ISyE 6740 – Spring 2021

Project Report

Team Member Names:

Noelle Richards

GTID: 903645972

Project Title:

Recipe Optimization for *Animal Crossing: New Horizons*

Problem Statement:

Animal Crossing: New Horizons (ACNH) is a Nintendo Switch that took the U.S. by storm when it was released in March 2020. According to information published by Nintendo itself, it is the second-best selling Switch game of all time ("Financial Results”).

In the game, the player creates a character and meets Tom Nook, a racoon with a predatory business sense. He allows the player to purchase an island getaway and sets them up with the basics, such as a tent. The player can then progress nonlinearly through the game, doing as they please.

A major component of the game is the economic system, almost entirely controlled by the aforementioned Nook using the currency of Bells. He is the only merchant in town and any item a player wishes to sell or buy must come from him (or his two nephews, Timmy and Tommy) at static prices. Another primary part of the game is the crafting of items. The player unlocks recipes and can use them to create items, using materials found around the island. As an intrepid data science student, I found myself wondering if I could optimize my use of materials to maximize my profit.

With some research (read: game hours), I have refined my initial idea further. Given a set of materials inputted by the user, I want to build a model that maximizes the difference between the revenue of selling an item and the revenue of selling the materials as is.

Data Source:

On Kaggle, there is a dataset of all the possible recipes in ACNH and their required materials, as well as their sell prices. I will need to add all possible materials and their sell prices. The dataset is very sparse with the dimensions of 595x24, with not all columns relevant for this project. Each recipe is broken down as follows:

Name	#1	Material 1	#2	Material 2	#3	Material 3	#4	Material 4	#5	Material 5	#6	Material 6	Buy	Sell
flower stand	1	red roses	2	pink roses	2	orange roses	2	white liles	1	yellow lilies	4	purple windflowers	NFS	200

Materials are listed as well as the quantity of each material needed. At most, a recipe will require six different types of materials. The data set can be found at the following link:

https://www.kaggle.com/datasets/jessicali9530/animal-crossing-new-horizons-nookplaza-dataset?select=recipes.csv

I verified the dataset using a cursory Google search and found that 25 recipes had been added since the dataset was last updated. I also found that while the recipes were accurate, the sell prices were not. I used Nookipedia (a "community-drive *Animal Crossing* wiki encyclopedia") to look up each of the 615 recipes, verify their correctness, and record the sell price.

I wanted maximize profit, not just revenue, which in this case means subtracting out the sell price of the raw materials (i.e. if the player doesn't do any crafting and just sells what is in their inventory). This information was not available in the Kaggle dataset, so I used the recipe dataset to find all the unique materials used, then looked up the sell price of each on Nookipedia.

The final issue I noticed with the dataset is that some recipes require other recipes in order to be crafted. For example, a DIY Workbench requires:

- 5 wood
- 5 hardwood
- 5 softwood
- 3 iron nuggets
- 1 Mini DIY Workbench

The Mini DIY Workbench is a separate recipe on the list requiring:

- 3 wood

- 3 hardwood
- 3 softwood
- 2 iron nuggets.

This program assumes that the player is only inputting the raw materials, rather than other recipes, as it is more likely that the player will have those in their inventory. In order to account for this in my data, I added all the raw materials for the intermediate recipes to the final recipes, i.e. a DIY Workbench now requires:

- 8 wood,
- 8 hardwood
- 8 softwood
- 5 iron nuggets

Methodology:

The program solves the following optimization problem:

$$max \quad \boldsymbol{SX} - \sum_{i=1}^{n} \boldsymbol{R}_i x_i \boldsymbol{P}$$

$$s.t. \quad \sum_{i=1}^{n} \boldsymbol{R}_i x_i \leq \boldsymbol{K}$$

$$x_i \geq 0, \qquad \forall i = 1, \ldots, n$$

$$x_i \in \mathbb{Z}, \qquad \forall i = 1, \ldots, n$$

Variable Definition:

- $\boldsymbol{S} = [s_1, \cdots, s_n]^T$: Vector of the sell price of each recipe
  - s_i is the sell price of recipe *i*
- $\boldsymbol{X} = [x_1, \cdots, x_n]^T$: Vector for the amount of each recipe that should be made in an optimal solution
  - x_i is the number of times recipe *i* should be made
  - k_j is the number of material *j* to be used in recipes
- $\boldsymbol{P} = [p_1, \cdots, p_m]^T$: Vector of the sell price of each material
  - p_j is the sell price of material *j*

