OPTIMIZING PRODUCTION FOR A PLASTICS MANUFACTURER USING LINEAR PROGRAMMING

D. CAMPBELL, L. FREY, S. FRYE, P. MARMORINO, P. PANOSOT, AND J. WINKLER

1. Introduction

Optimizing production schedules is an essential task in any factory setting. This is less challenging in factories with high-volume production of a small number of items. Scheduling efficiency is more essential and requires more care in factories in which many different parts are being produced. Our industry partner, Alpha Systems, asked our PIC Math team to develop an automated production scheduling program for their tanks division in which various quantities of more than 500 different tanks are regularly manufactured.

Alpha Systems is a plastics-based manufacturing company in which innovation and development are at the core of company culture. In the tanks division, some of the production scheduling thus far has been done manually, relying on significant institutional knowledge. We were tasked with streamlining and optimizing the production scheduling in the rotational mold section of the tanks division. Our main focus was to optimize the production while still allowing for client and customer satisfaction to be prioritized. An automated production schedule can improve efficiency and allow for future growth. A set production plan also allows Alpha Systems to expand more easily with a shorter transition period.

The model created helps to maximize production and create a more efficient production process overall. This is necessary because without a maximized production process revenue will never be maximized. Our model creates a mold schedule for rotational molding. It shows when molds will need to be run and when they will need to be switched out in order to meet production requirements. The model also takes into consideration that Alpha Systems requested that two weeks worth of inventory be held in the warehouse to help meet demand and to have a turnover of warehouse stock every two weeks. This is important because for small ticket items it is more efficient to produce per order rather than to have two weeks supply in inventory, where it may sit for multiple months. The model then prioritizes keeping sufficient inventory stocked for high-volume products.

This paper is organized as follows. Section 2 discusses Alpha Systems as a company. Section 3 of this paper summarizes our results. Section 4 goes over the programming language and libraries that we used to write and solve the linear programming problems. The mathematical details of the three linear programming problems are discussed in sections 5 and 6. We conclude in section 7 and discuss possibilities for future work.

Date: May 5, 2020.

2. Alpha Systems

Alpha Systems is a multimillion dollar company that was initially founded in 1984. As a contributor to the construction industry, the company produces a wide range of products for recreational vehicles and manufactured housing as well as the marine, automotive and retail industries. These products include but aren't limited to roofing, flooring, adhesives, shutters, freshwater and holding tanks. Furthermore, Alpha Systems uses three key manufacturing methods. The oldest of these methods, rotational molding, is utilized to manufacture both high quality freshwater and holding tanks. The next method, injection molding, was implemented in 1997[1] and injection molded shutters can now be seen adorning homes all throughout the country. The final method, blow molding, was acquired in 2009 as a means of more efficiently producing freshwater and holding tanks for the recreational vehicle industry, which has been an immense success. Their mission is to recurrently develop products that allow other industries to build finished products of which are in high demand.

Throughout the course of several months, all contact was conducted by means of the two most qualified individuals at Alpha Systems regarding the project goal. The primary individual, Jeff Taber, is the well respected Director of Manufacturing through which we first began orchestrating the project. As the project progressed, requiring more in depth understanding of the manufacturing process in the tanks division, we began consulting with Mike Odiorne, the exceptionally dedicated Tanks Division Plant Manager. Mike was able to provide us with all of the relevant numerical data and details regarding equipment and aspirations for production, which became the basis of our model.

Alpha Systems was interested in receiving a model that would maximize production, specifically in the tanks division. We focused on optimizing rotational mold production because Alpha Systems had already optimized their blow molding process due to the fact that there are not as many molds to run and mold changes are not practical. Rotational molding is performed by a machine with 4 arms that hold 6 different molds. Each mold is filled with liquid polyethylene plastic and as the plastic hardens the mold is rotated continuously to evenly distribute the plastic, which takes about 60 minutes. This process creates a hollow tank that is then put onto an assembly line for further processing, where employees install add-ons to create more than 500 different products. Because many products use the same molds, we disregarded add-on pieces and focused on optimizing production based on molds. Not only was the number of molds included in our model, but the time for changing a mold, about 1 hour per mold, plus a penalty of 6 hours for stopping the arm of the machine, was also factored into the production time.

