Genetic Algorithms and Evolutionary Computing
Template for project report

Stagl, Martin Johann
r0776982@kuleuven.be

Hoang, Phuong Thao
r0776845@kuleuven.be

January 5, 2020

1 Existing genetic algorithm

Perform *a limited set* of experiments by varying the parameters of the existing genetic algorithm (population size, probabilities, ...) and evaluate the performance (quality of the solutions).

1. Data set(s) used & explain why you selected these data set(s):
   For our initial performance estimation of the existing simple genetic algorithm we use the provided *131 cities benchmark* from the course page in Toledo.
   This has several reasons:

   (a) We are given an optimal route and can examine how 'far' we are away from the existing optimum.
   (b) The position of points is fixed, thus does not change over our parameter estimation runs.
   (c) The benchmark data set is the smallest from the given ones with known optimal path, and therefore we hope that the running time until convergence will be not too high.

2. Parameters considered and intervals/values:
   According to the book [ES15] the best-known method in the category of non iterative tuning is the frequently used parameter 'optimisation' through a systematic comparison of a few combinations of parameter values.

   Regarding this, we limit the parameters we want to examine within our experiments to the following:

   (a) Population size: 32, 64, 96, 128
   (b) Maximum number of Generations: 30, 40, 60, 80, 100
   (c) Probability of crossover: 5%, 90%, 95%, 100%
   (d) Probability of Mutation: 5%, 10%, 95%
   (e) Percentage of Elite: 0%, 5%

3. Performance criteria used:
   As performance criteria we regard both **computation time** of the whole simple genetic algorithm without visualization on a windows 10 64bit machine with i7-5500 2.40GHz quad-core CPU with 8 GB DDR3 RAM and fixed seed of 776982 to ensure reproducibility and **mean minimum path length** over *ten* runs.

4. Test results:
   The resulting minimal paths together with the variable combination and resulting computation times are stored as a table in the comma separated values file
   $'/template/results\_ParameterTuning_Task1.csv'$. Some results are also shown in *Table 1*.

NIND	MAXGEN	ELITIST	PR_CROSS	PR_MUT	Time (in s)	Shortest-Path	Generations
32	40	0	0.05	0.05	0.0608	2339.77	40
32	100	0.05	0.95	0.95	0.8872	1886.84	100
64	30	0	0.05	0.05	0.0961	1901.86	30
64	80	0	0.05	0.05	0.2258	1654.31	80
96	60	0	1	0.1	1.6169	2169.68	60
96	100	0	0.05	0.05	0.3789	1663.42	100
128	30	0.05	0.05	0.05	0.1614	1776.15	30
128	100	0.05	0.05	0.95	0.8568	**1195.51**	100

Table 1: Sample of Results Task 1, in bold the overall best achieved result

5. Discussion of test results:
   From the evaluation results we can clearly see that with increasing maximum number of generations the traveling route decreases. Also the more individuals per population the smaller is the route.
   On the other hand, when increasing the crossover rate we get worse results. This can be a hint that the chosen crossover operator namely *'alternating edges crossover'* for adjacency representation may not be the best option and therefore we will investigate it in the following sections. This is highly important as the crossover operator mainly functions as a tool for exploitation, thus tries to converge the population to a specific point in landscape.
   From the first boxplot we examine that with increasing mutation probability the variance of the route length increases. As mutation operators are mostly used to provide exploration of more areas of the solution landscape it is crucial to find a good rate that provides enough exploration of the search space but not prohibits exploitation.
   The data also show that the maximum number of generations per run is always generated. In the following section we provide three different stopping criteria and use it in combination with the best parameter-options obtained from this experiment.

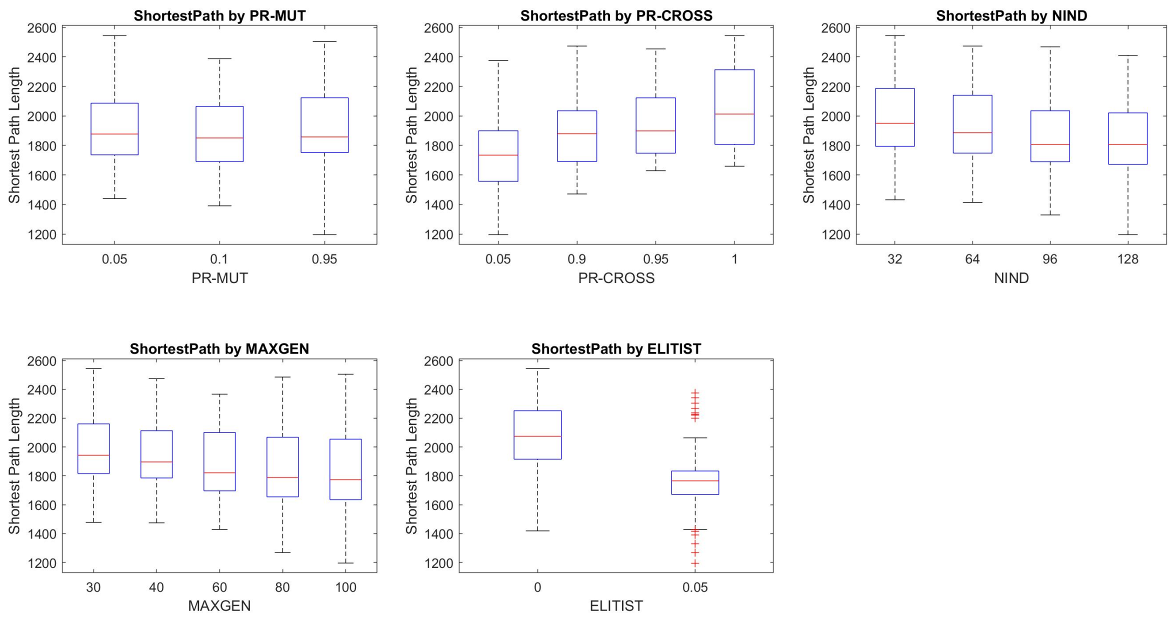


Figure 1: Performance aggregated per variable and variable value

2 Stopping criterion

Implement a stopping criterion that avoids that rather useless iterations (generations) are computed.

