This project can be accessed via: https://github.com/McLeodolf/stiglerDiet.

Computational Intelligence for Optimization Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

The Stigler Diet

Group 10

Cláudia Rocha, number: 20191249

Rita Soares, number: 20220616

Leonhard Allgaier, number: 20220635

May, 2023

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

INDEX

1. Introduction

The aim of this project is to solve Stigler's Diet Problem using optimization through Genetic Algorithms. Our team intends to create a diet using 5 ingredients that meets the minimum nutritional requirements and minimizes the price through the evolution of generations. In this problem, we are taking into consideration the nutrient requirements and prices of 77 ingredients in the context of 1939.

The code developed for this project is derived from the Charles Genetic Algorithm Library that was originally created during classes. It has been further generalized to handle both minimization and maximization problems, with the aim of making it as versatile and abstract as possible.

This project was equally built by the members of the group.

2. Methodology

In the development of this project, 8 files were created. First, in *data.py*, it's imported the list of nutrients with the respective quantities and a list with the data regarding each commodity, as well as its price, weight and nutritional characteristics.

The function for the initialization of the population can be found in the file *population_initial.py* it aims to ensure variety by checking if every ingredient is present at least once. Besides the parameters 'population_size', 'individual_size', 'data' and 'nutrients', that help define our problem, there's also a Boolean parameter called 'constraints', that, when True, ensures that every individual contains the minimum requirements of nutrients.

The individuals are composed of 5 integers, each representing the index of the corresponding ingredient on the data list (Table 1 – Annex). This encoding was chosen for two main reasons: its simplicity and some genetic operators demand numeric individuals.

In *selection.py* the following functions can be found: Ranking Selection, selection probability is dependent on the position of the fitness on a ranking and not on the value itself; Roulette Selection, the probability of a given individual is proportionate to its fitness; and Tournament Selection, *n* individuals are randomly selected with repetition to compete, the one with best fitness is selected.

Regarding *crossover.py*, 3 functions were defined: One Point Crossover, "uses the single point fragmentation of the parents and then combines it at the crossover point to create the offspring" [1]; Shuffle Crossover, that "firstly randomly shuffles the genes in the both parents but in the same way, then it applies the 1-point crossover technique by randomly selecting a point as crossover point and then combines both parents to create two offspring. After performing 1-point crossover the genes in offspring are then unshuffled in same way as they have been shuffled" [1]; and Arithmetic Crossover, generate two offspring based on the application of a formula evolving both parents and a constant, alpha that varied between 0 and 1.

The mutation functions can be found on the file *mutation.py* which includes Scramble Mutation, select two random positions from a random individual and shuffle the genes between them; Swap Mutation, two random positions from an individual are selected and then those genes are switched; and Inversion Mutation, select two random positions from a random individual and inverses the gene order between them.

The file *diet_algorithm.py* contains the function for the evolution of the population. Here we are allowed to choose the number of individuals in the population, the probabilities that result in crossover or reproduction and/or mutation as well as which selection method, operators and presence of elitism, to later try several combinations and get insights on the GA.

The *utils.py* only implements our fitness function.

Lastly, *visualizations.py* contains the information from all the combinations performed in the previous step and visualization functions for a deeper analysis.

3. Fitness Function and Its Challenges

The main goal of this algorithm is to minimize the total cost of a meal that respects the minimum intake of nutrients. To achieve this the fitness function was defined as the total cost, to ensure the nutrients were also taken into consideration a penalty system was applied, for each value that didn't respect the minimum required one value is added to the fitness.

During the first trials of the algorithm a problem was noticed, a lack of diversity on the best solutions found, where several ingredients would be repeated. To solve this, we decided to reward the variety in a population, by applying a penalty system based on the frequency of each ingredient in the population, this value is added to the fitness since it is a minimization problem. After some attempts, it came to attention the requirement to normalize this penalty value because the algorithm would become unstable, and the fitness function would keep increasing. A MinMax approach was used, otherwise the weight of this penalty would overshadow the main aim, the minimization of costs and the nutrients constraints.

Furthermore, a second study was performed, which only consisted of the calculation of the total cost and penalty by lack of diversity. In this scenario, the nutrients constraints were applied on the initial population and on the crossovers.

