Evolutionary Algorithms: Final report

Erikas Švažas (r0769473)

January 3, 2021

1 Metadata

- **Group members during group phase:** Kopko Jakub and Schellekens Ellen
- **Time spent on group phase:** 15 hours
- **Time spent on final code:** 110 hours
- **Time spent on final report:** 20 hours

2 Modifications since the group phase

2.1 Main improvements

1. the algorithm was modified according to the 3-islands approach to improve diversity and obtain lower distance tours for the TSP. Different EA strategies were created for every sub-population.
2. added Nearest Neighbour (NN) heuristic for initialization, which significantly improved convergence time.
3. additionally to the order crossover, heuristic greedy (HGreX) and sequential consecutive crossover (SCX) operators were implemented. Not only did these operators lead to a better-fitness solution, but also the convergence rate was drastically improved. This is because HGreX and SCX operators take into account the distance matrix while making recombination decisions.
4. additionally to the inversion mutation, inversion displacement mutation operator was implemented. This operator creates an additional level of exploration for each individual due to a combination of two separate mutation operators - inversion and displacement, leading to better performance of the EA.
5. $(\lambda+\mu)$ elimination was changed by (λ, μ) elimination: in the case of $(\lambda+\mu)$, the population can be dominated by one individual which is much better than others leading to an early convergence of the EA. This is not the case for (λ, μ) elimination as individuals from old population are not copied to new a new generation.
6. added adaptive control of mutation and recombination probabilities. The mutation control strategy is different for every sub-population, recombination probability depends on the fitness feedback from the EA.
7. added migration scheme between different sub-populations. The scheme starts only if any of the sub-populations has converged or a fixed number of iterations have passed. At every iteration, 2 out of 3 sub-populations are randomly selected for migration. These sub-populations exchange $2-4$ individuals.
8. added local search operator *2-opt*, which significantly improved exploitation phase of the EA. To make *2-opt* faster additional improvements such as Don't look bits (DLB) and Fixed radius search (FRS) were implemented. The operator starts after 3 iterations from the start of the migration phase.
9. added methodology to adapt parameters based on the problem-size. As *2-opt* requires significant computational time, smaller number of individuals are improved with *2-opt* for larger problems. Also, smaller population and offspring sizes are used in case of large problems.
10. updated termination criterion for the use with islands scheme and *k-opt*. Only if any of the sub-populations converges, the EA (consisting of recombination and mutation operators) is extended by the migration and local search operators. After the start of the migration stage, the stopping criterion of the main loop is restarted and updated to take into account the difference between best-fitness and mean-fitness of every sub-population.

2.2 Issues resolved

Issue 1: the implementation in the group phase was still able to get stuck in a local minimum, probably due to the progressing loss of the diversity of the population. One of the main reasons is inadequate exploitation pressure put on by the elimination operator. The previously used $(\lambda + \mu)$ strategy can lead to an early convergence as one individual might have a much better fitness score than others, thus I implemented (λ, μ) scheme. To maintain a diversity of the population, I have implemented a diversity-promoting 3-island model where sub-populations undergo different evolution. All modifications significantly benefited the solutions for all benchmark problems.

Issue 2: the implementation in the group phase lacked efficiency to successfully find the best-fitness individual as fast as possible. The addition of the NN method during the initialization significantly improved the convergence rate. Also, additional SCX and HGreX crossover operators not only have a much higher convergence rate than previously used order crossover but also provide better-fitness solutions, which can be improved even further by the *2-opt*. To control the exploration-exploitation balance during evolution, adaptive strategies for mutation and recombination probabilities were used. However, this efficiency problem is directly related to the identified issue 1 and the provided solutions such as a better elimination scheme and diversity-promotion methods are essential for the issue 2 as well.

3 Final design of the evolutionary algorithm

The scheme of a final design of the implemented EA is shown in figure 1. Next, an extensive discussion of different scheme parts will be covered.

