Stigler's Diet Problem:
A Genetic Algorithm Approach

Diogo Pitadas Fernandes – 20220507

GitHub Repository

Computational Intelligence for Optimization

Nova Information Management School

1. Problem definition

In 1945, “Stigler posed the following problem: For a moderately active man (economist) weighing 154 pounds, how much of each of 77 foods should be eaten on a daily basis so that the man’s intake of nine nutrients (including calories) will be at least equal to the recommended dietary allowances (RDAs) suggested by the National Research Council in 1943, with the cost of the diet being minimal?”. Being RDAs the levels of intake of essential nutrients deemed adequate to meet the known nutrient needs of practically all healthy persons (Garille et all. ,2001).

The aim of this project is to build a genetic algorithm that offers the best possible solution to the problem. Therefore, the original problem dataset was used, containing 77 food items, and their respective nutrient content and price (Yam, 2021) as well as the 1943 RDA nutrient values (Garille et all. ,2001).

2. Problem instance

Given the problem definition above, let us define a set *J* = {1, 2,…,77} of the available 77 food items. For each *i* in *J* let p(i) be the price of the food item and q(i) be the quantity chosen. Thus, the search space (S) and fitness function (f) can be defined by the following equations:

$$S = \{(q(1), q(2), \ldots, q(77)) \mid 0 \leq q(i) \leq 100,\ \forall i \in J\} \qquad f = \sum_{i=1}^{77} q_i \cdot p_i$$

q_i - quantity of food i
p_i - price of food i

Figure 1 - Search space and fitness function equations.

Note that the number of times a food can be chosen in the representation was set to 100, in an attempt to maintain a healthy number of combinations that allow the algorithm to explore different and varied diet options, while keeping the running time and computational resources needed at reasonable level.

To make the model reach a local optimum with a total cost as low as possible, but also respecting the nutrient thresholds, a penalty was added to the fitness function for diets that didn’t contain the right amount of nutrients. Five variations were created. They differ in two main aspects. The way to calculate the deviation from the diet nutrient content from the target nutrients - a more simplified, and computationally expensive approach with probably faster convergence, focusing on the overall diet balance rather than individual nutrient levels. It encourages the algorithm to achieve a balanced diet in terms of overall nutrients. On the other hand, a more granular and nuanced approach that can help avoiding scenarios where some nutrients are excessively consumed while others are insufficient.

The other variation aspect is whether the penalty was added only in case of nutrient shortage or in case of nutrient excess too. This was done to encourage the algorithm to meet nutritional requirements (as stated in the problem definition) and, at the same time, retrieve a balanced diet, without many possibly unhealthy nutrient excesses, namely calories.

Equation 1 in figure 2, shows the penalty formula for when the nutrient deviation is calculated by subtracting the sum of target nutrients by the current diet nutrient content. The penalty is only added if the subtraction result is positive, indicating a nutrient deficit. Equation 2 shows the formula where nutrient deviation is also calculated with the sum but in this case the penalty is added regardless nutrient shortage or excess. Equation 3 shows the formula for when the nutrient deviation is calculated for each nutrient but only added if there´s a nutrient shortage. Equation 4 it´s similar but adds the penalty whether exists a nutrient shortage or excess. Equation 5 follows the same nutrient

deviation method as the previous two but adds two times the weight to shortages and one time for excess.

The multiplications were added to increase the penalty weight and make the shortage and excess scenarios much less desirable and thus, less probable to be included in the final solution.

For this instance, the daily target nutrient values were multiplied by 365, to make the algorithm return a yearly balanced diet.

$$\text{Penalty} = f + 1000 \cdot \max\left(0, \sum_{j=1}^{n} t_j - \sum_{j=1}^{n} c_{i,j} \cdot q_i\right) \quad (1)$$

$$\text{Penalty} = f + 1000 \cdot \left|\sum_{j=1}^{n} t_j - \sum_{j=1}^{n} c_{i,j} \cdot q_i\right| \quad (2)$$

$$\text{Penalty} = f + \sum_{j=1}^{n} 1000 \cdot |t_j - c_{i,j} \cdot q_i| \ , \text{if } \mathbf{t_j} > \mathbf{c}_{i,j} \quad (3)$$

$$\text{Penalty} = f + \sum_{j=1}^{n} 1000 \cdot |t_j - c_{i,j} \cdot q_i| \ , \text{if } \mathbf{t_j} \neq \mathbf{c}_{i,j} \quad (4)$$

$$\text{Penalty} = \begin{cases} f + \sum_{j=1}^{n} (2000 \cdot |t_j - c_{i,j} \cdot q_i|) & \text{if } \mathbf{t_j} > \mathbf{c}_i \\ f + \sum_{j=1}^{n} (1000 \cdot |t_j - c_{i,j} \cdot q_i|) & \text{if } \mathbf{t_j} < \mathbf{c}_i \end{cases} \quad (5)$$

Code names:
1 - fitness_sum_shortfall, 2 - fitness_sum_excess_shortfall,
3 - fitness_each_shortfall, 4 -fitness_each_excess_shortfall,
5 - fitness_each_excess_shortfall_weighted

q_i - quantity of food i
$c_{i,j}$ - nutrient j amount in food i
t_j - nutrient j amount in target nutrients list

Figure 2- Penalty variations equations.

3. Genetic Operators

For the project, three selection methods, nine different crossover operators and eight mutation operators were tested, as shown in table 1.

Operator Type	Name	Description
Selection	Fitness proportionate selection	Choose an individual from a population based on their fitness, with higher fitness individuals having a higher probability of selection.
Selection	Tournament	Randomly chooses n individuals from the population and selects the best individual based on fitness. Tournament size can be specified.
Selection	Ranking	Begins by sorting the population based on fitness in descending order, assigns selection probabilities proportional to the rank of each individual's fitness, and selects an individual based on these probabilities.
Crossover	Single Point	Selects a random point to split the chromosomes of both parents and creates offspring by combining the segments from each parent.
Crossover	K Point	Similar to single point but with k randomly chosen crossover points
Crossover	Uniform	Swaps genes between the two parents with a given probability to create offspring.
Crossover	Adapted PMX	Similar to the classical PMX, this function exchanges segments between parents based on two random crossover points but does not ensure the uniqueness of genes in the offspring
Crossover	Ordered	Similar to the classical approach of ordered crossover, but utilizes index-based operations to perform crossover
Crossover	Average	Calculates the average of each parents' genes and rounds it to the nearest integer. Returns only one offspring

Crossover	Cycle	Similar to the classical approach of cycle crossover, but utilizes index-based operations to perform crossover
Crossover	Int Arithmetic	Similar to classical arithmetic crossover but uses a rounded-to-the-nearest-integer approach
Crossover	Fitness dependent	Applies a chosen crossover operator n times and selects the two best offspring based on fitness
Mutation	Flip food	Inspired by its flip bit counterpart, this function randomly selects a segment of the chromosome and flips each gene: 0 to a random integer between 1 and 100, and non-zero values to 0.
Mutation	Swap	Selects two random positions in the chromosome and swaps their values.
Mutation	Inversion	Selects a random segment of the chromosome and reverses the order of the genes within that segment.
Mutation	Shuffle	Selects a random segment of the chromosome and randomly shuffles the genes within that segment.
Mutation	Shift	Selects a random segment of the chromosome and shifts it left or right by a random number of positions (between 1 and 5), to the left or to the right (50% chance)
Mutation	Add or remove	Selects between 1 to 20 genes and adjusts them by adding values (1 to 3) if nutrient content is below the target or sets genes to 0 if above the target.
Mutation	Gaussian Adaptation	Randomly adds a value from a normal distribution (mean 0, standard deviation 10) to each gene with a 50% chance, ensuring values are within the range of 0 to 100.
Mutation	Fitness Dependent	Applies a chosen mutation operator n times and selects the best individual based on fitness.

Table 1 - Genetic operators tested.

4. Configurations testing

Ideally all possible combinations of the genetic operators and respective parameters would be tested and compared against each other. Since that would be extremely time consuming and computationally intensive, I decided to adopt a step wise approach. That is, starting with random operators, running multiple configurations, and keeping all the operators fixed but one. After analyzing the results, I elected the best operator and changed the previous random operator to the newly elected one. This process was done iteratively, until the best operator was selected for each type.

Given that the fitness function and crossover operators significantly influence algorithm performance and are highly problem-dependent, the fitness function was selected first. Subsequently, various crossover operators were tested. The optimal population size and crossover probabilities were then determined. Following this, the optimal tournament size was identified and included while evaluating the selection methods. Subsequently, different mutation operators and mutation probability were assessed. Finally, the effect of including or excluding elitism in the algorithm was examined.

The selection factors were solution fitness, speed of convergence and the quality of the solutions found, considering factors such as average food variety and nutrient balance (both excesses and deficiencies). The analysis was complemented with statistical tests. I used a mean comparison parametric test (ANOVA) since each configuration ran 30 times, adhering to the central limit theorem. To find pairwise mean differences, a post hoc Tukey test was performed. For each step all the visualizations, statistical output and additional indicators tables can be consulted in the Annex.

5. Results

5.1 Fitness function

The performance of the five fitness functions tested can be divided into two groups. There is a statistically significant difference in performance between the groups, but not within the members of each group. On one hand we have the functions that only penalize nutrient shortfall, either calculating it per nutrient or by sum of nutrients. On the other hand, there's the functions that penalize both shortfall and excess. This group starts with very high fitness and plateau around the 500th generation, while the other starts lower and after the 500th generation continues to improve slightly.