Alpha systems not only wanted to optimize their manufacturing schedule, but also wanted to optimize storage. The goal was to have roughly two weeks worth of product in storage at all times. This was not always achievable; however, this is a rare occurrence with our model. In the uncommon case that production falls short of demand, Alpha systems utilizes outsourcing. This is a scenario we accounted for in our model. When demand is unable to be met a message is output to inform Alpha Systems that outsourcing is needed.

Warehouse organization was also reconsidered due to Alpha systems expanding their current warehouse. It was expressed that a lot of time is spent retrieving raw products from the warehouse and transporting them back to the production line. Products were grouped into two different categories: raw product and finished product. These groups were then subdivided into fast, moderate, and slow moving products. These groupings help to distinguish where products should be stored in the warehouse to minimize travel times on forklifts. Giving them these new groupings allows them the ability to reorganize the warehouse to optimize production and minimize time on the forklift.

3. Results

We constructed two main programs to optimize production and scheduling in Alpha System's tanks division. The first runs an integer linear programming problem to optimize production of each mold, while the second runs two separate linear programming problems to find the optimum scheduling for mold changes as well as to determine which molds should be run on which machine at what time.

The program optimizing production entailed an integer linear programming problem with over a thousand variables and a similarly large number of constraints. This model was implemented in Python and is solved by the popular CBC solver, which employs a branch and cut technique to solve the linear systems. This model can be run on a standard home computer and is guaranteed to find a feasible solution for any remotely reasonable real-world input.The model usually manages to keep the warehouse stocked up with less than two-weeks supply, but there will usually be a few products in any given week for which demand cannot be met given the limited number of molds available at Alpha Systems. Although Alpha Systems stores a huge surplus of slow-moving products in the warehouse, our model finds that this is not optimal and the marginal time cost of running small production runs as needed for low-volume products is less than the benefits of focusing on the fast-moving products.

The program optimizing scheduling required two integer linear programming problems to be done sequentially. Both steps required tens of thousands of variables, but only several hundred constraints. The increased size of these problems meant that the CBC solver was no longer sufficient to produce a solution in a reasonable amount of time. The powerful solver Gurobi was used instead for these steps in the optimization, although Python was still used to implement the model. Gurobi is efficient enough for these larger linear programming problems to be run on a personal computer as well. After these linear programming problems are solved an hourly schedule for every machine is produced, detailing exactly which molds should be used and for how long in order to minimize the time wasted changing molds.

4. Programming

With the sheer number of variables and constraints in the model, it was evident that a computer would be required to solve it. Typical tools for data analysis include R and Python. We opted for Python for several reasons. Both Python and R have a similar

capacity to handle mathematical operations required for data analysis. The differences were in the available libraries, community support, and PIC Math members' previous experience with the languages. While R has a deep and mature set of statistical libraries, Python has more general libraries for things such as linear programming, symbolic programming, artificial neural network training, and tools to compile the language down to machine language instructions. The programming community size of Python is much larger than R, with Python beating Java in programmer attention as of the Stack Exchange Incorporated survey in 2019, with 41.1% of respondents saying they use the language with development work, with R at only 5.8%. [2] This means that online searches to resolve technical bugs would have a higher rate of success. This also means that the skill of knowing the Python language is much more transferable between problem domains. These made Python the better choice for our purposes.

The Python libraries used were pandas, openpyxl, PuLP, Gurobipy, and cx_Freeze. Pandas provides some data structures used in R such as data frames. Openpyxl provides functions to convert between Excel and Python data types. PuLP is a library for linear programming problems. Gurobipy is an interface to the Gurobi software. Finally, cx_Freeze was used to package our program into an executable file that Alpha Systems could use without installing Python.

