Grocery Gus or: How I Learned to Stop Worrying and Love the Log Card

Author: Gus Lipkin

Date: 21 April 2021

Table of Contents

Introduction and Background

Author's note

Defense contractors generally don't like when you go into too much detail about what you do or how their company works, even if it is for a paper for school. As such, I will be describing where I work and my job as a supermarket.

My Origin Story

I was originally hired as an intern with MiGrocery with the express purpose of collecting information on items that were delivered to be stocked in our canned goods department. Each canned good is supposed to have a log card with it. Each log card describes where the product was made and who handled it along with any climate conditions so we know if the product is safe for the sales floor. I was given a list of SKUs and a login for a piece of software we will call GrocerDocs which allows for realtime tracking of each SKU and a serial number for each unit of that SKU.

Unfortunately, the people who stock the warehouse and are not very organized and items of the same SKU do not always end up stocked together. In addition, just because GrocerDocs says an item is in a location does not mean it will actually be there when I look for it and each item may not have its required log card with it. Someone else could have taken the item without updatingGrocerDocs or taken the log card and not uploaded it to GrocerDocs, or both.

GrocerDocs will tell me where each item is, but there's no way to check and see all of the locations I will need to visit without checking the location atribute for each SKU which could have several hundred serial numbers scattered about the warehouse where I work and on the sales floor that I do not have access to. I quickly realized that going SKU by SKU and searching all the different locations that for each specific serial number was inefficient. I found a way to export GrocerDocs' complete inventory and used R to extract a list of every serial number for every SKU I needed to find and order the list by location and export this information to a spreadsheet. I also added some columns to help me better keep track of product as it moved around the warehouse and so I knew which items I had already looked for and which ones I had uploaded. With my new program, I could now sync with GrocerDocs every morning to get the newest locations. Once I showed this to my supervisors, they thought I was a "tech wizard". Thus began the onslaught of spreadsheet generator requests.

The Spinoff Series

My supervisor, the canned food department head, wanted a spreadsheet that had the item names and SKUs down the left and the item serial numbers listed horizontally with each serial number cell highlighted based on its status. She had recently spent several days creating this spreadsheet but since item status can change, it was no longer up to date and she needed a new one. Creating it by hand would take me several days each time, but automating it would only take several days once. Once completed, she showed the other department managers and the bakery manager decided he wanted one too. Then, the district manager decided he wanted me to make a spreadsheet that was very similar but only listed items on the sales floor. The catch was that he wanted a single file for each department with a worksheet for every store. While there is no list of what each store should have on the sales floor, I could make a good effort at figuring it out by comparing across stores and making sure every store has the same number. Unfortunately for him, it was my day off and I was not going to accept my regular hourly rate. Ultimately, we agreed on a bonus of five hours of pay for coming in and creating the document.

At this point, my supervisor, the canned goods department head, realized that I had not done any work for her in several weeks because I was working on generating spreadsheets of all sorts for other department heads. She wants me back on the log card project. By creating the location sorted spreadsheet of every item I need to find, I have already optimized the process as much as I can.

The Dilemma

Is it more economical to keep me on the log card project full time, keep me on the log card project and loan me out occassionally, or switch me to special projects full time?

Analysis

Expected Marginal Utility

As we will establish in *Profit Maximization*, I provide no added value to the log card project over any other person performing the same task. I will concede, of course, that I already know which buttons to press to update my spreadsheets and no one else does, but those tasks are easy to teach in under fifteen minutes and can therefore be ignored. Because I provide no added value to the project, when my supervisor asks me to work on log cards the company has a marginal utility of zero while I am doing that work.

What is the company's expected marginal utility for each unit of time I am on a special project? This is relatively easy to answer because of the nature of the special projects I am assigned to. Each special project has progress that can be measured by tasks completed or percent completion when done by hand such as transcribing part numbers into a spreadsheet or trying to find duplicate part numbers. However, when writing a program that does these tasks it is exceptionally rare that a computer running the program would take as long as a human would to do the same task.

`Assumption 1: A computer program will always take less time to complete a given task than a human will`

Of course, that program does take time to write. It takes me about a week to write a fully-fledged solution to any special project. If I am able to re-purpose existing code for a new project, that time is cut down to just one or two days.

`Assumption 2: I will take a week to finish writing a program to complete any given task`

It should now be clear that if any special project is expected to take more than a week to complete, I should be tasked with writing a program to do the task. If the task is expected to be repeated, I should be asked to write a program if the total time spent on the task is more than a week. It may take me more time in the beginning, but at some point the programming time and task time will reach a breakeven point at which point the program will be more efficient.

