Optimization of the Student Diet

Me 575

Wesley Holt and Bradley Davis

Abstract

A college student’s diet is unique and complex. As college students, we are always broke, often need late night energy, and usually don’t have much time to prepare meals. Our project was to optimize the college student’s diet. We minimized cost given a constraint function of recommended food groups such as vitamins, calories, fiber, protein, etc.. As grocery pick-up services and online grocery shopping grow in popularity, an algorithm that finds the optimal diet could be used to create a shopping list to be sent to the local grocery store, saving the customer both time and money.

We were able to minimize our costs and obtain an optimal grocery list multiple ways. These included modeling all design variables as continuous, a genetic algorithm, and branch and bound methods. Our best and final method runs fmincon repeatedly and eliminates irrelevant design variables (foods) until it has a small pool to work with. We then run a genetic algorithm to find the optimum foods to buy with minimal cost while accounting for health constraints and a satisfaction constraint to make sure you like the foods you're getting.

Introduction

The goal of this project was to create a useful optimizer that helps a student understand how to eat healthy on a limited budget. As we began our research and development a few possible useful applications we thought of could be implementing an optimizer like ours into a grocery pick up, where someone could put in their budget and get food recommendations based on their past preferences and monetary limitations.

As we began researching other similar work that had been done in this field we found three relevant journal articles in our research. In the first article, *College Students and Eating Habits: A Study Using An Ecological Model for Healthy Behavior*, the factors that contribute to how college students determine their diets are analyzed. The second article, *Linear Programming: A Mathematical Tool for Analyzing and Optimizing Children's Diets During the Complementary Feeding Period*, outlines an attempt at using mathematical tools to optimize young children's diets at an early stage in life. Finally, in the third article, *Optimal Individual Diet*, an optimization project similar to the one we did is described, where the optimal diet is determined by minimizing cost subject to a variety of nutritional constraints.

Throughout this paper we will show the different methods with which we were able to accomplish our goal of optimizing a low cost healthy diet, how we progressed through the problem and handled unexpected difficulties and finally arrived at a finished product.

Methodology

Our optimization problem can be summarized as the following:

Minimize *cost*

With respect to *the different foods to be purchased*

Subject to *nutrition and satisfaction constraints*

Our objective function took in the amounts of each food (design variables) and outputted the cost in dollars to purchase that amount of food. The nutrition constraint function needed to output value(s) representing how healthy a given list of foods is. Likewise, our satisfaction constraint function needed to output a value(s) representing how satisfied the user would be with a given list of foods.

Design Variables

We used data from the USDA's *2015-2016 FNDDS At A Glance - FNDDS Nutrient Values* database, which contained a list of over 8,000 different foods and drinks, with the nutrition facts for each. The design variables are inherently discrete—one can only purchase discrete quantities of most foods. Some of our initial optimizers, however, treat the variables as continuous.

The Nutrition Model

The first challenge in our project was to develop a model to quantify the nutrition of a given diet. Our first model incorporated the FDA Recommended Daily Value Nutrition Facts. Because we are targeting this optimization tool for college students, we used the minimum and maximum recommended nutrition fact quantities for a male young adult. Examples of these types of constraints for a 1-day diet included 2,000 < *total calories* < 3,000, or *grams of saturated fat* <

30. A given diet would be nutritionally feasible if the total nutrition fact values of all the foods met the FDA guidelines.

As we tested this model, we realized that the optimizer would maximize 1 or 2 design variables and leave almost all others untouched. The “optimal” diets it would find would be very unbalanced from a nutrition standpoint (e.g. 2 pounds of bananas, and less than an ounce of everything else). To counteract this, we improved upon our model by adding food group constraints. A food group constraint would force the optimizer to choose diets that had significant portions of foods in more than just one food group (e.g. must have 4 oz of vegetables, 4 oz of dairy, etc.).