1. **Stopping criterion & explain why you selected this criterion**
   We decided to implement three different stopping criteria and try to find the most suitable one for the traveling salesman problem through parameter search. Our approach is based on stopping the execution if for **n_percentage** of generations the difference (in absolute value) of the stopping evaluation functions $E_n(T)$ and $E_{n-1}(T)$ stays within the interval $[0, delta]$.
   Then to be optimized parameters per criterion are *delta*, which regulates the interval in which the obtained value from the stop function can lie and *n_percentage*, which regulates how many of following generations can be within the interval defined by the stop function and delta.

   (a) The first criterion stops the GA if $E_1(T) = min(F)$ stays in the interval $[E_1(T) - delta, E_1(T) + delta]$ for n_percentage of following generations. This criterion simply states that the algorithms stops if the best obtained individual does not change over several generations, this is a rather basic stopping criterion and triggers if exploitation of the best sub-population is maximal.

   (b) The second criterion stops the GA if $E_2(T) = mean(F)$ stays in the interval $[E_2(T) - delta, E_2(T) + delta]$ for n_percentage of following generations. This stopping criterion is triggered when the mean path length of the population does not change a lot (depending on delta). This criterion considers the whole population and may trigger if exploitation of all local minima is maximal.

   (c) The third criterion stops the GA if $E_3(T) = mean(min(F))$ stays in the interval $[E_3(T) - delta, E_3(T) + delta]$ for n_percentage of following generations. This criterion is triggered when the improvement of the best individual per generation flattens.

2. **Test results (incl. performance criteria and parameter settings)**

   For our parameter optimization we use as population size: 128, as maximum number of Generations: 100, as probability of crossover: 5% as probability of mutation: 95% as percentage of elite: 5% as the best parameters examined from Task 1 and the following parameter sets:

   (a) stoppingCriteria: 1,2,3

   (b) n_percentage: 0.2, 0.4, 0.6

   (c) delta: 50, 70, 100

   The resulting minimal paths together with the variable combination, the resulting computation times and the number of generations actually computed are stored as a table in the comma separated values file
   '/template/results_ParameterTuning_Task2.csv'. Some results are also shown in *Table 2*.

stop-pingCrite-ria	n_percentage	delta	Time (in s)	Shortest-Path	**Genera-tions**
1	0.4	100	0.8836	1195.51	100
1	0.2	50	0.9861	1345.06	79
1	0.6	100	0.1190	1997.48	11
2	0.2	100	0.9155	1137.57	100
2	0.2	70	0.2953	1507.11	33
2	0.4	50	0.1821	1996.15	17
3	0.6	50	0.2446	1690.70	25
3	0.4	70	0.2443	1771.57	17
3	0.2	100	0.1493	1965.36	13

Table 2: Sample of Results Task 2 ordered by Criteria and ShortestPath

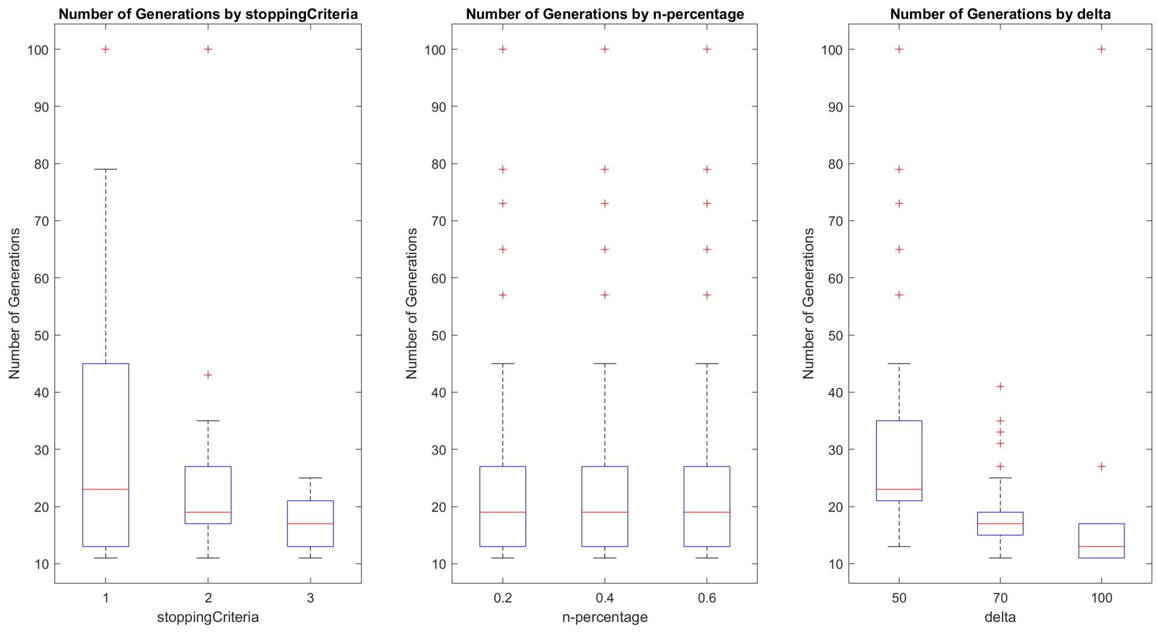


Figure 2: Number of Generations until stopping criteria is triggered aggregated per parameter value

