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 30, 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* [**?**] 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 [**?**].
- 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

[**?**], "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.

2.1 Why this evolutionary approach

According to [**?**], 'Even the simple swapping problem is NP-hard.' As shown in Section 2.2, the problem at hand can be considered a swapping problem. Due to the nature of NP-hard problems, brute-forcing a solution is not a viable option. The search space is large and only grows with the number of floors and people in the elevator, which also contributes to many local minima and maxima. The problem we are dealing with is to be considered a real-world scenario where finding the most optimal solution is not necessarily required; instead, the priority is to reach an acceptable solution within a reasonable amount of time. Therefore, the choice of a genetic algorithm was made. By using a population of genes, a good coverage of the search space is achieved. By using a mutation rate and smart selection algorithms, the algorithm can escape local minima and maxima. Ultimately, the most important aspect is reaching a solution in a reasonable amount of time and the solution does not need to be the optimal.

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 **??**, below. For more insight into presentation of algorithms, you can advise [**?**].

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`

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

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 the fitness function used in an evolutionary algorithm. An example, following the principles of [**?**], 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:

Each experiment consists of two JSON files: one representing the building which specifies where all people are located, and another containing the population with a set of initial genomes. These experiments are saved to files rather than being generated randomly each time to ensure the ability to run the same experiment multiple times for consistency.

One specific building which already had result data available was used to compare our algorithm's results against previously known results.

To thoroughly evaluate the algorithm we decided to test it on buildings of different sizes (small, medium, large) and with corresponding population sizes. Given the number of experiments required to be run we selected buildings with 10, 50, and 100 floors containing populations of 10, 50, and 100 people respectively. Additionally the specific building mentioned earlier which has 21 floors and 10 people was included.

Buildings were generated with constraints ensuring that each floor could hold between 0 and half of the maximum population. While it was theoretically possible for all people to be on just two floors in practice the distribution was more balanced. For example in the small 10-floor building with 10 people the distribution looked as follows:

From		To
0	→	6
0	→	8
3	→	5
4	→	7
5	→	7
6	→	2
7	→	0
8	→	4
9	→	0
9	→	2

After generating and validating all buildings and populations we proceeded to run the experiments. Each experiment was conducted on a computer with an AMD Ryzen 7 7800X3D and took roughly ten seconds to a minute. In totally 150 experiments were run.

Due to time constraints the only parameter that was adjusted during the experiments was the mutation rate to limit the number of results to analyze. To ensure the reliability of results each combination of building and population was tested five times allowing us to identify potential anomalies. Experiments were conducted with two mutation rates, 10% and 60%, to observe differences in algorithm performance. The thought process behind the very high mutation rate was to semi-simulate the effects of a longer generation limit without significantly increasing runtime.

The experimental procedure is outlined as follows:

- Generate the building and population, or load a pre-existing experiment.
- Run the experiment generating a CSV file with results and a PNG with a graph.
- Re-run the same experiment with different parameters.
- Analyze the results to evaluate the algorithm's performance.

5 Results and Analysis

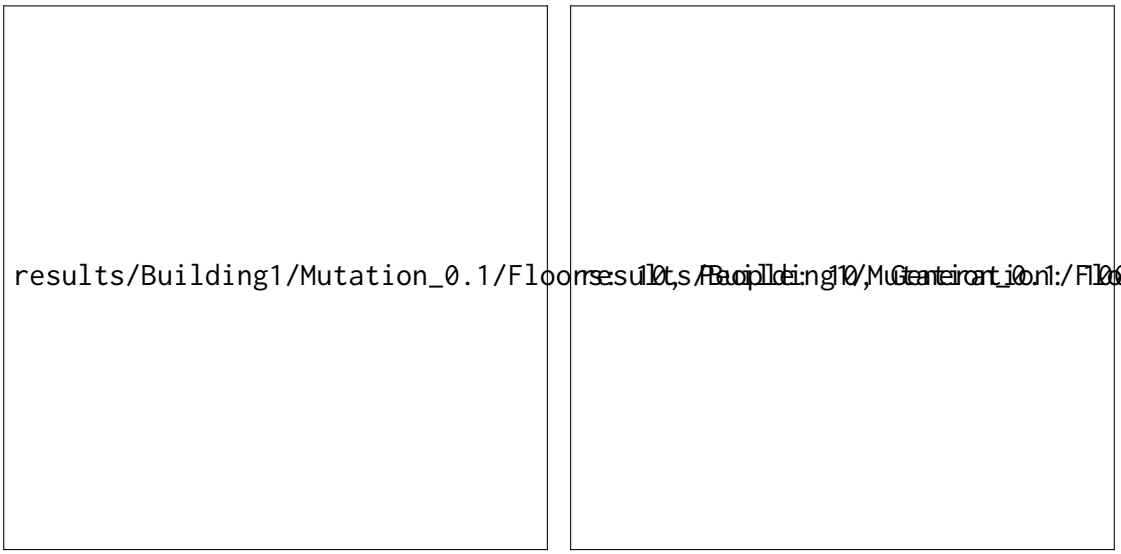
The following results are displayed using graphs to highlight some of the insights we have taken from the experiments we ran. Blue line and text represents our simplest crossover that swaps the last halves of two parents to create two children described in(3. X1). The red represents the second crossover that selects genes from each parent based on their fitness scores, described in (3. X2). The orange represents the