Since the problem definition doesn't include any nutrient excess mention, I decided to choose one of the functions that only focus nutrient shortage. Even if there's no statistical difference between the function that uses sum and the other that penalizes per nutrient, the first one could have more tendency to generate diets where not all target nutrients are met. Actually, the nutrient shortage average of only 30 runs is far greater when using the sum and almost zero when using the individual penalization. For those reasons, and to be more in line with the problem definition, I've chosen the latter.

5.2 Crossover operator

From all operators tested the fitness dependent crossover was the best performer, being statistically different from all the others. Along with average crossover and adapted arithmetic crossover, it was able to find a solution with no nutrient shortage. In terms of chromosome modification and recombination, this type of crossover uses any other crossover method, but adds an extra layer, enhancing the probability of returning offspring with better fitness. Although due to its repetitive nature it also greatly raises the running time.

When I first tested against the other operators, the crossover method was the single-point crossover. Later I tested it with all crossover operators and the best performer was the uniform operator. All proved statistically different from each other and all, but cycle crossover, retrieved solution without nutrient shortage, on average. The uniform operator had the lowest fitness, on average, and it was faster to converge. I believe that swapping genes between parents creates a lot of diversity, considering the representation adopted. And, on top of that, creating ten different offspring and selecting the best two, ensures that the randomness isn't too destructive for the gene pool, inhibiting the algorithm to improve.

It would be interesting to analyze the effect of lowering or raising the probability of gene swap between the two parents, but that level of granularity falls outside the scope of this project.

5.3 Population size

Until now, in all the tests made, the population parameter was set to 100 individuals. I decided to explore other options with the intent of optimizing this parameter. The population sizes tested ranged between 10 and 450 individuals.

The analysis showed, as expected, that larger populations tend to converge to solutions with better fitness, probably because a larger population tends to have more diversity and prevents premature convergence while balancing exploration of the solution space.

Statistically, there was no difference between a population of 200 and the others above (max tested was 450). Making this number the best candidate but since having a bigger population greatly increases the running time and the mean fitness difference between 200 individuals and 100, wasn't that great, a compromise was made, and the 100 individual population was kept.

5.4 Crossover Probability

Until now, in all the tests conducted, the crossover probability was set to 0.85. The probabilities tested ranged from 0.4 to 0.95 in 0.05 increments. The overall tendency shows that higher crossover probabilities tend to produce solutions with better fitness. More frequent crossover enables more extensive exploration of the solution space. However, if the probability is too high, the algorithm might struggle to maintain good solutions once they have been found, increasing the risk of disrupting highly fit individuals.

Since the crossover operator iterates through the offspring and selects the best, this reduces the risk of disruption. This may explain why, in this particular case, a very high crossover probability is beneficial. Nonetheless, since there is no statistical difference between 0.85 and higher probabilities, the parameter was kept at 0.85.

5.5 Selection Methods

Before comparing selection methods I determined what was the optimal tournament size. The overall tendency shows that the best values are between 2 and 8, after that the fitness tends to get worse. Bigger tournament sizes decrease the selection pressure, allowing weaker individuals to dominate and reproduce, thus reducing the overall quality of the solutions over time. Since there was no statistically significant difference between the aforementioned values, 4 was chosen because on average produced better solutions.

The three selection methods tested had similar performance, although they tested all statistically different. Fps was slightly faster than the other two, but in terms of fitness tournament selection had the best performance.

5.6 Mutation operator

From the operators tested, the best one is add or remove mutation. This mutation changes the genes in function of the chromosome nutrient balance, in an attempt to encourage the algorithm to pick foods that haven't been picked in case of nutrient shortage and removing food in case of excess. Since the operation is done in a random segment of the chromosome it doesn't prove too destructive, and even if it was there's already all the other operators that would lower the probability of that individual to propagate.

Flip food mutation is a simplified version. It doesn't take into consideration nutrient content but inherits the same encouragement idea. Since there's no statistically significant difference between them, I chose flip food because it's less computationally intensive.

5.7 Mutation Probability

Until now, in all the tests conducted, the crossover probability was set to 0.15. The probabilities tested ranged from 0.01 to 0.5 in 0.5 increments. The overall tendency shows that higher mutation probabilities tend to produce solutions with better fitness. Since the solution space is quite large in this problem instance a higher chance of mutation enables more exploration and avoids the algorithm getting stuck in a local optimum, by introducing diversity in the population.

The fitness improvement becomes more visible with probabilities above 0.20. Since there was no statistical difference between 0.2 and any tested probability above it, I opted for a more conservative rout and adopted the 0.2 probability.

5.8 Elitism

Elitism often proves beneficial by ensuring that the fittest solutions are preserved to continue influencing the evolving population.

However, in this instance, it had no discernible effect on the algorithm; there was no statistically significant difference observed between running the algorithm with elitism or without it. This may be due to the use of a fitness-dependent crossover operator. Fitness-dependent crossover prioritizes better solutions by ensuring that higher fitness solutions contribute more to the offspring generation. This mechanism may be naturally preserving the best solutions without the explicit need for elitism, which might explain the lack of significant improvement when elitism is employed.

6. Conclusions

The final algorithm configuration is: population with 100 individuals; tournament selection with tournament size 4; fitness dependent crossover with uniform crossover and a probability of 0.85; and flip food mutation with a probability of 0.20. This configuration was able to reach a final solution of 4.8 $ yearly diet cost without any nutrient shortage.

The global optimum is known for the problem at hand – 39.93 $. When comparing the reached solution I noticed that must have been an error in the algorithm development. After debugging I found the issue was with target nutrient insertion. Instead of the 3000 calories and 5000 IU for Vitamin A, I inserted 3 and 5 respectively. Since every aspect of the algorithm was tested and every decision tailored to different nutrient amounts, when changing the values to the correct ones the algorithm gets stuck in a local optimum very far from the global optimum. This indicates that the operators chosen are sensitive to the specific nutrient values used for the problem, and not robust against changes in these values. To correct this the operators and parameters need to be tested again and reselected.

Regarding project limitations, there were more parameters that could be tested and optimized, for example, the number of crossover points in k point crossover or the uniform crossover probability. Additionally, there were variations in generation numbers across experiments due to running time constraints. Generation variation can negatively impact algorithm convergence because inconsistent generation numbers mean that different experiments may not allow the algorithm to reach a steady state or optimal solution. Although, I consider that the generation variation didn't have a great impact on the results because the charts indicate a fast convergence and there wasn't convergence failure in any of the experiments. Nevertheless, more generations could always lead to a better solution in the genetic algorithms' context.

Nonetheless, while using the wrong nutrient values, I consider the algorithm to be able to reach a desirable solution with smooth convergence.

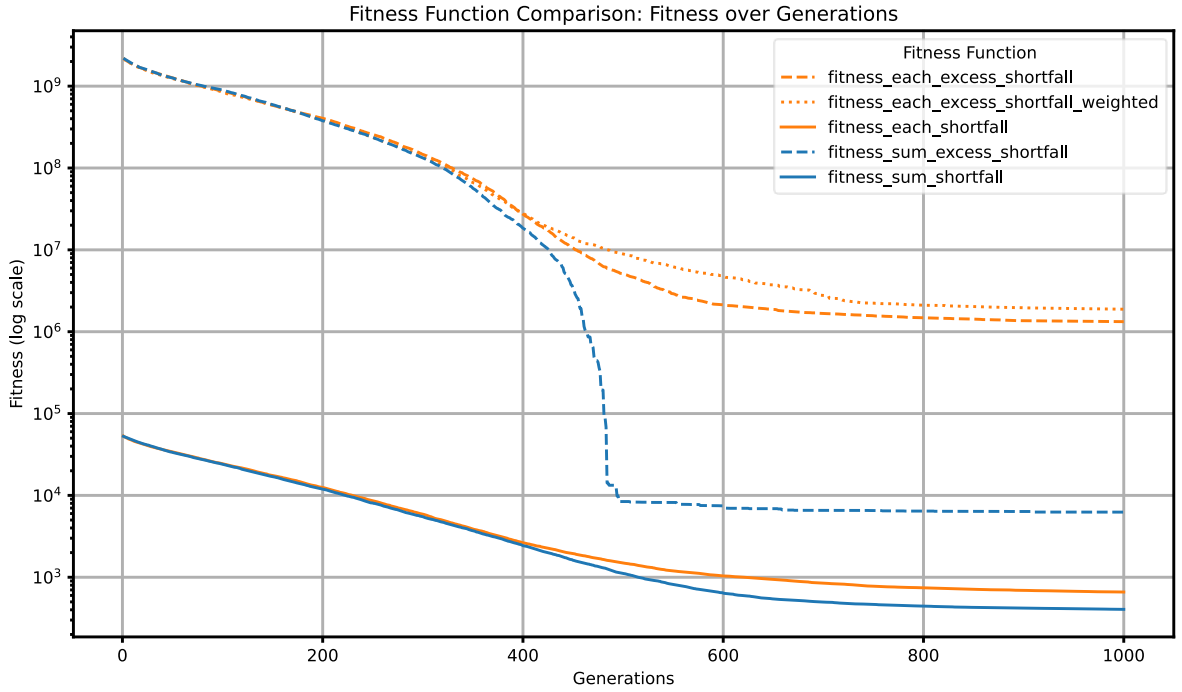
References

Garille, S. G., & Gass, S. I. (2001). Stigler's diet problem revisited. *Operations Research*, *49*(1), 1-13.

