Group 24 Assignment Report

An Evolutionary Algorithm for an elevator to deliver as many people as possible to their desired floor

Oskar Sundberg Linus Savinainen
Samuel Wallander Leyonberg Gustav Pråmell
Joel Scarinius Stävmo

September 27, 2024

Contents

1 Introduction

The recommended software for typesetting assignment reports is LaTeX. It will allow you to prepare high-quality documents, especially in the area of Computer Science. This document can serve as a template for reports. Each section begins with brief instructions in red text. All the instructions in red, as well as the dummy text, should be removed in the final version to submit. The LaTeX source of this file includes examples of using the most needed commands and environments. You can find plenty of other examples with explanations in many web forums and discussion groups on the Internet. The easiest way to edit your report is to use `https://www.overleaf.com/`. Overleaf does not require any setup on your computer, and it is free to create an account.

The book *Writing for Computer Science* [1] is a useful assistance on how to write properly and present your work when it comes to Computer Science topics. It is a strong recommendation to follow its guidelines and limit the usage of AI tools to generate text. Keep in mind that the examiner is an expert in Evolutionary Computation and therefore, any false information generated by an AI tool is easily notable. Such case may lead to failing the assignment.

The introduction should briefly introduce the assignment and its purpose.

This report addresses an investigation for optimizing an elevators route to improve the efficiency of picking up and delivering passengers to their desired floors in the least amount average waiting time. In large buildings with a lot of floors, efficient elevators are crucial for managing traffic and must serve every one in a reasonable time. Poorly designed elevator systems will lead to long waiting time, unnecessary stops and unsatisfied users. Elevator technology have made progress during the years, but many elevators still struggle with finding an efficient way to serve passengers.

The hypotheses that evolutionary algorithms will be able to find near-optimal or optimal routes for elevators aiming to maximize the number of passengers served while minimizing travel distance and/or travel time is the core focus in the report. Different strategies and hyperparameter are experimented with to enable demonstration of how evolutionary algorithms can be used in such a problem.

This paper will focus on a statically defined problem, i.e., there will not be more passengers coming to the elevator as time goes on, unlike a normal elevator.

2 Optimize elevator routing

The second section should present the problem you tackle using your evolutionary approach. Overall, this section should include:

- The mathematical formulation considered in your study. Some problems have a clear mathematical model (e.g., Travelling Salesman Problem), while others do not (e.g., n-Queens). Based on the problem you chose, search the literature and find a proper way to present the problem.
- One paragraph that briefly presents at least 3 published academic works where any evolutionary approach is used to solve the problem. It would be wise to cite here works that influenced your algorithm. This practice saves you time from looking for additional academic resources. You can find more information about reading and searching in the literature in [2].
- The motivation behind the evolutionary approach you decided to develop. A good practice would be to align the motivation with some literature gap found in the academic works you presented above. However, this is not mandatory. You can motivate your selection on the characteristics of the algorithm making it proper for the problem.

Note: Change the section's title to match the name of the problem you chose for your assignment.

2.1 Mathematical formulation

2.2 Similar published academic work

[3], "Each vertex of a graph initially may contain an object of a known type. A final state, specifying the type of object desired at each vertex, is also given. A single vehicle of unit capacity is available for shipping objects among the vertices. The swapping problem is to compute a shortest route such that a vehicle can accomplish the rearrangement of the objects while following this route."

Mapping the elevator problem to the swapping problem.

- Vertices are represented as floors in the elevator.

- The object is defined as a person.
- Initial state consists of N persons waiting at the floors.
- The final state is reached when all persons are at their destination floor.
- The elevator is a vehicle with a max capacity that moves the persons between floors, i.e. a single vehicle shipping objects among the vertices.
- The elevator problem is aiming to reduce the waiting time of the persons in the elevator and thereby reduce the distance traveled.

According to [3], "Even the simple swapping problem is NP-hard."

2.1 Why this evolutionary approach

3 Algorithm

The third section should present the evolutionary approach you developed. You can divide this section into subsection. In any case, you should mention the following details:

Evolutionary approach. Clearly describe the algorithm you developed. You should clearly explain the evolutionary operators you used and what modifications you did to match the problem. It is extremely important to present also a pseudocode of your algorithm. An example is given in 1, below. For more insight into presentation of algorithms, you can advise [4].

To typeset pseudocode in LaTeX you can use one of the following options:

- Choose ONE of the (`algpseudocode` OR `algcompatible` OR `algorithmic`) packages to typeset algorithm bodies, and the algorithm package for captioning the algorithm.
- The `algorithm2e` package.

You can find more information here: `https://www.overleaf.com/learn/latex/Algorithms`

Solution representation. Clearly describe the solution representation you used. You can use figures to improve the comprehensibility of this part.

