients through tubing up through the lid of the bucket and down through the rockwool. This setup is referred to as a hydroponic drip system. In addition, there is a full-spectrum fluorescent light

(a) (b)

Figure 3: System with (a) open and (b) closed cover.

above, and an air pump that feeds into the bucket that airates the water. There is also a removable shield that prevents the bright light from bothering people in the room.

The electronics enclosure contains one Arduino Uno microcontoller, which controls the power to the water pumps and light, and the control circuit as described in Fig. 2. The Arduino is connected via an Ethernet cable to the internet, allowing it to query our server periodically to receive control parameter updates.

4 Method of Solution

The core of Seed is the dynamic control of the hardware subsystem (water pumps, lights) via online learning. After constructing our prototype Seed system, we tested the viability of optimizing the health of a plant using a machine learning algorithm. Our investigation consisted of a set of experiments conducted over the course of a month. By analyzing the data collected from these experiments, we were be able to get an idea of how well the proposed plant-health optimization framework performs in practice, and what the limitations of our system are.

4.1 Experimental Purpose

The aim of these experiments was to study how plant health reacts to different levels of nutrient solution, and to assess whether plant health is able to be optimized by varying nutrient solution dosage over time.

4.2 Experimental Setup

A total of five English Ivy plants were analyzed in our experiments: four grown hydroponically (*hydroponic plants*), and one grown in a pot and soil at a window sill (*control plant*). The experimental plants were further divided into two groups: plants that did not have their nutrient level varied (*control hydroponic plants*), and those that had their nutrient level varied by the learning algorithm (*experimental plants*). All plants were bought as mature adults, all living in the same soiled pot. The plants were separated by removing the soil from the pot and disentangling the root systems, producing five independent plants. The experimental plants were placed into our system, and the control plant was repotted using the original soil. The hydroponic plants were named *plant0*, *plant1*, *plant2*, and *plant3*, and will henceforth be referred to by these names.

4.3 Experimental Procedure

The nutrient level for the experimental plants was varied via pulse-width modulation (PWM) of the water pumps. Every T_{PWM} Mseconds, all four water pumps were turned on. For plant i, the i^{th} water pump was turned off $n_i T_{PWM}$ seconds after being turned on, where $n_i \in [0, 1]$ is defined as the nutrient level for plant i. Thus, n_i controls how long nutrient solution is delivered to plant i during each PWM cycle, so varying n_i will vary the total amount of water and nutrients delivered to plant i. The light cycle for experimental plants was fixed as a simple 12-hours on, 12-hours off schedule. Every day, the hydroponic light turned on at 8 a.m. and turned off at 8 p.m. The nutrient solution was comprised of MaxiGrow nutrient powder dissolved tap water. Every week, the solution was drained and refilled with 3 gallons of water mixed with 6 teaspoons of MaxiGrow.

The control plant was placed at a window sill on the second story of a building. The plant was watered every other day, to the point where the top of the soil was slightly damp. The plant received approximately 10 hours of sunlight daily through the window.

Every T days, each plant was evaluated on performance using two metrics: leaf color (p_{color}) and leaf growth rate (p_{growth}). See the succeeding sections for detailed descriptions. For each experiment, we designated one performance metric, $p_{opt} \in \{p_{color}, p_{growth}\}$, as the target of our optimization algorithm. That is, p_{opt} are samples of the objective function for the online learning algorithm (see section 4.5). The measured values for both p_{color} and p_{growth} were recorded in a spreadsheet, and p_{opt} was also recorded in the database for processing. After the p_{opt} values were stored, the learning algorithm output new values of n_i for each experimental plant. This was the way the system "learned" from performance of its constituent plants.

4.4 Performance Measurement

4.4.1 Leaf Color

As described in [**?**], the color of a plant's leaves is related to the amount of chlorophyll the leaves contain. Generally speaking, greener leaves have more chlorophyll. Therefore, one way to quantify the health of a plant is to measure the average wavelength of light λ_{avg} reflected by the leaves and compare to the wavelength of pure green light $\lambda_{green} = 495\ nm$. [3] Since the leaves of our target plant typically reflect light at greater than 500 nm, we can simply measure how small λ_{avg} is in order to assess how close it is to λ_{green}. In particular, the Leaf Color performance metric is given by $p_{color} = \frac{1}{\lambda_{avg}}$.

