CHEME 6800 - Computational Optimization
Greenhouse light optimization

December 2018

Terry Greene - tmg77

Kritika Kashyap - kk933

Eesha Khanna - ek542

Michael Eaton - me442

Yuval Kaminer - yk734

Abstract

Growing crops in greenhouses can extend the growing season if conditions such as light, temperature, humidity, and CO_2 concentration, within the greenhouse remain favorable for the plants. To achieve a successful harvest in such environments, it is necessary to supplement natural conditions with artificial means. Modern commercial greenhouses have become excellent at this, they are often filled with equipment to monitor and control heating, cooling, lighting, humidity, and a host of other variables. One of the keys to a steady crop yield is to maintain the optimal level for each of these throughout the greenhouse. Variability in any of these factors across a growing area can cause undesired variability in the crop yield across that area. This project will combat that variability in lighting by optimizing the positions of light fixtures within a greenhouse, to minimize the variance of light intensity across a space.

1. Table of Contents

2. Introduction

Light energy is an essential resource for plant growth. However, in some regions and during certain seasons natural light may not be sufficiently available to grow a crop. To combat this, growers have long relied on additional light sources to augment solar radiation. Providing supplemental lighting to a greenhouse crop increases the upfront costs through the purchase of lighting and labor, and increases the energy costs associated with operating a greenhouse. In recent years however, advancements in LED technology have significantly reduced the costs of LED lighting but in order to remain competitive, growers seek to optimize their operations further. Standardizing the intensity of light across their plants is an issue faced by growers.

Variable light intensity across a growing surface will lead to variability in plant yield. This problem can be solved through plant rotation, but that is manually intensive work and can be difficult to optimize. A potentially more cost effective option is to optimize lighting to minimize intensity variance. At approximately 93 million miles away, the Sun is great at averaging light intensity across a relatively small area, like a greenhouse. Light sources closer to the plant growth, like LEDs, are not as effective at equally distributing light. To achieve perfectly uniform light distribution from LEDs, they would have to be blanketed above the entire growing surface. For a greenhouse this is not a practical solution, as an efficient greenhouse should harness as much free natural lighting as possible. This solution could be prohibitively expensive requiring the purchase and maintenance of a large number of light fixtures. So the goal becomes to place a small number of non-obstructive light sources above a growing surface and control them in such a way that light intensity variance is minimized. This must be achieved while maintaining a light intensity between a minimum and maximum value.

Achieving minimal light varrience across a greenhouse canopy presents a complex optimization problem, one that this report aims to shed some light on. The first challenge of this project was identifying and relating the interaction between all the major parameters influencing light intensity variance. Once that was complete, a non-convex non-linear programing problem could be formulated. Therefore solving the problem presented many solutions at local optimums.

3. Background

Light optimization in greenhouses is a highly profitable achievement for cost reduction and profit increase. Agricultural yields are being maximized constantly and while the profits are always limited and uncertain due to high maintenance costs and large uncertainty in the market. Adding up those numerous facts, additional artificial light can greatly reduce uncertainty in natural light and add provide growing enhancements in relatively low light areas; a light optimization method that considers several types sources with different parameters as well as different positioning of the sources would add valuable insight to the problems faced by growers.

Light sources come in a variety of types, some have a very narrow beam of light, making them very focused and on the other end of the spectrum there are very dispersive sources that make a much more ambient light (non-directional), while usually the former tends to have much higher intensities than the latter. Another factor to consider in optimizing the light sources is the variety of the absorbed light by the target, if it is big, some of the plants in the greenhouse might become over exposed and burnt from excess light, while others will not achieve best possible growth conditions due to less than optimal light impact.

Taking into account the non-linear character of light attenuation from its source as the main function to describe the physics of the problem, and a statistical variance at points on a surface within the greenhouse, the optimal solution for the problem could be found by formulating and solving an optimization problem. The proposed model will take into account the physics of the problem and with mathematical tools will give an optimal solution for positions, of a specific number and type of lights, in order to get optimal light distribution in the greenhouse.

In this work the main parameter for variance analysis will be the light intensity hitting the targeted surface, the theoretical quantification of the intensity is by counts of light photons hitting the surface per unit of time:

$$\frac{Photons}{area \cdot time} = \frac{\mu mol}{m^2 \cdot Sec}$$

By taking a generalized version of the light intensity from a point source and general attenuation [figure 1] adding to it angle of spread from the source and impact angle on the target surface [figure 2] the physical definition of the problem is formulated.