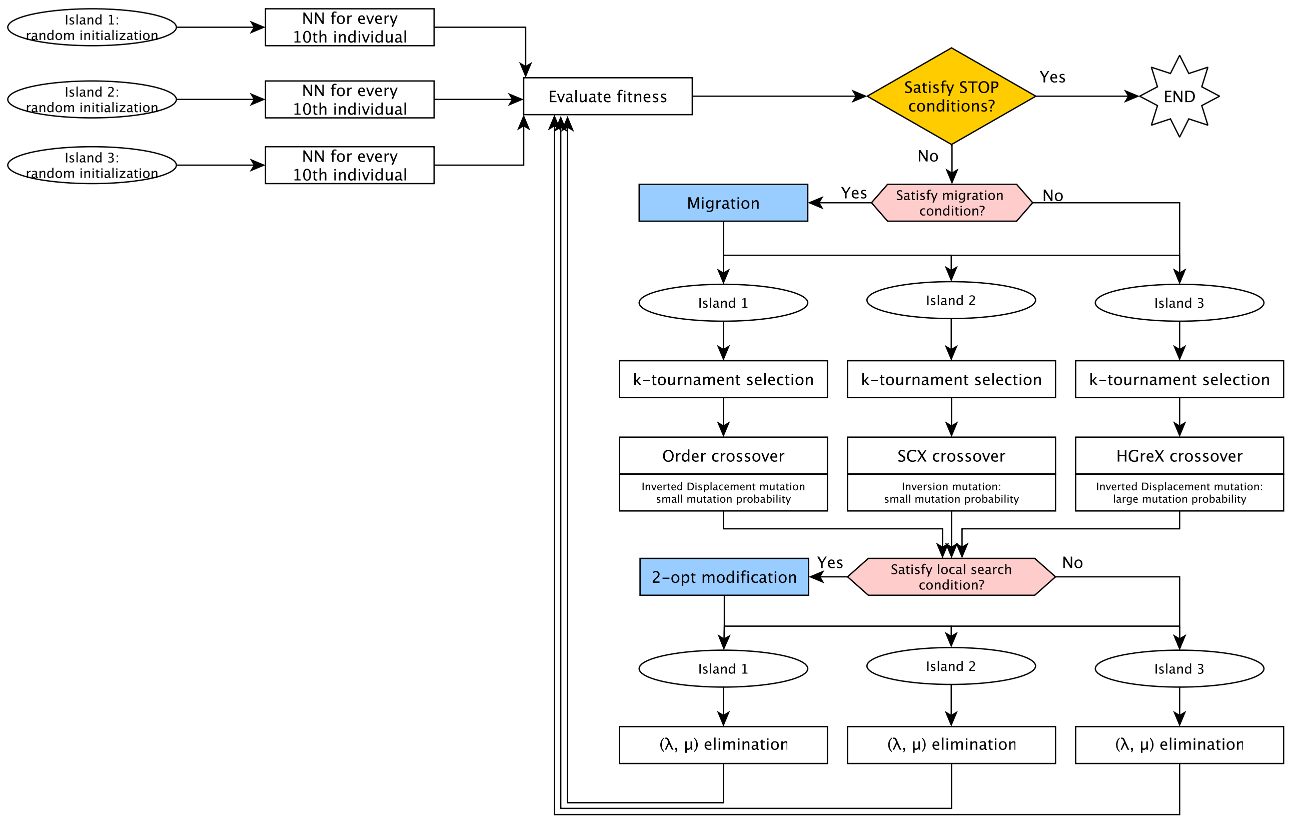

Island 1: random initialization
Island 2: random initialization
Island 3: random initialization
NN for every 10th individual
NN for every 10th individual
NN for every 10th individual
Evaluate fitness
Satisfy STOP conditions?
Yes
END
No
Migration
Yes
Satisfy migration condition?
No
Island 1
Island 2
Island 3
k-tournament selection
k-tournament selection
k-tournament selection
Order crossover
Inverted Displacement mutation small mutation probability
SCX crossover
Inversion mutation: small mutation probability
HGreX crossover
Inverted Displacement mutation: large mutation probability
2-opt modification
Yes
Satisfy local search condition?
No
Island 1
Island 2
Island 3
(λ, μ) elimination
(λ, μ) elimination
(λ, μ) elimination

Figure 1: Scheme of implemented EA with Islands approach.

3.1 Representation

The candidate solutions are represented by path permutation. Each element of the permutation represents a city and the order of the elements of the permutation denote the order that the cities are visited in. Path representation is a natural and compact way to represent solutions. As it is widely used in the literature, many efficient genetic algorithms can be found for this type of representation. Since it is a permutation, invalid solutions that visit the same city multiple times are ruled out by default. The implementation is done using *NumPy* arrays since these are more optimized for large-scale operations and can use of benefits of *NumPy* package. The datatype of the *NumPy* array is *uint16*, which means it will use unsigned integers that are 16 bits long. This amount of bits is the minimum number of bits needed to represent sufficiently large numbers for the scale of problems we want to solve and implies optimization of the speed and memory usage of the EA.

3.2 Initialization

The population is initialized by taking random permutations of the number of cities. Assuming *permutation()* function from *NumPy* package is sufficiently random, our initialization will be sufficiently diverse, since it is completely randomized. For the chosen elimination strategy (λ, μ), usually the ratio $\frac{\mu}{\lambda} \approx 5$ is maintained. As it is discussed in subsection 3.11, I implemented the grid cross procedure to obtain the most suitable λ, as well as population size μ. As a rule of thumb, the initial population should be diverse as much as possible, but the randomly generated population is far from being good quality.

