Evolutionary Algorithms: Final report

Simon Perneel

December 17, 2020

1 Metadata

- **Group members during group phase:** Thomas Feys & Timo Martens
- **Time spent on group phase:** 12.5 hours
- **Time spent on final code:** 40 hours
- **Time spent on final report:**

2 Modifications since the group phase

2.1 Main improvements

The following paragraphs briefly describe the main improvements since the group phase. A more detailed explanation is given in section 3.

Initialization of the population: The population is initialized now with an amount of heuristic good individuals. It leads the search into more promising areas of the search space and speeds up the optimization process. See section 3.2 and 3.7.

More and better mutation operators: In addition to swap mutation, insert mutation, scramble mutation and reverse sequence mutation have been implemented in order to increase randomness in the search. It results in more diversity in the population. See section 3.4.

Dpx crossover and cycle crossover: In addition the order mutation, distance preserving crossover and cycle crossover have been implemented in order to generate more diverse individuals. See section 3.5

Crowding: The (λ+μ)-elimination has been modified with a crowding algorithm to promote diversity and avoid premature convergence. See section 3.8.

Elitism : Elitism is used to ensure that the best 3 individuals of the population survive elimination and remain unmutated. See section 3.12

Code optimizations: Some improvements have been made to the code in order to reduce iteration time and obtain more overview in the code. This is explained more in detail at section 3.12.

2.2 Issues resolved

Handling problems with disconnected cities: With the first version, the algorithm did not work with cities that are not connected with each other. The algorithm could not deal with the infinite costs in the distance matrix. This problem is solved simply but effectively by replacing the infinite values in the matrix by extremely high costs. In this way, tours with unconnected cities are not selected to generate offspring and will rapidly be eliminated from the population.

Early convergence: The diversity of the population in the first version of the algorithm decreased too quickly. This led to an early convergence of the mean fitness and best fitness. Because of this, the algorithm often gets stuck in a local optimum. This early convergence was mainly due to the (λ+μ)-elimination scheme and the order crossover operator. These operators causes a loss in diversity. In addition, there was no diversity promotion implemented yet.

This problem was solved by using crowding as a diversity promotion scheme. Whenever an individual is promoted to the next generation, the algorithm removes a similar individual in the seed population. This is further explained in section 3.8. Also, the distance preserving crossover (DPX) has been implemented. This crossover operator makes use of instance-specific knowledge, while at the same time, it preserves diversity within the population. See 3.5 for a more detailed explanation.

Not optimized for larger tours: During the group phase the code was too slow for the larger tours. The algorithm ran out of time, with the optimal and heuristic value still quite far away. This was due to the lack of diversity, but also because the iterations took to long.

This problem has been adressed by a number of techniques. By use of a heuristic to initialize the population, the algorithm could start the search at a more promising area in the search space. Therefore, the algorithm yielded better results. The use of the heuristic to initialize the population creates a new problem; this field in the search space has a lot of local optima. Therefore, the alpha value has been set quite high from the beginning.

Now, the found result is always better than the heuristic value. However, for the larger tours, the found result sometimes is still too far away from the optimal value after 5 minutes.

3 Final design of the evolutionary algorithm

Goal: Based on this section, we will evaluate insofar as you are able to design and implement an advanced, effective evolutionary algorithm for solving a model problem.

In this section, you should describe all components of your final evolutionary algorithm and how they fit together.

3.1 Representation

The candidate solutions are represented by a permutation of (non-recurring) integers. The permutation represents the order in which the cities are visited. The permutation appears as a numpy array of integers in our code: e.g. [7, 6, 1, 5, 4, 2, 3]. This representation permits the use of common crossover and mutation operators and a mutation operator. It is also chosen because it is more intuitive and easier to work with than the adjacency representation, which also is a possible representation for the TSP.

3.2 Initialization

The population is largely initialized with random individuals (i.e. random permutations) in order to start with sufficient diversity. A smaller part of the population is initialized with heuristic good individuals. The tours of these individuals are made with the nearest neighbour algorithm. A tour starts with a random individual, and repeatedly visits the nearest city untill all have been visited. This draws the search from the beginning towards more promising solutions. To prevent these heuristic individuals from immediately taking over the entire population and removing all diversity, the mutation probability has been set high from the beginning.

3.3 Selection operators

Which selection operators did you implement? If they are not from the slides, describe them. Can you motivate why you chose this one? Are there parameters that need to be chosen? Did you use an advanced scheme to vary these parameters throughout the iterations? Did you try other selection operators not included in the final version? Why did you discard them?

3.4 Mutation operators

Which mutation operators did you implement? If they are not from the slides, describe them. How do you choose among several mutation operators? Do you believe it will introduce sufficient randomness? Can that be controlled with parameters? Do you use self-adaptivity? Do you use any other advanced parameter control mechanisms (e.g., variable across iterations)? Did you try other mutation operators not included in the final version? Why did you discard them?