Fitness function. It is also very important to mention the fitness function you used. In many cases, the objective function of the problem is not the same as

Algorithm 1 Example of an algorithm's pseudocode

Require: $n \geq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  $X \leftarrow x$
  $N \leftarrow n$
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow \frac{N}{2}$ ▷ This is a comment
    else if N is odd **then**
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

the fitness function used in an evolutionary algorithm. An example, following the principles of [5], is given below.

$$F = \sum_{i=1}^{d} x_i^2 \tag{1}$$

where x_i is the i-th gene (i.e., decision variable) in the solution and d corresponds to the number of decision variables in the problem.

Note: Change the section's title to match the name of the algorithm you developed for your assignment.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

4 Experimental part

This section describes the setup of experiments [6]:

- Provide the details of the hardware and software that you used.
- Describe the steps you carried out during your experiments.
- Detail the data you used for the evaluation of your algorithm.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

5 Results and Analysis

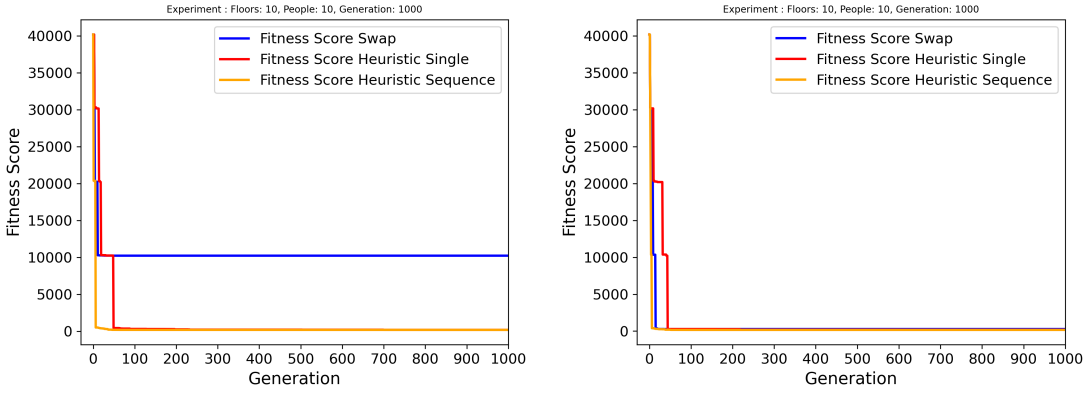
The following results are displayed using graphs to highlight some of the insights we have taken from the experiments we ran. Blue line and text represents our simplest crossover that swaps the last halves of two parents to create two children. The red represents the second crossover that selects genes from each parent based on their fitness scores. The orange represents the third crossover that selects contiguous segments of genes from each parent based on their fitness scores.

5.1 Building 1

When running the algorithm with this crossover we didn't see any improvements and the best score we achieved was also 182, see 1b. The first crossover reached the best value within fewer generations and seems to preform better in this instance. To be completely sure that we couldn't get improved results by changing to a different crossover without changing anything else we decided to try a much more advance crossover that had a great impact on.

The third crossover uses segment-based selection instead of selecting genes individually, this method selects contiguous segments of genes from each parent. The offsets determine how many genes to take from each parent based on their fitness scores. This one always performs well and never get stuck in local minima for long. The best score we achieved with this crossover was 172. We also tested the same building with more people inside, and we saw similar results with the third crossover always having good results and a great consistency. The simplest one also

performed similarly as the third one, but it wasn't as consistent. The second one is extremely unreliable it can perform very well, but it can also get stuck in local minima for a long time. Besides that it took more generations for all crossovers to reach a good solution because of the increased number of people. The third crossover often reaches a good solution quicker than the other two crossovers.



(a) Score/Arrived/Length: 10227/9/9, 182/10/11, 172/10/11.

(b) Score/Arrived/Length: 284/10/11, 187/10/14, 167/10/12.

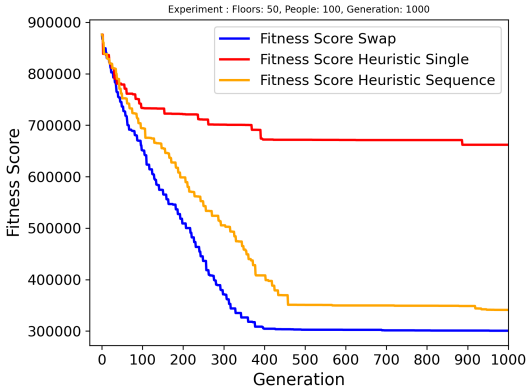
Figure 1: Building 1 comparing worst and best case.

5.2 Building 2

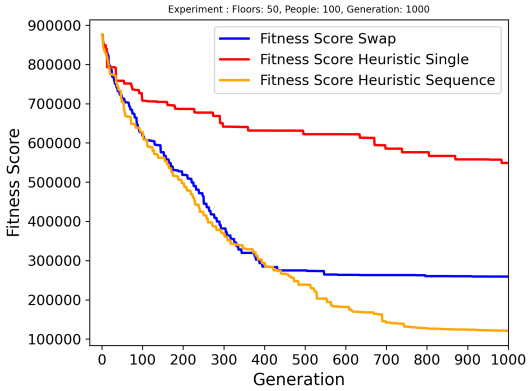
Samuel had some papers he wanted to compare these results with.