While the first two models for nutrition gave us reasonable, balanced diets, they greatly simplified all the complex factors that determine a human’s nutrition. Our third and most current model for nutrition is slightly more sophisticated, as it calculates a nutrition metric known as the Healthy Eating Index (HEI). The HEI is widely used in nutrition research today, and it offers a more complete view on the quality of a diet. An HEI score ranges from 0 to 100, and it measure diet quality by considering 13 different categories:

- Total fruits
- Whole fruits
- Total vegetables
- Greens and beans
- Whole grains
- Milk/dairy
- Total protein foods
- Seafood & plant proteins
- Fatty acid ratio
- Refined Grains
- Sodium
- Added sugars
- Saturated fats

We specifically used the HEI-2015, which is the most recent version of the index. The average American diet has an HEI score of 49. We chose to set the optimizer to require an HEI score of at least 65.

Each of our three models for nutrition (*Nutrition facts*, *Nutrition facts + food groups*, *HEI*) has advantages and disadvantages. Our first two models are linear, which made the optimization much simpler. However, they were less accurate at estimating diet quality. Our third model using HEI was more accurate, but it was non-linear, which made the optimization slower and more difficult.

The Satisfaction Score

To incorporate a satisfaction constraint to our diet optimizer, we had to be able to quantify how happy the user is with any given diet. We did this by assigning each food design variable a satisfaction score. To calculate the total satisfaction of a diet, we would multiply each design variable amount by its satisfaction score, and then sum up all the individual satisfaction scores.

Assigning satisfaction scores

We did not find much research on quantifying the satisfaction with a diet, so we came up with the following method of assigning satisfaction scores to each of our foods. We split up the metric into two components—a "favorites" subscore and a "popularity" subscore. The "favorites" subscore is determined by the user and is meant to reflect their personal food preferences. They go through a list of foods and select which ones are their favorites. The subscore for each food is a 1 if it is a favorite, and a 0 if it is not.

The "popularity" subscore is a measure of how often the food is bought and consumed by the general public. We used data from the 2015-2016 National Health and Nutrition Examination

Survey to determine the “popularity” subscore. In the survey, 8,506 participants were asked to recall all the foods and drinks they consumed in the past 24 hours. Over 5,000 unique foods and 120,000 entries were recorded in the survey, which comprised a large enough sample for our purpose.

The “popularity” subscore was determined by:

$$\frac{\log(n_i)}{\log(n_{i,max})}$$

where n is the number of times the food was reported as eaten.

Figure 1 shows the distribution of “popularity” subscores amongst all the foods.