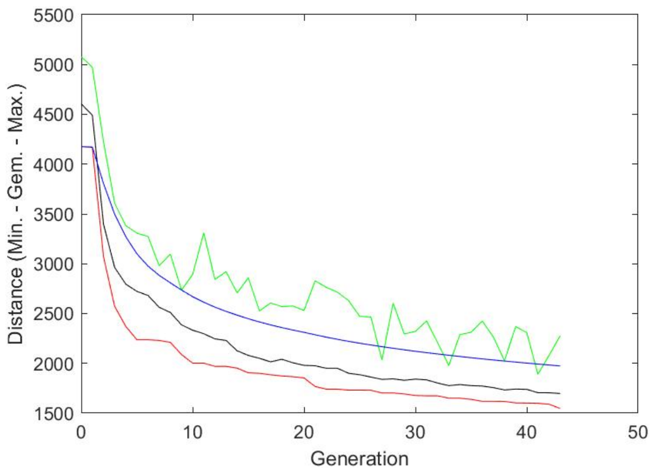


Figure 4: Distance vs Number of Generations: The green line is the fitness of the worst individual in generation t. The blue line is the average fitness in generation t, the black line is the mean best fitness E_3 and the red line the best fitness value so far.

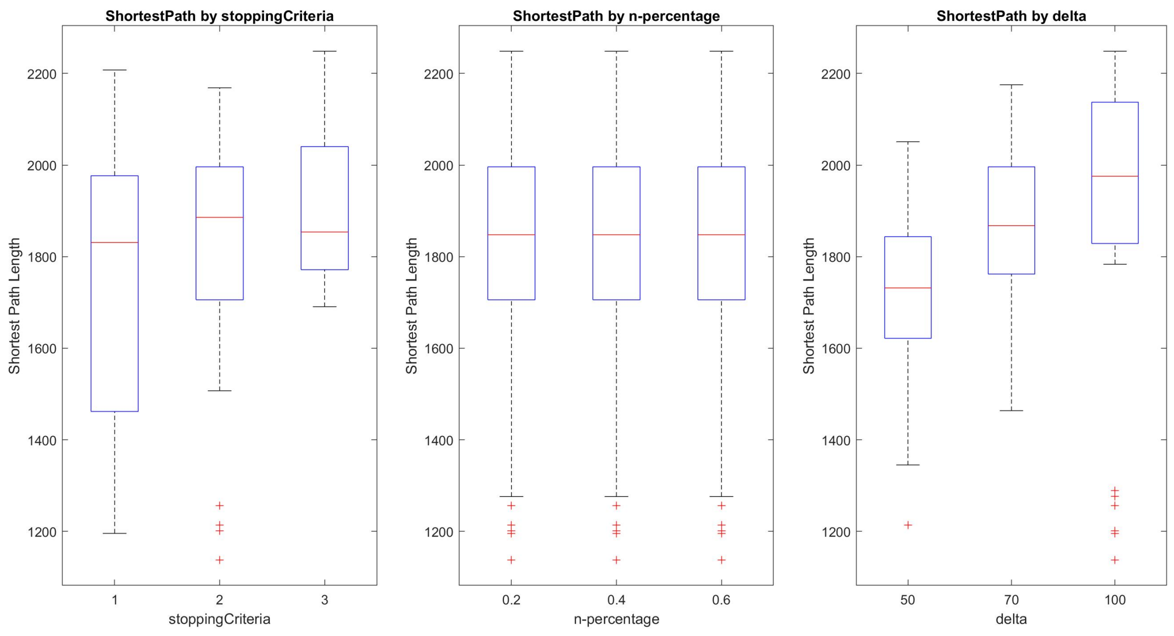


Figure 3: ShortestPath values aggregated per parameter value

Taking the best results of both runs into account, namely for Task 1 the parameter setting: $NIND = 128$ and $MAXGEN = 100$ and $ELITIST = 0.05$ and $PR\_CROSS = 0.05$ and $PR_MUT = 0.95$ and for Task 2 $NIND = 128$ and $MAXGEN = 100$ and $stoppingCriteria = 1$ and $n\_percentage = 0.2$ and $delta = 50$ we get highly significant performance differences tested both with a paired t-test as a parametric test and with the Wilcoxon test as an non-parametric test.

Paired t-test:
data: x and y
$t = -5.4365, df = 9,$ p-value $= 0.0004129$
alternative hypothesis: true location shift is not equal to 0

Wilcoxon signed rank test
data: x and y
$V = 0,$ p-value $= 0.001953$
alternative hypothesis: true location shift is not equal to 0

3. **Discussion of test results:**
The results show that it is possible with only a simple stopping criterion to reduce computation time by nearly a magnitude but not without loss of significant performance. We can also see that stopping method 3 has the highest stopping pressure limiting the number of generations.

3 Other representation and appropriate operators (main task)

Implement and use another representation and appropriate crossover and mutation operators. Perform some parameter tuning to identify proper combinations of the parameters.

1. **Representation**
As a representation for TSP problem we chose to use adjacency representation and path representation as these two are mostly used for graphical problems such as ours, where we give each city a unique number from *1 to N* and we try to find a permutation of these numbers so that the length of the path through these cities is the shortest.

The reason why we didn't use for example binary representation is that encoding the cities that way would be quite unnatural. Giving each city a unique number was the easiest and most intuitive choice. Also, during our lectures and exercises we talked a lot about compatible crossover and mutation operators and some of their variants were also a topic of scientific researches and they were proved to work quite well for TSP (`https://www.hindawi.com/journals/cin/2017/7430125/tab1/`).

Another reason for choosing adjacency and path representation is that the operators are in most cases interchangeable and compatible with both of the representations.

Each individual is represented as a permutation of *N* numbers. That's in total *N!* possibilities. BUT we have to take into account that for example the individuals *(1, 2, 3), (2, 3, 1)* and *(*3, 1, 2) represent the same path, which leads us to the *(N-1)!* possibilities.