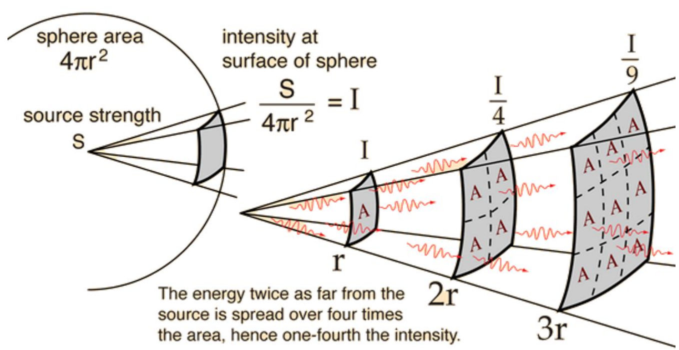

Figure 1: Light Attenuation From a Light Source

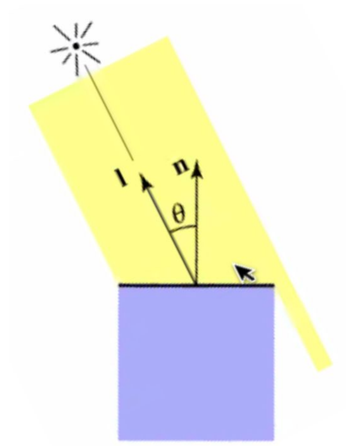

Figure 2: Light Impact Direction vs Perpendicular Normal to The Target Surface

4. Literature Review

Mathematical optimization models of greenhouses appropriate for their optimal operation have been proposed so that energy costs related to artificial lighting can be minimized. Many sources of artificial lights can be used as a supplemental for plant growth inside a greenhouse. These include incandescent lights, High Intensity Discharge (HID) lights, fluorescent tubes and Light-emitting diodes (LED). One of the main advantage that LED has over the other mentioned sources is that it can be customized to produce the wavelength of light desired. LED plant lights emit the red and blue wavelengths of the light spectrum that is the most important source of energy for plants. Other advantages of LED lights include its energy efficiency, long life and low emission of heat. Designing a method to achieve a good uniform illumination distribution of LED lights on a plane of interest plays an important role in ensuring that lack of adequate light does not limit the growth of plants in a greenhouse. In addition to applications in the realm of plant growth and greenhouses, uniform illuminance distributions are favorably perceived by people in terms of visual abilities, allow for lower average horizontal illuminance in places where light level requirements are not strict, and also provide a solution to glare related problems [9].

Uniform illumination has been looked at from different perspectives: lens design and optimization, where the LED is treated as having two optical components, namely the source of light and the lens that controls angle and outgoing intensity [1, 2, 6]; development of reflector hoods and embodiments around the LED lights that use optics principles to allow for uniform illumination [8]; integrated daylighting and electric lighting to achieve a uniform illumination through dynamic simulations that account for varying levels of daylight coupled with appropriate changes in the dimness of LEDs [5]; and optimization of various parameters such as geometry, source to target distance, intensities or dimming levels, spatial distribution and angles, placement (x,y coordinates) as well as number of LED light sources [7]. Regardless of the approach taken to achieve uniform illumination on a target plane, it has been established that multiple sources of light are needed [4].

Barbosa et al. (2017) solved a multiobjective optimization problem. The goal of the optimization process was to find a solution that satisfied both thermal dissipation and light efficiency. If the optimization goal was only to minimize the LED temperature, an obvious solution would be a luminaire with the fewest number of LEDs arranged in the largest possible heat sink area. Smaller the number of LEDs in the luminaire, lower would be the heat dissipation of the system. However, the number of LEDs and their layout on the luminaire directly affect the distribution of luminous flux on the target plane. Therefore a combination of lighting cost and heat transfer cost was taken as the objective function. Two deterministic algorithms were selected to solve this problem: quasi-Newton and Nelder–Mead. The quasi-Newton method is a derivative-based method that provides a balance between the simplicity of gradient descent and the speed of Newton's method. The Nelder–Mead method is a direct search derivative-free method. The performance of these two algorithms is highly dependent on the starting point. A heuristic method namely Biogeography based optimization (BBO) was also used to design the luminaire. Hybrid methods can achieve better results. Thus, the starting point of a deterministic algorithm could be the best individual found by the heuristic method [10].

Su et al. developed a workflow to optimize uniform illumination on a target plane using LED arrays of various configurations and intensities. A simulated annealing algorithm was used, which is a probabilistic technique that finds an optimum by comparing outcomes of neighboring states and compares those to the current state outcomes [1]. This method, like many others, allows for only two degrees of freedom for the placement of the light sources (i.e. x and y directions), whereas the optimization presented in this paper light source placement in a 3D space [1, 4, 7].

Three different frameworks have been developed to define the relationship between the light source and the target surface: the tailoring method, which develops nonlinear PDEs based on the required illumination distribution and the LED characteristics; the grid method, which discretizes the target surface into a matrix and treats each component as a point at which the radiation is a linear combination of the light contribution from all luminaries; and the simultaneous multiple surface method, which is typically used for extended sources of light when designing or optimizing optical lenses [2]. We use the grid method and assume point sources of light.

Ray tracing method optimizes the performance of LED illumination systems by modifying the LEDs' luminous intensity distribution curve (LIDC). Tsai et al. (2016) developed the one-time ray-tracing optimization method. With an initial design of a LED illumination system, the light source's luminous intensity distribution is first divided into several source zones (S-zones) and the target illumination region is divided into several target zones (T-zones). The resulting average illuminance at each T-zone resulting from each S-zone is described by a characteristic matrix. The characteristic matrix contains the ray-propagating property of the initial design of the illumination system structure. For optimizing the system property by modifying the LIDC of the light source, this method introduces a flux weighting array. With every change in the weight ratio of the light flux of the source zones, the illumination of the target region can be easily calculated by a simple manipulation of the characteristic matrix without performing a time-consuming ray-tracing simulation [11].

