Evolutionary Algorithms: Final report

Stijn Staring (r0620003)

December 8, 2020

1 Formal requirements

2 Metadata

- **Group members during group phase:** Rajat Sharma and Pieter-Jan Vrielynck

- **Time spent on group phase:** 13 hours
- **Time spent on final code:** 72 hours
- **Time spent on final report:** 12 hours

3 Modifications since the group phase

Goal: Based on this section, we will evaluate insofar as you are able to analyse common problems arising in the design and implementation of evolutionary algorithms and your ability to effectively solve them.

3.1 Main improvements

Diversity was instantaneous lost and the algorithm was therefore rapidly stuck and very early convergence was reached. –¿ much variation on solutions around a cost of 4000 for the best solution. random initialization of the population does not necessarily lead to diverse population –¿ have used NN with a random selection of cities to choose. Always different random start. Experimented in elimination step with diversity mechanisms. (overlap edges/ same routes)

To big exploitation pressure with lambda plus mu. Replaced by a diverse k tournament scheme. Element is removed from pop –¿ others more change to be selected. More randomness in the elimination and reduces the exploitation pressure and increases the exploration.

No measure taken to handle the ATSP in comparison with the normal TSP –¿ work with greedy operators that always cheapest. Take into account if inverse is very expensive. Implemented a 3 opt local search operator and not 2 opt. 2 opt is inversion, 3 opt is insertion of a subpath and is enhanced with two local searches.

scramble can destroy the solution a lot, so it is good to be used in the beginning iterations but in the ending iterations maybe you can use some better mutation. –¿ a very naive method that scrambels the route without knowledge. –¿ leads to variance solutions and unpredictability. Enhanced by greedy mutation.

The Sequential cross over operator was reasonable good. Changed to greedy cross over. Take overlap and instead of looking to best costs between two individuals, look at the best cost in the population.

Very high start cost, the algorithm had to bridge a big distance –¿ NN and 3 opt on initial solution.

The mean cost of the best solution after applying the genetic algorithm in the group phase was around 32500. Now the mean is already around 28300 after initialization of the population.

Calculation cost is still an issue in the bigger problems. Tried to reduce this issue by working with a smaller population size.

Best solution went up and down during the group phase. Solved by implementing elitism. The best solution is always retained over the generations.

Short description 1: State what modification you made (e.g., replaced top-λ selection with k-tournament selection). What aspect of your evolutionary algorithm did it improve?

⋮

3.2 Issues resolved

Recall the list of issues from the group phase. Describe how you solved these issues in the individual phase.

- Algorithm has a too high calculation load for the big TSP problems. Try

Short description 1: Describe the observation or problem from the group phase. Explain what caused this issue. How did you solve it (you can refer to the list of improvements)? Did fixing it significantly benefit your evolutionary algorithm? If you did not fix it: why not?

⋮

4 Final design of the evolutionary algorithm

Graph of the flow of the algorithm –¿ one heuristic and local search technique.

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.

4.1 Representation

How do you represent the candidate solutions? What is your motivation to choose this one? What other options did you consider? How did you implement this specifically in Python (e.g., a list, set, numpy array, etc)?

Each solution is an object with different features and methods. Objects handy to group features.

Routes are simple list. Lots of build in operators to work with lists. easy to modify. connection start and end made in cost calculation. Efficiency increased –¿ cost assigned to object.

List is easy to modify –¿ easy to calculate fitness.

4.2 Initialization

How do you initialize the population? How did you determine the number of individuals? Did you implement advanced initialization mechanisms (local search operators, heuristic solutions)? If so, describe them. Do you believe your approach maintains sufficient diversity? How do you ensure that your population enrichment scheme does not immediately take over the population? Did you implement other initialization schemes that did not make it to the final version? Why did you discard them? How did you determine the population size?

Have tried the heuristic nearest neighbours to initialize the population. After the initialization applied 3-opt local search technique.

NN explained ...

Local search –¿ 3 opt (high calculation load) see section 4.7

Rather small population to cut the calculation cost. For the small problem

Experimented with using NN for only half of the population and the rest randomly generated. Whereafter 3 opt was used. This only slowed down convergence, so used NN instead for the whole population. To improve diversity in the initial population, the NN is starting from a random start city that only can be selected once.

Because the use of the heuristic and local search –¿ started with a richer population and genetic algorithm was expected to do the small, last improvement until optimum.

Time to initialize 100 individuals in the 29 tour problem: ...