Yam Peleg. December 2021. Calculating An Optimal Diet. Version 1. Retrieved 10/05/2024 from https://www.kaggle.com/datasets/yamqwe/calculating-an-optimal-diet.

Annex

Annex 1 – Fitness Function Analysis



Anova Results:

	df	sum_sq	mean_sq	F	PR(>F)
fitness_func	4.00	1833951302034303488.00	458487825508575872.00	671.31	0.00
Residual	145.00	99031980097176944.00	682979173083978.88	NaN	NaN

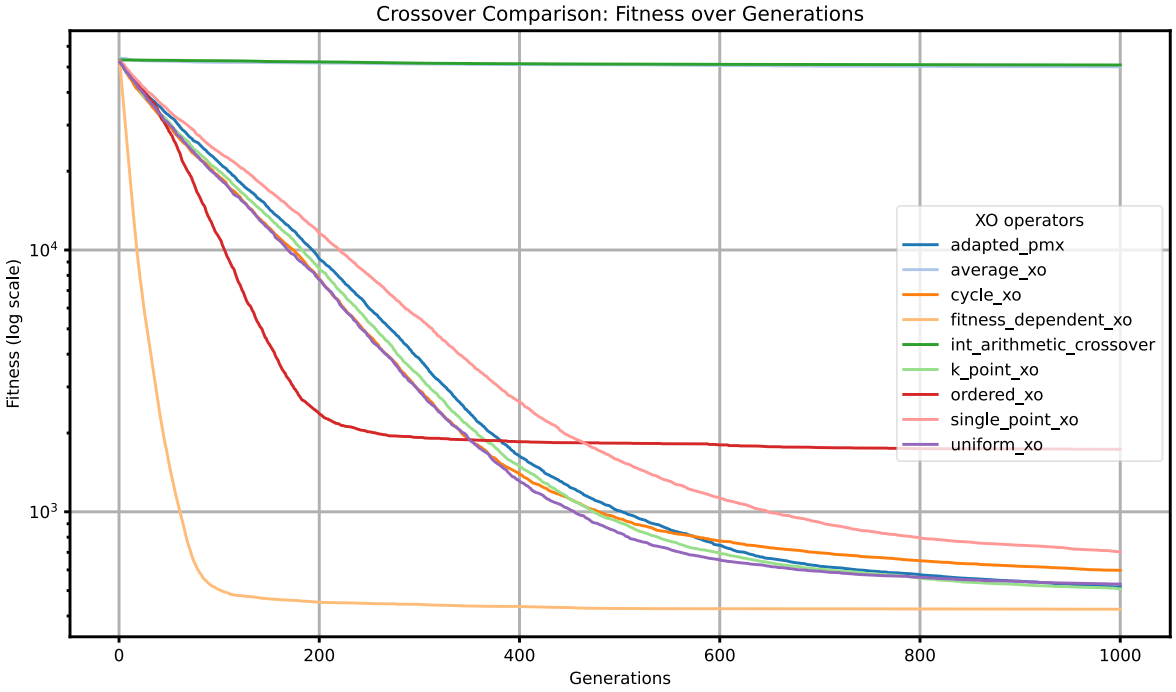
Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
fitness_each_excess_shortfall	fitness_each_excess_shortfall_weighted	-4436961.1417	0.965	-23076947.4066	14203025.1233	False
fitness_each_excess_shortfall	fitness_each_shortfall	-228421141.0274	0.0	-247061127.2923	-209781154.7624	True
fitness_each_excess_shortfall	fitness_sum_excess_shortfall	-3773105.3233	0.9806	-22413091.5883	14866880.9416	False
fitness_each_excess_shortfall	fitness_sum_shortfall	-228421470.2334	0.0	-247061456.4984	-209781483.9685	True
fitness_each_excess_shortfall_weighted	fitness_each_shortfall	-223984179.8857	0.0	-242624166.1507	-205344193.6207	True
fitness_each_excess_shortfall_weighted	fitness_sum_excess_shortfall	663855.8183	1.0	-17976130.4466	19303842.0833	False
fitness_each_excess_shortfall_weighted	fitness_sum_shortfall	-223984509.0918	0.0	-242624495.3567	-205344522.8268	True
fitness_each_shortfall	fitness_sum_excess_shortfall	224648035.704	0.0	206008049.4391	243288021.969	True
fitness_each_shortfall	fitness_sum_shortfall	-329.2061	1.0	-18640315.4711	18639657.0589	False
fitness_sum_excess_shortfall	fitness_sum_shortfall	-224648364.9101	0.0	-243288351.1751	-206008378.6451	True

Solution average values of 30 runs

fitness_func	fitness	food variety	nutrient_excess	nutrient_shortage
fitness_sum_shortfall	405.27	20.97	26836.97	-22666.10
fitness_each_shortfall	659.88	26.40	64187.72	-3.74
fitness_sum_excess_shortfall	6247.61	49.57	13339.48	-7605.90
fitness_each_excess_shortfall	1326581.02	46.07	3008.44	-1852.10
fitness_each_excess_shortfall_weighted	1881894.01	44.97	6632.97	-458.52

Annex 2 – Crossover Operator Analysis



Anova Results:

	df	sum_sq	mean_sq	F	PR(>F)
xo	8.00	100298763244.73	12537345405.59	8910.99	0.00
Residual	261.00	367214740.62	1406953.03	NaN	NaN

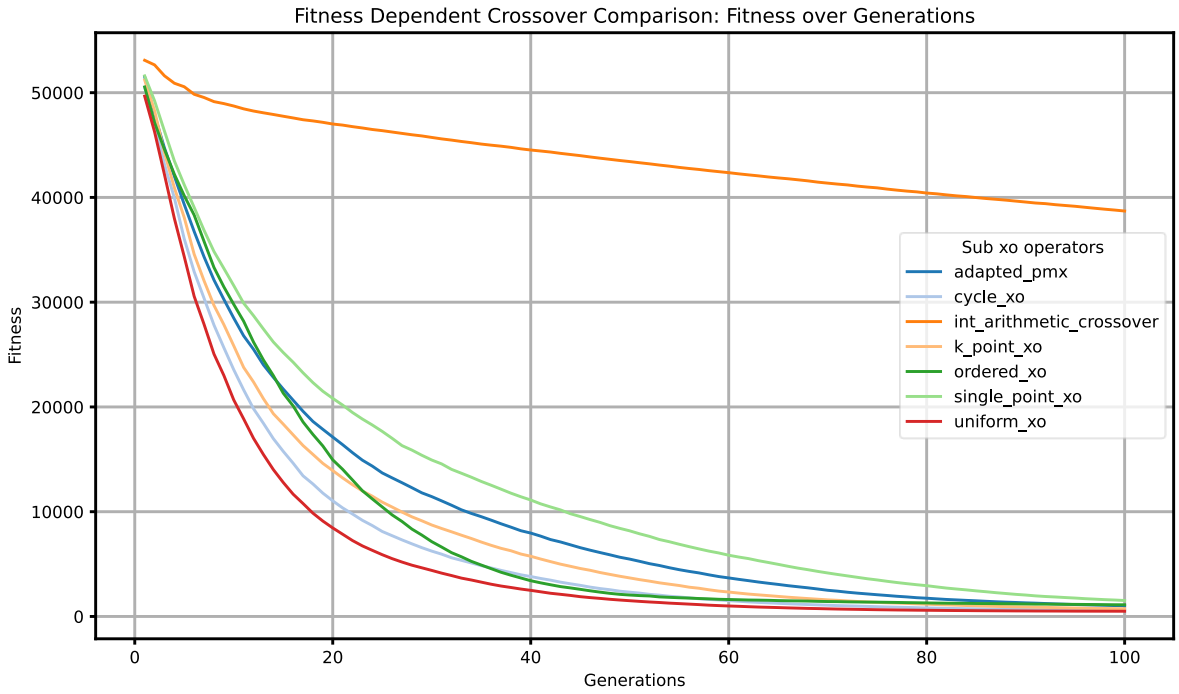
Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
adapted_pmx	average_xo	44925.6515	0.0	43967.6521	45883.6508	True
adapted_pmx	cycle_xo	-606.2152	0.5592	-1564.2146	351.7841	False
adapted_pmx	fitness_dependent_xo	-5137.4385	0.0	-6095.4379	-4179.4392	True
adapted_pmx	int_arithmetic_crossover	45455.8958	0.0	44497.8965	46413.8952	True
adapted_pmx	k_point_xo	-381.0779	0.9458	-1339.0772	576.9215	False
adapted_pmx	ordered_xo	-1257.1452	0.0018	-2215.1445	-299.1458	True
adapted_pmx	single_point_xo	958.2492	0.0499	0.2499	1916.2486	True
adapted_pmx	uniform_xo	-652.3579	0.4552	-1610.3573	305.6414	False
average_xo	cycle_xo	-45531.8667	0.0	-46489.866	-44573.8673	True
average_xo	fitness_dependent_xo	-50063.09	0.0	-51021.0893	-49105.0906	True
average_xo	int_arithmetic_crossover	530.2444	0.7267	-427.755	1488.2437	False
average_xo	k_point_xo	-45306.7293	0.0	-46264.7287	-44348.73	True
average_xo	ordered_xo	-46182.7966	0.0	-47140.796	-45224.7973	True
average_xo	single_point_xo	-43967.4022	0.0	-44925.4016	-43009.4029	True
average_xo	uniform_xo	-45578.0094	0.0	-46536.0087	-44620.01	True
cycle_xo	fitness_dependent_xo	-4531.2233	0.0	-5489.2227	-3573.224	True
cycle_xo	int_arithmetic_crossover	46062.1111	0.0	45104.1117	47020.1104	True
cycle_xo	k_point_xo	225.1373	0.9982	-732.862	1183.1367	False
cycle_xo	ordered_xo	-650.9299	0.4583	-1608.9293	307.0694	False
cycle_xo	single_point_xo	1564.4644	0.0	606.4651	2522.4638	True
cycle_xo	uniform_xo	-46.1427	1.0	-1004.1421	911.8566	False
fitness_dependent_xo	int_arithmetic_crossover	50593.3344	0.0	49635.335	51551.3337	True
fitness_dependent_xo	k_point_xo	4756.3607	0.0	3798.3613	5714.36	True
fitness_dependent_xo	ordered_xo	3880.2934	0.0	2922.294	4838.2927	True
fitness_dependent_xo	single_point_xo	6095.6878	0.0	5137.6884	7053.6871	True
fitness_dependent_xo	uniform_xo	4485.0806	0.0	3527.0813	5443.08	True
int_arithmetic_crossover	k_point_xo	-45836.9737	0.0	-46794.9731	-44878.9744	True
int_arithmetic_crossover	ordered_xo	-46713.041	0.0	-47671.0404	-45755.0416	True
int_arithmetic_crossover	single_point_xo	-44497.6466	0.0	-45455.646	-43539.6473	True
int_arithmetic_crossover	uniform_xo	-46108.2538	0.0	-47066.2531	-45150.2544	True
k_point_xo	ordered_xo	-876.0673	0.1034	-1834.0666	81.9321	False
k_point_xo	single_point_xo	1339.3271	0.0006	381.3278	2297.3265	True
k_point_xo	uniform_xo	-271.28	0.9936	-1229.2794	686.7193	False
ordered_xo	single_point_xo	2215.3944	0.0	1257.395	3173.3937	True
ordered_xo	uniform_xo	604.7872	0.5624	-353.2121	1562.7866	False
single_point_xo	uniform_xo	-1610.6072	0.0	-2568.6065	-652.6078	True