4. Results

To get a robust solution and reliable conclusions regarding our GA, all the following combinations were ran 30 times and calculated the median per generation of all of them, a measure that is not skewed by extreme values and allows for comparisons of the central tendency of the results. Additionally, the weight parameter of the penalty was also always maintained at 5 so it has a balanced impact on the fitness function while keeping the relevance of the price as the main parameter.

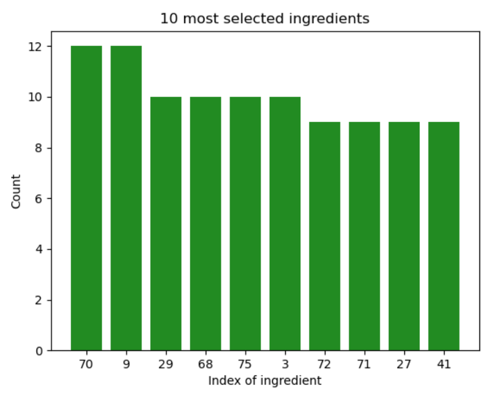

Figure 1.1 – Histogram of the top 10 most frequent ingredients in the initial population with restrictions switched to False

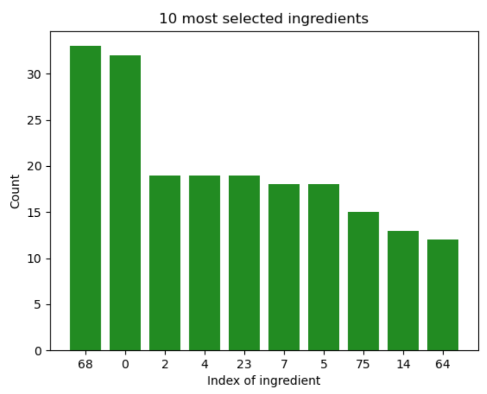

Figure 1.2 – Histogram of the top 10 most frequent ingredients in the initial population with restrictions switched to True

As mentioned, the nutrients restrictions are activated by a flag either on the fitness function or on the initial population and crossover. To decide which one to keep constant during testing the frequency of the top 10 ingredients is plotted (fig 1.1 and fig 1.2), this led to the conclusion that the application of the constraint on the initial population restricts our initial diversity, as it created a pattern that some ingredients such as 68 and 0 have a much higher frequency and are commonly pick, probably because of their nutrient's values. On the other hand, the restrictions on the fitness function seems to produce an equally diverse population, that is believed to lead to better results, as so the constrains will only be True for the fitness function.

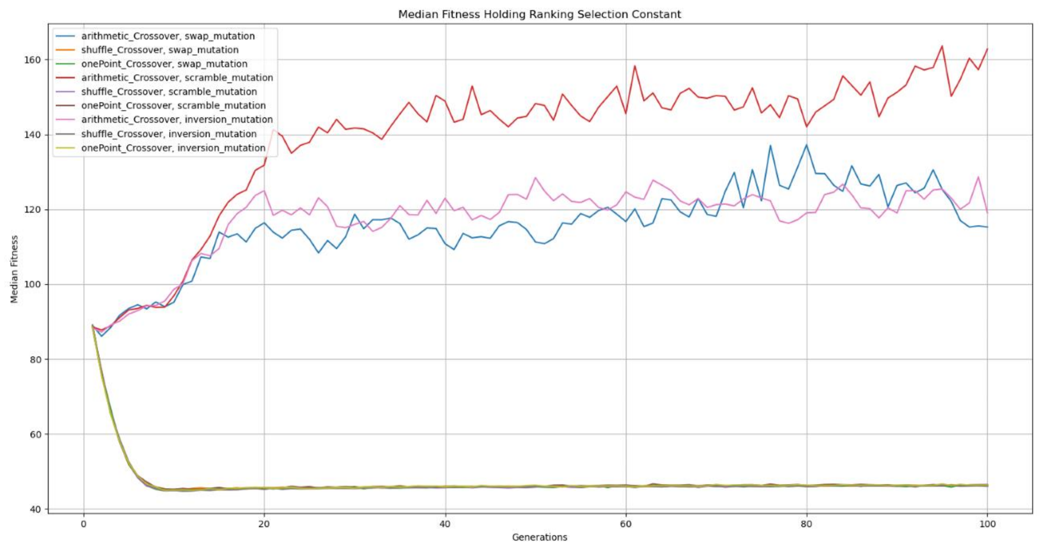

Figure 2.1 – Line Plot of the median fitness per generation for all combinations holding ranking selection constant over 30 trials with a penalty weight of 5, population of 150 through 100 generations and a reproduction probability of 0.95 and mutation of 0.1.