4.3 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?

Made use of competition based selection mechanism. The k tournament operator is chosen because the k value makes it easy to trade off exploration and exploitation and the operator can be easy and cheap implemented. The k -parameters are varied during the hypersearch.

Have chosen for k - tournament selection. A small adjustment with respect to the ones in the slides is the use of a variable k value. Start with small k –¿ more exploration . Later higher k, more exploitation. If more

individuals are selected, there is a higher change that among the selected individuals also a very good solution is chosen. This solution will rule out the rest which leads to more the same individuals used in the recombination operator and thus more exploitation towards this good solution.

Other operators tried are the exponentially decaying ranking with a decreasing s value over time. Results were similar.

Also tried the most simple random selection.

4.4 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?

Can try a decreasing probability of mutation with the increase of the amount of generations. You don't want the mutation operator to ruin your solutions.

4.5 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?

Implemented the inver-over evolutionary algorithm. Its power was its simpleness because the cross over was based on simple inversion of a sub part of an individual. This operator combines the advantage of unary and binary operators. Low cost and still a notion of the population. and thus was a cheap unary operator. The selection however of the segement was still population driven. Optimal solutions were consistently found for the tour29 problem.

downside –¿ inversion doesn't take the assymmetric cost matrix into account and is inpredictable. Therefore chosen for a greedy recombination opertor.

Overlap is copied to the offspring and rest of the edges are deleted. A random start city is chosen and the

4.6 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?

Tried ramdom elimination Tried lambda + mu Tried age based elimination Tried k-tournament elimination

Would have been better if would check the similarity of edges between different solutions and based on this increase the cost and so decrease the probability that an individual would be selected. The implementation that was made was however to calculation expensive to be efficient. Therefore the cheaper method of preventing the same route to be included in the survivors has been used.

4.7 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)?

NN developed and still used for the initialization of the population. Naive three opt by swapping, two opt is considered And efficient three opt based on two local searches that only considers the most potential cities for the three swap.

Let the local search run until convergence for the small problem.

4.8 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?

Say that added diversity promotion in the elimination, because than you have a direct effect. This is not the case when you add a diversity mechanism in the selection step, because cross over and mutation can influence the diversity of the population.

4.9 Stopping criterion

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

Say that first used the difference between the mean of the population and the best but is not good with diversity condition. What used as stopping criteria –¿ of the best solution stayed constant for 80 generations in the tour 29 problem.

4.10 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?

4.11 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?

A hyper search is run with theses parameters: ...

These are the parameters that can vary: ...

Here is a table with the final parameters: ...

The parameters for the bigger problem can be obtained in a similar way.

4.12 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.

Elitism is implemented. Talk about parallel implementation. The operators are good, but solution can be improved towards cheaper implementations of these operators.

5 Numerical experiments

Goal: Based on this section and our execution of your code, we will evaluate the performance (time, quality of solutions) of your implementation and your ability to interpret and explain the results on benchmark problems.

To obtain the numerical tests described here, the implementations of the bigger problems are changed to the disadvantage of quality. Local search removed/less search. population made smaller.

5.1 Metadata

What parameters are there to choose in your evolutionary algorithm? Which fixed parameter values did you use for all experiments below? If some parameters are determined based on information from the problem instance (e.g., number of cities), also report their specific values for the problems below.

Report the main characteristics of the computer system on which you ran your evolutionary algorithm. Include the processor or CPU (including the number of cores and clock speed), the amount of main memory, and the version of Python 3.

5.2 tour29.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? What is the corresponding sequence of cities?

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?

Solve this problem 1000 times and record the results. Make a histogram of the final mean fitnessess and the final best fitnesses of the 1000 runs. Comment on this figure: is there a lot of variability in the results, what are the means and the standard deviations?

5.3 tour100.csv

Have to simplify the implementation.

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?

5.4 tour194.csv

Have to simplify the implementation.

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?

5.5 tour929.csv

Have to simplify the implementation.

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?

6 Critical reflection

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

Describe the main lessons learned from this project. What do you think are the main strong points of evolutionary algorithms in general? Did you apply these strengths in this project? What are the main weaknesses of evolutionary algorithms and of your implementation in particular? Do you think these can be avoided or mitigated? How? Do you believe evolutionary algorithms are appropriate for this problem? Why (not)? What surprised you and why? What did you learn from this project?

7 Other comments

In case you think there is something important to discuss that is not covered by the previous sections, you can do it here.