Solution average values of 30 runs

xo	fitness	food variety	nutrient_excess	nutrient_shortage
fitness_dependent_xo	424.44	13.67	49190.80	0.00
k_point_xo	507.89	17.50	59715.32	-101.44
adapted_pmx	516.99	18.50	51975.87	-573.26
uniform_xo	529.68	17.83	59881.23	-62.57
cycle_xo	597.97	23.13	49297.05	-27.66
single_point_xo	703.01	27.93	66112.32	-22.84
ordered_xo	1732.50	48.20	41036.04	-887.83
average_xo	50121.05	76.03	2587972.92	0.00
int_arithmetic_crossover	50915.41	76.90	2493057.14	0.00

Annex 3 – Fitness Dependent Crossover Analysis



Anova Results:

	df	sum_sq	mean_sq	F	PR(>F)
dependent_xo	6.00	32163238146.51	5360539691.09	10950.74	0.00
Residual	203.00	99371347.92	489514.03	NaN	NaN

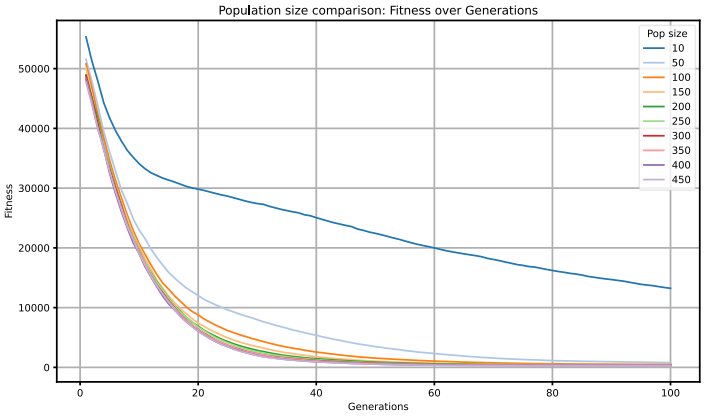
Multiple Comparison of Means - Tukey HSD, FWER=0.05

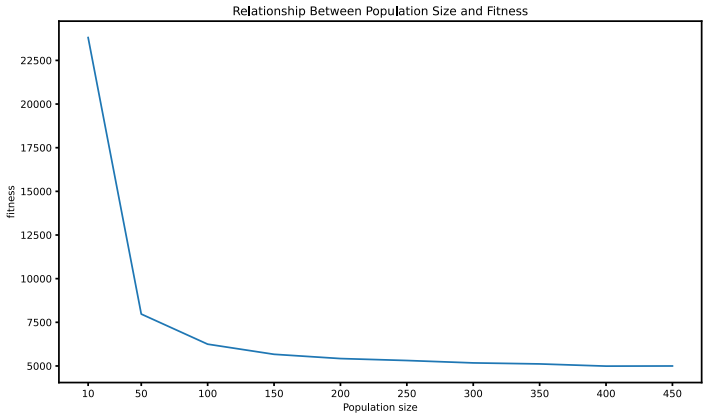
group1	group2	meandiff	p-adj	lower	upper	reject
adapted_pmx	cycle_xo	-2937.4179	0.0	-3475.4189	-2399.4168	True
adapted_pmx	int_arithmetic_crossover	33618.5086	0.0	33080.5075	34156.5096	True
adapted_pmx	k_point_xo	-1607.4435	0.0	-2145.4446	-1069.4425	True
adapted_pmx	ordered_xo	-1676.7618	0.0	-2214.7628	-1138.7608	True
adapted_pmx	single_point_xo	2271.9418	0.0	1733.9408	2809.9428	True
adapted_pmx	uniform_xo	-4054.9525	0.0	-4592.9535	-3516.9515	True
cycle_xo	int_arithmetic_crossover	36555.9264	0.0	36017.9254	37093.9275	True
cycle_xo	k_point_xo	1329.9743	0.0	791.9733	1867.9754	True
cycle_xo	ordered_xo	1260.6561	0.0	722.655	1798.6571	True
cycle_xo	single_point_xo	5209.3597	0.0	4671.3586	5747.3607	True
cycle_xo	uniform_xo	-1117.5346	0.0	-1655.5357	-579.5336	True
int_arithmetic_crossover	k_point_xo	-35225.9521	0.0	-35763.9531	-34687.9511	True
int_arithmetic_crossover	ordered_xo	-35295.2704	0.0	-35833.2714	-34757.2693	True
int_arithmetic_crossover	single_point_xo	-31346.5668	0.0	-31884.5678	-30808.5657	True
int_arithmetic_crossover	uniform_xo	-37673.4611	0.0	-38211.4621	-37135.46	True
k_point_xo	ordered_xo	-69.3183	0.9997	-607.3193	468.6828	False
k_point_xo	single_point_xo	3879.3853	0.0	3341.3843	4417.3864	True
k_point_xo	uniform_xo	-2447.509	0.0	-2985.51	-1909.5079	True
ordered_xo	single_point_xo	3948.7036	0.0	3410.7026	4486.7046	True
ordered_xo	uniform_xo	-2378.1907	0.0	-2916.1917	-1840.1897	True
single_point_xo	uniform_xo	-6326.8943	0.0	-6864.8953	-5788.8933	True

Solution average values of 30 runs

dependent_xo	fitness	food variety	nutrient_excess	nutrient_shortage
uniform_xo	497.76	17.70	53178.92	0.00
cycle_xo	587.61	22.53	56743.09	-6.40
k_point_xo	737.75	28.53	60746.08	0.00
adapted_pmx	1018.27	36.97	75806.46	0.00
ordered_xo	1115.59	41.30	61606.98	0.00
single_point_xo	1523.91	41.73	96321.84	0.00
int_arithmetic_crossover	38702.90	77.00	2047368.96	0.00

Annex 4 – Population Size Analysis





Anova Results:

	df	sum_sq	mean_sq	F	PR(>F)
pop	1.00	3383984393.69	3383984393.69	171.46	0.00
Residual	298.00	5881293924.55	19735885.65	NaN	NaN

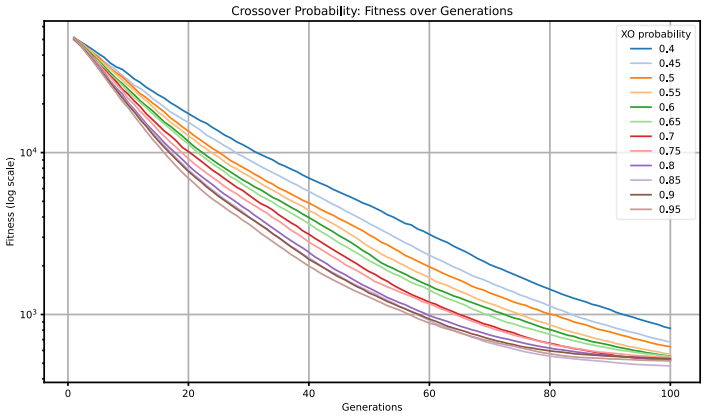
Multiple Comparison of Means - Tukey HSD, FWER=0.05