Several Python libraries for linear programming exist. We selected PuLP due to its popularity and price point as an open-source project. Additionally, we needed support for integer linear programming, something that some other linear programming libraries, such as SciPy, do not support. In the PuLP library, we still had the choice of what solver to use. We used the CBC solver as it uses a branch and cut method, a method useful for solving ILPs. Some commercial solvers (such as Gurobi, which we used to schedule production) may have shorter run times. For production, this was not an issue. However, for scheduling, this was an issue. This is why Gurobi was used to solve the two linear programming problems involved in scheduling, despite it not being a dedicated Python product. Gurobi has features such as visualization and timing iterations of optimization algorithm, which are extremely useful for optimizing the performance of very large linear programming models such as those used to establish a weekly production schedule. Gurobi is also extremely fast compared to other linear solvers, which was necessary for the scheduling problem.

Alpha Systems had its majority of its data in the form of Excel spreadsheets. One of the stated problems was to make sure the warehouse had a two-week supply and did not overflow. The products sold over time varied, sometimes wildly. The approach to handle this was to write a program in Python that could be compiled as an executable for the company to use. The program begins by prompting the user to input the starting week and the number of weeks the user desires to run the program. The starting week indicates the column in the sales data spreadsheet the program is to begin with, each column representing one week of sales. Next, spreadsheet data on molds, sales, and inventory is read into Pandas data frames. Using integer linear programming (ILP), a solution meeting our constraints is found. Finally, Openpyxl is used to write the production back to the spreadsheet. The program performs the above steps in a loop for the desired number of weeks, each time shifting the sales data by one column to read the next week's sales.

5. Formulating a linear programming problem to optimize production time

Alpha Systems maintains a stock of around 400 rotational molds, of which almost 300 are unique. Many molds can mold more than one product - once the base product is molded, different fittings an attachments can be added to the product (usually a holding tank or waste tank of some kind) by hand. Different products made by the same mold are considered the same by the program because its only purpose is to optimize what molds should be used and for how long. Most products have only one mold, but some high-volume products had multiple identical molds to keep up with demand. These molds can be mounted on any one of the machines that Alpha Systems has. Each machine has 4 arms, and on each arm up to six molds can be mounted. To simplify the problem, we assumed that each item takes one hour to go through the molding process.

We were given daily sales data for the division of Alpha Systems that runs the rotational molds and blow molds. Discussion with Alpha Systems directed us into modeling production on a weekly basis. Under normal circumstances Alpha Systems operates these molds 5 days a week, 24 hours a day, making a weekly production schedule more applicable than a daily schedule.

We were not provided any profit data to possibly provide a quantity to optimize. They stressed that their priority was to consistently fulfill demand in order to satisfy their customers and grow their business. As a result, this was also our model's top priority. Before the optimization begins, our model determines the products for which it is impossible to meet the demand that week, even when using inventory from the warehouse. For these products, the molds are run continuously so as to keep the deficiency in supply as little as possible. These products are essentially deleted from the linear programing problem.

Unfortunately, due to the lack of profit data, it was not immediately obvious what should be the objective function for the model. What we decided on was to prioritize products with higher sales, by assigning a fictional "profit" equal to the average weekly sales for that item. The objective function is therefore the sum of the "profit" made by running certain molds for certain times.