Different optimization techniques have been applied to minimize variation in illumination, ranging from simulated annealing algorithms to brute force methods that calculate illumination for different iterations and check if a threshold is met [1, 4, 7]. In general, most studies use two kind of objective functions for minimization - coefficient of variance of the root mean square error (CVRMSE) and variance/standard deviation of illumination levels at different grid points [1, 4, 6]. This project uses superposition of lambertian reflectance from each light source to allow for easy formulation of nonlinear equations and constraints [3, 6].

5. Problem formulation

The objective of the problem is to evenly distribute lighting resources in an indoor agricultural setting through optimizing the lighting configuration parameters. The optimal solution is subject to constraints that involve the geometry of the space and characteristics of the lighting fixtures. The optimization problem is formulated to minimize the variance of lighting intensity across a particular surface by choosing the positions of light fixtures. The light fixtures were modeled as point sources emitting in all directions, with locations defined by their (x,y,z) coordinates. This was a simplification assumed for the purpose of this project. It is also important to note that the room being modelled was assumed to have no outside daylight, which was another simplification made for this project.

The objective function to be minimized is the variance of light intensity values across a discretized surface:

$$\textbf{min } f(x_i, y_i, z_i) = \sum_{ijk} \left[\frac{(Iijk - IA)2}{(IA)} \right]$$

Where,

$$I_{ijk} = \frac{K_i}{r_{ijk}^2}$$

- "I_{ijk}" is the light intensity contribution of light fixture "i" on the (x,y) coordinates of the surface of interest, these coordinates are mapped to the j_{th} and k_{th} points on the discretized surface.

$$r_{ijk} = \sqrt{(X_i - X_p)^2 + (Y_i - Y_p)^2 + (Z_i - Z_p)^2}$$

- "r" is the distance between any point P on the surface of interest and the "i_{th}" light fixture. The coordinates defining the location of point P are (x_p,y_p,z_p) and the coordinates of the fixture are defined as (x_i,y_i,z_i).
- K_i = Light intensity constant of fixture i

With technological advancements in the LED industry, lighting fixtures can be programmed to dim their intensity. The dimming features can be included in the optimization problem by introducing variables for the K value shown above. The present formulation would allow this K value to become a variable, however this was not done in the present study.

Original Problem constraints

- Position constraints: The (x,y,z) positions of the fixtures should be within the dimensions of the room.

  $x_L <= x_i <= x_U$

  $y_L <= y_i <= y_U$

  $z_L <= z_i <= z_U$

- Fixture dimming constraints: The fixtures may only be dimmed to 50% of their full brightness. This is a realistic constraint to avoid impacting the full life of the product.

  $0.5 =< D_i <= 1$

- Minimum Intensity value constraint

  $I_{ijk} >= I_{min}$

To more easily formulate the problem in Julia, the light intensity values at each grid point were introduced as variables. The relationships between the locations of the point sources (x,y,z), and the light intensity at each grid point "I_{jk}" were formulated as constraints in order to model the physics correctly. These constraints require that the sum of the brightness contributions from each light source defined by Eq (1 and 2) equal the total light intensity value for a particular grid point.

$$\sum_{i}\left[I_{ijk}\right] = I_{jk}$$

This did not really add any flexibility to the problem because the relationship between variables is not an inequality but this was important in formulating the problem to JuMP and IPOPTs required inputs. It was also necessary to introduce I_avg as a variable and define the relationship between all of the grid point and the average for use in the objective function.

$$I_{Avg} = \sum_{jk}\left[I_{jk}\right]/(M)$$

In addition to working well with the optimization package, this problem formulation reduced complexity of individual terms in the objective function and reduced the difficulty of debugging. The next section will discuss the implementation of the optimization problem in JuMP, the Julia Optimization package that was used in this study.

6. Model and algorithm

JuMP is an open-source algebraic modeling language (AML) that has been used for the optimization of this non-linear problem. JuMP takes advantage of advanced features of the Julia programming language to offer unique functionality while achieving performance on par with commercial modeling tools for standard tasks. The main attraction of using JuMP is its ability to plug and play with optimization solvers written in C, C++ and Fortran. JuMP is a dynamic language with interactive command-line read-eval print loop C-like performance. It was important the the user be able to interact with solvers while they are running. This offers both control of the solution process and allows the reduction of overhead needed to regenerate a model, when solving a sequence of related instances. This motivated the implementation of the problem in Julia using JuMP. JuMP is designed to be extensible, allowing for developers both to plug in new solvers for existing problem classes and to extend the syntax of JuMP itself to new classes of problems [12].

However, AMLs do not solve optimization problems, they provide the problems to optimization routines called solvers. Ipopt is an excellent solver for derivative based nonlinear models which supports Hessian evaluations and gradient calculations.