5.3 Building 3

After running all the buildings with mutation rate of 0.1 we started testing out different values and quickly found out that our algorithm found a solution that serves all or almost all people much faster with a greater mutation rate. We ended up display result from runs with a mutation rate of 0.6. 3 shows how much faster our algorithm reach acceptable solutions than before. When mutation rate was low the best solutions gave us a fitness scores around 500 000 but with a high rate we were able to get scores around 200 000. The best run served 66 people when mutation was 0.1, the best run with mutation 0.6 served all 100 people. Reason behind this is that to be able to serve as many people as possible the genes have to mutate to become longer or shorter. With a higher mutation rate this happens in much quicker pace and therefore the algorithm is able to leave local minimums faster. If we had more time to experiment with different mutation rates we could



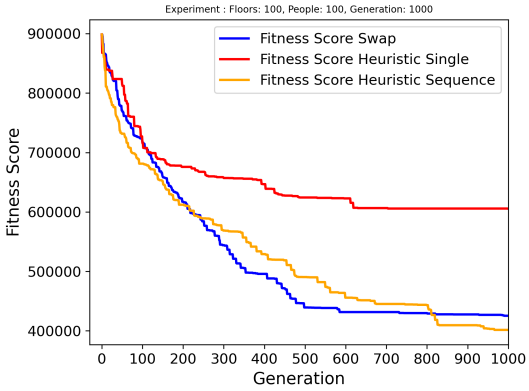
(a) Score/Arrived/Length:
300697/74/69, 661991/35/52, 341167/69/74.



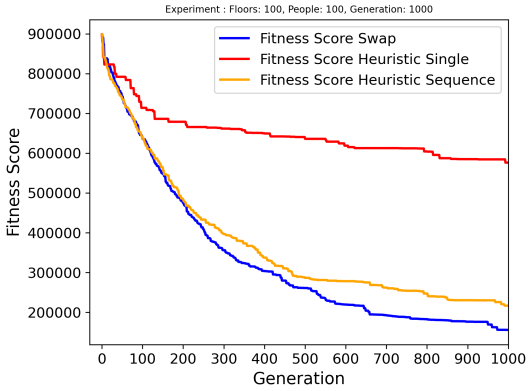
(b) Score/Arrived/Length:
259130/79/71, 548857/48/59, 121157/94/124.

Figure 2: Building 3 comparing worst and best case.

probably find an even better rate. In our situation mutation the genomes to become longer than the number of floors in the building is key to be able to serve all people. Therefore, a greater mutation rate is beneficial and less computational power is needed compared to increasing the population size or the number of generations. We tested increasing number of generations and received similar results as with a greater mutation rate.



(a) Score/Arrived/Length:
425352/67/80, 605508/44/102, 401562/66/75.



(b) Score/Arrived/Length:
155619/100/151, 576322/48/117, 216508/94/139.

Figure 3: Building 4 comparing different mutation rates.

Thoughts:

— The first crossover can perform really well and even beet the third one in some cases, but it also tends to get stuck in local minima in some cases. This means that it is inconsistent and that you can't trust it to always give you a good solution. It performs worse with fewer people in the buildings.

— Too much diversity makes the second crossover bad, maybe it would have been better with more elitism and less diversity from mutations and roulette wheel selection. This crossover also have problems performing well on larger buildings with more people.

— The third crossover is the best one, due to that it always performs well and never gets stuck in local minima for long. This one performs well on all buildings and with all amounts of people. It also performs well with different mutation rates.

6 Conclusions

In this section you should provide a concise summary of what has been done, the obtained results and some recommendations on how this study could be extended.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

References

[1] J. Zobel, *Writing for computer science*. Springer, 2014.

[2] J. Zobel, "Reading and reviewing," in *Writing for Computer Science*, pp. 19–33, Springer, 2014. https://doi.org/10.1007/978-1-4471-6639-9_3.

[3] S. Anily and R. Hassin, "The swapping problem," *Networks*, vol. 22, no. 4, pp. 419–433, 1992.

[4] J. Zobel, "Algorithms," in *Writing for Computer Science*, pp. 115–128, Springer, 2014. https://doi.org/10.1007/978-1-4471-6639-9_10.

[5] J. Zobel, "Mathematics," in *Writing for Computer Science*, pp. 131–143, Springer, 2014. https://doi.org/10.1007/978-1-4471-6639-9_9.

[6] J. Zobel, "Experimentation," in *Writing for Computer Science*, pp. 197—-215, Springer, 2014. https://doi.org/10.1007/978-1-4471-6639-9_14.