2. **Crossover operators & explain why you selected the operators**
   We worked with already implemented Alternating Edges Crossover in a combination with adjacency representation (default setting).

   As our own we implemented Partially Mapped Crossover *(pmx)* and Order Crossover *(orderx)* to complement the choice of path representation. These methods and their variations are heavily used in the literature.

   As we have expected, Order crossover works better than PMX because it preserves the ordering of the parents so the exploitation rate is higher.

   Lastly, we implemented Heuristic Crossover *(uhx)* - task 4. It works extremely well with path representation - leads to quicker decrease of path length and higher accuracy.

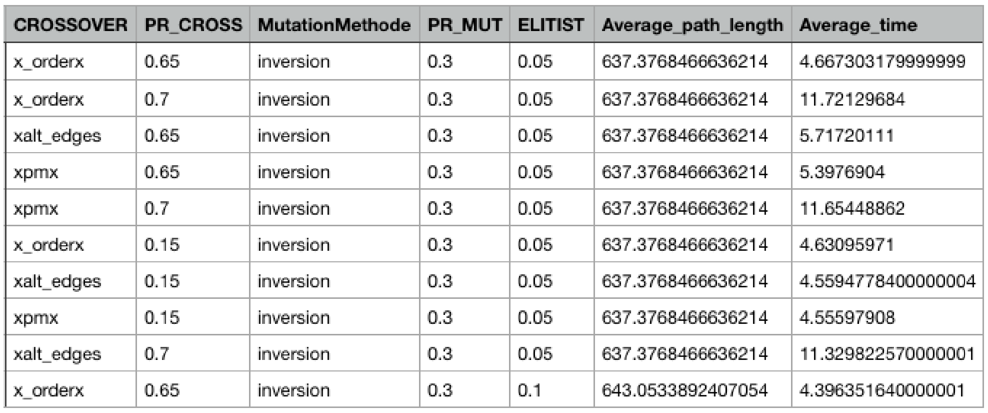
3. **Mutation operators & explain why you selected the operators**
   We've implemented all the mutation operators which are compatible with permutation representation and were mentioned during the lectures. That means: Swapping, Scramble, Insert, Inversion and Reciprocal exchange (already in the package). While swapping only works with path representation (because it can create loops), the others were used in a combination with both of our representations.

4. **Parameter tuning: parameters considered and intervals/values**
   In this task we are mostly focusing on representation and crossover and mutation operators. That's why we were playing with these parameters and we chose their following values so that we could get as many diverse combinations as possible (the rest of the parameters are fixed):

   (a) Representation: 1 - adjacency, 2 - path
   (b) Crossover operator: xalt_edges, xpmx, x_orderx
   (c) Probability of crossover: 15%, 65%, 70%
   (d) Mutation operator: inversion, reciprocal_exchange, scramble, insert, swapping
   (e) Probability of Mutation: 15%, 30%, 70%
   (f) Percentage of Elite: 1%, 5%, 10%

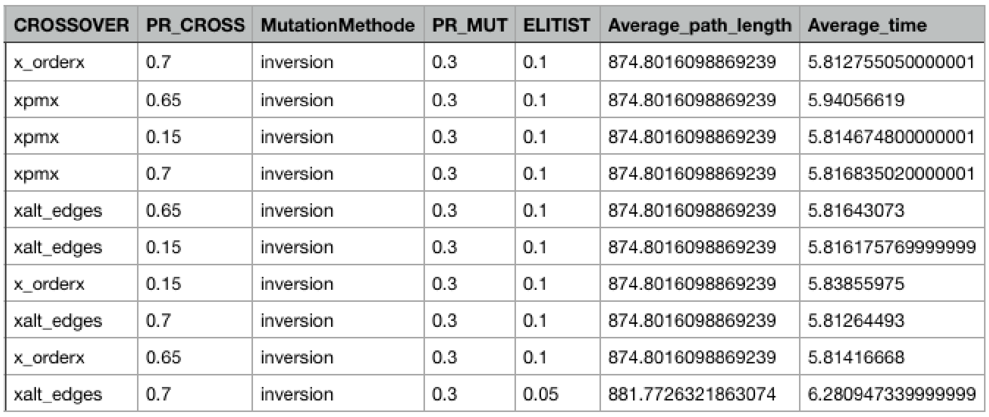
5. **Data set(s) used & explain why you selected these data set(s)**
   For this experiment we chose again *131 cities benchmark* from the course page in Toledo for the same reasons as in *Task 1.* This way we could also compare the results with SGA.

6. **Test results (incl. performance criteria and parameter settings)**
   The 10 best results (according to the path length) for adjacency representation were obtained while choosing our parameters this way:



CROSSOVER	PR_CROSS	MutationMethode	PR_MUT	ELITIST	Average_path_length	Average_time
x_orderx	0.65	inversion	0.3	0.05	637.3768466636214	4.667303179999999
x_orderx	0.7	inversion	0.3	0.05	637.3768466636214	11.72129684
xalt_edges	0.65	inversion	0.3	0.05	637.3768466636214	5.71720111
xpmx	0.65	inversion	0.3	0.05	637.3768466636214	5.3976904
xpmx	0.7	inversion	0.3	0.05	637.3768466636214	11.65448862
x_orderx	0.15	inversion	0.3	0.05	637.3768466636214	4.63095971
xalt_edges	0.15	inversion	0.3	0.05	637.3768466636214	4.5594778400000004
xpmx	0.15	inversion	0.3	0.05	637.3768466636214	4.55597908
xalt_edges	0.7	inversion	0.3	0.05	637.3768466636214	11.329822570000001
x_orderx	0.65	inversion	0.3	0.1	643.0533892407054	4.396351640000001