I experimented by applying the Nearest Neighbours (NN) heuristics during the initialization phase according to the algorithm in [9], where only half or fewer individuals are improved with the NN. The results for the benchmark problem $194.csv$ can be summarized as follows: there is a significant decrease in the convergence time with no positive effect on the fitness. However, as we seek to create efficient EA for problems up to 1000 cities, the application of NN during the initialization phase significantly decreases convergence time for the largest problems. As it will be shown in section 4, the implemented EA rarely converges for the largest benchmark problem and it is important to achieve the best possible solution within the limit of 5 minutes. Thus, NN application during the initialization plays an important role in achieving this goal.

On the other hand, there is a significant drawback of the application of NN in initialization - loss of the diversity in the initial population and the EA might quickly get stuck in a local minimum. One of the reasons can be the NN algorithm itself - the same tour will be constructed for the same starting point and there is a possibility that tours will be very similar for different starting points[9]. Also, other steps of EA discussed below in combination with NN might further increase the chances of getting stuck in a local minimum. To mitigate the early convergence issue, an Islands-based diversity promotion scheme was implemented where the 1 sub-population has a higher mutation rate than other sub-populations. Also, only $1/10$ of all initial population was improved with the NN algorithm to reduce the diversity loss and maintain randomness.

3.3 Selection operators

Selection should return high-fitness individuals more often than low-fitness individuals while preserving diversity. K-tournament selection operator offers a tool to adjust selection pressure - if k is relatively large, the selection will be biased towards individuals with the highest fitness value and the operator yields a random selection if $k = 1$. Contrary to alternatives such as roulette wheel selection, a low-fitness individual has a relatively large chance to be selected due to a random choice for participation in a tournament if k is small. Thus, k-tournament selection (without replacement) was a straightforward choice for the selection operator and the best k was determined experimentally as discussed in subsection 3.11.

3.4 Mutation operators

During the group phase of the project, we considered the swap and inversion mutations. We selected the inversion mutation as it introduces more randomness. This operator forms a basis for considering TSP so we can create all individuals by using a sequence of inversion mutations. In fact, this is a basis for *k-opt* local search operator discussed in subsection 3.7. By combining inversion and displacement mutation operators, an advanced mutation operator can be created - inverted displacement mutation[7]. This operator randomly selects two indices and inverts the sub-path in between them. During the second step, the sub-path is inserted to a randomly chosen position. Based on a study of mutation operators for TSP[7], both, inversion and inverted displacement, mutation operators outperform many other operators such as exchange, displacement, or inverted exchange operator. As implemented EA is based on an island model, both mutation operators are used for different sub-populations as discussed in subsection 3.8. I have also experimented with the insertion operator, but I did not experience major improvements. The proposed EA does not use self-adaptivity to control mutation probability. Instead, adaptive mutation control is implemented - every sub-population starts with 0.9 mutation probability, which linearly decreases. The final mutation probabilities for all sub-populations are as follows: first and second island - 0.05, third island - 0.15. The number of iterations to reach the lowest mut. probability differs based on the size of the problem as in table 1.

3.5 Recombination operators

During the group phase of the project, we considered several crossover operators such as cycle, edge, or order crossovers. According to the study of various crossover operators for TSP[5], the order crossover is the best choice from non-heuristic based crossover operators. The order crossover works as follows: in the first phase, part of one of the parents is randomly chosen. This part will be directly copied to the offspring. In the second phase - missing elements are copied based on the order they appear in the second parent. As a consequence, two children are created by repeating the process with the exchange of the roles of the parents. If the parent is a high-fitness individual, probably each of its sub-paths is also high quality and this information can be preserved during the first phase. The second stage preserves many edges from the second parent - it preserves at least edges that did not occur in the copied part of the parent to guarantee some balance between properties from each parent in the offspring. Just a few of all edges in both of the offspring does not exist in any of the parents.

As a promising alternative taking into account the cost matrix while making decisions, Sequential constructive crossover (SCX) can be suggested [3]. SCX starts by selecting the first gene in the first parent as the starting point and sequentially searching both of the parents for the *not yet visited* node. If this node does not exist, sequentially search in the list of ordered *not yet visited* nodes to obtain the node. When nodes from both parents are selected, compare edge costs between two parents and select the one with the lower cost. The process is repeated until full offspring is created. As SCX constructs offspring by sequentially selecting only the best edges, a significant improvement in the convergence rate can be expected. Not only SCX maintains many edges from any of the parents, but it can also induce new edges that might be better than the ones from the parents. Note that SCX preserves all common edges between parents. If there is little overlap between both of the parents, the number of newly imposed edges will be larger, but the offspring will probably have higher fitness.

It was shown that for the VRP problem, which is similar to TSP, heuristic-based crossover operator (HGreX) is the best among other popular operators [8]. In fact, HGreX is very similar to SCX operator, but in a case of non-existing sequential *not yet visited* node, it selects the best node from all available nodes not taking into account nodes location. Thus, even higher fitness offspring can be expected.