5.1. **Model formulation.**

Our decision variables were simply the number of hours to run each product's mold(s) for the week in question. The limited number of molds for each product provides a upper bound 3 to these decision variables. In addition, the total number of mold-hours is limited by the 4 machines they have (there are nowhere near enough machines to run all the molds simultaneously) (represented by 2. Alpha Systems specified that it is undesirable to run a mold for less than twenty-four hours, due to the time it takes to replace one mold with another (approximately 30 minutes to an hour). Our model accounted for this by introducing two sets of binary decision variables, β_i and γ_i. Different combinations of the values of β_i and γ_i allow for the program to distinguish between running a mold for 0 hours, running a mold for between 1 and 23 hours, and running a mold for at least 24 hours by including the two constraints 6 and 7

Although it would be a rare occurance, it is possible that there are not enough machines to satisfy demand for every product, even if there are enough molds. Because of this possibility, a set of binary decision variables α_i was utilized and added to the constraint 4

forcing demand to be satisfied. If demand cannot be met, α_i is forced to equal 1, and a penalty of 10 million is assigned to the objective function 1. This penalty is high enough that demand will always be satisfied if it is at all possible.

We were told that keeping no more than two week's worth of inventory in the warehouse was desirable. This provided us with an upper bound 5 on the new inventory in the warehouse, preventing the model from running wild by producing too much of a certain product. It is worth mentioning that in a daily inventory that Alpha Systems provided us as an example, this rule was violated for the majority of their products.

A mathematical description of the model is shown below.

Maximize

$$\sum_i p_i(v_i + i_i - d_i) - 10000000\alpha_i \tag{1}$$

Subject to

$$\sum_i v_i + 6 - 6\gamma_i \leq capacity_{max} \tag{2}$$

$$v_i \leq 120m_i \tag{3}$$

$$v_i + i_i \geq d_i - 10000\alpha_i \tag{4}$$

$$v_i + i_i - d_i \leq D_i \tag{5}$$

$$24 * \beta_i \leq v_i \tag{6}$$

$$v_i \leq 24 + 2400\beta_i - 24\gamma_i \tag{7}$$

Here, i corresponds to a mold, where the sum is over all molds (for which demand can be met), and each constraint holds for all i. In addition, p_i is the "profit" of each mold, v_i is the weekly production for each mold, i_i is the inventory for each mold at the beginning of each week, d_i is the weekly demand for that mold, and D_i is the desired inventory for that mold (which is equal to the maximum of the two-week average, the current contents, and 1). The constant $capacity_{max}$ is the maximum number of total hours that all the machines at Alpha Systems can be worked for (that are reserved for this division). Its numerical value has not been reproduced here for confidentiality. The 10000000, 10000, and 2400 are all "big M" type numbers (such as described in the popular textbook [3] by Hillier) that were chosen large enough that they are effectively infinity for the purposes of this model.

5.2. **Performance and limitations.** Overall, the model performs reasonably well. Alpha Systems makes many extremely low volume products. Because the model equates sales with profits, the end result is that low volume products are not kept in inventory (because the model will try not to "waste" production time on such low "profit" items) and only the minimum number required to fulfill demand that week will be produced. This works well because low volume products often go months without any orders, so keeping stock of them in the warehouse would be a waste of space, and aligns well with Alpha System's

goal to turn over inventory every few weeks. On the other hand, the model is constantly producing as much of the high-volume products as possible. This is also not a problem because the high demand for those products means that the excess stock in the warehouse experiences rapid turnover. The model also indicates that there are some products for which additional molds would be useful. For example, there are two products for which the long-term average demand slightly exceeds molding capacity.

The main limitation of the model is that it does not take into account the small amount of time (around 30 minutes to an hour) it takes to change the rotational mold. In fact, this time is much higher because in order to change one mold an entire arm (which can hold up to six molds) must be stopped. This was judged to be a reasonable sacrifice considering that the 1 hour it takes to mold each product is a very rough average and the variance in molding time would outweigh the time wasted changing molds. In addition, there is some free time built into the model because there is a six hour penalty for running a mold for less than 24 hours. The scheduling of which molds go on which machines at what time is done by a separate model, discussed in the next section.

6. Optimizing scheduling using linear programming

Alpha Systems expressed interest in having some kind of automated production scheduling that determines when, and on which machine a mold is run. Currently this task is performed manually using a calculator and spreadsheet which is time consuming. Automating this process would save a lot of time while also giving a better optimized production distribution among the molding machines.