The 10 best results (according to the path length) for path representation were obtained while choosing our parameters this way:



CROSSOVER	PR_CROSS	MutationMethode	PR_MUT	ELITIST	Average_path_length	Average_time
x_orderx	0.7	inversion	0.3	0.1	874.8016098869239	5.812755050000001
xpmx	0.65	inversion	0.3	0.1	874.8016098869239	5.94056619
xpmx	0.15	inversion	0.3	0.1	874.8016098869239	5.814674800000001
xpmx	0.7	inversion	0.3	0.1	874.8016098869239	5.816835020000001
xalt_edges	0.65	inversion	0.3	0.1	874.8016098869239	5.81643073
xalt_edges	0.15	inversion	0.3	0.1	874.8016098869239	5.816175769999999
x_orderx	0.15	inversion	0.3	0.1	874.8016098869239	5.83855975
xalt_edges	0.7	inversion	0.3	0.1	874.8016098869239	5.81264493
x_orderx	0.65	inversion	0.3	0.1	874.8016098869239	5.81416668
xalt_edges	0.7	inversion	0.3	0.05	881.7726321863074	6.280947339999999

Taking the best results from Task 1 and this task into account, we get significantly better performance with our new implementation tested both with a paired t-test as a parametric test and with the Wilcoxon test as an non-parametric test.

Paired t-test:
Adjacency Representation: $t = 148.4, df = 9, \text{p-value } < 2.2e-16$
Path Representation: $t = 48.625, df = 9, \text{p-value} = 3.299e-12$
alternative hypothesis: true location shift is not equal to 0

Wilcoxon signed rank test
Adjacency Representation: $V = 55, \text{p-value} = 0.001953$
Path Representation: $V = 55, \text{p-value} = 0.001953$
alternative hypothesis: true location shift is not equal to 0

7. **Discussion of test results**
As we can see from the above results, for both representations we obtained all 10 best results with mutation operator set to inversion and probability of mutation quite high (30%). The probability of mutation is not changing. It seems like the choice of PR_MUT=0.3 provides us with the optimal exploration while probability of a crossover vary quite a lot.
While adjacency representation works better with low elitism = 5%, for path representation the suitable choice is higher elitism of 10%. Also, we noticed that all three crossover operators appear in the both tables. That means that their exploitation performance in our example is comparable.

To our surprise the results for path representation were quite bad (when looking at the path length) in comparison with adjacency representation. However, for the average computational time it is the other way around. The results of adjacency representation are getting close to the optimal path length which is 564. We will get better results for path representation using heuristic crossover *uhx* introduced in the next task.

Overall, the results were much better than the ones obtained by SGA. But we have to mention that this was ran together with the Nearest neighbour initialization heuristic which helped to improve the performance quite a lot.

4 Local optimisation

Test to which extent a local optimisation heuristic can improve the result.

1. Local optimisation heuristic & explain why you selected this heuristic:
   We decided to implement two different approaches of heuristic optimization. The first one can be used as an alternative to random initialization and the second one to use the distance between as a heuristic for the crossover operator.

   (a) **Nearest neighbour initialization heuristic**:
   In this heuristic, the salesman starts at some city and then visits the city nearest to the starting city, and so on, only taking care not to visit a city twice. At the end all cities are visited and the salesman returns to the starting city.

   (b) **Unnamed Heuristic Crossover**
   This crossover operator was presented in [VYPS09], operates heuristically in terms of starting with a random current city and copies it to child. It puts four pointers to right and left neighbors of current city in both parents and then compares which pointed city is nearest to current city and has not been copied to child yet. Such a node is selected as current city, copied to child and its pointer goes forward in its direction.

2. Test results (incl. performance criteria and parameter settings):
   As performance criteria we use the same as explained before but keep our focus mainly on the time differences between ordinary and heuristic implementations.

CROSSOVER	InitializationMethode	MutationMethode	Time (in s)	ShortestPath	Generations
'uhx'	2	'scramble'	7.2992	643.85	300
'uhx'	2	'reciprocal_exchange'	8.6763	646.43	300
'uhx'	1	'inversion'	5.9815	661.35	300
'xalt_edges'	2	'inversion'	6.2269	847.90	300
'xpmx'	2	'inversion'	6.0154	847.90	300
'x_orderx'	2	'inversion'	6.0042	847.90	300
'xalt_edges'	1	'inversion'	5.8538	1496.91	300