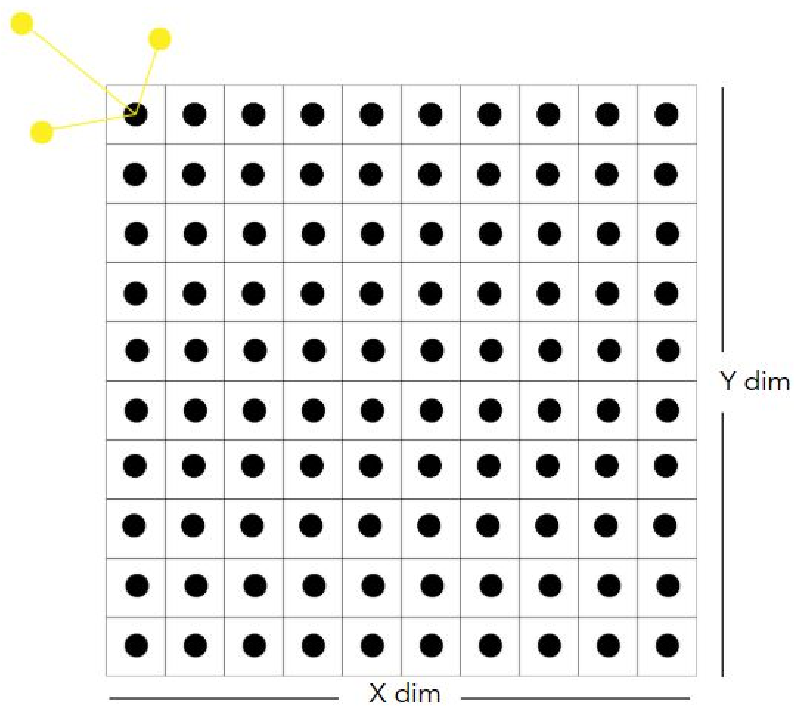

Figure 3: Schematic Representation for a 10×10 grid and 3 point sources.

One of the classic problems in greenhouse optimization is to optimally allocate supplemental lighting to the crops. The problem considers the variance of the sum of intensities from each point source of light onto each discretized grid point on a plane of interest. The aim is to seek the x, y and z position coordinates of the point sources in order to minimize the variance of intensity on the plane of interest. This is a non-linear optimization problem with linear and non-linear constraints. Variables involved in the optimization problem are as follows :

1. Three state variables
   - x : x position coordinate of the point source
   - y : y position coordinate of the point source
   - z : z position coordinate of the point source
2. One control variable
   - Imin : minimum Intensity
3. Three dummy variables
   - I : sum of intensities from all point sources on a given grid point
   - I_avg : average Intensity
   - f : objective function

The positions of light are affected by dimensions of the room, cosine of angle between the normal of **the point source on the horizontal plane and the line connecting the point source to the grid point**, position of grid points and the distance constraint between any consecutive light source. The solution algorithm implements Lambert's cosine law through constraints and then aims to minimize the variance of Intensity in the objective function.

Ipopt.jl was used to allow the user to explore the effects of varying the number of point sources involved in illuminating a given plane of interest of user defined dimensions. The JuMP code can be found in the Appendix-I. The model is solved with a different initial value for x, y and z of each point source. This ensures that the solutions of the state variables reach a unique minima for every point source and does not get stuck in a local optimum solution due to the non-convexity of the problem. Avoiding the initialisation of x, y and z leads to the concentration of all lights at a particular coordinate which defies the purpose of obtaining a uniform spread on a given plane. Hence this is an important step in the solution algorithm.

The code is written such that it loops through the coordinates of every grid point, calculate the sum of intensity at this point by looping through the coordinates of every point source. Another solving approach attempted was to customize a user defined function that would calculate the sum of intensities at each grid point from each point source and return the value of the intensity map across the plane of interest to the function. The next step is to call this user defined function into the objective function and minimizing it using Ipopt. However, this elegant solving style proved to be inefficient and the code crashed. The non-linear solver is unable to calculate the Hessian and gradient matrices for a function involving such intricate loops. The user defined function we made essentially becomes a black box problem for the non-linear solver which it cannot access and solve.

The failure of one solving technique helped the development of a better understanding of the way the non-linear solver optimizes the model and develop another solving technique on this path. Any function involving unknown variables has to be written as a constraint or an objective function according to the syntax of JuMP. A dummy variable "I" (intensity) was introduced for each grid point. "I" is constrained to be greater than or equal to a predefined minimum intensity "Imin" (control variable). A non-linear constraint is written to allocate the sum of intensity between every point source and every grid point to the dummy variable "I". The variable "I" stores the superposition of intensity for each grid point. This further helps calculating another dummy variable "I_avg" (average intensity) by summing over "I" and dividing by number of grid points. "I_avg" stores the value of average intensity for the plane of interest. Finally the last dummy variable "f" (objective function) was used by writing it as a non-linear constraint that allocates the value of the variance of intensity across a given plane to "f". "f" can be called into the non-linear objective function for the model created using JuMP and solved using Ipopt. The code written in this fashion in Julia leads to the successful completion of the optimization by giving desirable results.