3.5 Recombination operators

Which recombination operators did you implement? If they are not from the slides, describe them. How do you choose among several recombination operators? Why did you choose these ones specifically? Explain how you believe that these operators can produce offspring that combine the best features from their parents. How does your operator behave if there is little overlap between the parents? Can your recombination be controlled with parameters; what behavior do they change? Do you use self-adaptivity? Do you use any other advanced parameter control mechanisms (e.g., variable across iterations)? Did you try other recombination operators not included in the final version? Why did you discard them? Did you consider recombination with arity strictly greater than 2?

3.6 Elimination operators

Which elimination operators did you implement? If they are not from the slides, describe them. Why did you select this one? Are there parameters that need to be chosen? Did you use an advanced scheme to vary these parameters throughout the iterations? Did you try other elimination operators not included in the final version? Why did you discard them?

3.7 Local search operators

What local search operators did you implement? Describe them. Did they cause a significant improvement in the performance of your algorithm? Why (not)? Did you consider other local search operators that did not make the cut? Why did you discard them? Are there parameters that need to be determined in your operator? Do you use an advanced scheme to determine them (e.g., adaptive or self-adaptive)?

3.8 Diversity promotion mechanisms

Did you implement a diversity promotion scheme? If yes, which one? If no, why not? Describe the mechanism you implemented. In what sense does the mechanism improve the performance of your evolutionary algorithm? Are there parameters that need to be determined? Did you use an advanced scheme to determine them?

3.9 Stopping criterion

Which stopping criterion did you implement? Did you combine several criteria?

3.10 The main loop

Describe the main loop of your evolutionary algorithm using a clear picture (preferred) or high-level pseudocode. In what order do you apply the various operators? Why that order? If you are using several selection, mutation, recombination, elimination, and local search operators, describe how you choose among the possibilities. Are you selecting/eliminating all individuals in parallel, or one by one? With or without replacement?

3.11 Parameter selection

For all of the parameters that are not automatically determined by adaptivity or self-adaptivity (as you have described above), describe how you determined them. Did you perform a hyperparameter search? How did you do this? How did you determine these parameters would be valid both for small and large problem instances?

3.12 Other considerations

Did you consider other items not listed above, such as elitism, multiobjective optimization strategies (e.g., island model, pareto front approximation), a parallel implementation, or other interesting computational optimizations (e.g. using advanced algorithms or data structures)? You can describe them here or add additional subsections as needed.

Instead of recalculating the cost of an individual every iteration, the cost is only recalculated if the tour of an individual has changed. This seemed a trivial thing, but actually saves a lot of time each iteration. The same thing has been done for the edge set of a tour.

Separate class for GA parameters: All parameters of the algorithm have been grouped in a separate class to obtain more overview and cleaner code.

4 Numerical experiments

4.1 Metadata

The parameters that were used for the evolutionary algorithm are listed below. Most of them are fixed. Only the alpha value changes over time, but stays within the range of 0.3-0.6.

- population size λ: 250
- % initialized with heuristic individuals: 15%
- amount of offspring μ: 150
- k-tournament k: 5
- mutation probability α: 0.3-0.6
- recombination probability p_{rc}: 0.99

Specifications of the computer system used for the experiments:

- CPU: Intel Core i7-8550, 4 cores @1.80 GHz (code is not adapted for multi-threading)

- 8 GB main memory
- Python version 3.8

4.2 tour29.csv

The best tour length found for this problem was 27154.49. This seems to be the optimal solution for this problem. The corresponding sequence of cities of this tour is:

[5 0 1 4 7 3 2 6 8 12 13 15 23 24 26 19 25 27 28 22 21 20 16 17 18 14 11 10 9]

A typical convergence graph is shown in Figure 1. For this benchmark problem, a simple stop criterion has been used in order to speed up the experiment. Every 50 iterations the algorithm checks if there is any improvement. If there is not, the optimization process is stopped. The average time to find the optimal solution is around five seconds.