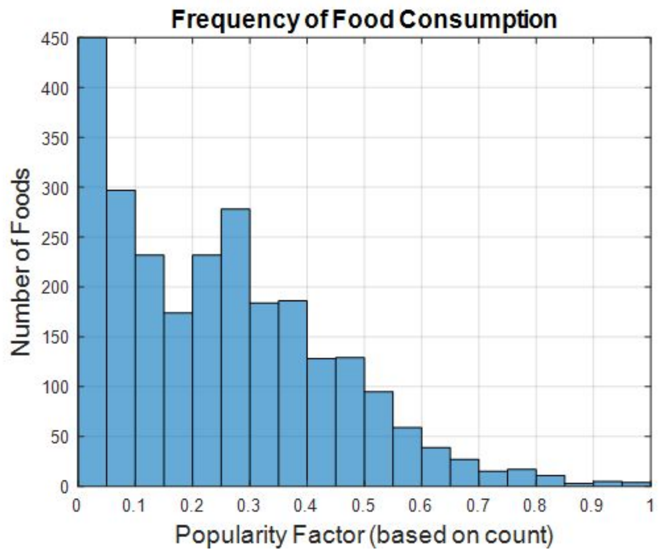


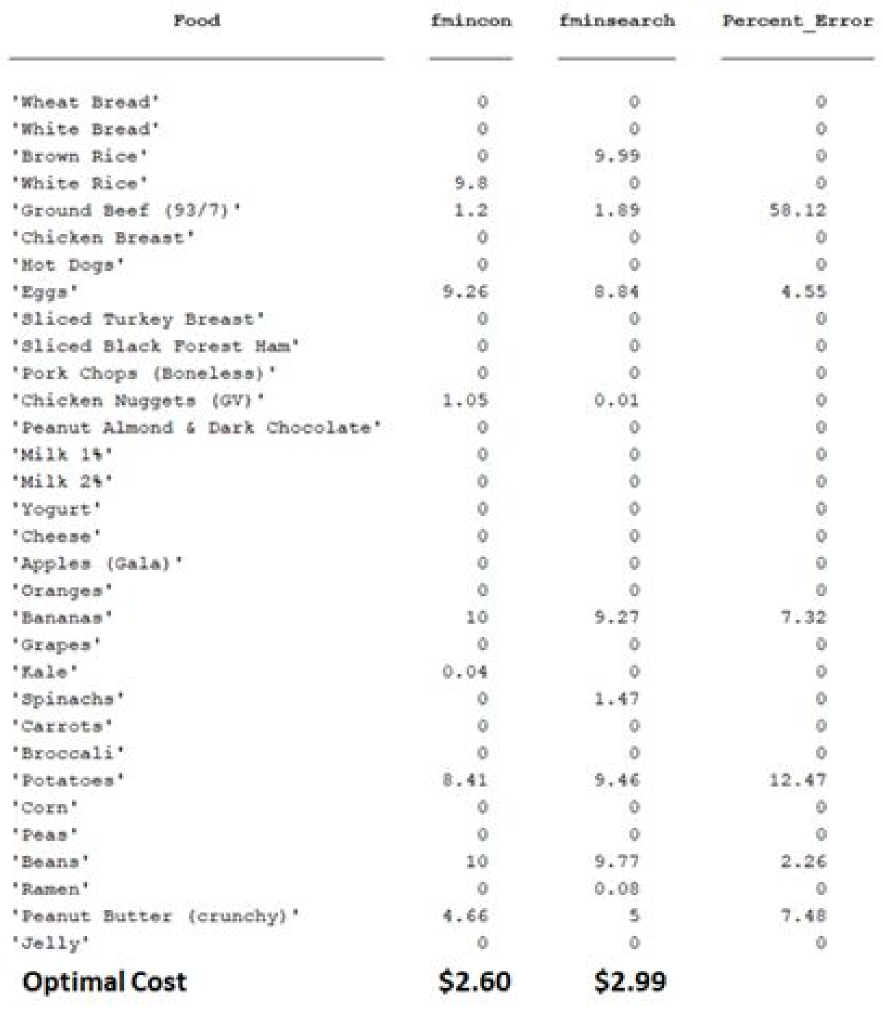
Figure 1: Food popularity quantified by frequency of consumption

The satisfaction score is the sum of the two subcores. Each subscore ranges from 0 to 1, so the satisfaction score ranges from 0 to 2.

Solving the Problem with Continuous Design Variables

Our first optimizer solved a simplified version of the problem. We used the basic nutrition model and treated each design variable as continuous. In an attempt to scale down this simplified problem, we selected 40 foods to be the design variables (instead of using all 8,000 in the database). We tested a few different algorithms, including MATLAB's constrained gradient-based optimizer, fmincon, and MATLAB's gradient-free Nelder-Mead algorithm, fminsearch. We found that fmincon greatly outperformed fminsearch, in both optimums found and function calls. Fmincon's optimal costs were on average 15% lower than fminsearch's optimal costs (see Table 1), and fmincon took about 5% of the function calls it took fminsearch to converge.