To calculate $\lambda_{avg,i}$ for a particular plant i, we chose one representative leaf l_i for color measurement [4], scanned it using a portable color scanner, and imported the image onto a Windows computer. The JPEG image from the scanner was cropped to isolate the leaf, then imported into MATLAB® and processed to convert each RGB pixel in the cropped image into a wavelength in nanometers. The algorithm for this conversion can be found in [**?**]. λ_{avg} was set as the average converted wavelength value over all pixels.

4.4.2 Leaf Growth Rate

The Leaf Growth Rate metric is simply the number of leaves the plant grew since the last time step. That is, $p_{growth} = L_t - L_{t-1}$, where L_τ is the number of leaves on the plant at time step τ. Note that since a plant may lose leaves, p_{growth} can be negative.

4.5 Learning Algorithm Parametrization

As mentioned previously, we aimed to optimize each experimental plant's performance, $p_{opt,i}$, by varying n_i. In the context of the gradient descent algorithm discussed in section 2.3, n_i are the feature vectors $\mathbf{x}_{i,t}$ and $p_{opt,i} = -f(n_i)$[5], where f is the theoretical performance function that models a plant's health as a function of its nutrient level.

4.6 Experiments

4.6.1 Experiment 1

In Experiment 1, we set $T_{PWM} = 7200$ (2 hours), $T = 1$, and $p_{opt} = p_{color}$. We designated *plant0* and *plant3* as the control hydroponic plants, with $n_0 = 0.125$ and $n_3 = 1$ (corresponding to 3 hours and 24 hours of daily nutrient solution volume per day). The experimental plants, *plant1* and *plant2*, were given initial values $n_1 = 0.479$ and $n_2 = 0.521$. In the gradient descent algorithm, we set $M = 14$ and $\alpha = 10$.

[3] By "pure green" we mean the color with the smallest wavelength that is widely considered a shade of green.

[4] The best measurement of $\lambda_{avg,i}$ would take the average wavelength over all the leaves on the plant, but this would be extremely tedious. Also note that l_i was the same for every time step t, so that for each plant the same leaf was scanned over the course of all experiments.

[5] The minus sign is to indicate that we are trying to *maximize* p_{opt}, but the general algorithm aims to *minimize* the objective function.

4.6.2 Experiment 2

In Experiment 1, we set $T_{PWM} = 7200$, $T = 3$, and $p_{opt} = p_{growth}$. We again designated *plant0* and *plant3* as the control hydroponic plants, with $n_0 = 0.125$ and $n_3 = 1$. The experimental plants, *plant1* and *plant2*, were given initial values $n_1 = 0.458$ and $n_2 = 0.542$. In the gradient descent algorithm, we set $M = 14$ and $\alpha = 0.1$. We had to stop updating n_1 and n_2 at $t = 4$, since further updates would have driven these values to zero.

5 Results

All numerical data collected is presented below in Fig. 4 and 5.