Each arm on a machine is independent from other arms on the same machine. That is, any arm can be stopped for mold changes without interrupting the production on other arms. However, the mounts on an arm are dependent in that any stop to a mount (running a mold) stops every mount on the same arm. Thus, each mold change reduces the maximum production capacity (within a time frame) by the number of mounts on the arm times the amount of time used for the mold change. For a near capacity, weekly production run, a typical number of mold changes (40-100) can lead to a significant reduction in maximum production capacity (240-600), which in turn can results in production not meeting demand.

With this in mind, the optimization is broken into two stages. The first stage allocates the production time for each product on each mount, with the objective being maximization of the production given the aforementioned constraints of the molding machines. The second stage minimizes the number of mold changes subject to the production equalling the solution of the first stage. Afterward, general programming is used to turn the production allocation into a schedule.

It should be noted that, in theory, it is possible to combine the two stages, or even the profit based and scheduling optimization, into one linear program. However, this would result in the model being significantly more difficult to solve, limiting our choice of solvers to only the most powerful (and expensive) ones. Another disadvantage is having the more difficult part of the problem 'holding back' the overall result. For example, for a given set of input, the solver might have no trouble finding an optimal solution for production

maximization, but only a good enough solution for the mold changes minimization in an acceptable time. Combining the two might result in only a good enough solution with respect to both of them.

6.1. **Model formulation.** To model the production allocation of the molding machines, we let the decision variables x_{ijk} correspond to the amount of time (in hours) to run a mold of product k on mount j of arm i. The assumption is that each mold is run at most only once without moving a mold from one mount to another, or dismounting and remounting the mold to the same mount later on. According to Alpha Systems, the mold change and molding times are both 1 hour. To keep track of the number of mold changes, we introduce binary variables a_{ijk} and add the constraints $x_{ijk} \leq h * a_{ijk}$, where h is the maximum available production time per mount, which forces a_{ijk} to be 1 whenever $x_{ijk} \geq 1$. Since the number of molds for a product is limited, the constraints $\sum_i \sum_j a_{ijk} \leq m_k$ is added, where m_k is the number of molds available for product k. To keep the total running hours on each mount under the maximum available time, we need the sum of the running time of every product on the mount to be less than or equal to the maximum available time minus the total time used for mold changing on the same arm, or

$$\sum_k x_{ijk} \leq h - \sum_j \sum_k a_{ijk}.$$

To summarize, let I be the number of arms, J the number of mounts per arm, and K the number of products. We then have:

$$x_{ijk} \geq 0 \tag{8}$$

$$x_{ijk} - h * a_{ijk} \leq 0 \tag{9}$$

$$\sum_i \sum_j a_{ijk} \leq m_k,\ k \in \{1, 2, ..., K\} \tag{10}$$

$$\sum_k x_{ijk} + \sum_j \sum_k a_{ijk} \leq h,\ i \in \{1, 2, ..., I\}, j \in \{1, 2, ..., J\} \tag{11}$$

To allow for the first mold on a mount to not count towards the total mold changes (suppose molds are mounted before the weekly production run) we introduce binary variables b_{ij} whose values are 1 if one or more molds are mounted to the mount j of arm i, and 0 otherwise. We then replace the constraints 11 with 14.

$$\sum_k a_{ijk} \geq b_{ij} \tag{12}$$

$$\sum_k a_{ijk} - h * b_{ij} \leq 0 \tag{13}$$

$$\sum_k x_{ijk} + \sum_j \sum_k a_{ijk} - b_{ij} \leq h,\ i \in \{1, 2, ..., I\}, j \in \{1, 2, ..., J\} \tag{14}$$

