THE STIGLER'S DIET PROBLEM

COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION PROJECT

2022/2023

PROFESSORS: LEONARDO VENNESCHI | BERFIN SAKALLIOGLU

STUDENTS: AFONSO REYNA - R20191197 | ANDRÉ SILVA - R20191226 | DINIS MELO - R20191190 | GONÇALO RODRIGUES - R20191300 | MAFALDA MARTINS - R20191220

NOVA IMS Information Management School

OPTIMZATION PROBLEM

The Optimization Problem chosen was the Stigler's Diet Problem, where the aim is to minimize the diet's overall cost while satisfying predefined nutritional requirements - which can be seen in the table below. The available nutritional data is based on information from 1944, and the values are given in dollars rather than per unit. However, there is information regarding the individual cost of each food item from 1939. Although originally designed for a minimization problem, our project has been developed to accommodate maximization problems as well.

Nutrient	Daily Recommended Intake
Calories	3,000 calories
Protein	70 grams
Calcium	.8 grams
Iron	12 milligrams
Vitamin A	5,000 IU
Thiamine (Vitamin B1)	1.8 milligrams
Riboflavin (Vitamin B2)	2.7 milligrams
Niacin	18 milligrams
Ascorbic Acid (Vitamin C)	75 milligrams

Nutrients' Daily Recommended Intake

Initially, our approach was to minimize the discrepancy between the overall nutrients in the diet solutions and the recommended daily intake values, while also minimizing the cost of the diet. However, we found that this multi-objective optimization approach was challenging to implement and inefficient.

We also realized that by ensuring the population (both parent and offspring) meets the daily intake requirements, the optimization objective primarily focuses on minimizing the cost. As we are dealing with an optimization problem, it is expected that implementing this constraint on the population will lead to solutions that gradually approach the daily requirements, as the emphasis is on minimizing the cost.

Thus, we expect that the difference between the total nutrients in the solution diet and the recommended diet will be minimal. Our final objective is to minimize the overall cost of the diet while ensuring that the solution meets the daily recommended intake levels for each nutrient. This approach allows us to simplify the problem and focus solely on minimizing the monetary cost.

Regarding the division of labour within our group, it was done as fairly as possible, during all steps of the project, such as the choice of the problem, the making of the code, as well as the report, and also in the analysis of our results. Our Git repository, which contains all the code made for this project, can be accessed through this link: https://github.com/LOLTUGALOL/CIFO_comCarinho.

METHODOLOGY

The first step was to define the representation of each individual from our population. We consider that an individual corresponds to a diet plan, in a way that its representation consists of a list, in which length is the same as the number of foods we have in our dataset, and thus each element of the representation will represent a quantity of each food to be inserted into the diet plan. When creating a new individual, these quantities are generated randomly with values that can range between 0 and 1, in this way we will allow the diet plan (individual) to be more flexible. The probability of the number 0 being assigned is of 0.7, while there is a 0.3 probability of generating a number between 0.1 and 1. The reason we don't generate a number between 0 and 1 is because we didn't want to include the probability of generating a zero twice.

Initially, we used a range between 0 and 3 for generating food quantities. However, the resulting diets were unrealistic for human consumption. As a result, we adjusted our approach to values between 0 and 1, in order to converge faster to a feasible diet.

It is important to note that an individual is only accepted in the population if it's nutrient values are equal or higher than the daily recommended intake values, through the function 'verify_macros'. This function consists of two steps:

1. Calculate the amounts of each nutrient an individual contains, by summing, for every food, its quantity multiplied by its price and by the amount of nutrient, provided in the dataset.
2. Then compare the resulting values with the corresponding daily recommended intake for each nutrient. If the result of any nutrient is lower than the recommended intake, the function returns 'False', which will mean that the individual is not accepted in the population, and a new one must be generated.

Not only did we restrain the initial population, we also restrained the offspring. If the offspring didn't satisfy the daily macros, the stochastic process of crossover and mutation would be performed until it was. A rule of thumb of 20 attempts was established so that the algorithm didnt get stuck creating unsatisfactory offsprings infinitely. If the loop was performed 20 times, the offsprings would assume the parents values.

It is worth noticing that we converted the price of each food to dollars because the nutrient values in the dataset are based on one dollar.