Table 1: Continuous optimization results with fmincon and fminsearch



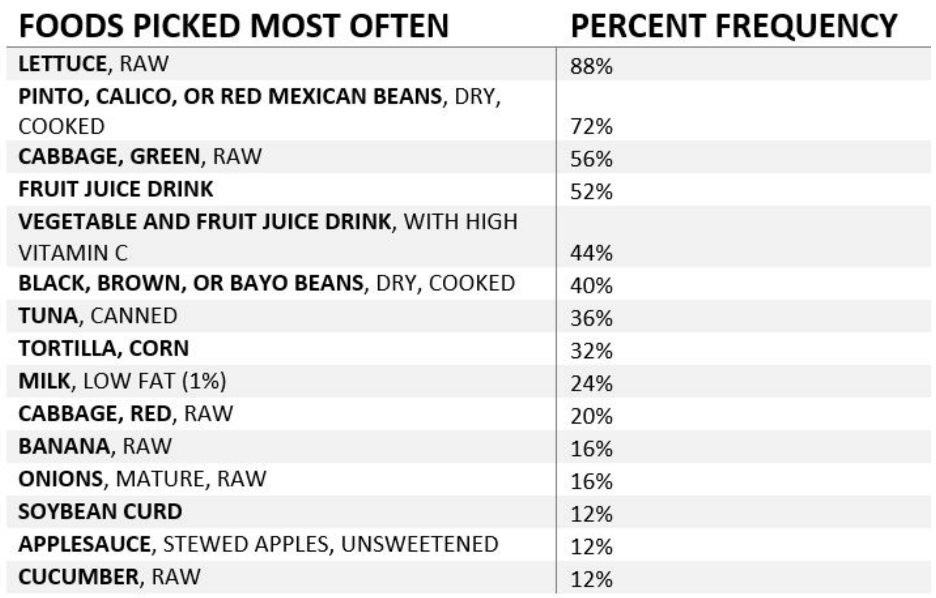
Food	fmincon	fminsearch	Percent_Error
'Wheat Bread'	0	0	0
'White Bread'	0	0	0
'Brown Rice'	0	9.99	0
'White Rice'	9.8	0	0
'Ground Beef (93/7)'	1.2	1.89	58.12
'Chicken Breast'	0	0	0
'Hot Dogs'	0	0	0
'Eggs'	9.26	8.84	4.55
'Sliced Turkey Breast'	0	0	0
'Sliced Black Forest Ham'	0	0	0
'Pork Chops (Boneless)'	0	0	0
'Chicken Nuggets (GV)'	1.05	0.01	0
'Peanut Almond & Dark Chocolate'	0	0	0
'Milk 1%'	0	0	0
'Milk 2%'	0	0	0
'Yogurt'	0	0	0
'Cheese'	0	0	0
'Apples (Gala)'	0	0	0
'Oranges'	0	0	0
'Bananas'	10	9.27	7.32
'Grapes'	0	0	0
'Kale'	0.04	0	0
'Spinachs'	0	1.47	0
'Carrots'	0	0	0
'Broccali'	0	0	0
'Potatoes'	8.41	9.46	12.47
'Corn'	0	0	0
'Peas'	0	0	0
'Beans'	10	9.77	2.26
'Ramen'	0	0.08	0
'Peanut Butter (crunchy)'	4.66	5	7.48
'Jelly'	0	0	0
Optimal Cost	**$2.60**	**$2.99**	

We then increased the size of our problem to 500 design variables and incorporated our more-accurate HEI nutrition model. The design variables Still treating the foods as continuous design variables, we tried using fmincon again to solve this problem, since it had success with our more simple problem. However, fmincon did not work as well with this new problem set-up. It was very slow, and it would often reach 10^7 or more function calls before meeting optimality constraints. We found that in about 10^5 function calls, about half of the food amounts were reduced to essentially zero. Keeping those design variables in the optimization problem was just unnecessarily slowing down fmincon. We then decided to write our own algorithm, called diet_optimizer_cont, to solve this problem. This optimizer follows the following steps:

1) Run fmincon for 10^5 function evaluations
2) Eliminate design variables with < 1 gram
3) Repeat until optimality is reached AND less than 30 design variables

This algorithm roughly halved the number of design variables every 10^5 iterations. By eliminating design variables along the way, we were able to converge to solutions much quicker. The optimum values varied significantly for different starting points, so we used Latin-hypercube Samples of 20 each time the optimizer ran to ensure an optimum value was reached. The optimizer would normally end up with around 15 to 30 of the same foods each time, which showed that it was decently consistent. The foods selected by the optimizer also matched our intuition on what an inexpensive, nutritious diet looks like. Table 2 shows the top foods chosen by the optimizer.