On the first trial every combination of selection, crossover and mutation with probabilities of 0.95 and 0.1, respectively, and maintaining elitism was tried, 150 individuals were considered through 100 generations. For the tournament selection the tournament size picked is 3 to ensure a balance between our penalties and giving a higher chance to the 'worst' individuals, this will be held constant throughout this study. The graphics in general show an early convergence (around generation 10th as seen in Figure 2.1 and 2.2, 2.3 in the Annex) for every combination except the ones evolving arithmetic crossovers, this is due to the introduction of a lot of variety through generations instead of maintaining the best ingredients, which leads the fitness to be unstable and keep increasing. Regarding the differences between the selection methods, the tournament and ranking selection are the ones that present the earliest convergence, stabilizing the fitness values before the 10th generation, while the roulette selection has its convergence around the 15th generation.

It's suspected that this early convergence is caused by the constraint defined in the selection, that each ingredient must be selected at least once, and the penalties of the variety, resulting in a good performance. Therefore, it was decided to run with less individuals in the initial population on the next trial - 70. The ranking selection is discarded because it's relation performance – computational expenses is not worth it.

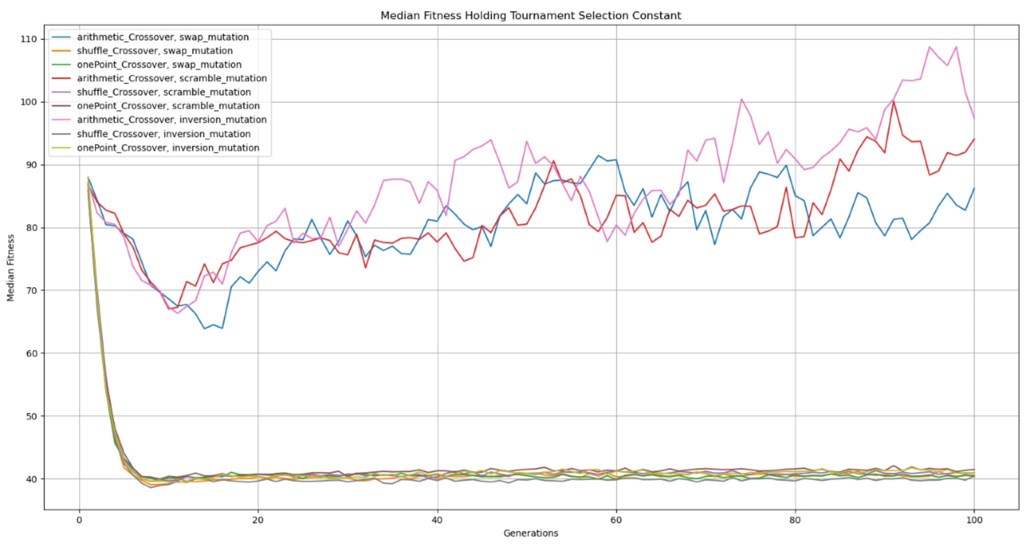

Figure 3.1 – Line Plot of the median fitness per generations for all combinations holding tournament selection (size of 3) constant over 30 trials with a penalty weight of 5, population of 70 through 100 generations and a reproduction probability of 0.95 and mutation of 0.1

As seen in figures 3.1 and 3.2 (Annex), reducing the population size did not alter significantly the convergence points, however it made the differences between combinations more evident, it is perceivable that for the Tournament selection, the combination with lower median of fitness is Shuffle crossover and Inversion mutation, and in Roulette selection is shuffle crossover. Although after convergence, the fitness function has the tendency to slightly increase in the roulette selection, with the use of elitism we can be sure that the best individual is kept until the end, so we do not consider the use of 100 generations to be a problem, as so no trial without elitism will be done, since there seems to be no apparent benefit from it.