7. Results and Discussion

In order to validate the model before any further case studies were carried out, a test case was developed. For this test case, a room of 100 units width, 100 units length, and 10 units height was used. The target surface was discretized into a 5 x 5 grid and the model was optimized for just one light source. One light source was chosen for the test case because it is easy to compute the result for this case analytically. The resulting location for the light source was (50, 50, 7.2), which made sense because the light was located at the center of the grid.

Another test case used before the case study was carried out was an optimization 4 light sources. For this test case, a room of 100 units width, 100 units length, and 10 units height was used. The target surface was discretized into a 10 x 10 grid and the model was optimized for just 4 light sources. Figure 4 shows the distribution of light sources. The symmetry of the solution validates the model further.

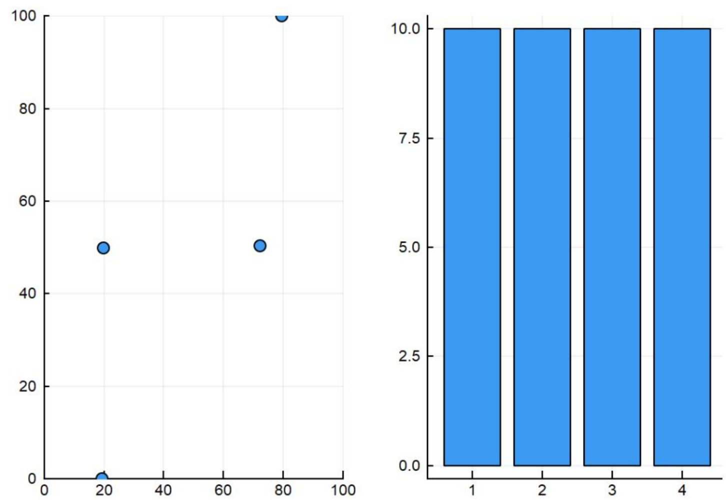

Figure 4: Left - X,Y Coordinates of Light Sources, Right - Z Coordinates of Light Sources

Iteration 1:

Once the model was validated, a case study with the following parameters was carried out:

- Dimensions of room: 100 units width (x-direction), 100 units length (y-direction), 10 units height (z-direction)
- Grid size: 10 x 10
- Number of target sources to optimize locations for: 10

Figure 5 shows the distribution of the intensities of light on the target surface, with white regions being less lit and darker regions being more lit. Additionally, figure 6 shows the X,Y locations of the light sources in plan view and figure 7 shows heights and the cones of light in section view.

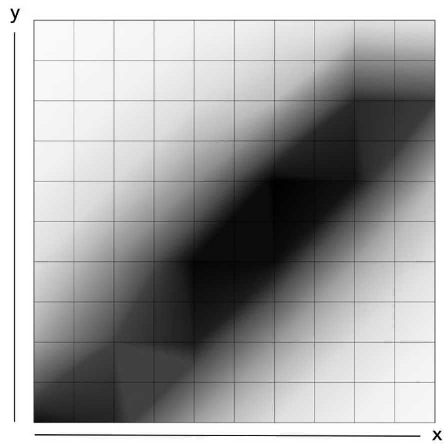

Figure 5: Light Intensity on Target Surface for Iteration 1

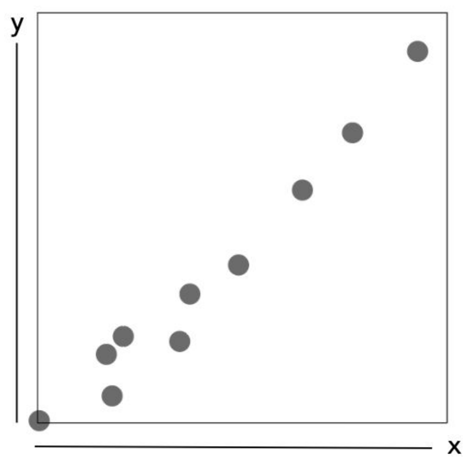

Figure 6: Position of Lights (Top View) for Iteration 1

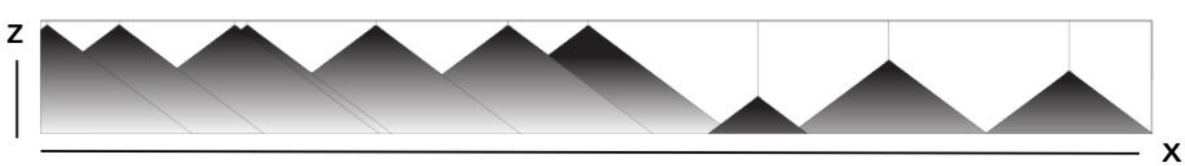

Figure 7: Position of Lights (Section View) for Iteration 1

Figures 5 and 6 show that the intensity and the light sources are concentrated along the diagonal of the room and the target surface respectively. This is perhaps a result of the symmetry of the problem.