Regarding our fitness function, it calculates the sum, converted into dollars, of the product of each food item's quantity in the solution and its corresponding price. This value corresponds to the price of the whole diet plan per day, which is what we aim to minimize.

When creating a new generation, there is a 0.9 probability of applying crossover and a 0.2 probability of applying mutation to the individuals. We also implemented the possibility of applying Elitism and Fitness Sharing to our algorithm. Elitism allows us to keep the best individual from one generation into the next one, while Fitness Sharing helps increase diversity in the population. The latter method is a good practice to encounter the premature convergence to a local optimum. In what concerns this last issue, we also considered using Restricted Mating, however, it would imply having one more parameter to consider, so we decided not to apply it, also due to the additional computational cost it would carry.

We included 3 selection methods. The first one was the Fitness Proportionate selection, where the higher the fitness value, the higher its probability of being chosen, in case of a maximization problem. It occurs the other way around in case of a minimization problem. The second was the Tournament selection, where the fittest individual among a subset of individuals of the population is chosen, and finally, the Ranking selection, which assigns selection probabilities to each individual, based on its relative ranks, and chooses one.

The mutations that were implemented were the Swap mutation, where two genes randomly selected are swapped, the Inversion mutation, where the order of a subset of genes of an individual is reversed, and finally, the Random mutation, which introduces random changes to the genes of an individual.

Regarding crossover, we implemented the Single Point crossover, where a crossover point is randomly selected and the genetic material is swapped between the two parents, generating two offsprings, the Multi Point crossover, which is the same as the first one, but in this crossover the number of the randomly selected crossover points is multiple. The last one was the Uniform crossover, which recombines the genes between the parents by randomly selecting the genes either from one parent or the other, with equal probability.

As we can see, several combinations using Elitism, Fitness Sharing, each crossover, selection and mutation can be created. In order to try to get the best solution possible, we performed 3 runs of every single combination of these methods, with 30 generations and with a population size of 70. We understand that the number of runs, as well as the number of individuals in the population and the number of generations, should be higher, however, due to its computational cost, this was what we belief to be a good tradeoff.

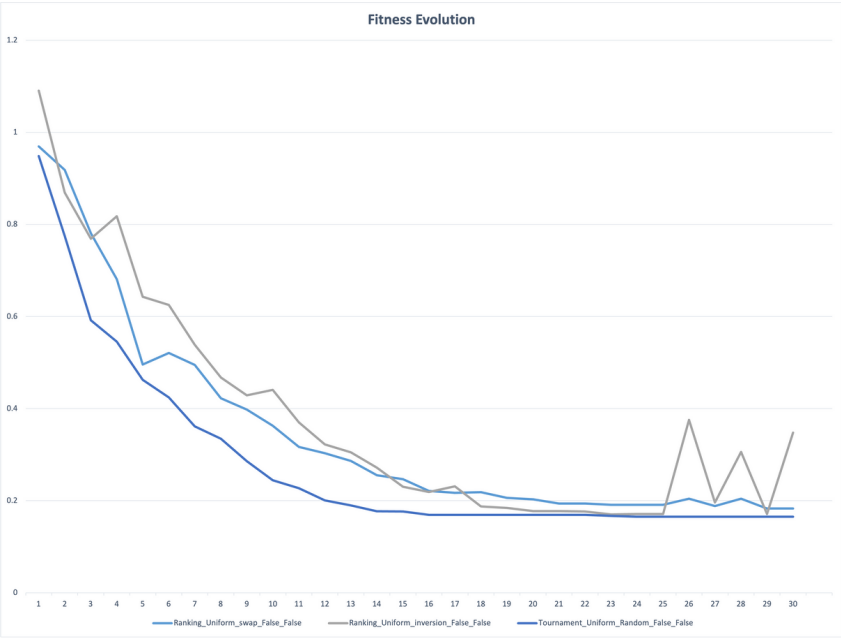
TESTS AND DISCUSSION

Some attempts were performed with Fitness Sharing enabled, but this method would present some difficulties which lead us to discard it from consideration. On the one hand, it altered the meaning of the fitness value, which would not represent the total price anymore. This was not a technical issue, but presented itself as an inconvenience as we had to alter the whole algorithm for this one change in meaning. On the other hand, on the trial runs we performed, the fitness values converged to 0 very fast with minimal change to the optimal representation while also giving unsatisfactory total price values.