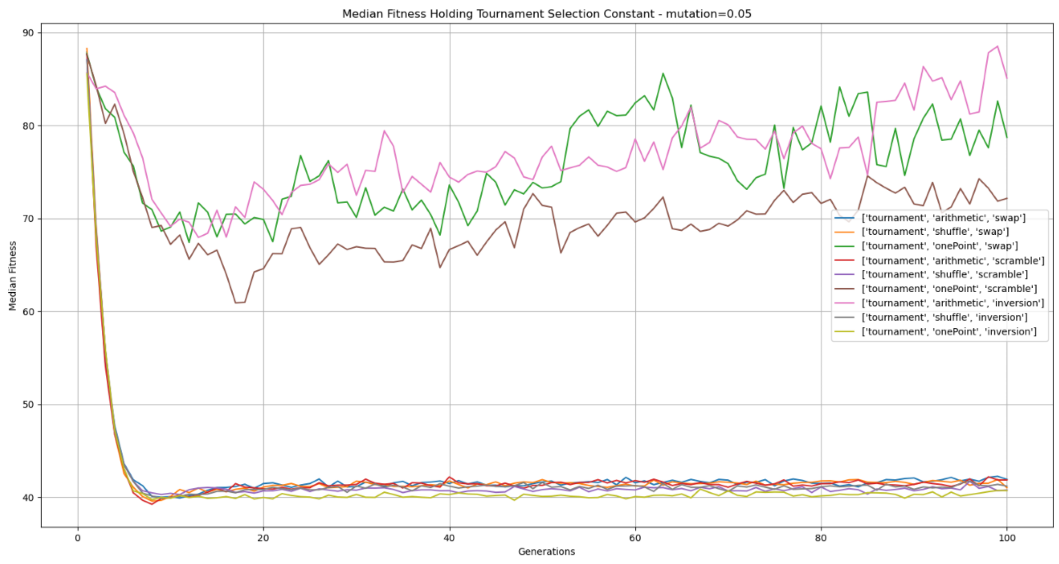

Figure 4.1 – Line Plot of the median fitness per generations for all combinations holding tournament selection constant over 30 trials with a penalty weight of 5, population of 70 through 100 generations and a reproduction probability of 0.95 and mutation of 0.05.

In the previous attempts, a mutation rate of 0.1 was used (Figures 2.1 – 2.3). A greedy search was performed, varying it the mutation probability with [0.05,0.2]. The values were chosen for the following reasons, a higher value can result in enhancing the capability of the algorithm to escape local optima by the disruptive changes and lead to a slower convergence. On the other hand, the lower value may preserve the good solutions and fine-tune them. After interpreting the results on figures 4.1 – 4.4 (Annex), it's concluded that, generally, the fitness functions behaviors are similar so values between those 3 should perform well.

Nr.	Population size	Generations	Crossover Probability	Mutation Probability	Selection	Crossover	Mutation	Fitness Value	List of ingredients	Normalized Sum of Nutrients	Price
[1]	70	100	0.95	0.1	Tournament	Shuffle	Inversion	33.8656	[40,45,45,46,52]	374.23	21.6
[2]	70	100	0.95	0.1	Roulette	Shuffle	Swap	34.1625	[49,4,52,4,46]	331.45	22.6
[3]	70	100	0.95	0.05	Tournament	Arithmetic	Scramble	42.9000	[41,40,49,52,45]	276.57	22.9
[4]	70	100	0.95	0.05	Tournament	One Point	Inversion	37.4385	[45,3,41,40,49]	192.93	24.9

Table 2 - Best results generated of 70 individuals evolving through 100 generations for 30 trials with the parameters described.

The chosen combinations based on the performance were [1] Tournament selection with shuffle crossover and inversion mutation with a 0.1 probability; [2] Roulette selection with shuffle crossover and swap mutation with a 0.1 probability; [3] Tournament selection with arithmetic crossover and scramble mutation with a probability of 0.05; and, lastly, [4] Tournament selection with one point crossover and inversion mutation with a probability of 0.05. As seen in table 2, the first two lists of ingredients contain the same ingredient twice.

The price was normalized according to the minimum intake of nutrients, so their impact is balanced and the one with higher sum of nutrients [1], it's also the one with the lower fitness, approximately 33.8, and a lower price of 21.6. Additionally, analysing figure 5 – Annex, we can observe that this combination is also highlighted, therefore, there's no question this is the most the suited reached combination for the problem. It contains apples, two times cabbage, carrots and sweet potatoes.

