THE STIGLER DIET PROBLEM

Computational Intelligence for Optimization

Project Repository
https://github.com/hubertober10/stiglerDiet.git

Hubert Oberhauser
20220628

Table of Contents

1. Introduction

The objective of this project is to address Stigler's Diet Problem by utilizing Genetic Algorithms for optimization. The goal was to design a diet consisting of 5 ingredients that satisfies the minimum nutritional requirements and minimizes cost through generational evolution.

This problem involves considering the nutrient requirements and prices of 77 ingredients as of 1939. The code for this project builds upon the Charles Genetic Algorithm Library, initially developed in a classroom setting. It has been further generalized to support both minimization and maximization problems, with the intention of enhancing its versatility and abstraction.

2. Methodology

In this project, seven files were developed. The data.py file imports a list of nutrients with their respective quantities and a list of commodities, including their prices, weights, and nutritional characteristics.(Table 1)

The main.py file executes the genetic algorithm with various mutation probabilities, reproduction probabilities, and either with or without a fitness constraint. It also saves the results to a CSV file for further analysis.The function for initializing the population is located in utility_functions.py, ensuring variety by checking that every ingredient appears at least once. The parameters population_size and individual_size help define our problem.

Individuals, each consisting of five integers, are randomly generated, with each integer representing the index of a corresponding ingredient in the data list (Table 1 – Annex). This encoding is chosen for its simplicity and because some genetic operators require

numeric individuals. The utility_functions.py file also contains all the functions for population evolution.

The selection.py file includes the Ranking Selection, where individuals are sorted based on their fitness, considering their rank rather than the value itself.

The crossover.py file defines the One Point Crossover function.

The mutation functions are found in mutation.py, which includes the Scramble Mutation, where two random positions from a random individual are selected, and the genes between them are shuffled.

Lastly, visualizations.ipynb contains information from all previous combinations and visualization functions for deeper analysis.

3. Challenges

The primary objective of this algorithm is to minimize the total cost of a meal while ensuring it meets the minimum nutrient intake requirements. To achieve this, the fitness function was defined as the total cost. A penalty system was implemented to account for nutrients: for each nutrient value that falls below the minimum requirement, a penalty is added to the fitness score.

During initial trials, a problem was identified where the best solutions lacked diversity, with several ingredients being repeated frequently. To address this, I introduced a reward system for variety within the population by applying a penalty based on the frequency of each ingredient. This penalty is added to the fitness score since it is a minimization problem.

After several attempts, it became evident that normalizing this penalty value was necessary because the algorithm became unstable, with the fitness function continually

increasing. A MinMax normalization approach was used to ensure that the penalty did not overshadow the primary goal of cost minimization and meeting nutrient constraints.

4. Results

To ensure robust solutions and reliable conclusions about the Genetic Algorithm, all combinations were run 30 times, and the median per generation was calculated. This measure is not influenced by extreme values and allows for comparisons of the central tendency of the results.

Additionally, the penalty weight parameter was consistently set at 5 to balance its impact on the fitness function while keeping the price as the main parameter.

Mutation probabilities of [0.05, 0.1, 0.2] were chosen for specific reasons. A higher mutation rate can help the algorithm escape local optima through disruptive changes but may lead to slower convergence. Conversely, a lower mutation rate can preserve good solutions and fine-tune them.

The price was normalized according to the minimum nutrient intake to balance their impact, ensuring that the ingredient with the highest sum of nutrients also has the lowest fitness score.

5. Conclusions

The first observation for improvement noted during the experiment is the need for a higher number of trials in future experiments to find an even more robust solution than the one identified in this study.

Secondly, the diversity constraint ensuring variation should be enhanced, as my approach did not completely prevent ingredient repetition in the final algorithm. Additionally, implementing more selection algorithms and crossover methods could improve the results.

A more extensive search for parameters such as mutation rate, crossover rate, generations, and number of individuals is recommended to explore a broader range of potential combinations.

For a more meaningful final evaluation, it would be beneficial to scale the nutrients using MinMax scaling, not just normalize them as was done in this case.

In Stigler's study, a diet costing $41,4741(Output 1) per year with 5 ingredients was found. Ideally, the final diet solution should include a greater variety of ingredients to meet the updated nutritional requirements, given that our algorithm is generalized and could work with an updated dataset. However, this study serves as a good example of the real-world application of Genetic Algorithms and their potential for research.