Table 3: Sample of Results Task 4 ordered by ShortestPath

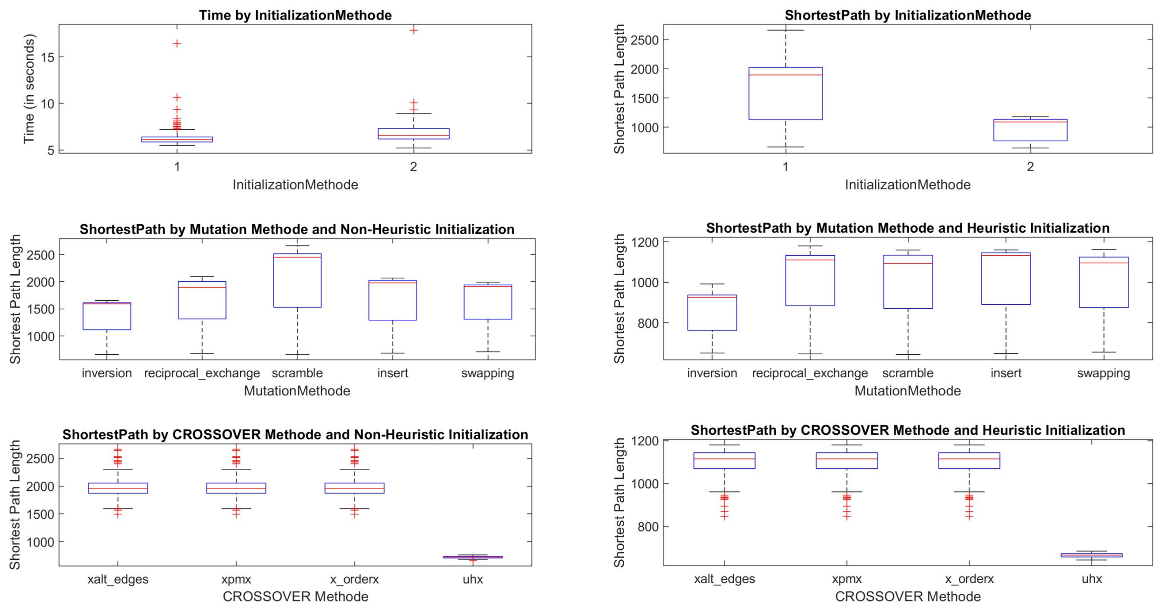


Figure 5: ShortestPath values aggregated per parameter value

Taking the best results from Task 1 and this task into account, we get highly significant better performance with our new implementation tested both with a paired t-test as a parametric test and with the Wilcoxon test as an non-parametric test.

Paired t-test:
data: x and y
$t = 156.98, df = 9, \text{p-value } < 2.2e-16$
alternative hypothesis: true location shift is not equal to 0

Wilcoxon signed rank test
data: x and y
$V = 55, \text{p-value} = 0.001953$
alternative hypothesis: true location shift is not equal to 0

3. **Discussion of test results**
From these results we can see that with the heuristic initialization we gain in average a 50 generation head start in comparison to random initialization without losing much of diversity as of the high percentage of the mutation rate. However, the better heuristics was by using it for the crossover. While it may happen that the average path length increases after using heuristic initialization, it always decreased when using heuristic crossover. Comparing heuristic crossover with non-heuristic variants we can see from the results that the overall minimal path length per generation rapidly decreases at the beginning while using heuristic crossover, in contrast to non-heuristic where it only slowly decreases.

Also from the statistical tests we can examine that these heuristic implementations result in a highly significant improvement over the simple genetic algorithm.

5 Benchmark problems

Test the performance of your algorithm using *some* benchmark problems (available on Toledo) and critically evaluate the achieved performance.

1. List of benchmark problems:
   For this Tasked we used all provided benchmark data sets from the course page in Toledo which had an optimal path attached.
   This has several reasons:

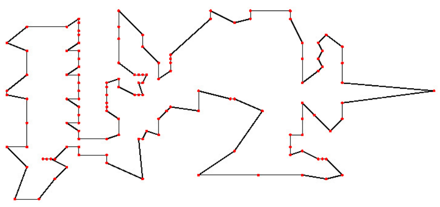
   (a) We are given an optimal route and can examine how 'far' we are away from the existing optimum.
   (b) The position of points is fixed, thus does not change over our parameter estimation runs.

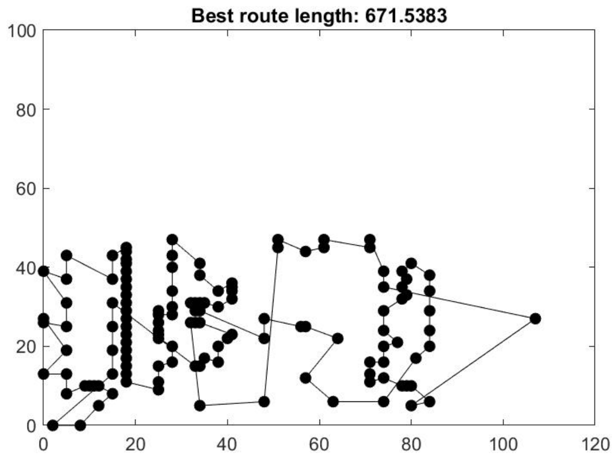
   The data sets are therefore the following.

   (a) 131 cities
   (b) 380 cities
   (c) 662 cities
   (d) 711 cities

2. Test results (incl. performance criteria and parameter settings)
   Taking the best parameter setting from *Task 1* and from the previous tasks combined the best parameter settings, which there are:
   $CROSSOVER = uhx$, $NIND = 300$, $MAXGEN = 300$, $ELITIST = 0.05$, $PR\_CROSS = 0.7$, $PR\_MUT = 0.3$, $InitializationMethode = heuristic$, $RepresentationMethode = path$, $MutationMethode = Scramble\ and\ SelectionMethode = fps$,
   we get highly significant better performance with our implementation tested both with a paired t-test as a parametric test and with the Wilcoxon test as an non-parametric test as can seen below.

   (a) 131 cities


(a) Optimal Path. It has length 564.



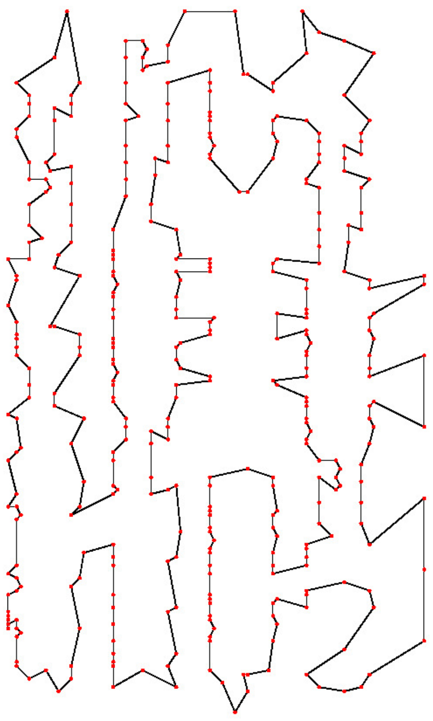
(b) Sample result of our algorithm

Figure 6: Optimal Path vs Sample result of our algorithm.