Each of the crossover operators discussed above is used for different EA strategies in the 3-islands approach as shown in the scheme 1. I have to point out that order crossover adds additional randomness compared to the SCX and HGreX operators - new relatively expensive routes will occasionally be created, which with further application of local search operator occasionally leads to a great solution. However, for the same reason, convergence becomes an extremely tough task - there is a significant difference between mean and best-fitness in a case of order crossover, especially around the minimum point.

Instead of self-adaptivity, an adaptive recombination strategy was implemented following approach defined in [6],[11], where f is the minimum fitness among two parents of interest and P_1 and P_2 are probabilities to be defined experimentally in subsection 3.11:

$$P_c = \begin{cases} P_1 - \frac{(P_1 - P_2)(f - f_{min})}{f_{avg} - f_{\min}}, & f < f_{avg} \\ P_2, & f \geq f_{avg} \end{cases} \tag{1}$$

The scheme ensures large recombination probability $(0.7 - 0.9)$ for $P_1 = 0.9$ and $P_2 = 0.75$, which changes according to the fitness feedback at each iteration. Note that probability is not restricted within the $[0, 1]$ interval - if the best fitness individual has a higher fitness score than any of the parents selected, the second factor will turn being negative and addition instead of subtraction will be performed, leading to a rec. probability > 1.

3.6 Elimination operators

The implementation is based on (λ, μ) elimination strategy. In the case of previously implemented $(\lambda + \mu)$, the population can be dominated by one individual which is much better than others leading to an early convergence of the EA. This is not the case for (λ, μ) elimination as individuals from the old population are not copied to new a new generation. Also, (λ, μ) strategy is the often preferred over the $(\lambda + \mu)$ strategy for the TSP[10].

3.7 Local search operators

As shown in figure 1, two local search operators were implemented - Nearest neighbour and $2-opt$. NN improves individual as follows: starting from the first city in path representation, the algorithm searches for the nearest *not yet visited* vertex. The algorithm continues until all vertices are visited. Application of NN during the initialization is discussed in subsection 3.2.

$2 - opt$ is a more sophisticated heuristic to improve individuals found by EA. The implemented $2 - opt$ works as follows: iteratively select two edges and replace them with the other two edges if the distance in between the selected vertices is reduced. Otherwise, select two other vertices. A straightforward implementation utilizes two for-loops where all edge combinations have to be considered. In the case of symmetric TSP, we can save half of the iterations by not considering the edge pair (20,5) when pair (5,20) has been already considered. Another speed-improvement technique which I implemented is *Don't look bits* - chances to find an improving move in current iteration are small if no improving change was found previously and vertex neighbors haven't changed. My implementation is based on an approach discussed in [1]. Every vertex has its own flag which is initially turned off. If the search for improvement starting at a specific vertex fails, then the corresponding flag is turned on. The flag can be turned off if this vertex is a successful endpoint of the exchange edges. Moreover, all vertices with *don't look bits* turned off are skipped when considering candidates for the improvement change. Finally, *Fixed radius search* improvement was implemented[2], which is based on the fact that inequity for $2 - opt$ move improvement: $d(x_1, y_1) + d(x_2, y_2) < d(x_1, x_2) + d(y_1, y_2)$ holds if at least one of the two conditions is satisfied: $d(x_2, y_2) < d(x_1, x_2)$; or $d(x_1, y_1) < d(y_1, y_2)$.

To illustrate the performance improvements, I run several experiments of the EA illustrated in fig. 1 with the straightforward implementation of the $2 - opt$ and $2 - opt$ with the *Don't look bits* and *Fixed radius search* modifications. On average of 5 independent runs on $194.csv$ problem, a straightforward implementation of $2 - opt$

improves 3.51 individuals per second and $2 - opt$ with modifications 11.03 individuals per second. Of course, there is a trade-off as individuals improved by the straightforward $2 - opt$ have higher-fitness values, but the modified approach also performs well and leave room for EA to evolve. The improvement of LSO for convergence speed is of the highest importance, especially for the largest problem. Note that the local search operator starts only after 3 iterations from the start of the migration and improves all individuals in the chosen island.

3.8 Diversity promotion mechanisms

To promote diversity, the islands-based EA approach was chosen as shown in fig. 1, where 3 sub-populations undergo different EA until the merging phase starts. Three different recombination and two different mutation operators were chosen for the evolution of sub-populations. Island 1 and Island 2 have a mutation rate that quickly decreases to 0.05, but Island 3 has a constant 0.7 mutation rate, which only after the first $15 - 30$ (depending on the problem size) iterations decreases to 0.15. Thus, Island 3 is responsible for exploration, and there is a large probability that good solutions will be transferred to other islands with much smaller mutation rates and will be preserved or improved. The migration stage consists of:

1. a random selection of 2 sub-populations for exchanging individuals;
2. a k-tournament selection (with $k = 3$) for the selection of individuals to migrate (without replacement). As the number of offspring is relatively small, $k = 3$ maintains the balance between low and high-fitness individuals to be selected for migration.