third crossover that selects contiguous segments of genes from each parent based on their fitness scores, described in (3. X3).

We started off with a fitness function that gave 10 points for each person served, here a higher fitness score was considered better. The reasoning behind this was to promote the algorithm to serve all people in the building. The problem with this was that it tended to give long genomes because it was more beneficial to serve all people than to serve them quickly, i.e. it made long trips to different floors when there was no one onboard and people were still waiting to get on the elevator. We then changed fitness function to one that gave huge penalties for all people that weren't served by the elevator when the route was finished. It also increased the score by one for each person not arriving to their destination for each floor traveled. This means that the lower score the better. It also made it possible for us to analyze the average waiting time and this was also a benchmark that most papers we found used for this kind of problem.

5.1 Building 1

Figure **??** shows results with 100 people in the building and the worst performance from 5 runs for each crossover on the smallest building with 10 people in it. Figure **??** displays the worst case in these runs where the blue crossover failed to serve one person and then received a huge penalty. We used figure **??** to highlight that with more people inside the orange crossover kept getting good results with great consistency. The blue one also performed as good as the orange one, but it wasn't as consistent. The red one was extremely unreliable it performed well sometimes, but it also got stuck in local minimums for long time periods. Besides that it took more generations for all crossovers to reach an acceptable solution because of the increased number of people. The orange and blue crossover often reached a good solution quicker than the red one, and it also tends to get worse the more people it has to serve.



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

(b) Score/Arrived/Length: 6727/100/45, 9113/100/61, 6880/100/52.

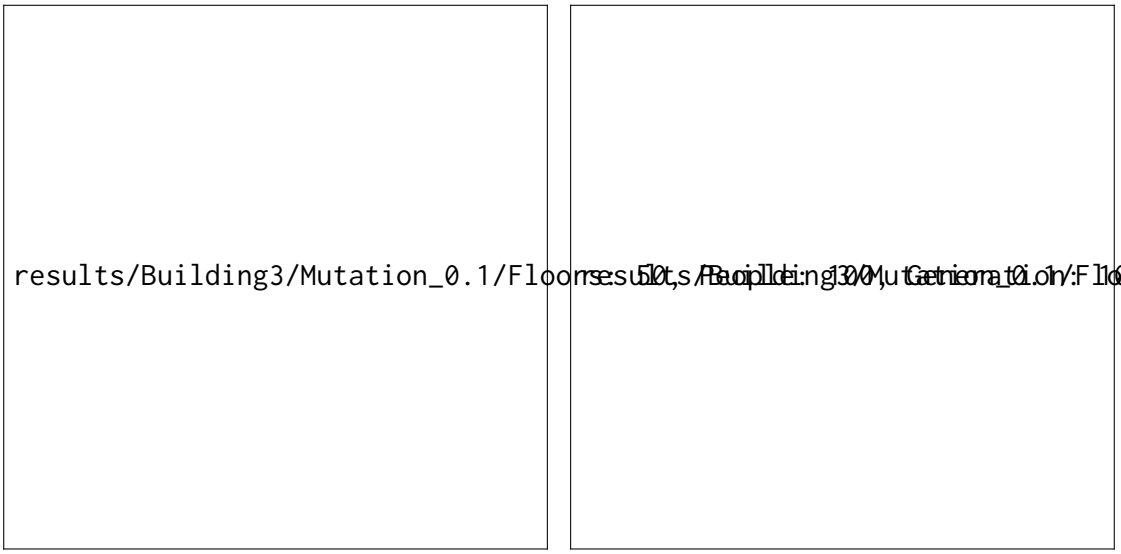
Figure 1: Building 1 comparing worst and best case.