Since IPOPT, the solver used for this project, is a local solver and since the problem is non-convex, it is important to note that the solution shown above is a local solution. For non-convex functions, several local solutions exist and in many cases, several equivalent local solutions also exist, especially when the problem being solved is symmetrical. This translates to the fact that the solution obtained by the solver is highly dependent on the specified initial points. Figures 8 and 9 show results from 2 local solutions obtained by the solver when the same parameters were specified and different initial points were used. Given that different solutions were obtained with very similar objective values, these figures highlight the importance of specifying "good" initial values in order to get closer to the global solution when using a local solver like IPOPT. As shown in figure 7, the result - that most of the lights are concentrated in the first half of the room where they are higher up (allowing for more diffused light), while in the other half of the room, there are less lights and they are lower down - further implies the existence of multiple equivalent solutions.

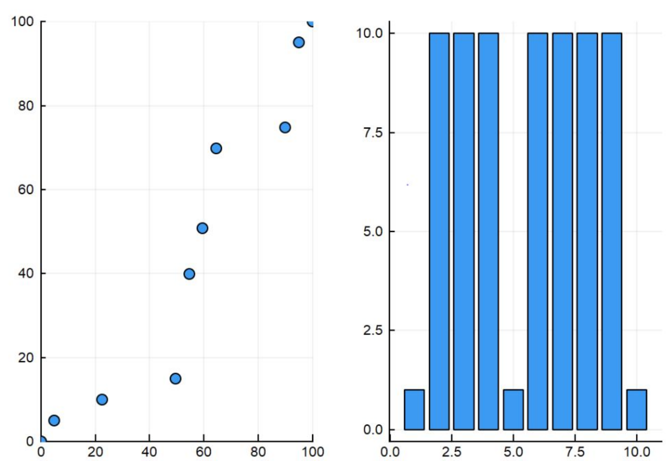

Figure 8: Left - X,Y Coordinates of Light Sources, Right - Z Coordinates of Light Sources

Objective value (variance) = 0.3026

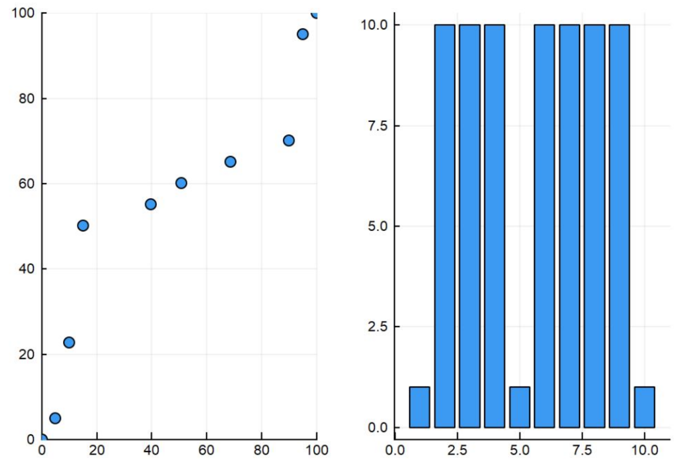

Figure 9: Left - X,Y Coordinates of Light Sources, Right - Z Coordinates of Light Sources

Objective value (variance) = 0.2994

Iteration 2:

After understanding the importance of initial points and the limitations of a local solver like IPOPT, an addition was made to the model that specified initial points which were spread out so as to prevent the solver from falling into the same local solutions for different points. The same parameters (as in iteration 1) were used and the updated model was run to get the results shown in figures 10, 11, and 12.