These assumptions, however, assume that MiGrocery has only one person working on these projects, either me or someone else. I will always take a week to complete a given task, but should they choose to hire a small army, essentially mult-threading the process, the task could be completed in a matter of hours. This problem has a natural solution that the total amount paid to employees to complete the task should be minimized regardless of it is paying one person for many hours or many people for one hour. Alternatively, if a task is crucial to company operations, they could choose to pay more people to complete it more quickly rather than wait for me to construct a solution.

Profit Maximization

Another, more concrete way, to calculate the value I add to the team is to try and maximize the companies profit by optimizing which projects I am put on and which projects I could be put on that someone else is put on.

The model of profit for a single given project is below. t is the unit of time at which a portion of the completed project is usable and adds value in operations. We start by finding the value of the portion of the project that was completed while I was writing my code where we start at time $t = 1$ because we are measuring actionable time units and not real time. We are taking the sum up to n which is just before the time that I am able to run my code and is greater than or equal to t. The next portion is the rest of the net present value function left intact where R_t represents the cash flow in time t. In this case, i represents the diminishing value of the project as time passes and the marginal rate of return of the project as it nears its completion.

$$\pi = \sum_{t=1}^{n}[\frac{R_t}{(1+i)^t}]$$
$$n \geq t$$

The second term represents the amount of value I provide by completing my code at time n and is also modeled using the net present value formula. This only has one term and is not summed because the cash flow R_n is reprentative of the value added by having the rest of the project completed.

$$\pi = \sum_{t=1}^{n}[\frac{R_t}{(1+i)^t}] + \frac{R_n}{(1+i)^n}$$
$$n \geq t$$

The third term represents the amount needed to pay the employees who work on the project by hand before my program is completed and also my pay for the same term. x is the number of employees including myself, p is their pay rate per hour, and h is the number of hours per time unit t.

$$\pi = \sum_{t=1}^{n}[\frac{R_t}{(1+i)^t}] + \frac{R_n}{(1+i)^n} - (xphN)$$
$$n \geq t$$

Something important to note here is that me writing the code in t time will always be more profitable than waiting for the other people to do the same task in $t+1$ time. In order to maximize this, we find the first derivative and find the maximum point of the line. We will leave the maximization here and move to balancing multiple projects.

$$\frac{d\pi}{dt} = \sum_{t=1}^{n}[-\frac{ln(i+1)r}{(1+i)^t}] - \frac{ln(i+1)r}{(1+i)^n}$$
$$n \geq t$$

From here, we can take this in two directions. The first would be to change the model to reflect that some projects are one-offs and some are repeated. The second would be to do a direct comparison with another modeled project to learn where my talents can best be applied.

One-offs vs Repeated Projects

A one-off project is something that once completed, is used once and then never again. A repeated project is performed multiple times. This could be every week, month, or other time period, or on an as needed basis. I've already established that profit for a one-off project is modeled with

$$\pi = \sum_{t=1}^{n}[\frac{R_t}{(1+i)^t}] + \frac{R_n}{(1+i)^n} - (xphN)$$
$$n \geq t$$

The first iteration of a repeated project will always stay the same as the above model because the company still wants its results as quickly as possible. If it is going to take me more than one period to complete, we can multiply the entire equation by the number of periods it will take me to complete the program. Let's call that number ρ.

$$\pi = \rho\{\sum_{t=1}^{n}[\frac{R_t}{(1+i)^t}] + \frac{R_n}{(1+i)^n} - (xphN)\}$$
$$n \geq t$$

Once I've finished the program, there is no diminishing value in waiting for the code to run. Thus, the only change to the function is added cost of paying someone to hit run on the program which is negligible.

Direct Comparison of Projects

This approach is relatively easy. To make this easier to understand, let's say there are two or more projects happening simultaneously and I can only work on one at a time. After estimating how long it will take me to write a solution based on prior knowledge of the tools needed or having written programs that can be adapted to the new tasks. We can put the profit maximization function to work and use the time it would take me to write the program as $t + 1$. Then, we do the same for the other projects in consideration. Whichever one has the highest potential profit is the one that I should be applied to. This can be taken a step further by taking into consideration whether or not the projects in question is a one-off or a repeated project.

Conclusion

I may be tooting my own horn, so to speak, but I do not think that MiGrocery could have hired a better and more productive intern. I have shown time and time again that I am able to use skills in my toolset to expertly optimize current workloads and to automate tasks that could take weeks to complete by hand. Through profit maximization techniques with estimated numbers, I have shown that I can be more than ten times as productive as an employee doing the same task by hand. In discussing expected marginal utility, we learned that it makes sense for me to take on any special project that is expected to take more than a week to complete or that is an ongoing project. Because I am so much more productive on special projects, it makes sense for me to be switched to those full time rather than kept on log cards and loaned out to special projects on occasion.