Comparison of the Quality of Solutions Generated by Baseline and Novel Functions

Introduction

The purpose of this project is to produce a function that can predict the demand of a logistics network based on 13 demand indicators. These indicators are multiplied with 13 parameters and have a bias added to them to calculate the final demand value. The algorithms examined are made to find the best parameters in the range of [-100, 100] for which the difference between the demand estimate and its actual value will be the lowest.
We will examine the novel function by comparing it with a baseline function. For the purposes of this solution, we assume that the reader possesses foundational knowledge on evolutionary algorithms and their main components. The proposed solution differs from the baseline on the mutation operator, where the decision was made to use a gaussian distribution to draw an amount which is added to the gene value. The idea is to introduce a small amount of change to the gene (Eiben et al., 2015, p. 57).

The Proposed CI Solution

The baseline solution is an existing one produced as a solution for the laboratory sessions. It uses tournament selection for parent selection and then applies recombination on two candidates to produce two children. The children are mutated using swap mutation and the new population consists only of the children generated in the previous one.
For my proposed CI Solution, I have applied a Gaussian distribution as a mutation operator (Fig.1), as there is empirical evidence presented (Fogel et al., 1990) that Gaussian mutation is a very effective search operator, more so than crossover or crossover and mutation combined. Due to the nature of this operator, we can be sure that most of the time, the change introduced will be small, as the probability that a small number will be drawn is greater than the probability of drawing a larger number. This is achieved by adding to a random allele of the gene a value drawn randomly from a normal distribution. This distribution, visualised in Fig. 2, uses the existing allele value itself as a mean (μ) and a dynamic standard deviation (σ) of

$$\sigma = (t_{max} + 0.1) - (t_{current} / 10),$$

also called step size. The above calculations make sure that σ always remains greater than 0. Most of the values will be drawn between –σ and +σ, as can be observed on the diagram. Due to the dynamic nature of the standard deviation used here, the amount of change introduced will decrease as the generations progress, as the σ will be getting a smaller value with each generation.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Fig. 1 General form of the normal distribution's density function

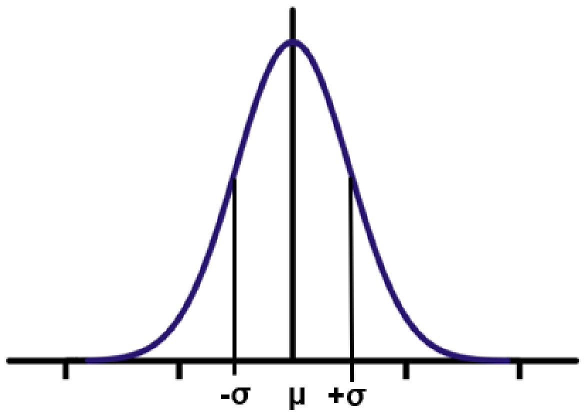
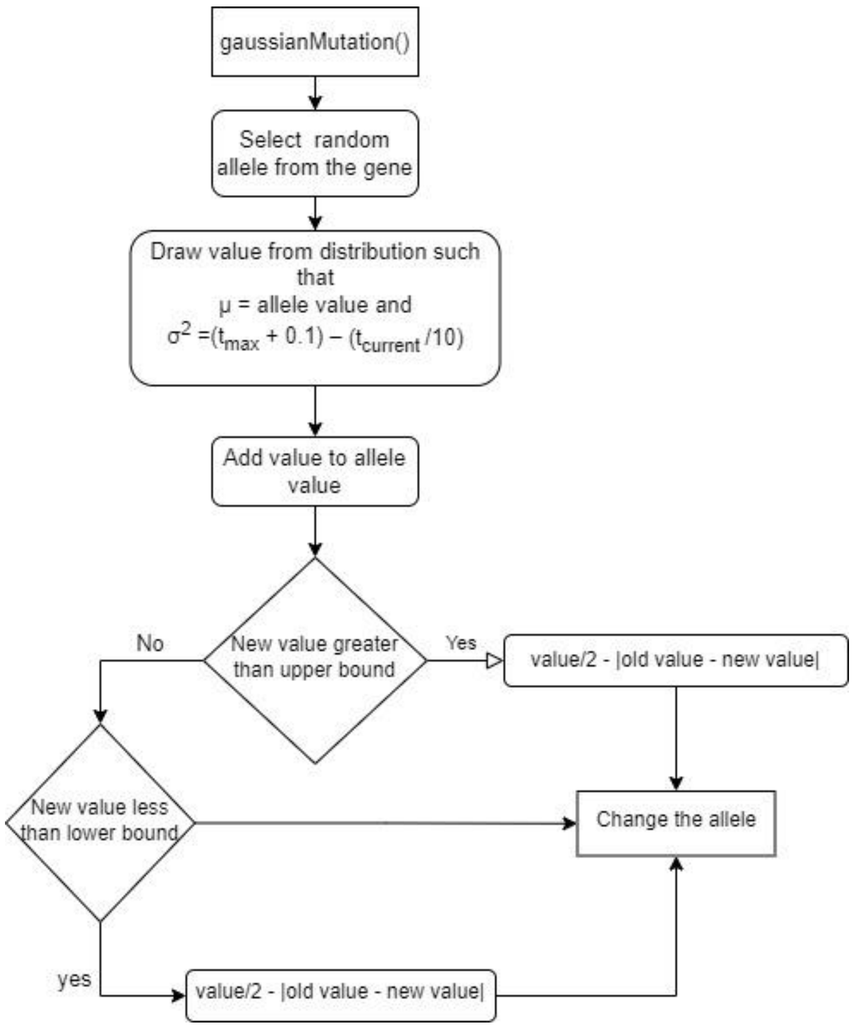


