**UNIVERSITY OF EXETER**

**MSC DATA SCIENCE WITH ARTIFICIAL INTELLIGENCE**

**COURSEWORK ON NATURE INSPIRED COMPUTING (ECMM 409)**

**NOVEMBER 2022**

**PROJECT REPORT**

**DESCRIPTION OF EXPERIMENTS**

The experiments were carried out to see the effect the parameters have on fitness. The hyperparameters being varied are population size (p), tournament size (t), and mutation rate (m). I carried out tests on the algorithm keeping two (2) of these variables constant while varying the third variable. This approach was chosen to see the effect the third variable has on the fitness of the solution. Other conditions under which the experiments/algorithms were carried/designed out include single-point crossover, penalization of solutions with fitness greater than the allowable weight of 285kg.

In the first instance, population size (p) and mutation rate (m) were kept constant while varying the tournament size (t). The results are recorded in Table 1.1.

**QUESTIONS**

**Question 1: Which combination of parameters produce the best result?**

Table 1.1 below shows the results I obtained from several experiments carried out with different parameter combinations using the algorithm developed. The combination of parameters with the best results is stated below. Fig 1 shows two graphical illustrations (a and b) of the results obtained using varying tournament sizes while keeping population size constant. The best results were obtained (as shown in fig 1b) when the parameters below were set:

* Population size: 100
* Tournament size: 10
* Mutation rate: 2

**Table 1.1: Table of experimental results**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variables | | | Results | | |
| Population size (*p*) | **Tournament size *(t)*** | **Mutation rate *(m)*** | **Weight** | **Fitness / Value** | Comment |
| 100 | 2 | 1 | 283.8 | 4418.0 |  |
| 100 | 4 | 1 | 284.5 | 4433.0 |  |
| 100 | 8 | 1 | 284.8 | 4439.0 |  |
| 100 | 10 | 1 | 283.8 | 4432.0 |  |
| 100 | 2 | 0 | 281.6 | 3851.0 | Effect of no mutation |
| 100 | 4 | 0 | 280.0 | 3645.0 |
| 200 | 4 | 0 | 284.0 | 3892.0 |
| 100 | 2 | 2 | 283.7 | 4381.0 |  |
| 100 | 4 | 2 | 283.0 | 4424.0 |  |
| 100 | 8 | 2 | 285.0 | 4434.0 |  |
| 100 | 10 | 2 | 284.7 | 4445.0 | Best results |
| 100 | 2 | 4 | 284.8 | 4316.0 |  |
| 100 | 2 | 8 | 281.5 | 4168.0 |  |
| 100 | 4 | 4 | 283.6 | 4360.0 |  |
| 100 | 4 | 8 | 282.6 | 4219.0 |  |
| 150 | 2 | 1 | 289.6 | 4411.0 |  |
| 200 | 2 | 1 | 284.4 | 4312.0 |  |
| 250 | 2 | 1 | 284.5 | 4352.0 |  |
| 300 | 2 | 1 | 284.7 | 4377.0 |  |
| 350 | 2 | 1 | 284.7 | 4365.0 |  |
| 200 | 4 | 1 | 284.7 | 4442.0 |  |
| 200 | 4 | 2 | 282.0 | 4403.0 |  |
| 200 | 4 | 4 | 283.8 | 4363.0 |  |
| 500 | 2 | 1 | 285.4 | 4331.0 |  |
| 500 | 4 | 4 | 298.8 | 4276.0 |  |

\* *Seed used - 2345*

Graphical user interface, application

Description automatically generated**Figure 1**

**Question 2: Why do you think this is the case?**

The reason for this could be based on the selection pressure and the small variation in mutation. A tournament size of 10 from a population of 100 gives rise to slightly low selection pressure, and hence individuals with suboptimal fitnesses can be selected. This behaviour has the potential to drive better results. However, higher tournament sizes (20 and above) will result in selection pressures that will become quite high and therefore, will not yield good results.

For mutation, Michalewicz [1] clearly explains that the effect a mutation operator has on candidate solutions is the introduction of some form of variability to the child solutions, and so enables the algorithm to explore the search space of candidate solutions to arrive at near-optimal solutions. However, the mutation rate is high, the algorithm becomes a random search.

**Question 3: What was the effect when you removed mutation?**

In my experiments, when the mutation rate was set to zero (0), that is, no mutation, I observed that the fitness converged very quickly, after just a few hundred fitness evaluations, at ‘fitnesses’ between 3700 to about 3900, having weights between 281.6 and 284 (Table 1.0). Also, see Fig 2. These fitnesses are not the optimal solutions and hence, this shows the importance of mutation. This is because a mutation in a child solution introduces new variations which can lead to solutions with better fitnesses. In addition, the fitness and weights of the resulting solutions get worse as the tournament size increases.

Chart, line chart

Description automatically generated

**Question 4: If you were to extend your EA to work with a multi-objective version of this problem, which functions in your program would you change and why?**

On a multi-objective version of this problem, the functions that will need to be altered are the replacement and fitness functions. To explain further, one of the most used multi-objective genetic algorithms is the NSGA-II, and it involves the use of Pareto ranking and crowding measures [2]. The replacement function of a multi-objective genetic algorithm employs a non-dominating sorting procedure, where a rank is assigned to each solution [3]. Cvörnjek et al [3] explain that for any two-child solutions, solution 1 has a better rank than solution 2 if solution 1 is better than solution 2 in at least one criterion and solution 1 is not worse than solution 2 in any criteria. Thus, solution 1 dominates solution 2.

**Observation**

During my experiments, I observed that the solutions with the best fitnesses were those having a population size, p of 100. The fitness decreased with increased values of p and at p=500, the results quickly converged.

Chart, line chart

Description automatically generated

**Conclusion**

In summary, the experiments conducted has shown the advantage of choosing the right combination of tournament size, population, and mutation rate in genetic algorithms (GAs). Setting these hyperparameters very high lead to the algorithm reaching suboptimal solutions and converging quicker. This exercise also demonstrated the effect of the removal of mutation in GAs, which was shown to also produce undesirable solutions.

**References**

[1] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Heidelberg, Berlin: Springer Berlin. [Online]. Available from: <https://ebookcentral.proquest.com/lib/exeter/reader.action?docID=6532646>

[2] H. Ishibuchi, Y. Nojima and Tsutomu Doi, “Comparison between Single-Objective and Multi-Objective Genetic Algorithms: Performance Comparison and Performance Measures”*,* 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 1143-1150, doi: 10.1109/CEC.2006.1688438.

[3] Cvörnjek, Nejc and Brezočnik, Miran and Jagrič, Timotej and Papa, Gregor. “Comparison Between Single and Multi Objective Genetic Algorithm*”.* 2014