I spent a lot of time experimenting with the application of fitness sharing during the selection before the local search operator starts. Neither Euclidean nor Hamming distance are suitable to compare 2 tours for the path representation, thus I considered Manhattan and correlation-based Kendall Tau distances. At first, I wished to distinguish two scenarios (single index shift and path reversal) from entirely different representation and correlation-based metric seemed to be a great choice. But I concentrated only on a Manhattan distance as the population is random at first and the difference of a reversed path and the path itself is not big, even though, the distance matrix is asymmetric. I did not include fitness-sharing approach to the final scheme as it significantly slowed down the execution and did not provide better solutions than the EA with 3-islands in fig. 1.

3.9 Stopping criterion

There are two main stages of the convergence in the implemented EA - before the migration starts and after:

1. at first, at least one sub-population should try to almost converge without the interaction between sub-populations. The evolution is maintained until a number of iterations of the evolutionary cycle exceed a given limit (12) or the difference between the best and the average fitness is less than a given fraction of the fitness of the average individual. In mathematical form: $f(x_{best}) - f(x_{average})| < tol * f(x_{average})$. To check the relative improvement within different generations, the average of best-fitness over multiple populations was used with $tol = 10^{-3}$. If any sub-population satisfies stopping criterion 1, migration starts.
2. all sub-populations should try to converge. The previous convergence criterion is restarted with $tol = 10^{-5}$. I noticed a significant difference between mean and best-fitness for each iteration (especially, for the island 1 with the order crossover), and the stopping condition was extended by the condition taking into account only the current iteration - the difference between the best-fitness and mean-fitness of all populations should be less than 5% of the mean-fitness value. Thus, for the second stage relative improvement, maximum number of iterations and best-mean fitness difference convergence conditions are combined.

3.10 The main loop

The main loop of the evolutionary algorithm is visualized in scheme 1. The algorithm starts by random initialization followed by the NN application for every 10^{th} individual. Each of the sub-population evolve differently:

- Island 1 - order crossover, inverted displacement mutation;
- Island 2 - SCX crossover, inversion mutation;
- Island 3 - HGreX crossover, inverted displacement mutation.

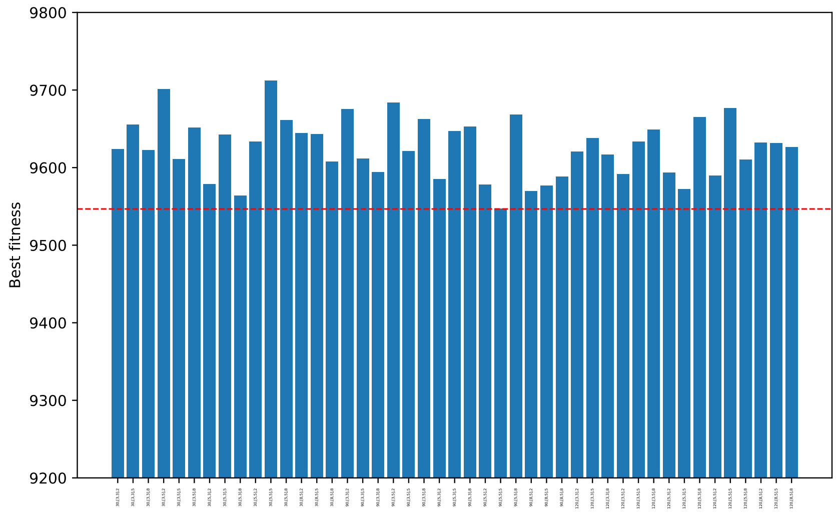
The evolution is continued until the first stopping criterion is satisfied - at least one subpopulation is about to converge or the maximum number of iterations before the start of the merging phase has been reached. Then, the migration phase is added to the EA, where only 2 randomly selected islands participate. The individuals for migration are selected by the k-tournament selection ($k = 3$). After 3 iterations from the start of the migration phase, the local search is also added to the EA, where all individuals of the randomly selected islands undergo improvements by the $2 - opt$ operator. The main loop continues until the second stopping criterion is satisfied.

3.11 Parameter selection

$2 - opt$ is the main bottleneck for the largest benchmark problem. To obtain satisfactory results within the limit of 5 minutes, all parameters discussed below are defined by the problem size and values for different problem sizes are discussed in the subsection 4.1. Parameters not defined by adaptivity are as follows:

- k_1 for k-tournament selection in selection of parents before the start of migration.
- k_2 for k-tournament selection in selection of parents after the start of migration.
- offspring size λ: evolution of the EA with large λ requires a significant amount of computation time. Also, λ defines other parameters such as $\mu = 5 \cdot \lambda$ and offspring size for each of the sub-populations $\lambda_i = \lambda/3$.
- number of elements to swap between sub-populations during the merging stage.