5.1 Experiment 1

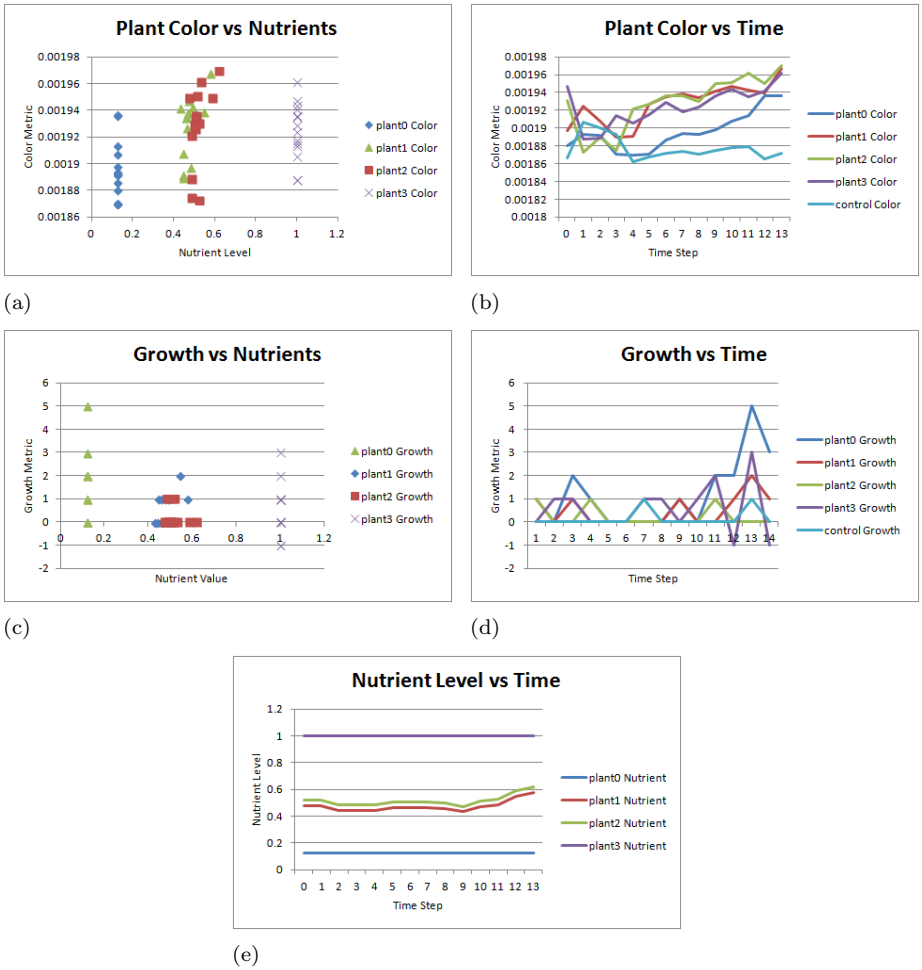

Figure 4: Results from experiment 1 with a timescale of approximately two weeks.

5.2 Experiment 2

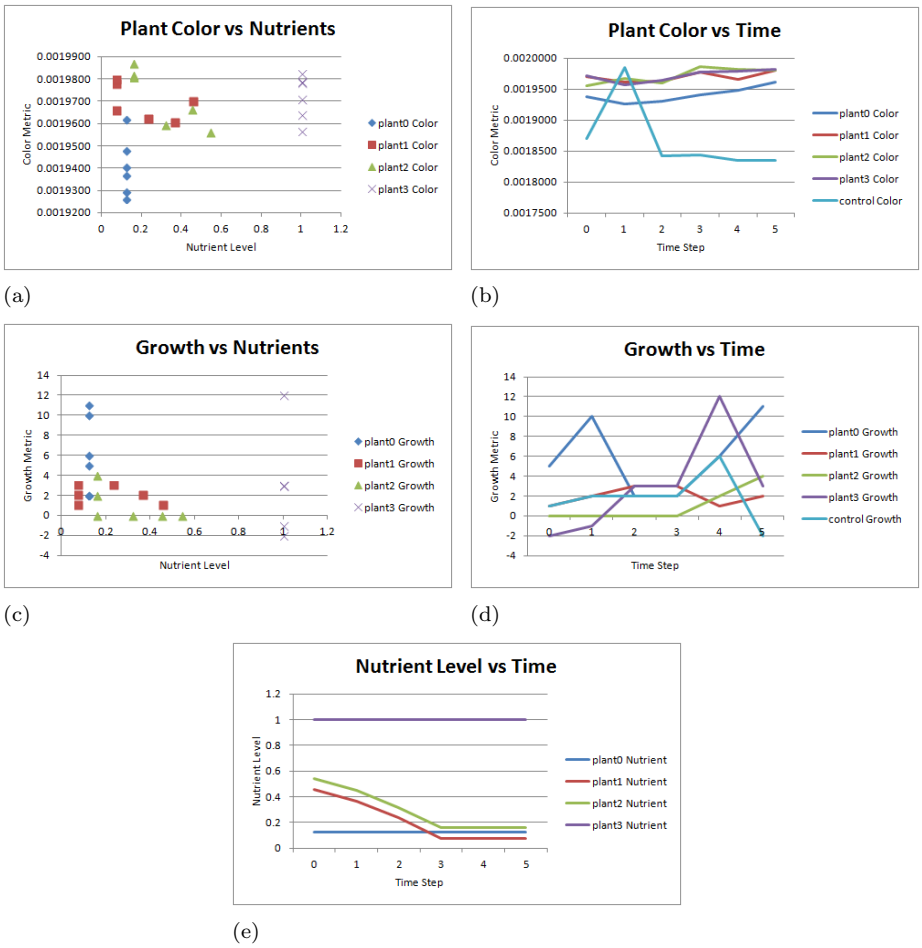

Figure 5: Results from experiment 2 with a timescale of approximately two weeks.