For the first stage we maximize the overall production with the upper bound for the production of each product being the desired production given by the optimal solution of

the profit based optimization. Thus, letting v_k be the desired production of product k,

$$\sum_i \sum_j x_{ijk} \leq v_k,\ k \in \{1, 2, ..., K\}. \tag{15}$$

To ensure that production meet weekly demand, we add another set of binary variables c_k and force it to equal to 1 when the production of product k is less than $min\{v_k, d_k\}$, where d_k is the weekly demand of product k as shown in 16.

$$\sum_i \sum_j x_{ijk} + I * J * h * c_k \geq min\{v_k, d_k\}. \tag{16}$$

The value $min\{v_k, d_k\}$ is used because it is the number determined by the profit optimization program to be sufficient at meeting the weekly demand if it is at all possible. Also note that $I * J * h$ is the weekly production capacity of all the molding machines combined.

Since the objective of the first stage is to maximize production, we let the objective function be the sum of all the productions on all mounts and arms with a huge penalty for each product not meeting demand.

The complete program for the first stage is as follows,

Maximize

$$Z = \sum_i \sum_j \sum_k x_{ijk} - I * J * h \sum_k c_k$$

Subject to

$$\begin{aligned}
x_{ijk} - h * a_{ijk} &\leq 0 \\
\sum_i \sum_j a_{ijk} &\leq m_k,\ k \in \{1, 2, ..., K\} \\
\sum_k a_{ijk} &\geq b_{ij} \\
\sum_k a_{ijk} - h * b_{ij} &\leq 0 \\
\sum_k x_{ijk} + \sum_j \sum_k a_{ijk} - b_{ij} &\leq h,\ i \in \{1, 2, ..., I\}, j \in \{1, 2, ..., J\} \\
\sum_i \sum_j x_{ijk} &\leq v_k,\ k \in \{1, 2, ..., K\} \\
\sum_i \sum_j x_{ijk} + I * J * h * c_k &\geq min\{v_k, d_k\}
\end{aligned}$$

and

$$\begin{aligned}
x_{ijk} \geq 0, \quad a_{ij} \geq 0, \quad b_i \geq 0, \quad c_k \geq 0 \\
(x_{ijk} \text{ integer}, \quad a_{ij},\ b_j,\ c_k \text{ binary}).
\end{aligned}$$

For the second stage, we set v'_k to $\sum_i \sum_j x_{ijk}$ from the optimal solution of the first stage, then replace constraints 15 with

$$\sum_i \sum_j x_{ijk} = v'_k,\ k \in \{1, 2, ..., K\} \tag{17}$$

Since the objective of the second stage optimization is to minimize the mold changes, the complete formulation becomes:

Minimize

$$Z = \sum_i \sum_j \sum_k a_{ijk} - \sum_i \sum_j b_{ij}$$

Subject to

$$\begin{gathered}
x_{ijk} - h * a_{ijk} \leq 0 \\
\sum_i \sum_j a_{ijk} \leq m_k,\ k \in \{1, 2, ..., K\} \\
\sum_k a_{ijk} \geq b_{ij} \\
\sum_k a_{ijk} - h * b_{ij} \leq 0 \\
\sum_k x_{ijk} + \sum_j \sum_k a_{ijk} - b_{ij} \leq h,\ i \in \{1, 2, ..., I\}, j \in \{1, 2, ..., J\} \\
\sum_i \sum_j x_{ijk} = v'_k,\ k \in \{1, 2, ..., K\}
\end{gathered}$$

and

$$\begin{gathered}
x_{ijk} \geq 0, \quad a_{ij} \geq 0, \quad b_i \geq 0 \\
(x_{ijk} \text{ integer}, \quad a_{ij},\ b_j \text{ binary}).
\end{gathered}$$

6.2. **Possible Improvements.** Assume that the molds are not mounted before the weekly production run so that the first mold to be mounted on each mount takes up the usual 1 hour from the available time, and that they are left mounted at the end of the week so that it can be run right away the following week.