The extensive grid-search was deployed on the *194.csv* benchmark problem to find the best configuration of the parameters discussed above. As can be seen of the averaged best-fitness values over 3 independent runs (left fig. 2), the highest-fitness solution was obtained when $k_1 = 5$ and $k_2 = 5$, $\lambda = 90$ and 5 elements were transferred in-between different sub-populations. With the chosen parameter values, same iterative search procedure for the best pair of recombination parameters P_1 and P_2 in eq. 1 was performed (right fig. 2). Even though the best fitness score was obtained for $P_1 = 0.75$ and $P_2 = 0.4$, article [11] suggests the use of $P_1 = 0.9$ and $P_2 = 0.6$. Also, the rule of thumb is as follows - crossover rates should not be lower than 0.6 [4]. Thus, the proposed EA uses $P_1 = 0.9$ and $P_2 = 0.6$ recombination parameters.



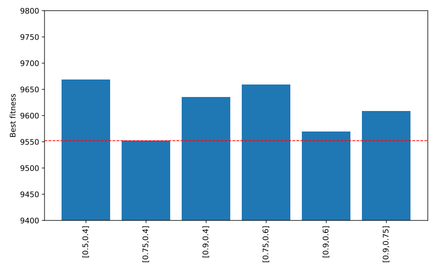


Figure 2: Averaged grid-search results on the *194.csv* problem (3 runs). Left: results for different parameter configurations. Parameters are: offspring size λ, k_1 and k_2 and number of elements to move in-between sub-populations. Right: results for the different configurations of recombination parameters p_1 and p_2 in eq. 1.

4 Numerical experiments

4.1 Metadata

The specs of the computer on which all timings in this report were measured: 2.7 GHz Dual-Core Intel Core i5 with 8GB of RAM. The python version 3.8 was used.

Problem specific parameters are summarized and defined in table 1. These include offspring number λ, population size $\mu = 5 \cdot \lambda$, number of islands to improve with the $2-opt$, number of individuals to migrate between sub-populations, and number of iterations when mutation probabilities will take it's smallest value for different sub-populations. With the increasing problem size, the $2-opt$ operator requires more computation time, thus it is important to modify parameters that solutions close to optimal ones can be found within 5 minutes.

Problem size	λ	$2-opt$ islands	migration ind.	mutation iterations
$n < 225$	90	2	5	15, 15, 30
$225 <= n < 330$	48	2	4	14, 14, 22
$330 <= n < 400$	42	2	4	12, 12, 18
$400 <= n < 500$	48	1	4	12, 12, 20
$500 <= n < 650$	36	1	4	12, 12, 20
$650 <= n < 800$	30	1	4	12, 12, 20
$800 <= n < ...$	30	1	3	10, 10, 13

Table 1: Summary of parameters not determined by adaptive control: number of different islands to improve with $2-opt$, number of individuals to migrate between sub-populations and iterations required for mutation probabilities take its lowest possible values for all 3 islands.

4.2 tour29.csv

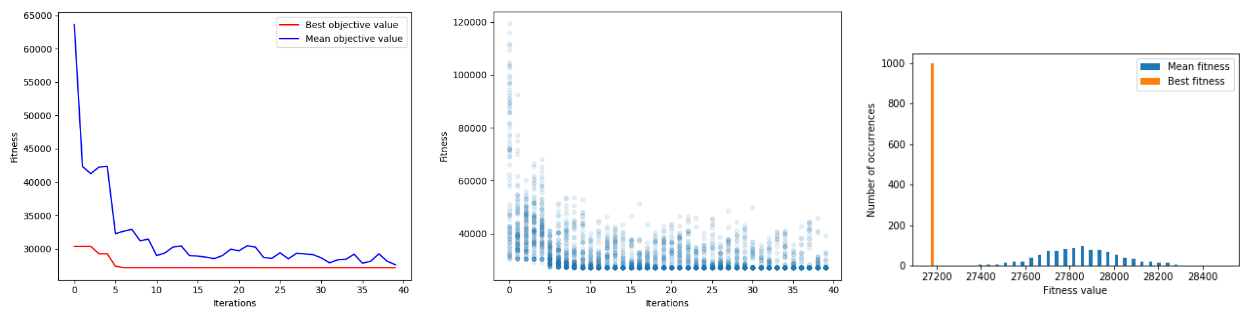


Figure 3: Numerical results for the benchmark *29.csv* problem: the convergence of the best and mean objective values (left), the diversity of the population during different generations (middle) and the histogram of mean and best fitness over 1000 runs (right). Greedy heuristics value is 30350.