Fig. 2 Visualisation of the normal distribution curve.

For the purpose of doing the necessary calculations in the context of the algorithm, the Apache Commons Math API has been utilised. This provides us with the NormalDistribution class, which enables us to customise the normal distribution by setting our own mean and standard deviation (Apache, 2022). An alternative, java's Random, is only is only able to generate values from a distribution with μ=0 and σ=1, which would not be useful to us (Oracle, 2020).



gaussianMutation()
Select random allele from the gene
Draw value from distribution such that
μ = allele value and
$\sigma^2 = (t_{max} + 0.1) - (t_{current}/10)$
Add value to allele value
New value greater than upper bound
Yes
value/2 - |old value - new value|
No
New value less than lower bound
Change the allele
yes
value/2 - |old value - new value|

Fig. 3 Gaussian Mutation used in the novel solution.
Where $\mu \in \mathbb{R}$, $\sigma^2 \in \mathbb{R} > 0$.

Experimental methodology

Several experiments were conducted to explore the difference in efficiency and quality of solutions between the two algorithms. The algorithms were optimized using the “train” instance of the Demand prediction problem and tested using the “test” instance of the problem.
With this experiment I aim to test the alternative hypothesis: After 80 function evaluations, the novel method (Algorithm B), finds a candidate solution with better fitness on the DemandPrediction instance “test” than the baseline method (Algorithm A) does.
Its null hypothesis would be: After 80 function evaluations, the novel method (Algorithm B), does not find a candidate solution with better fitness on the DemandPrediction instance “test” than the baseline method (Algorithm A) does.

The two algorithms were run a combined total of 80 times and the solution with the best fitness after 100 generations was recorded on a table. The data was recorded in an Excel spreadsheet.

Analysis of experimental results

I used Welch’s t-test approach to compare the performance of the two algorithms based on solution quality. In figures 4 and 5 we can observe the data drawn visualised as histograms.