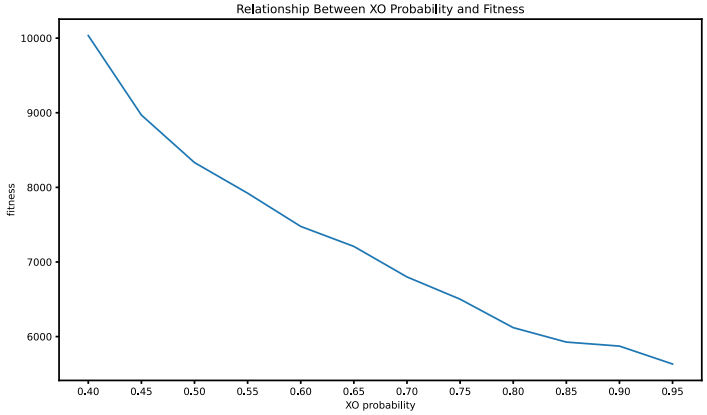
group1	group2	meandiff	p-adj	lower	upper	reject
10	50	-15834.9981	0.0	-16429.5383	-15240.458	True
10	100	-17559.1293	0.0	-18153.6695	-16964.5891	True
10	150	-18139.4009	0.0	-18733.9411	-17544.8607	True
10	200	-18383.4882	0.0	-18978.0284	-17788.948	True
10	250	-18494.788	0.0	-19089.3282	-17900.2479	True
10	300	-18632.5658	0.0	-19227.106	-18038.0256	True
10	350	-18690.7302	0.0	-19285.2703	-18096.19	True
10	400	-18816.2129	0.0	-19410.7531	-18221.6727	True
10	450	-18808.6757	0.0	-19403.2159	-18214.1355	True
50	100	-1724.1312	0.0	-2318.6713	-1129.591	True
50	150	-2304.4028	0.0	-2898.9429	-1709.8626	True
50	200	-2548.4901	0.0	-3143.0302	-1953.9499	True
50	250	-2659.7899	0.0	-3254.3301	-2065.2497	True
50	300	-2797.5677	0.0	-3392.1078	-2203.0275	True
50	350	-2855.732	0.0	-3450.2722	-2261.1919	True
50	400	-2981.2148	0.0	-3575.7549	-2386.6746	True
50	450	-2973.6776	0.0	-3568.2177	-2379.1374	True
100	150	-580.2716	0.0623	-1174.8118	14.2686	False
100	200	-824.3589	0.0006	-1418.8991	-229.8187	True
100	250	-935.6587	0.0	-1530.1989	-341.1186	True
100	300	-1073.4365	0.0	-1667.9767	-478.8963	True
100	350	-1131.6009	0.0	-1726.141	-537.0607	True
100	400	-1257.0836	0.0	-1851.6238	-662.5434	True
100	450	-1249.5464	0.0	-1844.0866	-655.0062	True
150	200	-244.0873	0.951	-838.6275	350.4529	False
150	250	-355.3871	0.665	-949.9273	239.153	False
150	300	-493.1649	0.2019	-1087.7051	101.3753	False
150	350	-551.3293	0.0952	-1145.8694	43.2109	False
150	400	-676.812	0.0122	-1271.3522	-82.2718	True
150	450	-669.2748	0.014	-1263.815	-74.7346	True
200	250	-111.2998	0.9999	-705.84	483.2403	False
200	300	-249.0776	0.9445	-843.6178	345.4626	False
200	350	-307.242	0.8233	-901.7821	287.2982	False
200	400	-432.7247	0.3788	-1027.2649	161.8155	False
200	450	-425.1875	0.4051	-1019.7277	169.3527	False
250	300	-137.7778	0.9992	-732.3179	456.7624	False
250	350	-195.9421	0.9887	-790.4823	398.598	False

Solution average values of 30 runs

pop	fitness	food variety	nutrient_excess	nutrient_shortage
450	320.26	8.83	49805.18	-0.00
400	329.44	8.77	49837.01	0.00
350	334.27	9.20	47650.72	-0.00
250	360.51	10.43	47225.31	0.00
300	361.75	10.43	47391.60	-0.00
200	415.25	13.63	47136.74	0.00
150	430.14	14.07	56735.22	0.00
100	495.61	18.13	53566.78	0.00
50	802.02	26.53	60455.16	0.00
10	13229.96	68.60	878345.15	0.00

Annex 5 – Crossover Probability Analysis





Anova Results:

	df	sum_sq	mean_sq	F	PR(>F)
xo_prob	1.00	583517878.34	583517878.34	2181.90	0.00
Residual	358.00	95741867.29	267435.38	NaN	NaN

Multiple Comparison of Means - Tukey HSD, FWER=0.05

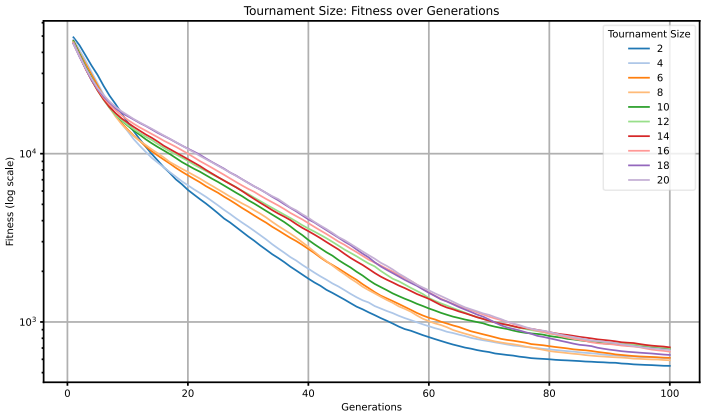
group1	group2	meandiff	p-adj	lower	upper	reject
0.4	0.45	-1065.4676	0.0	-1414.1476	-716.7876	True
0.4	0.5	-1702.5336	0.0	-2051.2136	-1353.8536	True
0.4	0.55	-2112.5186	0.0	-2461.1986	-1763.8386	True
0.4	0.6	-2558.0905	0.0	-2906.7705	-2209.4105	True
0.4	0.65	-2826.145	0.0	-3174.825	-2477.465	True
0.4	0.7	-3235.1646	0.0	-3583.8446	-2886.4846	True
0.4	0.75	-3533.3009	0.0	-3881.9809	-3184.6209	True
0.4	0.8	-3914.7528	0.0	-4263.4328	-3566.0728	True
0.4	0.85	-4108.3344	0.0	-4457.0144	-3759.6544	True
0.4	0.9	-4162.0272	0.0	-4510.7072	-3813.3472	True
0.4	0.95	-4402.1057	0.0	-4750.7857	-4053.4257	True
0.45	0.5	-637.066	0.0	-985.746	-288.386	True
0.45	0.55	-1047.0511	0.0	-1395.7311	-698.3711	True
0.45	0.6	-1492.623	0.0	-1841.303	-1143.943	True
0.45	0.65	-1760.6774	0.0	-2109.3574	-1411.9974	True
0.45	0.7	-2169.697	0.0	-2518.377	-1821.017	True
0.45	0.75	-2467.8333	0.0	-2816.5133	-2119.1533	True
0.45	0.8	-2849.2853	0.0	-3197.9653	-2500.6053	True
0.45	0.85	-3042.8669	0.0	-3391.5469	-2694.1869	True
0.45	0.9	-3096.5597	0.0	-3445.2397	-2747.8797	True
0.45	0.95	-3336.6382	0.0	-3685.3182	-2987.9582	True
0.5	0.55	-409.985	0.0072	-758.665	-61.305	True
0.5	0.6	-855.5569	0.0	-1204.2369	-506.8769	True
0.5	0.65	-1123.6114	0.0	-1472.2914	-774.9314	True
0.5	0.7	-1532.631	0.0	-1881.311	-1183.951	True
0.5	0.75	-1830.7673	0.0	-2179.4473	-1482.0873	True
0.5	0.8	-2212.2192	0.0	-2560.8992	-1863.5392	True
0.5	0.85	-2405.8008	0.0	-2754.4808	-2057.1208	True
0.5	0.9	-2459.4936	0.0	-2808.1736	-2110.8136	True
0.5	0.95	-2699.5721	0.0	-3048.2521	-2350.8921	True
0.55	0.6	-445.5719	0.002	-794.2519	-96.8919	True
0.55	0.65	-713.6264	0.0	-1062.3064	-364.9464	True
0.55	0.7	-1122.646	0.0	-1471.326	-773.966	True
0.55	0.75	-1420.7823	0.0	-1769.4623	-1072.1023	True
0.55	0.8	-1802.2342	0.0	-2150.9142	-1453.5542	True
0.55	0.85	-1995.8158	0.0	-2344.4958	-1647.1358	True
0.55	0.9	-2049.5086	0.0	-2398.1886	-1700.8286	True
0.55	0.95	-2289.5871	0.0	-2638.2671	-1940.9071	True