6. Appendix

Index	Ingredient	Index	Ingredient
0	Wheat Flour (Enriched)	39	Salmon, Pink (can)
1	Macaroni	40	Apples
2	Wheat Cereal (Enriched)	41	Bananas
3	Corn Flakes	42	Lemons
4	Corn Meal	43	Oranges
5	Hominy Grits	44	Green Beans
6	Rice	45	Cabbage
7	Rolled Oats	46	Carrots
8	White Bread (Enriched)	47	Celery
9	Whole Wheat Bread	48	Lettuce
10	Rye Bread	49	Onions
11	Pound Cake	50	Potatoes
12	Soda Crackers	51	Spinach
13	Milk	52	Sweet Potatoes
14	Evaporated Milk (can)	53	Peaches (can)
15	Butter	54	Pears (can)
16	Oleomargarine	55	Pineapple (can)
17	Eggs	56	Asparagus (can)
18	Cheese (Cheddar)	57	Green Beans (can)
19	Cream	58	Pork and Beans (can)
20	Peanut Butter	59	Corn (can)
21	Mayonnaise	60	Peas (can)
22	Crisco	61	Tomatoes (can)
23	Lard	62	Tomato Soup (can)
24	Sirloin Steak	63	Peaches, Dried
25	Round Steak	64	Prunes, Dried
26	Rib Roast	65	Raisins, Dried
27	Chuck Roast	66	Peas, Dried
28	Plate	67	Lima Beans, Dried
29	Liver (Beef)	68	Navy Beans, Dried
30	Leg of Lamb	69	Coffee
31	Lamb Chops (Rib)	70	Tea

32	Pork Chops	71	Cocoa
33	Pork Loin Roast	72	Chocolate
34	Bacon	73	Sugar
35	Ham, smoked	74	Corn Syrup
36	Salt Pork	75	Molasses
37	Roasting Chicken	76	Strawberry Preserves
38		Veal Cutlets	

Table 1

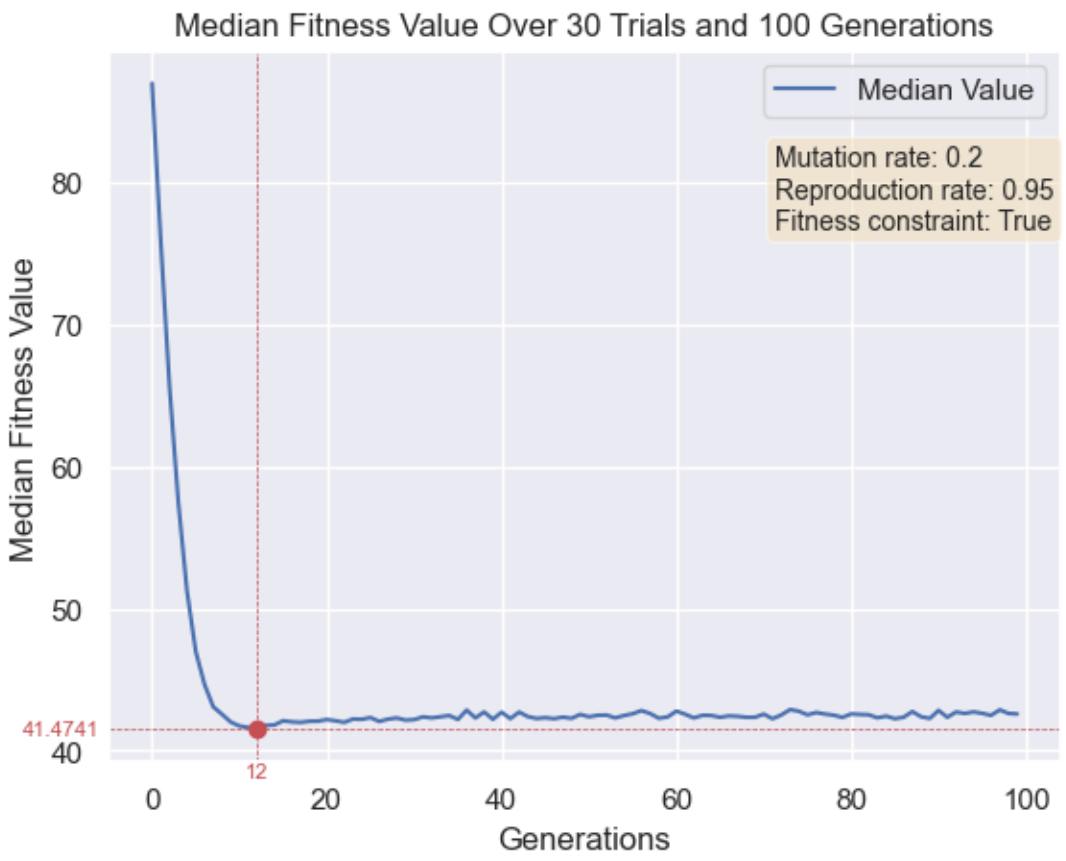

Figure 1