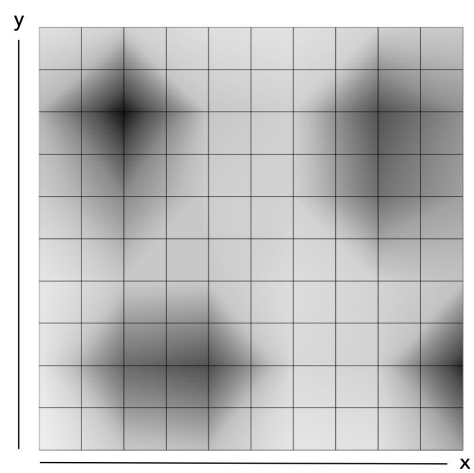

Figure 10: Light Intensity on Target Surface for Iteration 2

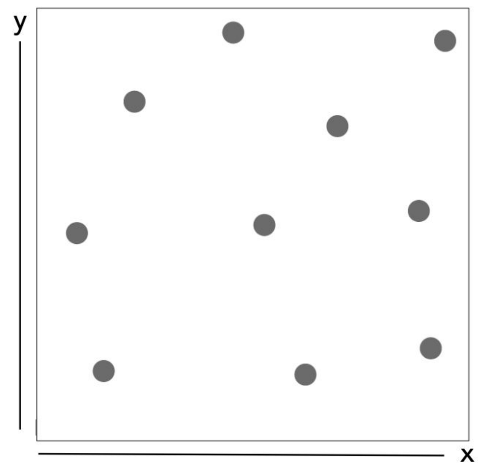

Figure 11: Position of Lights (Top View) for Iteration 2

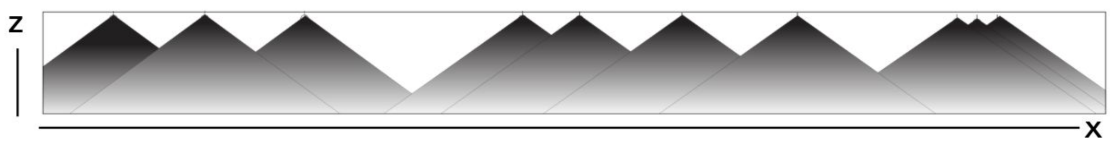

Figure 12: Position of Lights (Section View) for Iteration 2

The variance for iteration 2 was also slightly lower than the variance for iteration 1. Therefore, once the limitation of the solver was understood, the solutions started becoming "better."

8. Uncertainty

In the process of developing an algorithm for light distribution across a greenhouse, several assumptions and simplifications were made. Assumptions and simplifications were necessary to reach a sound solution to the described problem within the project time constraints. Concerning the artificial light sources, two significant assumptions were made. It was assumed that each light added to the system emanated uniformly from a point source; the second assumption was that a light source is a zero dimensional object, meaning that it does not have physical limitations or disturbances such as shading or projected light with specific framework angles. In practice any artificial light would cover a three dimensional space and put off light at variable intensities. These unfortunate realities were removed from the problem formulation due to complexity they introduced into the problem. However, it can be assumed to have minimal impact because the area of the light is insignificant in comparison to the greenhouse floor. Additionally it can be assumed that a light emitted from a bulb will closely match the manufacturer specifications when averaged over time.

Other simplifications made while formulating this problem can be similarly rationalized away due to large impacts on complexity but assumed small impacts on solutions. For example the model developed did not account for variations in ambient natural light. Sunlight throughout a greenhouse can obviously vary at any given time due to cloud cover, or other plants. Over the course of a growing season or even a day though, the sun's photons across the growing canopy should have very small variances (assuming no monuments obstructing them). Similarly it was assumed the plant canopy would exist at a uniform height. With proper optimization of light sources and other factors influencing growth, this should be approximately true. This however would not be true in greenhouses growing more than one crop. Which is an example of complexity that could potentially be later introduced into the problem formulation.

9. Future work

A proposed next steps in future projects would be to incorporate a two stage stochastic solver into the current solution. The solver will have a first step of solving the strategic level or here-and-now, optimizing the light conditions for the averaged natural light and the lights boundary conditions and all the sources' parameters. The second step would be operational level optimization which takes into account the actual natural light and adjusts the light optimization, by calculating for the entire daily light integral. Then making changes in the light source intensities and the duration of operations.

Other proposed future steps for enhancing the optimization of supplemental light in a greenhouse are, changing the light parameters to fit multiple different light types. These include different source power, spread angle from the source, and dimmable light that could make the supplemental light much more adaptive and easily optimized on a day's light integral. An optimization problem like light in a greenhouse is a problem that the complexity for optimization could be almost endless, but can create a large benefit as a product, because experimenting with different types of lights, intensities or reflectors to achieve optimum lighting conditions is very costly. So by increasing the complexity of the optimization process, aspects that would normally would not be examined can be added and calculated through minor additional solver time.

An aspect that was not taken into account in this work is actual sized and physical limitations of greenhouses. Future work could examine whether a square shaped building is the preferred shape. Perhaps lighting or other constraints that would favor a different shape. Further limitations may also need to be placed on lighting fixture height due to the height of the plants or maximum height limitations due to the structure. Variation of the shape could theoretically easily implemented in the code as it takes into account the general arbitrary dimensions and calculates with them as boundaries.

One last aspect that could be implemented into the optimization work is the addition of outside daylight as a variable. The current problem is a simplified version of the real scenario and assumes that the growing surface receives zero light from the outside. In reality, light from natural sources would be diffused throughout the greenhouse and contribute significantly to the ambient lighting of the greenhouse. The variability of these sources should be accounted for in a more robust solution.

The presented problem of light optimization could easily include more features, and constraints, any additional component added to the optimization problem though increase the complexity and the time to solve. However these factors can be added to computational optimization much faster and cheaper than physical testing would allow. The work presented provides the optimizational framework of a real life complex nonlinear, non-convex problem. Working off of this framework, other external constraints and considerations could be added with relative ease.