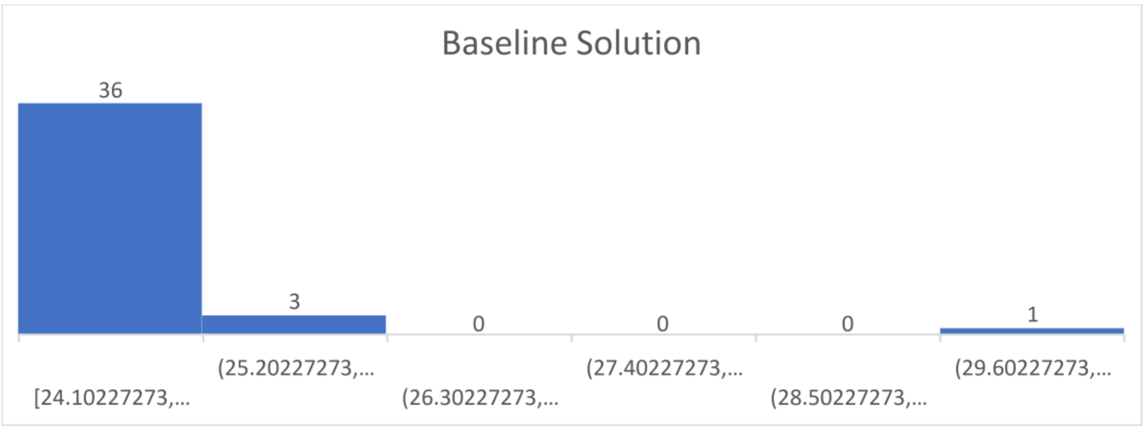


Fig.4 Histogram of Baseline solution results

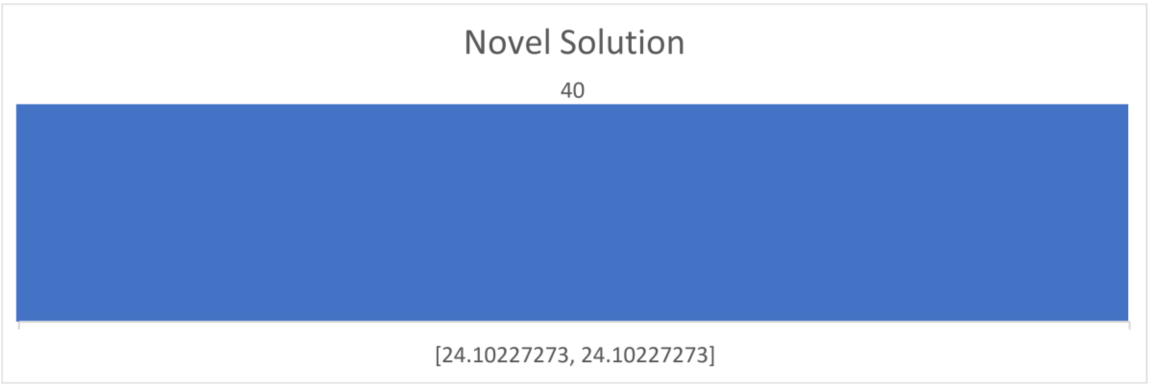


Fig.5 Histogram of Novel Solution results

Something that can be observed from the raw data, is that the novel solution always found the same value of 24.102272727272727, while the baseline arrived to a larger minimum four times.
By averaging out the result for each algorithm, we get 24.58332868 for algorithm A and 24.10227273 for algorithm B. By running the t-test function for the data, we find that the p-value is 0.003740737. Comparing that with the significance level α = 0.05, we can see that p-value < α, therefore we can reject the null hypothesis and accept the alternative.

References

Eiben, A.E. and Smith, J.E. (2015) *Introduction to Evolutionary Computing*, *SpringerLink*. Springer Berlin Heidelberg. Available at: https://link.springer.com/book/10.1007/978-3-662-44874-8 (Accessed: *December 12, 2022).*

Fogel, D.B. and Atmar, J.W. (1990) "Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems," Biological Cybernetics, 63(2), pp. 111–114. Available at: https://doi.org/10.1007/bf00203032.

Apache Commons (2022) Apache Commons Math 3.6.1 API. Available at: https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/index.html (Accessed: December 13, 2022).

Oracle (2020) Java platform, Standard Edition Java API Reference, Random (Java SE 15 & JDK 15). Available at: https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/util/Random.html (Accessed: December 13, 2022).