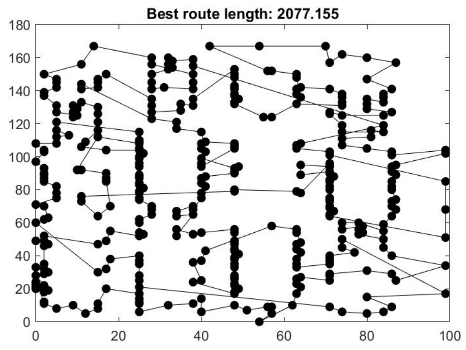
Paired t-test:
$t = 32.732, df = 9$, p-value $= 1.141e - 10$
alternative hypothesis: true location shift is not equal to 0

Wilcoxon signed rank test
$V = 55$, p-value $= 0.001953$
alternative hypothesis: true location shift is not equal to 0

(b) 380 cities



(a) Optimal Path. It has length 1621.



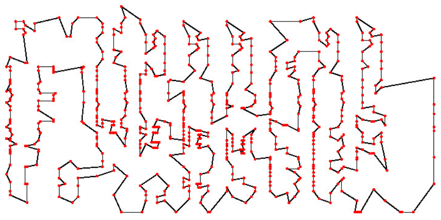
(b) Sample result of our algorithm

Figure 7: Optimal Path vs Sample result of our algorithm.

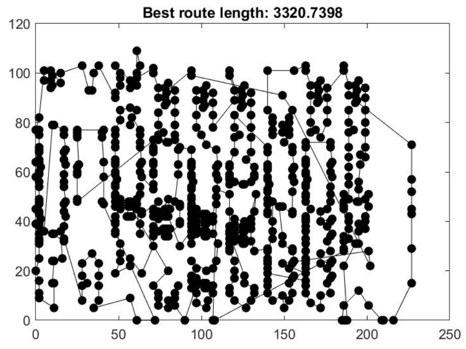
Paired t-test:
$t = 59.216, df = 9, \text{p-value} = 5.628e - 13$
alternative hypothesis: true location shift is not equal to 0

Wilcoxon signed rank test
$V = 55, \text{p-value} = 0.001953$
alternative hypothesis: true location shift is not equal to 0

(c) 662 cities



(a) Optimal Path. It has length 2513.



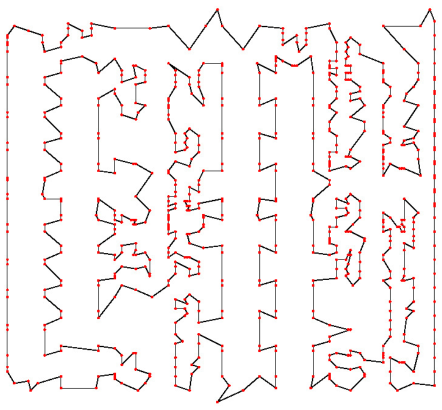
(b) Sample result of our algorithm

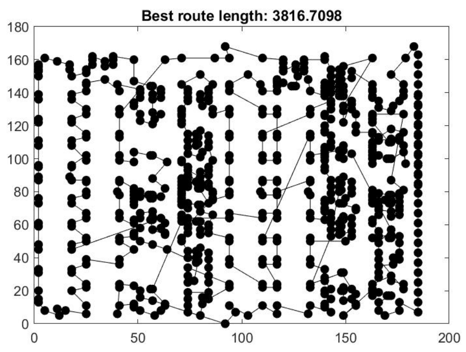
Figure 8: Optimal Path vs Sample result of our algorithm.

Paired t-test:
$t = 70.846, df = 9, \text{p-value} = 1.124e - 13$
alternative hypothesis: true location shift is not equal to 0

Wilcoxon signed rank test
$V = 55$, p-value $= 0.001953$
alternative hypothesis: true location shift is not equal to 0

(d) 711 cities


(a) Optimal Path. It has length 3115.



(b) Sample result of our algorithm

Figure 9: Optimal Path vs Sample result of our algorithm.

Paired t-test:
$t = 62.037, df = 9$, p-value $= 3.706e - 13$
alternative hypothesis: true location shift is not equal to 0

Wilcoxon signed rank test
$V = 55$, p-value $= 0.001953$
alternative hypothesis: true location shift is not equal to 0

3. **Discussion of test results**
   The test results showed that with our implementation and with the best parameter settings, evaluated in the previous tasks, we could achieve quite good results throughout the benchmark tests.
   All the results have a very low percentage of deviation from the optimal path. This can also be seen from the images(right side) which closely relate the optimal paths shown in the images to the left above.

6 Other task(s)

We decided to experiment with parent selection and survivor selection.

1. **Description of implementation**
   For parent selection we used SUS which was a part of the initial MATLAB tsp package. As rival selections we chose to implement Fitness proportionate selection and Tournament selection without replacement. It's proven in many articles that this choice of a parent selection works well for TSP problem. These methods (*fps, tourwithoutrepl*) were then called before the recombination and mutation in a function *select*.

   For survivor selection (aka replacement strategy) we chose to implement Round-robin tournament. The reason behind this was that this algorithm doesn't care about the number of offspring in comparison to number of parents like it is for example in (μ, λ) method. Also, in Round-robin tournament we can play with the parameter q which tells us with how many opponents will each individual have to compete. The choice of q is important because its value decides how much fitness pressure will there be. With smaller q we allow less fit individuals to be selected.