In addition to this data, we also took pictures of each plant every day to visually document growth. The initial and final pictures for the control plant and *plant0* are presented in Fig. 6 and 7.

6 Analysis

6.1 Experiment 1

In Experiment 1, it is clear that each hydroponic plant improved over time, for both p_{color} and p_{growth}. It would be tempting to interpret this as a success for our learning algorithm, but Fig. 4(e) reveals that the nutrient level wasn't varied significantly over time. The hydroponic plants all grew well with the amount of nutrients they were given. However, *plant0*, which received the least amount of nutrients, exhibited the largest amount of improvement for both metrics. On

(a) (b)

Figure 6: Initial (a) and final (b) of control plant

(a) (b)

Figure 7: Initial (a) and final (b) of *plant0*

the other hand, the control plant performed the worst for color, and among the worst for growth.

The performance vs. nutrient graphs (Fig. 4(a) and (c)) demonstrate a ma-

jor weakness in our model (see 7.3.1). We see many different performance values mapped to the same nutrient level. This is a result of the natural progression of leaf growth over time, independent of the precise nutrient level. It appears these plots do not tell us anything about the relationship between performance and nutrient level.

6.2 Experiment 2

Experiment 2 differed from Experiment 1 in two major ways. First we measured performance every three days instead of daily, to allow plants to properly react to changes in nutrient level. In addition, we used leaf growth as the target for optimization.

We see similar trends to Experiment 1 in terms of leaf color (Fig. 5(a) and (b)), but the remaining plots exhibit some important differences.

Unlike in Experiment 1, the nutrient level varied significantly over time (Fig. 5(e)), to the point where we had to manually stop the algorithm from updating to prevent nutrient level from approaching zero. The reason for this can be seen in Fig. 5(c), where most of the high-growth data lies near low nutrient values. The most notable result of the large nutrient variation can be seen in the trace for *plant2* in Fig. 5(d). Growth is static for several timesteps, but as the nutrient level is decreased, growth increases. However, *plant1*, which also had its nutrient value varied, did not exhibit such improvement.

7 Conclusions

7.1 Successes

7.1.1 System Functionality

Our biggest success was building a fully functional automated plant care system. The electronics and software controlling the water pumps, lights, and air pump of the hydroponics system work exactly as intended, allowing for remote control of all these components. In addition, we were able to grow four very healthy-looking English Ivy plants using our automated hydroponics system, which required very little attention to take care of. The control plant grown in the window sill was substantially less healthy than the hydroponic plants by the end–the leaves were turning yellow and falling off, while the hydroponic plants all had green, vibrant leaves (see Fig. 6 and 7).

7.1.2 Evidence of Health Improvement via Machine Learning

As discussed in section 6, there was evidence of long-term plant improvement resulting from the learning algorithm. In particular, in Experiment 2, for the first three time steps *plant2* grew no new leaves. However, after n_3 was algorithmically lowered to a small value (0.16), it began to grow new leaves. We stopped collecting data after time step 5, but visually it is clear *plant2* has continued to improve since the end of the experiment. By observing the data for *plant0*, which received a constantly small amount of nutrients, we conjecture that English Ivy responds well to a small amount of water and nutrients. We can say our algorithm "learned" this fact by observing *plant1* and *plant2* and examining

the relationship between their leaf growth $p_{growth,1}, p_{growth,2}$ and nutrient levels n_1, n_2.

7.2 Failures

7.2.1 Noisy Data

Although there are some patterns apparent in our results (e.g. Fig 4 shows average increase in p_{color} over time), there is obviously a lot of noise in performance measurements. Notably, the performance vs. nutrient level plots look meaningless, and it is clear that nutrient level is not the only independence variable controlling performance. This reveals that our growth model is oversimplistic and overall does not produce salient data.