5. Conclusions

The first observation or improvement identified during the experiment's execution, which should be considered in future experiments, is conducting a higher number of trials to discover an even more robust solution than the one found in this study. Secondly, the diversity constraint that ensures variation should be improved, as our approach was unable to completely prevent ingredient repetitions in the final algorithm. Additionally, we recommend conducting a more extensive search for parameters such as mutation rate, crossover rate, generations, number of individuals, among others, in order to explore a larger search space of potential combinations. Moreover, for a more meaningful final evaluation, it would be beneficial to scale the nutrients, not just normalize them as done in our case, but using MinMax scaling. If researchers in future studies thoroughly analyse certain operators like arithmetic crossover, they may choose to exclude them beforehand since they are not suitable for implementing our specific problem.

In Stigler's study, he found a diet that cost $39,93 per year with 5 ingredients, while our solution with 4 ingredients was $21.6 in 1939 dollars. Nowadays, with the inflation rate, it would be equivalent to $457.92 per year. Ideally, our final diet solution should have more variety of ingredients to meet the further nutritional requirements that have changed in the past 84 years given that our algorithm is generalized and would work with an updated dataset, however, it is a good example of the optimization of Genetic Algorithms in the real-world context and the potential it brings to research.

6. References

[1] J. Umbarkar, A., & Sheth, P. D. (n.d.). Crossover operators in Genetic Algorithms: A review

[2] Vanneschi, L., & Silva, S. (2023b). *Lectures on intelligent systems*. Springer International Publishing AG.

7. 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 - Corresponding ingredient to the index on the data list. Encoding used for this study.

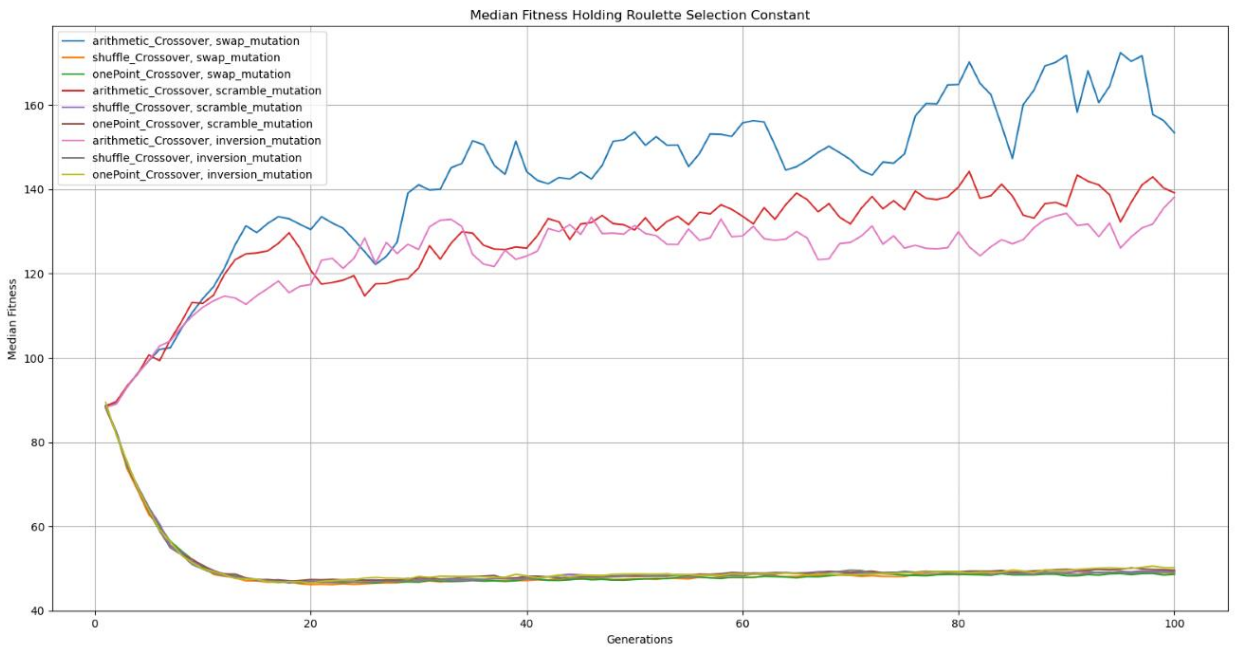

Figure 2.2 – Line Plot of the median fitness per generation for all combinations holding roulette selection constant over 30 trials with a penalty weight of 5, population of 150 through 100 generations and a reproduction probability of 0.95 and mutation of 0.1.