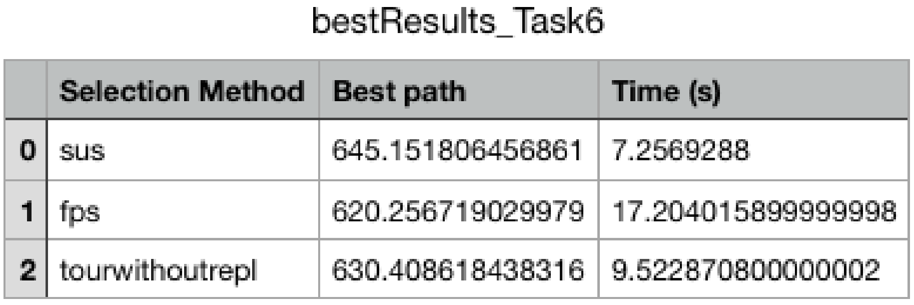
On the other hand if $q = \textit{population size}$ then the selection is deterministic. We placed this method *rrt* inside *gatbx* folder and it is called within *reins* function which reinserts offspring in the population after recombination and mutation.

2. **Description of the experiments**
   Experiments were again done on *131 cities benchmark problem.* We went with path representation with these fixed parameters (chosen according to the previous results):

   (a) Crossover operator: Unnamed Heuristic Crossover
   (b) Probability of crossover: 70%
   (c) Mutation operator: inversion
   (d) Probability of Mutation: 30%
   (e) Percentage of Elite: 5%

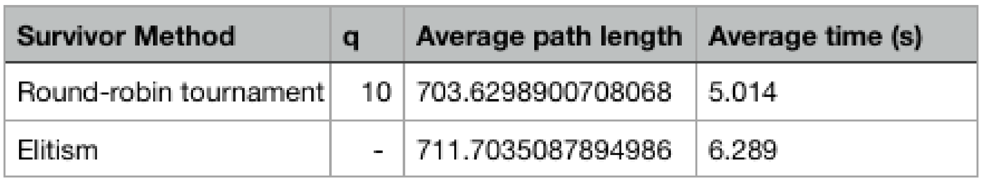
   For each parent selection method we ran tests 10 times and selected the best results according to path length. After receiving these results we added the survivor selection with the same parameters as stated above and with the best parent selection method. In addition, we lowered the MAXGEN from 300 to 100 (explanation below).

3. **Test results**



bestResults_Task6

	Selection Method	Best path	Time (s)
0	sus	645.151806456861	7.2569288
1	fps	620.256719029979	17.204015899999998
2	tourwithoutrepl	630.408618438316	9.522870800000002



Survivor Method	q	Average path length	Average time (s)
Round-robin tournament	10	703.6298900708068	5.014
Elitism	-	711.7035087894986	6.289

4. **Discussion of test results**
   The best result according to path length was obtained by choosing Fitness proportionate selection. There's only 11% deviation from the optimal path. But if we look at the computational time, FPS is the worst out of our three selection methods. To balance the computational time and accuracy, the best choice seems to be Tournament selection without replacement.

   We've noticed that using a Round-robin tournament for survivor selection led to quicker convergence to the optimal path. To see that more clearly we decided to lower the maximum number of generations to 100 instead of 300. We ran tests with different qś and came to a conclusion that setting q=10 is the most suitable for our benchmark problem. We also compared it with the results with already implemented elitism and displayed the average values in the second table above.

Conclusion

Concluding we want to give an overview of our finding in a table and present the paired t-test results among the best results of each task compared to the SGA provided as a base implementation on the 131 cities benchmakr data set.

Algorithm Used	P-Value	Min Distance	Mean Min Distance	Mean Time (in sec)
SGA	-	1158.95	1212.1	0.84
Best Task 1 Stopping Criteria	0.0004129	1345.00	1572.9	0.51
Best Task 3 Adj	2.2e-16	616.19	637.4	4.67
Best Task 3 Path	3.299e-12	828.52	874.8	5.81
Best Task 4 Heuristic	2.2e-16	646.43	662.5	9.21
Best Task 6 Heuristic	1.057e-10	660.24	703.6	7.02

Table 4: P-Values using paired T-Test compared to the base implementation given which was the simple genetic algorithm

We can see that our additional effort lead to significant better results than obtained from the base implementation.

Time spent on the project

1. For each student of the team: estimate how many hours spent on the project (NOT including studying textbook and other reading material).
   (a) Martin Johann Stagl (41 hours):
   - Setup and limited set of experiments (2 hour)
   - Implementation and evaluation of stopping criterion (5 hours)
   - Implementation and evaluation of another representation methode (2 hours)
   - Implementation and evaluation of local optimisation heuristic for initialization and crossover (8 hours)
   - Implementation and evaluation of performance tests on benchmark problems (6 hours)
   - Writing the report and documenting the code (10 hours)
   - Meeting, where we discussed our findings and problems: 4 time 2 hours each (8 hours)

   (b) Hoang Phuong Thao (41 hours):
   - Setting up path representation (2 hours)
   - Implementation of crossover operators and mutation operators (8 hours)
   - Implementation of parent selection and survivor selection (5 hours)
   - Setting tests for parameter tuning and writing scripts for the analysis of results (8 hours)
   - Writing the report and documenting the code (10 hours)
   - Meeting, where we discussed our findings and problems: 4 time 2 hours each (8 hours)

2. Briefly discuss how the work was distributed among the team members.
   During our first meeting we discussed how we wanted to distribute the workload. We aimed for an equal distribution of workload and also wanted that both of us know about the work of the other person. We then scheduled three more meetings in which we discussed our results and tried to help each other if there where some problems with the implementation.

References

[ES15] A. E. Eiben and James E. Smith. *Introduction to Evolutionary Computing.* Springer Publishing Company, Incorporated, 2nd edition, 2015.

[VYPS09] G. Vahdati, M. Yaghoubi, M. Poostchi, and M. B. N. S. A new approach to solve traveling salesman problem using genetic algorithm based on heuristic crossover and mutation operator. In *2009 International Conference of Soft Computing and Pattern Recognition*, pages 112–116, Dec 2009.