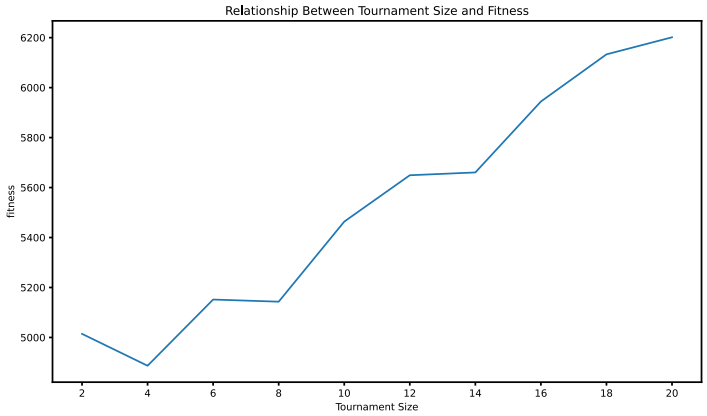
```
 0.6   0.65  -268.0545 0.3251  -616.7345    80.6255  False
 0.6    0.7  -677.0741    0.0 -1025.7541  -328.3941   True
 0.6   0.75  -975.2104    0.0 -1323.8904  -626.5304   True
 0.6    0.8 -1356.6623    0.0 -1705.3423 -1007.9823   True
 0.6   0.85 -1550.2439    0.0 -1898.9239 -1201.5639   True
 0.6    0.9 -1603.9367    0.0 -1952.6167 -1255.2567   True
 0.6   0.95 -1844.0152    0.0 -2192.6952 -1495.3352   True
0.65    0.7  -409.0196 0.0074  -757.6996   -60.3396   True
0.65   0.75  -707.1559    0.0 -1055.8359  -358.4759   True
0.65    0.8 -1088.6078    0.0 -1437.2878  -739.9278   True
0.65   0.85 -1282.1894    0.0 -1630.8694  -933.5094   True
0.65    0.9 -1335.8822    0.0 -1684.5622  -987.2022   True
0.65   0.95 -1575.9607    0.0 -1924.6407 -1227.2807   True
 0.7   0.75  -298.1363 0.1789  -646.8163    50.5437  False
 0.7    0.8  -679.5882    0.0 -1028.2682  -330.9082   True
 0.7   0.85  -873.1698    0.0 -1221.8498  -524.4898   True
 0.7    0.9  -926.8626    0.0 -1275.5426  -578.1826   True
 0.7   0.95 -1166.9411    0.0 -1515.6211  -818.2611   True
0.75    0.8  -381.4519 0.0186  -730.1319   -32.7719   True
0.75   0.85  -575.0335    0.0  -923.7135  -226.3535   True
0.75    0.9  -628.7263    0.0  -977.4063  -280.0463   True
0.75   0.95  -868.8048    0.0 -1217.4848  -520.1248   True
 0.8   0.85  -193.5816 0.8022  -542.2616   155.0984  False
 0.8    0.9  -247.2744 0.4548  -595.9544   101.4056  False
 0.8   0.95  -487.3529 0.0004  -836.0329  -138.6729   True
0.85    0.9   -53.6928    1.0  -402.3728   294.9872  False
0.85   0.95  -293.7713 0.1966  -642.4513    54.9087  False
 0.9   0.95  -240.0785 0.5032  -588.7585   108.6015  False
------------------------------------------------------------
```

Solution average values of 30 runs

xo_prob	fitness	food variety	nutrient_excess	nutrient_shortage
0.85	480.76	17.33	49155.01	0.00
0.95	516.55	17.80	55926.42	-0.00
0.70	520.51	19.43	58155.79	-0.00
0.80	528.43	19.27	52655.49	0.00
0.90	533.46	18.97	49169.22	-0.15
0.75	542.10	19.30	54677.73	0.00
0.65	550.69	20.30	58412.79	0.00
0.60	551.89	21.50	50933.57	-0.07
0.55	569.06	22.23	52669.22	0.00
0.50	632.04	24.80	60445.50	0.00
0.45	674.18	27.20	56929.29	0.00
0.40	821.30	33.03	54693.51	0.00

Annex 6 – Tournament Size Analysis





Anova Results:

	df	sum_sq	mean_sq	F	PR(>F)
tour_size	1.00	57313306.55	57313306.55	216.48	0.00
Residual	298.00	78895224.85	264749.08	NaN	NaN

Multiple Comparison of Means - Tukey HSD, FWER=0.05

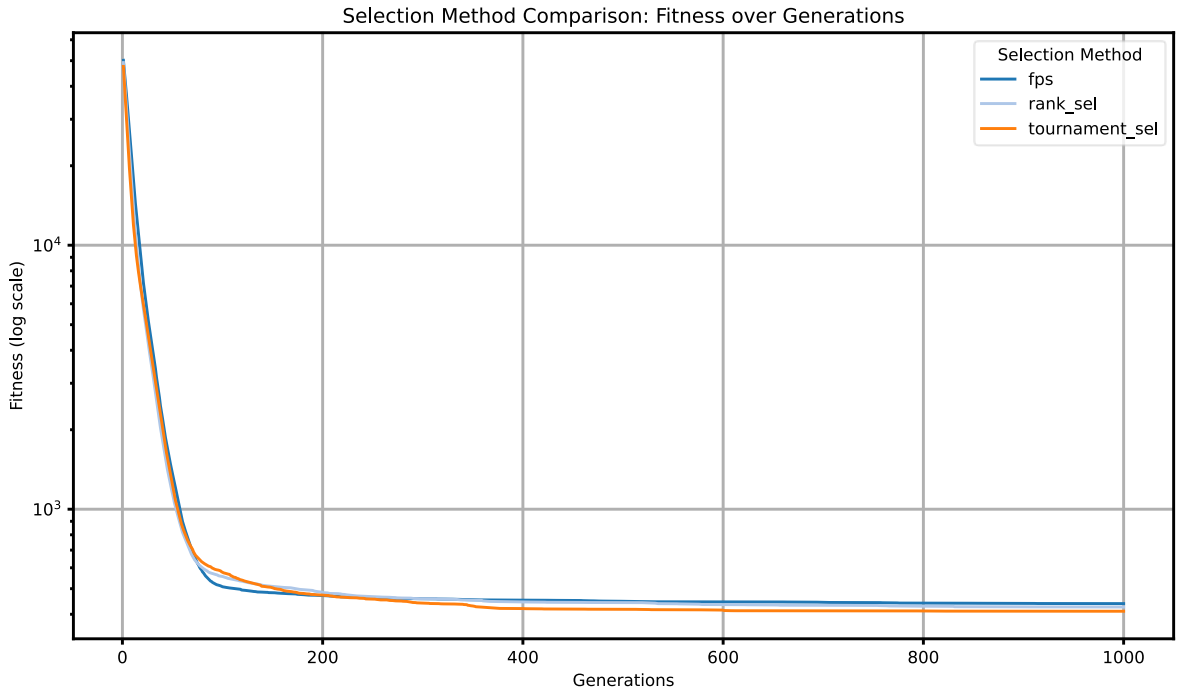
group1	group2	meandiff	p-adj	lower	upper	reject
2	4	-127.6374	0.994	-550.1713	294.8964	False
2	6	137.3171	0.9898	-285.2167	559.851	False
2	8	128.7214	0.9936	-293.8124	551.2552	False
2	10	449.2145	0.0271	26.6806	871.7483	True
2	12	634.8534	0.0001	212.3195	1057.3872	True
2	14	646.0521	0.0001	223.5183	1068.5859	True
2	16	929.9664	0.0	507.4326	1352.5003	True
2	18	1118.7269	0.0	696.1931	1541.2608	True
2	20	1187.0292	0.0	764.4954	1609.5631	True
4	6	264.9546	0.6001	-157.5793	687.4884	False
4	8	256.3588	0.6454	-166.175	678.8927	False
4	10	576.8519	0.0008	154.3181	999.3857	True
4	12	762.4908	0.0	339.957	1185.0246	True
4	14	773.6895	0.0	351.1557	1196.2234	True
4	16	1057.6039	0.0	635.07	1480.1377	True
4	18	1246.3644	0.0	823.8305	1668.8982	True
4	20	1314.6667	0.0	892.1328	1737.2005	True
6	8	-8.5957	1.0	-431.1296	413.9381	False
6	10	311.8973	0.358	-110.6365	734.4312	False
6	12	497.5362	0.0079	75.0024	920.0701	True
6	14	508.735	0.0058	86.2011	931.2688	True
6	16	792.6493	0.0	370.1155	1215.1831	True
6	18	981.4098	0.0	558.876	1403.9436	True
6	20	1049.7121	0.0	627.1783	1472.2459	True
8	10	320.4931	0.3187	-102.0408	743.0269	False
8	12	506.132	0.0062	83.5981	928.6658	True
8	14	517.3307	0.0046	94.7969	939.8645	True
8	16	801.245	0.0	378.7112	1223.7789	True
8	18	990.0055	0.0	567.4717	1412.5394	True
8	20	1058.3078	0.0	635.774	1480.8417	True
10	12	185.6389	0.9263	-236.8949	608.1727	False
10	14	196.8376	0.8971	-225.6962	619.3715	False
10	16	480.752	0.0123	58.2181	903.2858	True
10	18	669.5125	0.0	246.9786	1092.0463	True
10	20	737.8148	0.0	315.2809	1160.3486	True
12	14	11.1987	1.0	-411.3351	433.7326	False
12	16	295.1131	0.4407	-127.4208	717.6469	False
12	18	483.8736	0.0113	61.3397	906.4074	True

```
12     20  552.1759 0.0017   129.642  974.7097   True
14     16  283.9143 0.4991 -138.6195  706.4482  False
14     18  472.6748 0.0151    50.141  895.2087   True
14     20  540.9771 0.0023  118.4433   963.511   True
16     18  188.7605 0.9188 -233.7733  611.2943  False
16     20  257.0628 0.6418  -165.471  679.5966  False
18     20   68.3023    1.0 -354.2315  490.8361  False
-----------------------------------------------------
```