Figure 3 shows a convergence graph of the algorithm performed on the benchmark dataset with 29 cities. We can see that EA required 38 iterations to converge and the best-fitness solution is even better than the optimal one (27200). However, this value is achieved in less than 6 iterations, but the population has not converged for more than 30 additional iterations as the mean fitness show large variations. These variations can be explained by several factors:

- island 3 has a high mutation rate for the first 30 iterations;
- order crossover used in island 1 adds additional randomness compared to crossover operators in other islands as new more expensive routes are created without taking into account cost matrix.

Fitness value better than greedy heuristic fitness is achieved from the start of the EA. This is because of the NN application during the initialization phase. As the migration phase starts from the 4^{th} iteration, the convergence of the mean fitness is shortly interrupted during the interval of $3-5$ iterations and continued since the 6^{th} iteration as $2-opt$ is added to the EA. Even though NN during initialization was applied, an extremely diverse population is still maintained after the end of the first iteration. As the algorithm progresses, the diversity of the algorithm decreases - more and more individuals have the best fitness score. After running 1000 independent runs (fig. 3), same best-fitness of 27154.488 solution was obtained for all experiments. Even though there is a large variability of the mean fitness value of the population, there is no variability of best-fitness value for all 1000 experiments.

The statistics for the mean fitness value can be summarized as: min = 27320.427, max = 28500.674, mean = 27863.097, std = 179.48. Note that the maximum mean fitness value is less than 5% from the best fitness value as required by the second stopping criterion. The averaged time to converge for 1000 experiments was 7.821 seconds and the python process on average requires 88.90625 MB of memory. The best solution is:

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

4.3 tour100.csv

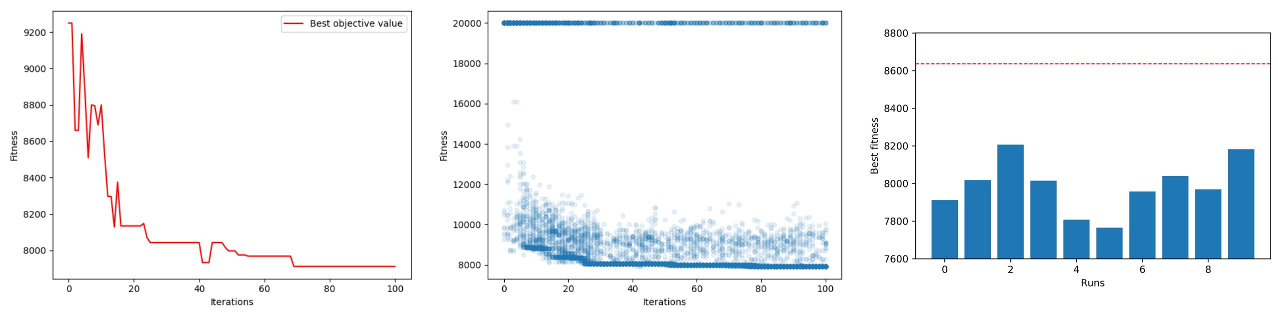


Figure 4: Numerical results for the benchmark *100.csv* problem: the convergence of the best and mean objective values (left), the diversity of the population during different generations (middle) and the variance in results over 10 runs (right). Greedy heuristics value is 8636.5 (red dashed line). Note that *inf* edges were represented with the value of 20000 in the plot of diversity.

Benchmark problem *100.csv* contains many (25%) disconnected cities. Such edges are assigned infinity cost value, thus should be removed within the first iterations of the EA. From the diversity plot in figure 4, it is clear that the amount of edges with inf cost value decreases as EA evolves, however they are never removed from the population entirely. Also, population diversity around the best fitness value stagnates and a similar standard deviation is maintained (excluding inf edges). This imposes that the best fitness can not be within 5% from the mean fitness value and the second convergence criterion will not be satisfied. The evolution of EA stops only when the maximum number of iterations has passed (100). This implies that EA is not capable to remove inf edges within the maximum number of iterations (100) and more iterations are required. However, note that the best fitness score is constant for the last 30 iterations, thus the convergence criterion based on the relative improvements is satisfied.

Why inf edges are not entirely removed from the population within the first iterations? I experimented by adding flags for crossover and mutation operators to set offspring as one of the parents if it contains inf edges. However, as the benchmark problem contains a lot of these edges, the diversity of the population is drastically decreased and early convergence close to the greedy heuristic value is obtained. As order crossover does not take into account cost and the amount of inf edges is significant, inf edges will be produced within further iterations of the EA. Best fitness scores for 10 independent runs are shown in right fig. 3. The best tour length I found is 7765.145 and the python process, on average, required 89.546875 MB.