Table 2: Most frequently selected foods and percent of attempts selected in

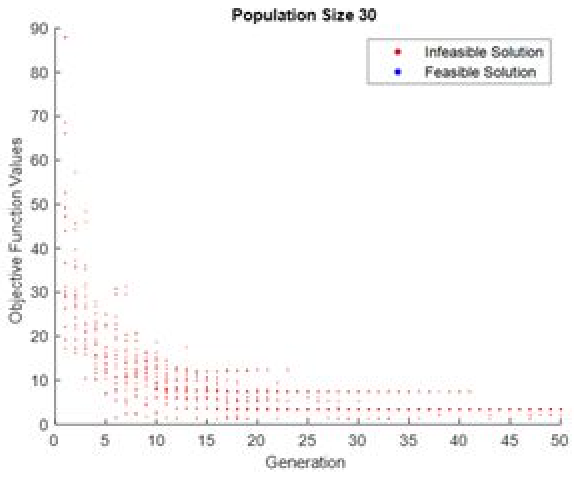


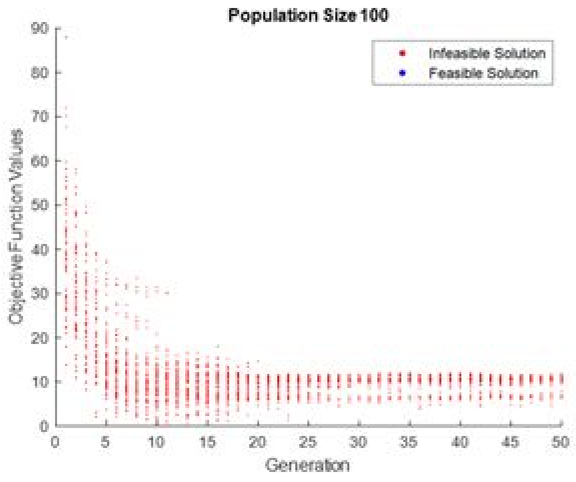
FOODS PICKED MOST OFTEN	PERCENT FREQUENCY
LETTUCE, RAW	88%
PINTO, CALICO, OR RED MEXICAN BEANS, DRY, COOKED	72%
CABBAGE, GREEN, RAW	56%
FRUIT JUICE DRINK	52%
VEGETABLE AND FRUIT JUICE DRINK, WITH HIGH VITAMIN C	44%
BLACK, BROWN, OR BAYO BEANS, DRY, COOKED	40%
TUNA, CANNED	36%
TORTILLA, CORN	32%
MILK, LOW FAT (1%)	24%
CABBAGE, RED, RAW	20%
BANANA, RAW	16%
ONIONS, MATURE, RAW	16%
SOYBEAN CURD	12%
APPLESAUCE, STEWED APPLES, UNSWEETENED	12%
CUCUMBER, RAW	12%

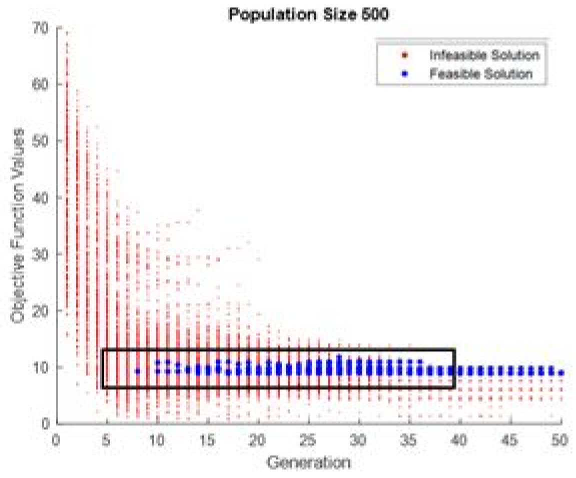
Solving the Problem with Discrete Design Variables

After achieving satisfactory results from the continuous-design-variable problem, we tried to solve the problem by assuming the foods can only be bought in discrete quantities, which is more realistic. Solving the discrete problem turned out to be more difficult than solving the continuous problem. We first tried two different genetics algorithms—one written by us (myGA), and one that was open-source (GA). These genetic algorithms turned out to be very slow and inconsistent. If the constraints were too tight and if the population size was too small, they would often have a hard time finding feasible solutions, as shown in figure 2. The population size could not be too large however, or the algorithm would converge painstakingly slowly, even for only 40 design variables. This issue only grew worse as we tried increasing the numbers of design variables.