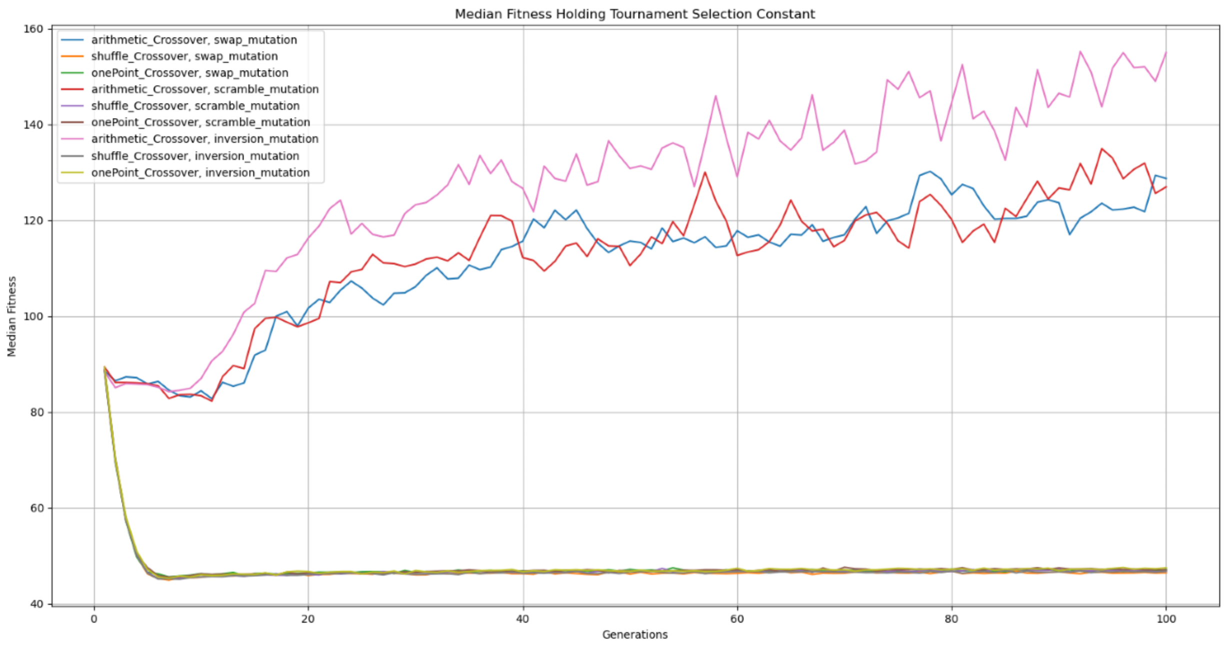

Figure 2.3 – Line Plot of the median fitness per generation for all combinations holding tournament selection (size of 3) constant over 30 trials with a penalty weight of 5, population of 150 through 100 generations and a reproduction probability of 0.95 and mutation of 0.1.