5.2 Building 2

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

5.3 Building 3

As mentioned earlier the red crossover tends to perform worse the more people in the building instances and Figure ?? highlights this phenom again. In figure ?? the blue crossover actually outperform the orange one, that got stuck in a local minimum for a long time. The orange one also was slower than the blue one. In addition, figure ?? then shows a case where the blue one performed similar to how it performed in ?? and the orange one reached an extraordinary low score of 121157 without getting stuck in a local minimum. With other words this pinpoints that the orange crossover have a good worst case and great best case which makes it the most reliable crossover of the three.



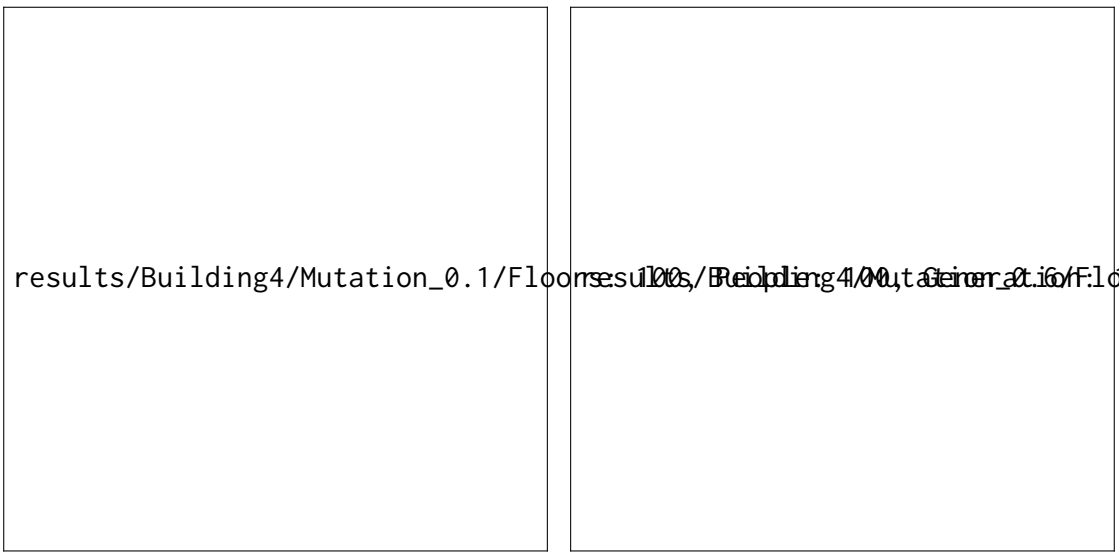
(a) Score/Arrived/Length: 300697/74/69, 661991/35/52, 341167/69/74.

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

Figure 2: Building 3 comparing worst and best case.

5.4 Building 4

After running all the buildings with mutation rate of 10% 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 display result from runs with a mutation rate of 60%. **??** shows how much faster our algorithm reach acceptable solutions than before. When mutation rate was low the best solutions gave us 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 10%, the best run with mutation 60% 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 than number of floors. With a higher mutation rate this happens in much quicker pace and therefore the algorithm is able to leave local minimums faster. 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. If we had more time to experiment with different mutation rates we could probably find an even better 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.

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.

The paper mainly looked at different crossover functions impact on our genetic algoritmes abilty to find solutions for various buildings configurations. For the statistics the conclusions drawn are:

Building	Crossover function
1	Vilken som var bäst
2	Vilken som var bäst
3	Vilken som var bäst

Because the algorithm is not forced to find a solution where all people reach their destination floor, our implementations include a time penalty discussed in (3.x), this to incentivize the algorithm to deliver all people to there destion floor. Further research on the impact of this time penalty in relation to the number of people in the building and their configurations and configuerations .

A finding from the paper is that a higher mutation rate seems to be better. To deturmen this further research in this area is needed. Is a higher mutation rate better? If so, what is the optimal range?

In addition, this report does not focus on the selection algorithm for the problem. Further research on which selection algorithm will perform best for the problem at hand would be of interest. 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.