Solution average values of 30 runs

tour_size	fitness	food variety	nutrient_excess	nutrient_shortage
2	548.56	18.97	65989.47	0.00
8	592.05	18.53	100126.79	-0.00
4	608.79	20.43	103304.20	0.00
6	611.85	21.37	80377.97	-0.00
18	636.86	18.53	97336.26	-7.27
16	666.21	20.87	95925.59	0.00
20	680.50	20.93	106838.34	0.00
10	691.36	22.67	119881.78	0.00
12	699.38	24.47	106113.29	-0.00
14	706.62	22.43	92160.70	0.00

Annex 7 – Selection Methods Analysis



Anova Results:

	df	sum_sq	mean_sq	F	PR(>F)
selection	2.00	257960.60	128980.30	40.34	0.00
Residual	87.00	278152.11	3197.15	NaN	NaN

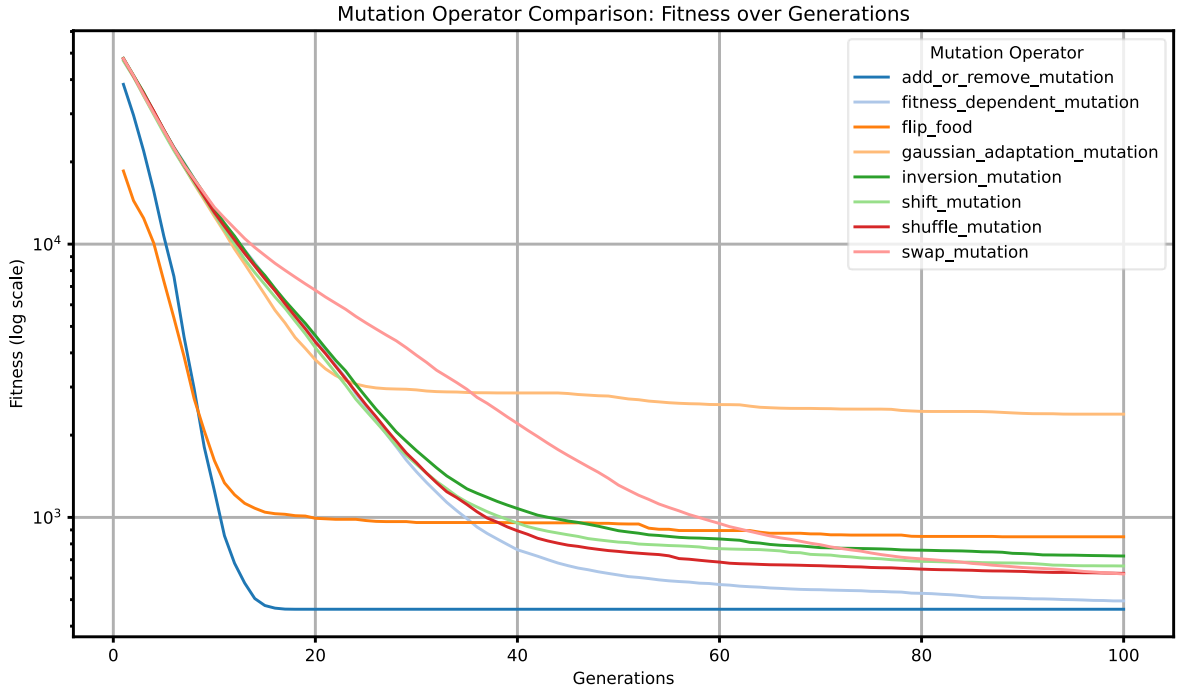
Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
fps	rank_sel	-87.556	0.0	-122.3681	-52.744	True
fps	tournament_sel	-128.3267	0.0	-163.1388	-93.5147	True
rank_sel	tournament_sel	-40.7707	0.0175	-75.5828	-5.9587	True

Solution average values of 30 runs

selection	fitness	food variety	nutrient_excess	nutrient_shortage
tournament_sel	410.69	11.37	85743.05	0.00
rank_sel	426.17	13.23	63939.64	0.00
fps	438.97	13.87	52726.27	0.00

Annex 8 – Mutation Operator Analysis



Anova Results:

	df	sum_sq	mean_sq	F	PR(>F)
mutation	7.00	409745778.24	58535111.18	533.61	0.00
Residual	232.00	25449552.04	109696.34	NaN	NaN

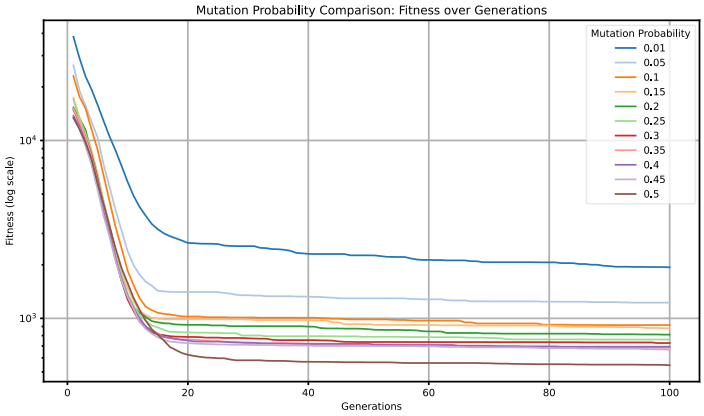
Multiple Comparison of Means - Tukey HSD, FWER=0.05

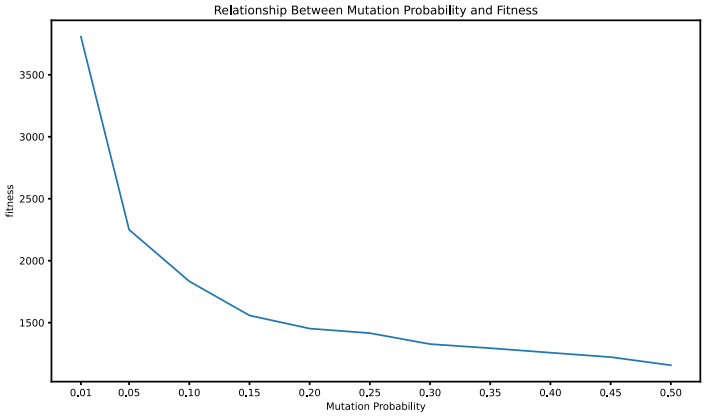
group1	group2	meandiff	p-adj	lower	upper	reject
add_or_remove_mutation	fitness_dependent_mutation	2377.6025	0.0	2116.0231	2639.1818	True
add_or_remove_mutation	flip_food	-151.6598	0.6387	-413.2391	109.9195	False
add_or_remove_mutation	gaussian_adaptation_mutation	3672.0968	0.0	3410.5174	3933.6761	True
add_or_remove_mutation	inversion_mutation	2594.4107	0.0	2332.8313	2855.99	True
add_or_remove_mutation	shift_mutation	2386.5052	0.0	2124.9259	2648.0845	True
add_or_remove_mutation	shuffle_mutation	2440.8311	0.0	2179.2518	2702.4105	True
add_or_remove_mutation	swap_mutation	3194.1682	0.0	2932.5889	3455.7476	True
fitness_dependent_mutation	flip_food	-2529.2623	0.0	-2790.8416	-2267.6829	True
fitness_dependent_mutation	gaussian_adaptation_mutation	1294.4943	0.0	1032.915	1556.0736	True
fitness_dependent_mutation	inversion_mutation	216.8082	0.186	-44.7711	478.3875	False
fitness_dependent_mutation	shift_mutation	8.9027	1.0	-252.6766	270.4821	False
fitness_dependent_mutation	shuffle_mutation	63.2287	0.9957	-198.3507	324.808	False
fitness_dependent_mutation	swap_mutation	816.5658	0.0	554.9864	1078.1451	True
flip_food	gaussian_adaptation_mutation	3823.7566	0.0	3562.1772	4085.3359	True
flip_food	inversion_mutation	2746.0705	0.0	2484.4911	3007.6498	True
flip_food	shift_mutation	2538.165	0.0	2276.5857	2799.7443	True
flip_food	shuffle_mutation	2592.4909	0.0	2330.9116	2854.0703	True
flip_food	swap_mutation	3345.828	0.0	3084.2487	3607.4074	True
gaussian_adaptation_mutation	inversion_mutation	-1077.6861	0.0	-1339.2654	-816.1068	True
gaussian_adaptation_mutation	shift_mutation	-1285.5916	0.0	-1547.1709	-1024.0122	True
gaussian_adaptation_mutation	shuffle_mutation	-1231.2656	0.0	-1492.845	-969.6863	True
gaussian_adaptation_mutation	swap_mutation	-477.9285	0.0	-739.5079	-216.3492	True
inversion_mutation	shift_mutation	-207.9055	0.2314	-469.4848	53.6739	False
inversion_mutation	shuffle_mutation	-153.5795	0.6235	-415.1589	107.9998	False
inversion_mutation	swap_mutation	599.7576	0.0	338.1782	861.3369	True
shift_mutation	shuffle_mutation	54.3259	0.9983	-207.2534	315.9053	False
shift_mutation	swap_mutation	807.663	0.0	546.0837	1069.2424	True
shuffle_mutation	swap_mutation	753.3371	0.0	491.7578	1014.9164	True