To simulate these scenario, add dummy variables a_{ijd} to the model, and let every expressions that sums over a_{ijk} also include a_{ijd} where appropriate. If mount j_0 on arm i_0 is empty, fix $a_{i_0 j_0 d} = 1$. Suppose mold k_1 is already mounted on mount j_1 of arm i_1, simply fix $a_{i_1 j_1 k_1} = 1$. If we wish to produce product k_1, the model will choose $1 \leq x_{i_1 j_1 k_1} \leq p_{k_1}$ over running the mold k_1 at some other mount.

6.3. **Limitations.** The main limitation of this model is that it does not account for possible conflict when mold changes on different arms happen at the same time. Alpha Systems stated that they only have one worker who performs the mold changes, thus the model should only allow one mold change at a time. The difficulty stems from the fact that the point in time where any mold change on a mount happens is also determined by preceding mold changes on other mounts on the same arm; a relationship that is extremely difficult to model with linear programing.

The second limitation comes from the assumption that each mold is mounted only once. In fact, it is possible to increase production by switching some molds to different mounts throughout the production run. The difficulty is similar to the above case in that we would have to determine when a mold is run, whereas the current model only determines how long a mold is run. However, in our test with Alpha Systems sales data, this limitation only becomes an issue when the desired production is near maximum capacity, and even then, the production time lost due to this issue is only around 1% of the desired production (the majority of the lost time is due to mold changes).

7. Conclusion and Future Work

7.1. **Conclusion.** We were able to automate the production of Alpha Systems by three linear programming problems.This was done using the programming language Python, but the program can be packaged as an executable so that no programming experience is required to use it. In addition to automating the production and scheduling, our program records inventory and identifies which products require special attention, either by outsourcing production or running the factory overtime. We hope that our program proves helpful to the tanks division of Alpha Systems in the long run and performs better than time-consuming scheduling by hand, while simplifying the maintenance of company records. With a flexible setup allowing for new products and new molds to be included in Alpha System's lineup, the program should remain usable barring major changes in production methods. Such future changes would likely require only minor adjustments to the program. If the program is effective enough, a similar scheduling program for another division of Alpha Systems could be constructed, as discussed in the remainder of this paper.

7.2. **Future work.** The shutters and siding division of Alpha Systems, similarly to the tanks division, had no automated scheduling algorithm in place and instead managed production by hand, and hence expressed interest in a similar program for the shutters and siding division. Although it is possible that work could be done on that in the future, our early analysis of the problem suggests a much more difficult linear programming problem would be in store.

The injection molds are run on an array of about twenty machines. Each machine is different, and can only operate a limited selection of molds. This is in contrast to the rotational molds, where each machine is functionally identical and hence the assignment of mold to machine does not matter in the initial optimization.

Holding tanks are utilitarian objects and hence only came in one color. Shutters and siding, however, are decorative, and Alpha Systems has hundreds of color options for each

of those. This would add additional complexity to the model, since the number of products would be much higher and there would not be distinct molds for each product. In addition, changing injection molds is a much lengthier project than for changing rotational molds and that would have to be accounted for in the model.

8. Acknowledgements

- PIC Math is a program of the Mathematical Association of America (MAA). Support for this MAA program is provided by the National Science Foundation (NSF grant DMS-1722275) and the National Security Agency (NSA).
- Dr. Peter Connor, for overseeing this research project as a PIC Math faculty member.
- Alpha Systems, Jeff Taber, and Mike Odiorne, for allowing us to tour the premises and supplying us with all the necessary data to make this project possible.

References

[1] Alpha Systems *Welcome to Alpha Systems, LLC*, (n.d.). Retrieved from https://www.alphallc.us/about/

[2] Stack Overflow, *Stack Overflow Developer Survey 2019*, (2019). Retrieved from https://insights.stackoverflow.com/survey/2019

[3] Hillier, F. S., *Introduction to operations research*, Mcgraw-Hill Education (2014)