10. References

1. Su Z., Xue D., and Ji Z. “Designing LED array for uniform illumination distribution by simulated annealing algorithm.” *Optics Express Vol. 20, No. S6.*

2. Zhang K., Su C.Y., Feng Z.Y., Wang W.J., Zhang C.H. “Freeform surface lens design for uniform illumination of a rectangular area.” *The Society of Light and Lighting.*

3. Cassarly W.J., and Hayford M.J. “Illumination Optimization: The Revolution has Begun.” *SPIE - The International Society for Optics and Photonics.*

4. Kopparapu, S.K. “Lighting design for machine vision application.” *Image and Vision Computing.*

5. Pandharipande A., and Caicedo D. “Daylight integrated illumination control of LED systems based on enhanced presence sensing.” *Energy and Buildings.*

6. Wang K., Wu D., Qin Z., Chen F., Luo X., and Liu S. “ New reversing design method for LED uniform illumination.” *Optics Express Vol. 19, No. S4.*

7. Ramane D., and Shaligram A. “Optimization of multi-element LED source for uniform illumination of plane surface.” *Optics Express Vol. 19, No. S4.*

8. Nicole S. et al. “Horticulture lighting system for providing uniform illumination from an elevated height.” *WO Application.*

9. Narendran N., Freyssinier J.P., and Zhu Y. “Energy and user acceptability benefits of improved illuminance uniformity in parking lot illumination.” *The Society of Light and Lighting.*

10. Barbosa J.L.F., Simon D., Calixto W.P. “Design Optimization of a High Power LED matrix luminaire.” *Energies 2017,10,639.*

11. Tsai K.F., Chu Shu-Chun. “Characteristic matrix operation for finding global solution of one-time ray-tracing optimization method.” *Optics Express Vol. 24, No. 19.*

12. Dunning I, Huchette J, Lubin M. “JuMP: A Modeling Language For Mathematical Optimization.” *arXiv:1508.01982v3 [math.OC] 15 August 2016.*

I. Appendix – I: Computer code

```
using JuMP, Ipopt

 # Creating the Model using JuMP
m = JuMP.Model( solver=Ipopt.IpoptSolver() )

# No. of point sources
P = 10

# lets specify no. of grid points. N is the no. of x grid point. M is
the no. of y grid points.
N = 10;
M = 10;

# So we have N*M grid
L = N*M;

# Constant of proportionality for Lambert's Cosine Law
C=100


# Lets specify the size of the room
Xdim = 100;
Ydim = 100;
Zdim = 10;

# Lets specify the variables

# Control variable
Imin =0.02

# Dummy Variables
@variable(m,I_avg>=0)
@variable(m,f>=0)
@variable(m,I[1:L]>=Imin)



# State Variables
```

```
@variable(m, x[1:P]>=0)
@variable(m, y[1:P]>=0)
@variable(m, z[1:P]>=1)

# Dimensioning Constraint
@constraint(m, con[i in 1:P], 0<=x[i]<=Xdim)
@constraint(m, don[i in 1:P], 0<=y[i]<=Ydim)
@constraint(m, ron[i in 1:P], 0<=z[i]<=Zdim)


# Initializing xmatrix
xmatrix    = zeros(N,1)

# Lets get all the x-coordinates
count = 0;
for i = 1:N
  xmatrix[i] = Xdim*(1/N)*(0.5 + count)
  count = count+1;
end

# println((xmatrix))

# Initializing ymatrix
ymatrix    = zeros(M,1)

# Lets get all the y-coordinates
count = 0;
for i = 1:M
  ymatrix[i] = Ydim*(1/N)*(0.5 + count)
  count = count+1;
end

# println((ymatrix))

# Now let's get all the combinations of coordinates for our plane of
interest
using Iterators
arr = Any[]
       for p in product(xmatrix,ymatrix)
                  push!(arr,[y for y in p])
       end
coordinate_matrix = hcat(arr)
```

```
# Non linear constraint for calculating intensity using Lambert's
Cosine law and allocating it to the dummy variable I
for l = 1:L
  @NLconstraint(m, sum((C*z[k]/((coordinate_matrix[l][1]-x[k])^2 +
(coordinate_matrix[l][2]-y[k])^2 + z[k]^2)^1.5 ) for k = 1:P ) ==
I[l] )
end

# Calculating average intensity across the plane and assigning it to
dummy variable I_avg
@NLconstraint(m, (sum(I[j] for j=1:L)/L) == I_avg)

# Calculating variance and assigning it to dummy variable f
@NLconstraint(m, sum((I[i]- I_avg)^2/L for i = 1:L) == f)

# Minimizing Objective
@NLobjective(m, Min, f)

# Distance Constraint between consecutive lights
for j = 1:P-1
    for i = j+1:P
        @constraint(m, x[i] >=  x[j] + 0.5 )
    end
end

for j = 1:P-1
    for i = j+1:P
        @constraint(m, y[i] >= y[j] + 0.5)
    end
end


# Lets set the initial values for x,y,z coordinates of light sources
count = 0
for i = 1:P
  setvalue(x[i], count)
  count = count + 6
end

count = 0
for i = 1:P
  setvalue(y[i], count)
  count = count + 6
```

```
end

count = 0
for i = 1:P
  setvalue(z[i], count)
  count = count + 1
end

status = solve(m)

println("got ", getobjectivevalue(m), " at ",
[getvalue(x),getvalue(y),getvalue(z),getvalue(I)])

writecsv("Objective.csv", float(getobjectivevalue(m)))
writecsv("xvalues.csv", float(getvalue(x)))
writecsv("yvalues.csv", float(getvalue(y)))
writecsv("zvalues.csv", float(getvalue(z)))
writecsv("Ivalues.csv", float(getvalue(I)))
```