Figure 2: Solutions of genetic algorithm by population size







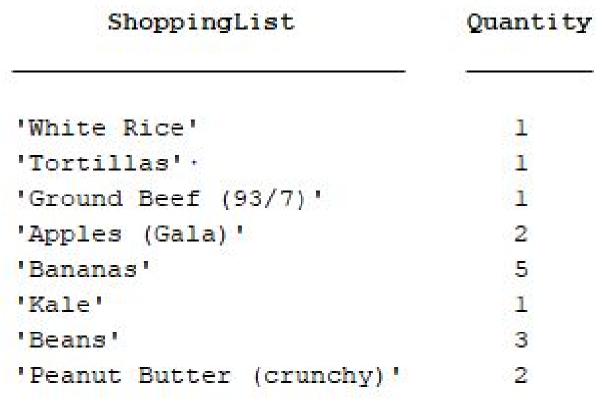
If we used our basic, linear nutrition model (nutrition fact and food group constraints), a branch-and-bound algorithm proved to be very effective. We used MATLAB's intlinprog, which is a mixed-integer optimizer that uses branch-and-bound techniques. With a linear cost objective and linear nutrition and satisfaction constraints, adding more design variables had little impact on the efficiency of the optimizer. We were able to find consistent optimal diets from over 500 design variables very quickly.

Results

The results of our optimizer can be shown in the outputs which consist of a list of foods with their respective quantities and a price. Using the branch and bound method we found a consistent 4 day price of about $16.40. Due to effects of bulk pricing models this becomes a little cheaper as we add more days and more mouths into the equation as well which our model only partially captures.

The food and corresponding quantities outputted in the form of a “shopping list” are shown in figure 1. At a glance the foods seem balanced among the food groups and are not surprising as foods like rice, beans, and kale are known for giving a lot of bang for your buck.

Table 3: Branch and bound optimal food results for 4 days



```
        ShoppingList            Quantity
______________________________  ________

'White Rice'                       1
'Tortillas'                        1
'Ground Beef (93/7)'               1
'Apples (Gala)'                    2
'Bananas'                          5
'Kale'                             1
'Beans'                            3
'Peanut Butter (crunchy)'          2
```

Our results with the branch and bound method were also very consistent whereas the results of our genetic algorithm varied. Our run time with branch and bound was also much faster than the genetic algorithm though slightly slower than a gradient based approach. Speed and efficiency are very important if this model is to be brought to market. Nobody likes a product that makes them wait for results.

Further experimentation and curiosity motivated us to see what the pareto front looks like as we format the project into a multiobjective optimization problem. The pareto fronts were created by using the diet_optimizer_cont algorithm for continuous design variables. As shown below in figures 3 and 4 nutrition and satisfaction (as we have framed them) both have a proportional relationship to cost in our area of interest (far right where the figure diverges is not of interest). This makes sense that more nutritional value costs more and better tasting food cost more. As an example, we see a sharp rise in the cost for HEIs greater than 65, so if one is more concerned with the saving cost than having the healthiest diet, then they could ensure to not go for HEI greater than 65 in our algorithm.

Figure 3: Pareto front of cost and nutrition

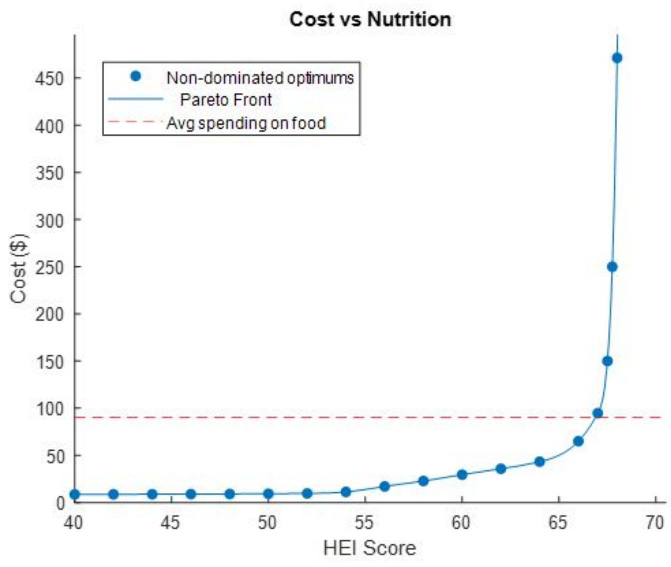
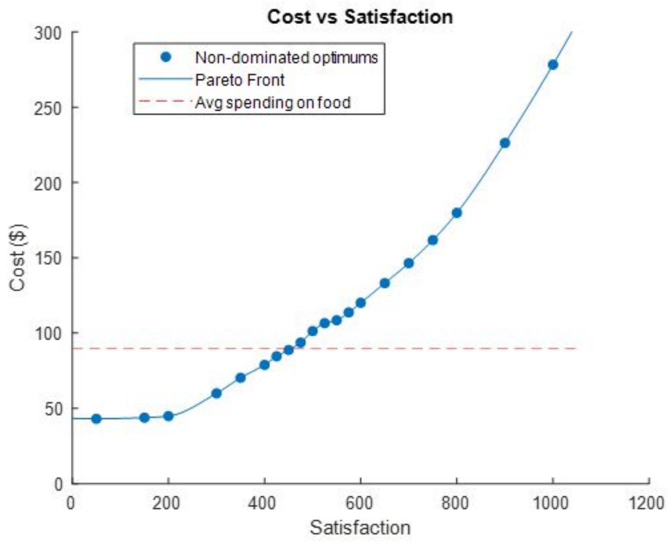