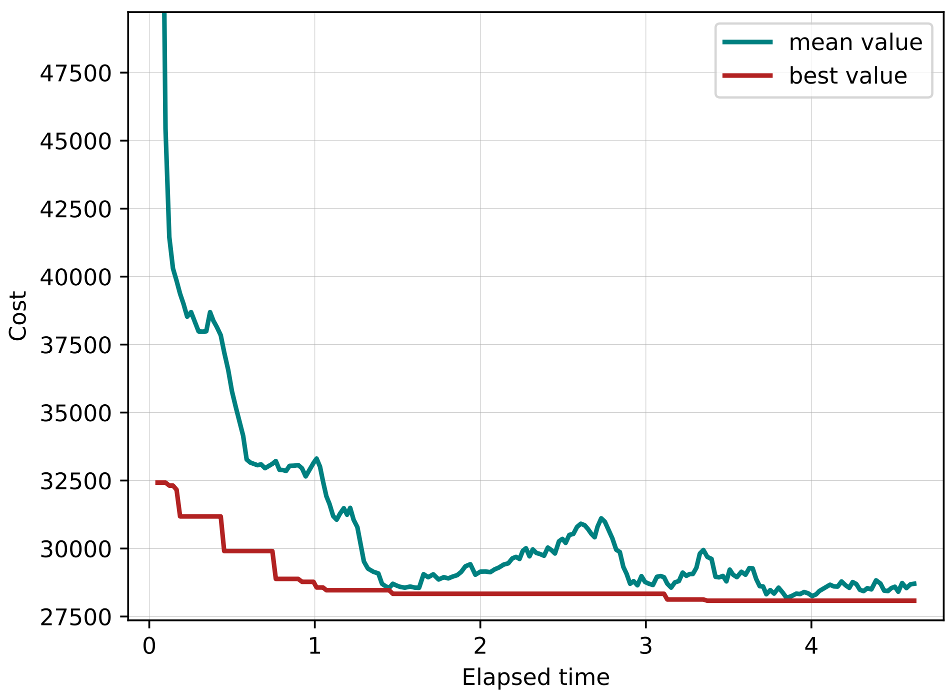
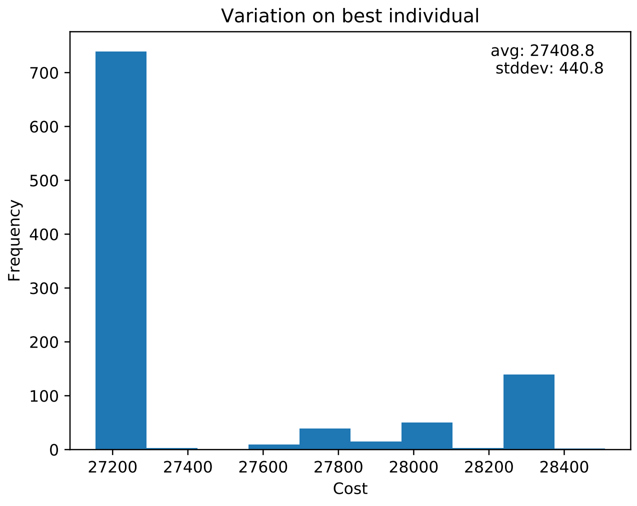
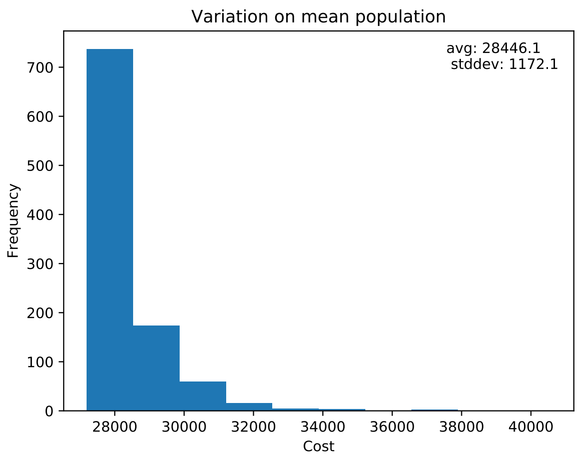


Figure 1: Typical convergence graph of tour29.csv

Figure 2 shows the histograms that are obtained when running the problem 1000 times with the GA:



(a) variation on the best individual



(b) variation on the mean individual

Figure 2: Histograms that shows the variation on the best individual and mean individual when the problem is run 1000 times

4.3 tour100.csv

Run your algorithm on this benchmark problem (with the 5 minute time limit from the Reporter). Include a typical convergence graph, by plotting the mean and best objective values in function of the time (for example based on the output of the Reporter class).

What is the best tour length you found in each case?

Interpret your results. How do you rate the performance of your algorithm (time, memory, speed of convergence, diversity of population, quality of the best solution, etc)? Is your solution close to the optimal one?

4.4 tour194.csv

Run your algorithm on this benchmark problem (with the 5 minute time limit from the Reporter). Include a typical convergence graph, by plotting the mean and best objective values in function of the time (for example based on the output of the Reporter class).

What is the best tour length you found?

Interpret your results. How do you rate the performance of your algorithm (time, memory, speed of convergence, diversity of population, quality of the best solution, etc)? Is your solution close to the optimal one?

4.5 tour929.csv

Run your algorithm on this benchmark problem (with the 5 minute time limit from the Reporter). Include a typical convergence graph, by plotting the mean and best objective values in function of the time (for example based on the output of the Reporter class).

What is the best tour length you found?

Interpret your results. How do you rate the performance of your algorithm (time, memory, speed of convergence, diversity of population, quality of the best solution, etc)? Is your solution close to the optimal one?

Did your algorithm converge before the time limit? How many iterations did you perform?

5 Critical reflection

Goal: Based on this section, we will evaluate your understanding and insight into the main strengths and weaknesses of your evolutionary algorithms.

The main strengths of a GA is in my opinion that it can be used in various representations and thus problems. It can come up relatively fast with good solutions and the basic concept is easy to understand. I think the main weakness of a GA is the tuning of the different parameters. There are a lot of parameters to be considered (k in k-tournament, α for mutation, λ for population size, μ for amount of offspring, ...). It is time-consuming and sometimes difficult to find out what the best values for these parameters are. One can try to implement self-adaptivity, where the different parameters adapt with every new generation. But there is no 'one size fits all' solution and self-adaptivity is still subject of many studies today.