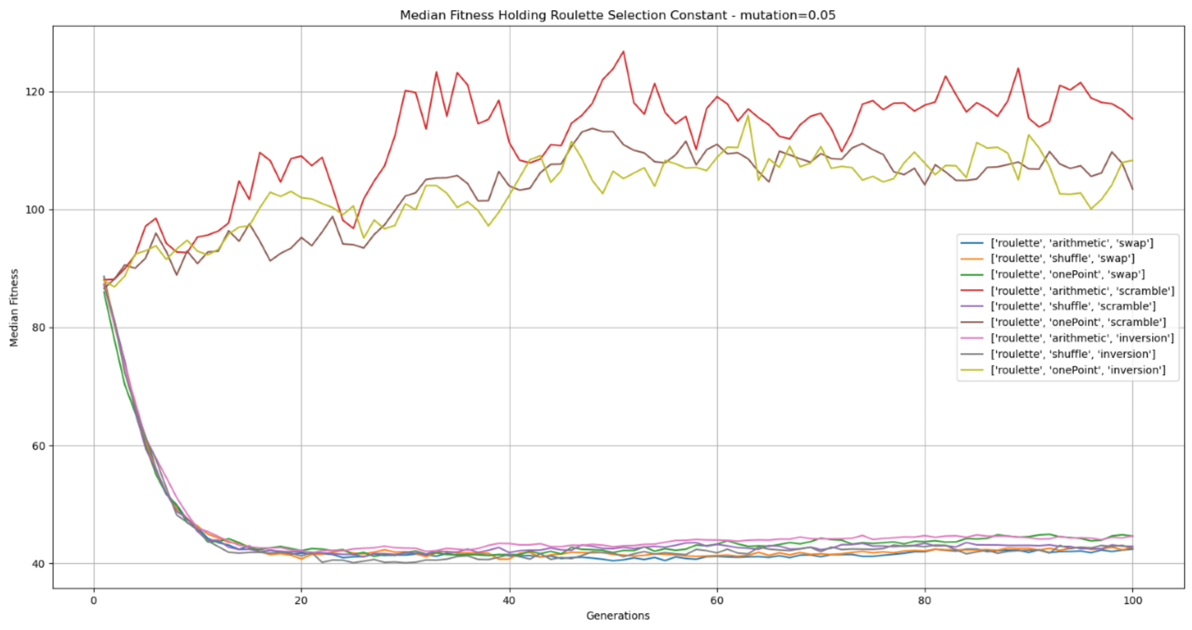

Figure 4.2 – Line Plot of the median fitness per generations for all combinations holding roulette selection constant over 30 trials with a penalty weight of 5, population of 70 through 100 generations and a reproduction probability of 0.95 and mutation of 0.05.

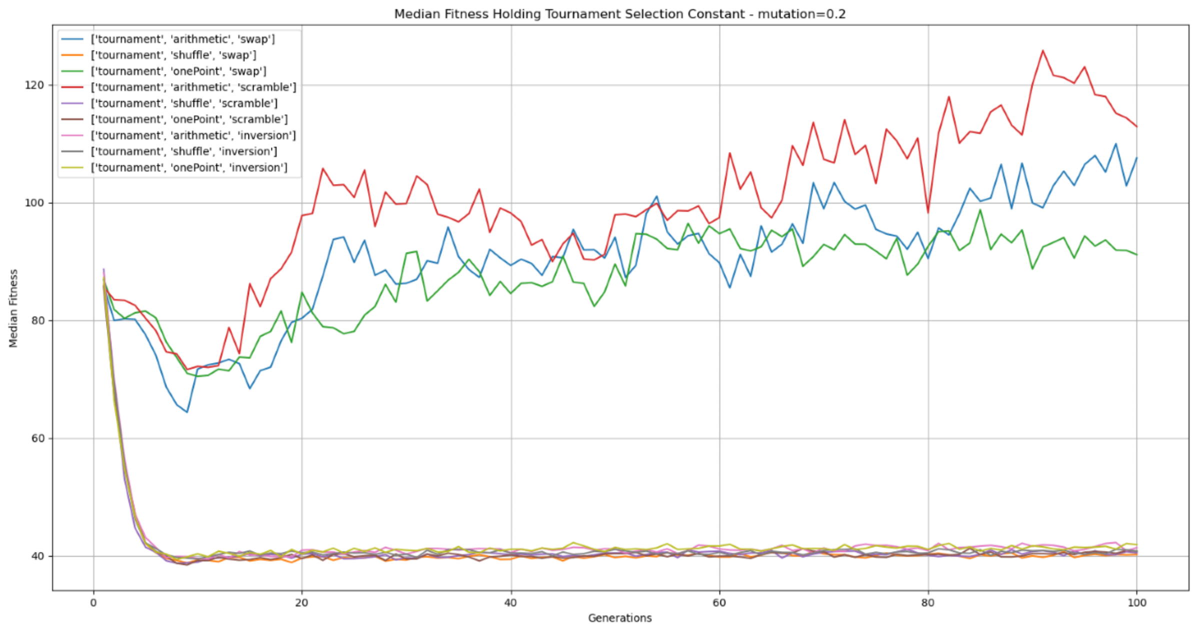

Figure 4.3 – Line Plot of the median fitness per generations for all combinations holding tournament selection (size of 3) constant over 30 trials with a penalty weight of 5, population of 70 through 100 generations and a reproduction probability of 0.95 and mutation of 0.2.