Figure 4: Pareto front of cost and satisfaction



Relevance:

We were extremely pleased with the results that our optimizer was able to find. We were able to produce grocery lists according to preference that would give a person all of their recommended daily nutrition for about $4 a day. Considering the average American spends $10-$21 dollars a day on groceries (when they don't eat out) this tool has the potential to really help those with less income save money and still support themselves and their families.

Conclusions

While the algorithm using intlinprog had the best results, it is limited to linear-objective, linear-constraint optimization problems. Our next step in developing the diet optimizer is to combine the efficiency of gradient-based optimizers with the ability of gradient-free optimizers to handle discrete design variables to solve nonlinear problems. One idea on how to do this is to expand on our diet_optimizer_cont algorithm. After the algorithm warm-starts fmincon a sufficient number of times and decreases the amount of design variables to less than 30, we could input those design variables into a particle swarm or genetic algorithm to acquire discrete quantities of foods to buy. This could then become the ideal grocery shopping list.

If someone were to continue this or a similar project in the future, we would like to recommend possible steps to take moving forward. One would be to consider the need students have for late night or early morning energy and account for caffeine, coffee, energy drinks, etc. or try to maximize energy given a budget constraint. One way to do this would be through a pareto front. This could take into account cost, energy, nutrition, satisfaction or any combination of these factors. That kind of a weighted optimal might be more relevant to a less health conscious crowd such as college students.

We also see this idea we've started as a product that could be taken to market. Partnering with large retailers or companies like amazon that are trying to disrupt the grocery market is one possibility. We could potentially scale the idea of optimizing diets according to preferences and budget and use machine learning to improve customer experience. Another possibility would be to take the product directly to market in the form of an application. Given our target market we would likely want to launch through a 3rd party pay/freemium model, meaning revenue comes from advertisements and eventually you hope to convert some free customers to paying customers through a complimentary product such as a health program or healthy food delivery service like Blue Apron.

References

Food cost database:

Info: https://fns-prod.azureedge.net/sites/default/files/resource-files/UserGuide_0.pdf

Download: https://www.fns.usda.gov/resource/cnpp-data

Nutrition facts database:

Info:

https://www.ars.usda.gov/ARSUserFiles/80400530/pdf/fndds/FNDDS_2015_2016_factsheet.pdf

Download:
https://www.ars.usda.gov/northeast-area/beltsville-md-bhnrc/beltsville-human-nutrition-research-center/food-surveys-research-group/docs/fndds-download-databases/

Food intake database:

Info: https://wwwn.cdc.gov/nchs/nhanes/2015-2016/DR1IFF_I.htm#SEQN

Download:
https://www.ars.usda.gov/northeast-area/beltsville-md-bhnrc/beltsville-human-nutrition-research-center/food-surveys-research-group/docs/wweia-documentation-and-data-sets/

Abstract Articles:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6315356/
https://journals.lww.com/jpgn/fulltext/2003/01000/linear_programming__a_mathematical_tool_for.6.aspx
https://aip.scitation.org/doi/pdf/10.1063/1.4902450?class=pdf

Other articles and sources

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2954450/

https://www.foxbusiness.com/lifestyle/what-is-the-average-grocery-bill-for-one-person