- $\boldsymbol{R} = \begin{bmatrix} r_{11} & \cdots & r_{1m} \\ \vdots & \ddots & \vdots \\ r_{n1} & \cdots & r_{nm} \end{bmatrix} \rightarrow \boldsymbol{R}_i = [r_{i1}, \cdots, r_{im}]$: A matrix of consisting of all recipes and their required materials
    - $\boldsymbol{R}_i$ is a vector of the required materials for recipe *i*
- $\boldsymbol{K} = [k_1, \cdots, k_m]^T$: Vector of the amount of each material available, to be inputted by the user
- *n* : Total number of recipes
- *m* : Total number of materials (types, not quantities)

Initially, I was anticipating having to use column generation to avoid having to iterate through each possible recipe combination, but the dataset is small enough that the computation time is only a few seconds. There are two csv files that the program uses. The first is "items.csv" which contains the columns "Material" and "Sell Price". This is where the vector $\boldsymbol{P}$ comes from. The other file is named "recipes.csv" and it contains the following columns: "Name", "Material #i" and "#i" for i=1,…,7, and "Sell Price".

Another deviation from my original supposed methodology is that my objective equation was $max \quad \boldsymbol{SX} - \boldsymbol{KP}$. This would mean that the price of raw materials would be subtracted from the revenue, regardless of whether they were used. For example, a golden candlestick requires 2 golden nuggets to craft, and is sold for 20,000 Bells. However, the 2 golden nuggets would sell for 20,000 without any crafting. Using the old objective equation, my program recommended crafting that was unnecessary because the addition in the first term would cancel out the cost of those raw materials in the second term. (i.e. without crafting the candlestick, the objective equation would be 0-20,000=-20,000. With crafting the candlestick, the objective value would be 0.) By using the revised equation, I only subtract the sell price of the materials actually used in the recommended equations, allowing for leftover materials if it makes no sense to use.

Evaluation and Final Results:

The ultimate measure of success for this project is if it works. In more quantifiable terms, this optimization should make more money than a default action of the player. In this case, I will assume the default action is to make as much of the most expensive item possible. The optimization solution should suggest to the player how much of each recipe they should make and that amount should be worth more of the in-game currency (Bells) than the default action.

In order to calculate the default action's profit, I found all the feasible recipes and the highest sell price among them. I then calculated how many times one could make the default recipe and the profit.

To test the optimization model, I generated 30 random vectors (with values from 0 to 10) ***K*** as input from the user, found the number of Bells expected from the default action, and compared to the number of Bells expected from the optimization solution. The below table shows the results:

Trial	Optimization Profit	Default Profit	Difference
1	135440	0	135440
2	171894	26756	145138
3	107544	0	107544
4	164725	53512	111213
5	377203	208575	168628
6	200059	40134	159925
7	108888	19800	89088
8	111734	0	111734
9	131843	0	131843
10	110598	13378	97220
11	105237	19800	85437
12	160512	13378	147134
13	355024	208575	146449
14	229063	0	229063
15	398784	208575	190209
16	108877	0	108877
17	148075	0	148075
18	199968	80268	119700
19	167323	0	167323
20	290408	208575	81833
21	125748	0	125748
22	425328	208575	216753
23	184742	80268	104474

Trial	Optimization Profit	Default Profit	Difference
24	148521	0	148521
25	113518	0	113518
26	349185	208575	140610
27	179084	0	179084
28	192621	93646	98975
29	122758	13378	109380
30	106199	0	106199

Based on this metric, the optimization program is an overwhelming success. There are a surprising amount of 0's in the default column but after some investigation, I determined most of them were "golden" recipes, which make a net zero profit and often have the highest sell price.

Secondary measures of success will be user friendliness and computation time. If it takes more time to input all the materials a player has into the model than it does to just make items and get more materials, it won't be worth using. The current solution is that all raw materials (140) are listed in a csv file named "userMats.csv", making it easy to CTRL-F or sort and filter for the materials the player wants to input. The program assumes that any space left blank means that there is none of that material available.

My original goal is to create some sort of process that allows a lay person to input their materials and returns the results of the optimization in less than 5 minutes. I chose 5 minutes because I personally would give up on the program at that point and just do my own crafting and selling, regardless of optimization. The optimization averages about 1.5-2 seconds to run fully, including showing the default alternatives. There are only two steps for the end user:

1. Enter available materials into "userMats.csv" and save
2. Open Anaconda prompt (or something else that will run Python scripts) and enter `python Project.py` while in the appropriate directory.

While these two steps take some computer literacy, a cursory survey of friends and family told me that most people would feel comfortable once trained on it or just reading the "README.txt" file.

Overall, I have achieved my optimization, user friendliness, and computational goals with this project and plan to continue using it to profit in-game.

Citations:

"Financial Results Explanatory Material 3rd Quarter of Fiscal Year Ending March 2022" (PDF). Nintendo. Retrieved February 3, 2022.].

Nookipedia. (2022, April 2). Main page. Nookipedia. Retrieved May 1, 2022, from https://nookipedia.com/wiki/Main_Page