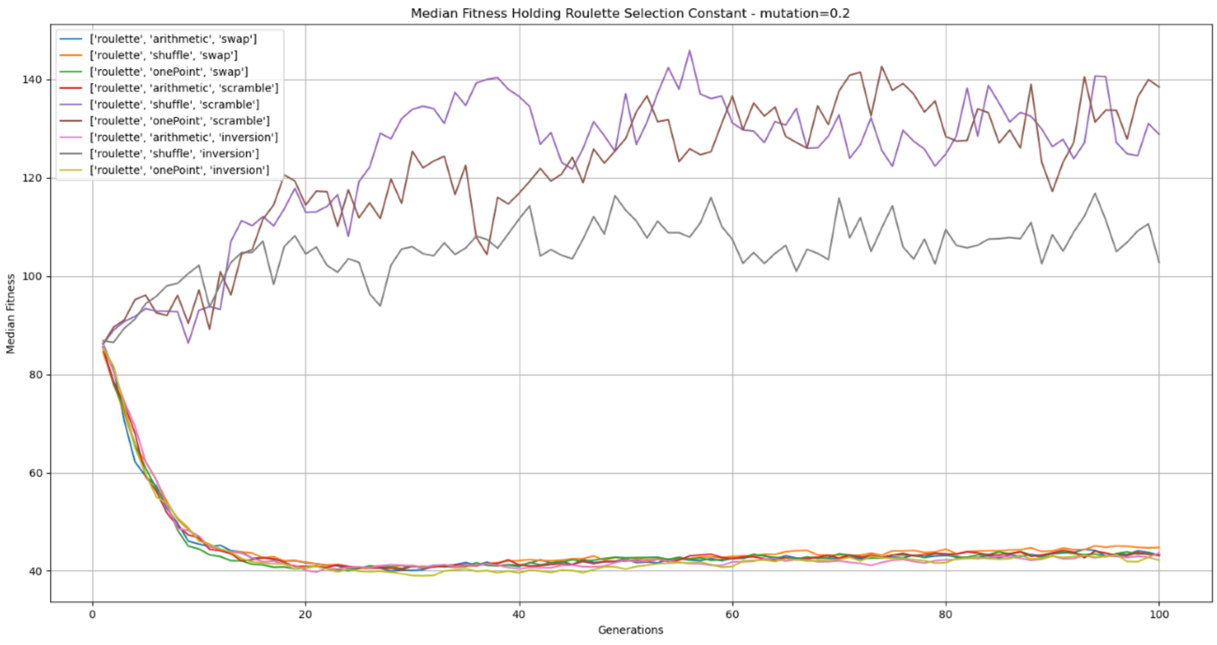

Figure 4.4 – Line Plot of the median fitness per generations for all combinations holding roulette selection constant over 30 trials with a penalty weight of 5, population of 70 through 100 generations and a reproduction probability of 0.95 and mutation of 0.2.

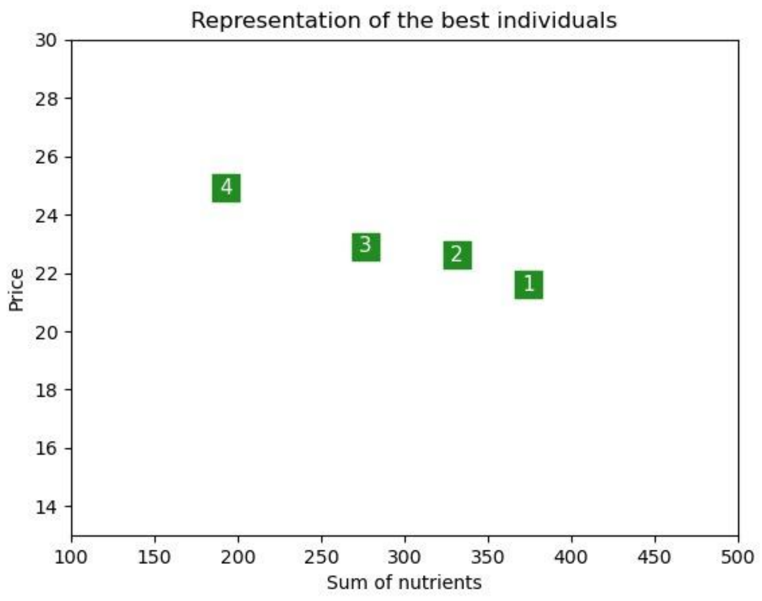

Figure 5 – Scatter plot of total of the normalized nutrients per total price of the individuals with the best fitness for the 4 best combinations. [1] Tournament Selection with Shuffle crossover and inversion mutation with a 0.1 probability; [2] Roulette selection with Shuffle crossover and Swap mutation with a 0.1 probability; [3] Tournament selection with Arithmetic crossover and Scramble mutation with a 0.05 probability; and, lastly, [4] Tournament selection with the operators being One Point crossover and Inversion mutation with a probability of 0.05