7.2.2 Statistical Insignificance

Apart from the success described in subsection 7.1.2, we cannot say that the experimental plants overall benefitted from the "intelligently" varied nutrient level with statistical significance. With such a small sample size and variability of the data, we cannot attribute stastical signifiance to any discoveries about the utility of our learning algorithm. In order to draw truly scientific conclusions, we would need hundreds of plants and a more rigorous method of evaluating health (e.g. using a chlorophyll meter on each leaf).

7.3 Future Work

7.3.1 Time Sensivity

One of the most glaring oversights in our plant growth model is the assumption that plant health at some particular time t_0 depends only on the amount of water it receives within a day of t_0. While we knew this was an unrealistic view, we underestimated the influence of time on the fidelity of our data.

There are two major ways time should be factored into the model: *plant reaction time* and *plant age*.

Plant reaction time is the minimal time delay between when a plant's environment is changed (say, nutrient level is lowered) and when the plant's appearance reflects that change. For example, it could take a whole week before an unwatered plant begins to wilt. We attempted to account for this delay in Experiment 2 by only taking measurements every 3 days, but this was just a heuristic based on observation of how quickly leaves grow. It is clear that for our algorithm to be truly robust, we need the exact time delay associated with every performance metric, either given as an input parameter or dynamically estimated by some means.

Plant age is simply how long the plant has been alive. We thought we controlled for this by buying already-mature plants, but it is clear by examining the data and visually inspecting the plant's growth that plants generally get healthier as time goes on, provided they are not neglected. In order to examine the relationship between nutrient level and health, it is necessary to consider the natural performance improvement brought by age.

7.3.2 Higher-Dimensional Data

The algorithm described in section 2.3 is formulated with a feature space of arbitary dimension n. In our protoype we considered only the one-dimensional problem, where $\mathbf{x}_{i,t} = n_i \in \mathbb{R}$–only nutrient level is controlled. However, if we decided to control more parameters, such as temperature and p.H., the algorithm would not change, but performance could be boosted. When targeting more parameters, every plant has a finer grain, more realistic mathematical representation, allowing for better practical performance. In addition, more paramters strengthens the ability to simulatanously accomodate plants living in different environments. To understand this aspect, consider two plant systems, one in Alaska and one in the Florida keys, which are given the same amount of nutrients every day. Under our experimental model, since only nutrient level matters, the plants will be treated equally by the algorithm. However, we know that plants grow differently depending on temperature and humidity, and Alaska is colder and drier than the Florida keys. Incorporating temperature and humidity measurements will permit the algorithm to differentiate between these environments and treat the plants accordingly.

7.3.3 Easy Performance Measurement

In order to collect data for our algorithm, we needed to manually scan and count leaves. These are boring and unattractive tasks for the end-user, so to get closer to a commercial product we need to simplify user feedback. One way of accomplishing this is to make performance a heurstic: have the user rate the plant on a scale from 1-10 based on a few qualitative aspects. There are drawbacks to this approach, but it would streamline performance measurement and make Seed more user-friendly.

7.3.4 Non-Convexity

Currently, we assume that the objective function f is convex (or concave), i.e. has a unique optimal point. However, there is no reason to believe that the "true" performance function is convex, regardless of which performance metric we consider. For non-convex problems, optimizating is much trickier, but it would be wise to consider the general non-convex problem when developing a new algorithm.

7.4 Closing Comments

All considered, we consider our project a success. We set out to create a network-based learning framework for automated hydroponics, and our prototype is precisely this. While so far we only have hints that our algorithm actually improves plant health, the gradient descent methodology is based on decades of research in convex optimization, so it is reasonable to believe that our strategy could be proven effective by a wider scale, more carefully conducted study. As a remotely controlled, dependable hydroponics system, our product works exactly up to spec. The question remains of how well Seed scales to thousands or millions of participants, and whether the improvements suggested in 7.3 are feasible to incorporate into the existing framework. Our prototype is only a glimpse into

what is possible with this concept–one day, a system like Seed could very well revolutionalize indoor gardening and bring fresh produce to the masses.

8 References