Solution average values of 30 runs

mutation	fitness	food variety	nutrient_excess	nutrient_shortage
add_or_remove_mutation	461.22	3.77	114453.50	0.00
fitness_dependent_mutation	494.80	18.57	59835.64	0.00
swap_mutation	620.60	22.07	82557.45	-0.00
shuffle_mutation	623.99	20.23	47640.40	-0.00
shift_mutation	664.07	21.70	43900.37	0.00
inversion_mutation	722.29	18.20	50571.97	-0.00
flip_food	849.39	8.07	147832.30	-0.00
gaussian_adaptation_mutation	2387.07	26.20	237622.93	0.00

Annex 5 – Mutation probability Analysis





Anova Results:

	df	sum_sq	mean_sq	F	PR(>F)
mut_prob	1.00	103641173.27	103641173.27	274.24	0.00
Residual	328.00	123959548.50	377925.45	NaN	NaN

Multiple Comparison of Means - Tukey HSD, FWER=0.05

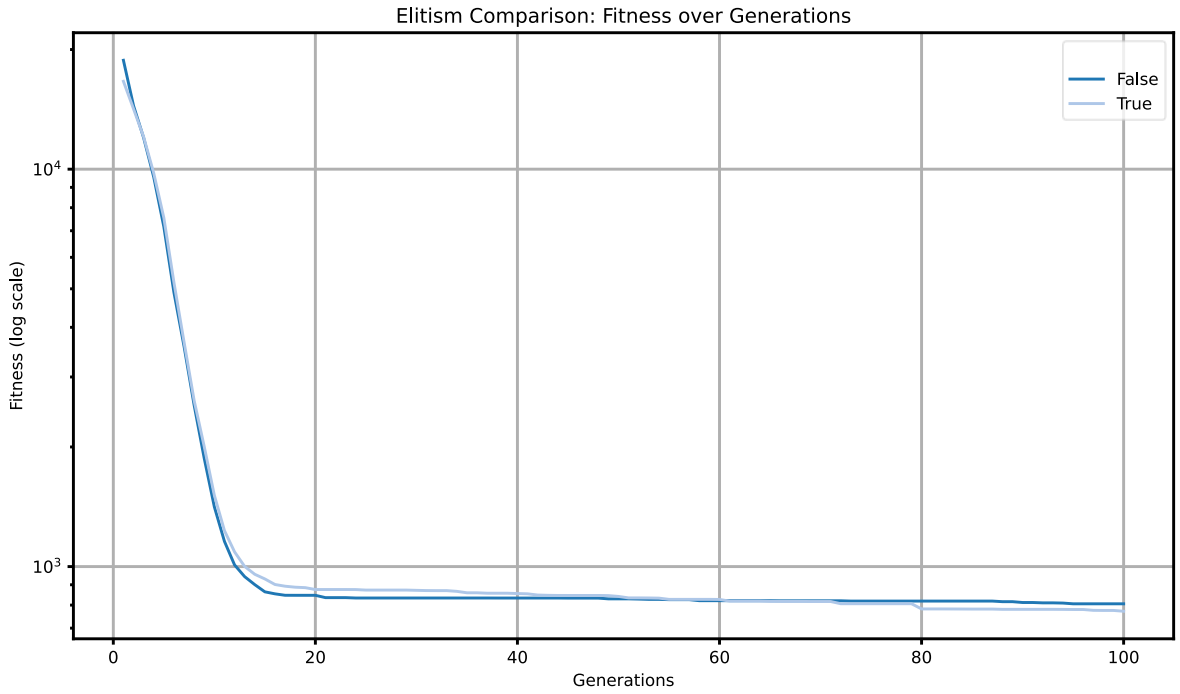
group1	group2	meandiff	p-adj	lower	upper	reject
0.01	0.05	-1556.3847	0.0	-1885.6226	-1227.1468	True
0.01	0.1	-1972.8704	0.0	-2302.1083	-1643.6325	True
0.01	0.15	-2248.4242	0.0	-2577.6621	-1919.1863	True
0.01	0.2	-2354.1069	0.0	-2683.3448	-2024.869	True
0.01	0.25	-2391.0663	0.0	-2720.3042	-2061.8284	True
0.01	0.3	-2479.4654	0.0	-2808.7033	-2150.2275	True
0.01	0.35	-2512.3751	0.0	-2841.613	-2183.1372	True
0.01	0.4	-2548.8328	0.0	-2878.0707	-2219.5949	True
0.01	0.45	-2584.4985	0.0	-2913.7364	-2255.2606	True
0.01	0.5	-2649.8512	0.0	-2979.0891	-2320.6133	True
0.05	0.1	-416.4857	0.0025	-745.7236	-87.2478	True
0.05	0.15	-692.0395	0.0	-1021.2774	-362.8016	True
0.05	0.2	-797.7222	0.0	-1126.9601	-468.4843	True
0.05	0.25	-834.6816	0.0	-1163.9195	-505.4437	True
0.05	0.3	-923.0807	0.0	-1252.3186	-593.8428	True
0.05	0.35	-955.9905	0.0	-1285.2284	-626.7526	True
0.05	0.4	-992.4481	0.0	-1321.686	-663.2102	True
0.05	0.45	-1028.1138	0.0	-1357.3517	-698.8759	True
0.05	0.5	-1093.4665	0.0	-1422.7044	-764.2286	True
0.1	0.15	-275.5538	0.1985	-604.7917	53.6841	False
0.1	0.2	-381.2365	0.0093	-710.4744	-51.9986	True
0.1	0.25	-418.1959	0.0024	-747.4338	-88.958	True
0.1	0.3	-506.595	0.0001	-835.8329	-177.3571	True
0.1	0.35	-539.5048	0.0	-868.7427	-210.2669	True
0.1	0.4	-575.9624	0.0	-905.2003	-246.7245	True
0.1	0.45	-611.6281	0.0	-940.866	-282.3902	True
0.1	0.5	-676.9808	0.0	-1006.2187	-347.7429	True

```
0.15    0.2  -105.6827  0.994  -434.9206   223.5552  False
0.15   0.25  -142.6421 0.9466    -471.88   186.5958  False
0.15    0.3  -231.0412 0.4536  -560.2791    98.1967  False
0.15   0.35  -263.9509 0.2535  -593.1888     65.287  False
0.15    0.4  -300.4086 0.1101  -629.6465    28.8293  False
0.15   0.45  -336.0743 0.0408  -665.3122    -6.8364   True
0.15    0.5   -401.427 0.0045  -730.6649   -72.1891   True
 0.2   0.25   -36.9594    1.0  -366.1973   292.2785  False
 0.2    0.3  -125.3585 0.9781  -454.5964   203.8794  False
 0.2   0.35  -158.2682 0.8982  -487.5061   170.9697  False
 0.2    0.4  -194.7259 0.7054  -523.9638    134.512  False
 0.2   0.45  -230.3916  0.458  -559.6295    98.8463  False
 0.2    0.5  -295.7443 0.1237  -624.9822    33.4936  False
0.25    0.3   -88.3991 0.9986   -417.637   240.8388  False
0.25   0.35  -121.3089 0.9828  -450.5468    207.929  False
0.25    0.4  -157.7665 0.9001  -487.0044   171.4714  False
0.25   0.45  -193.4322 0.7138  -522.6701   135.8057  False
0.25    0.5  -258.7849 0.2809  -588.0228     70.453  False
 0.3   0.35   -32.9098    1.0  -362.1477   296.3281  False
 0.3    0.4   -69.3674 0.9998  -398.6053   259.8705  False
 0.3   0.45  -105.0331 0.9943   -434.271   224.2048  False
 0.3    0.5  -170.3858 0.8458  -499.6237   158.8521  False
0.35    0.4   -36.4577    1.0  -365.6956   292.7802  False
0.35   0.45   -72.1233 0.9998  -401.3612   257.1146  False
0.35    0.5  -137.4761 0.9582   -466.714   191.7618  False
 0.4   0.45   -35.6657    1.0  -364.9036   293.5722  False
 0.4    0.5  -101.0184 0.9958  -430.2563   228.2195  False
0.45    0.5   -65.3527 0.9999  -394.5906   263.8852  False
------------------------------------------------------------
```

Solution average values of 30 runs

mut_prob	fitness	food variety	nutrient_excess	nutrient_shortage
0.50	545.29	7.13	196979.45	0.00
0.45	665.72	6.40	177874.37	0.00
0.35	682.79	5.50	105679.23	0.00
0.40	692.87	6.27	131920.06	0.00
0.30	726.56	8.27	174696.44	-0.00
0.25	759.58	5.80	115512.73	0.00
0.20	809.97	6.30	97386.09	0.00
0.15	880.52	8.73	171594.03	0.00
0.10	916.69	8.17	144896.17	0.00
0.05	1225.78	9.83	100145.23	0.00
0.01	1936.58	14.73	167804.17	0.00

Annex 9 – Elitism Analysis



t-Test Results

t-statistic: 0.07039376807142313
p-value: 0.9438825980683059

Solution average values of 30 runs

	fitness	food variety	nutrient_excess	nutrient_shortage
elitism				
True	772.02	8.03	152911.51	0.00
False	806.01	6.33	108214.57	0.00