To facilitate the evaluation of the optimal algorithm among various combinations, a decision was made to record the data associated with these combinations for subsequent analysis in Excel. The Best Fitness value from each run was utilized to compute the Average Best Fitness for each combination. This enabled us to identify the top 10 Genetic Algorithms that demonstrated superior performance for our specific problem. The table below showcases these top 10 algorithms.

Algorithm	Selection	Crossover	Mutation	Elitism	Average Best Fitnes
GA1	Tournament	Uniform	Random	False	0.1650
GA2	Uniform	Ranking	Inversion	False	0.1692
GA3	Ranking	Uniform	Swap	False	0.1829
GA4	Tournament	Multi point	Inversion	False	0.1908667
GA5	Ranking	Uniform	Random	False	0.195393102
GA6	Tournament	Uniform	Inversion	True	0.198433333
GA7	Tournament	Multi point	Random	True	0.201288103
GA8	FPS	Multi point	Inversion	False	0.2017
GA9	Tournament	Uniform	Random	True	0.207655759
GA10	Tournament	Uniform	Swap	False	0.2087

To comprehensively assess the performance of our configurations, it would have been prudent to employ robust statistical tests, such as the Kolmogorov-Smirnov test. These tests would have allowed us to determine the statistical significance of the differences between the results obtained from different configurations. *"understand whether the differences between the results obtained by the two configurations are statistically significant or not*" [2](Vanneschi Silva Book).
After this statistical test we could have considered also the Success Rate, SR = successful runs/ performed runs, but in this problem we had difficulty defining a successful run.



Subsequently, we analyzed the performance of the three most promising algorithms using a line graph. This graph plotted the Best Fitness value on the y-axis and the number of generations on the x-axis. The purpose of this analysis was to examine the behavior and trends exhibited by these algorithms over a span of 30 generations. We can identify that Tournament_Uniform_Random_False_False is the best GA, and shows more stability when compared with the others GA.

FINAL DIET

Having all the GA in considerations and after all the solutions presented we can present a final diet plan for Stingler. With this diet plan he will be able to sastify all the macros required while also being able to save the major amount of money. The daily price of the diet will be 13,5 cents. The aliments present in the final diet are: Wheat Flour (Enriched): 1.0 lb. , Evaporated Milk (can)': '2.9 oz.','Onions': '0.3 lb.', 'Spinach': '0.1 lb.', 'Sweet Potatoes: 0.5 lb. ,Navy Beans, Dried': '0.7 lb.'}.

CONCLUSION

In conclusion, the application of Genetic Algorithms to the Stigler's diet problem has demonstrated their effectiveness in addressing complex optimization challenges. The Stigler's diet problem, with its vast number of variables and constraints, serves as an ideal testbed for evaluating the capabilities of Genetic Algorithms in finding optimal solutions.

The experimental results obtained through the application of Genetic Algorithms to the Stigler's diet problem have showcased their ability to find near-optimal solutions efficiently.

However, it is worth noting that we also encountered some of the algorithm's limitations. The performance of the algorithm relies heavily on the selection of appropriate genetic operators, population size and its stochastic initialisation, as well as the termination criteria and restraints. Furthermore, the computational complexity of Genetic Algorithms increases exponentially with the size of the problem, necessitating careful consideration of computational resources and time constraints. This proved a problem when an exhaustive Grid Search of parameters was performed, specially when considering the constraint of requiring solutions to be within satisfactory macro bounds.

It is undeniable that we fell short of the optimum achieved by Stigler himself of 39$ per year, having achieved a minimum of 49$ a year. It is important to note, however, that different algorithms were used in both approaches.

Furthermore, one aspect that could have been better implemented in our perspective is the application of the restraint. In hindsight, the minimum daily intake restraint was slightly too "brute force", which inevitably lead to longer run times and issues with infinite loops.

Nevertheless, we believe that we achieved good results with our implementation of the algorithm.

REFERENCES

https://developers.google.com/optimization/lp/stigler_diet [1]

https://elearning.novaims.unl.pt/pluginfile.php/159770/mod_resource/content/1/Vanneschi-Silva-Book.pdf [2]

https://ictactjournals.in/paper/IJSC_V6_I1_paper_4_pp_1083_1092.pdf [3]

https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm [4]