4.4 tour194.csv

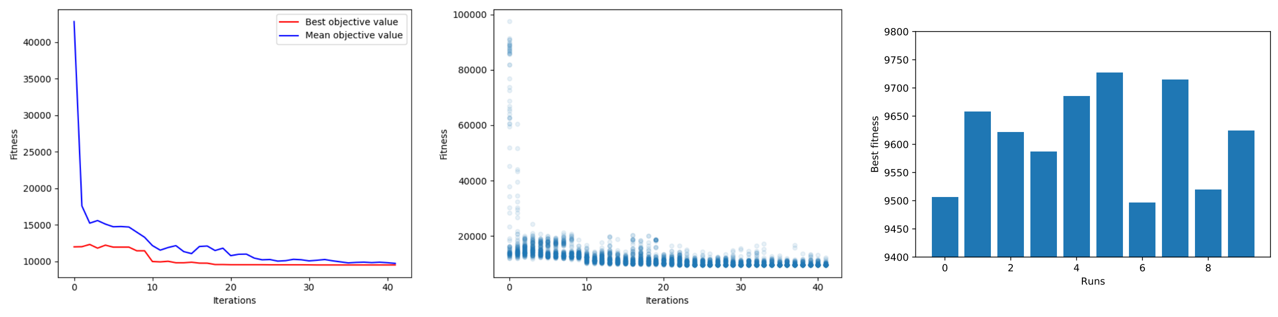


Figure 5: Numerical results for the benchmark *194.csv* problem: the convergence of the best and mean objective values (left), the diversity of the population during different generations (middle) and the variance in results over 10 runs (right). Greedy heuristics value is 11385.01

Convergence and diversity reduction graphs are shown in figure 5. As parameter selection was conducted based on this benchmark problem, the convergence graph is the most convincing from all benchmark problems tested - the mean fitness value steadily converges to the best fitness as diversity among individuals iteratively decreases. Because of the NN application in the initialization, best fitness is approximately equal to the greedy heuristic value up to the point when local search operator $2-opt$ is added to the EA (8 iterations). Note that mean fitness exhibits many fluctuations until the mutation rate for the island 3 is set to the smallest possible value (0.15) and mean fitness can finally approach the best fitness - EA converged. Best fitness scores for 10 independent runs are shown in fig. 5 (right). The best tour length is 9496.973 and the python process, on average, required 90.93359375 MB of memory and 239.0715 seconds.

4.5 tour929.csv

Convergence and diversity reduction graphs are shown in figure 6. Note that parameters according to the table 1 were used. The situation is similar to other cases - the best fitness individual is a better solution than the one obtained by simple greedy heuristics from the end of the first iteration. From the mean fitness curve in the convergence graph, it is clear that merging starts after 3 iterations and is quickly followed by the application of $2-opt$. Also, the diversity graph indicates extensive diversity at the beginning of the EA and substantial reduction before the convergence. Even though the case in fig. 6 converged before exceeding the 5 minutes limit, it is a rare case and 8 out of 10 runs (right fig. 6) did not converge before the end of a time limit. Best fitness scores for 10 independent runs are shown in right fig. 6. The best tour length is 100562.459 and the python process, on average, required 106.859375 MB of memory and 289.0715 seconds to converge.

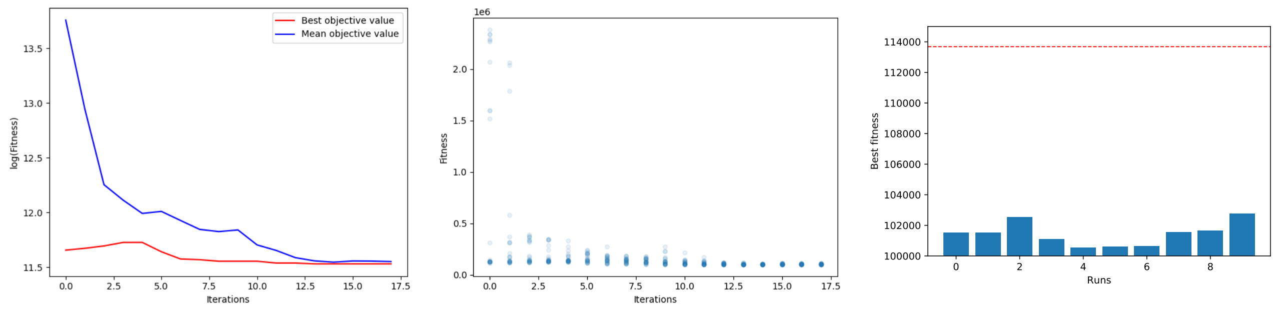


Figure 6: Numerical results for the benchmark *929.csv* problem: the convergence of the best and mean objective values (left), the diversity of the population during different generations (middle) and the variance in results over 10 runs (right). Greedy heuristics value is 113683 (red dashed line).

5 Critical reflection

5.1 Discussion on the proposed EA

Main advantages (+) and disadvantages (−) of the proposed EA are:

+ algorithm is able to obtain solutions close to the optimal ones;
+ algorithm obtains solutions with a small standard deviation;
+ amount of memory required is problem size-dependent and scalable;
+ population diversity is maintained for a large number of iterations;
+ application of $2-opt$ significantly improves the quality of the solutions;
− exploration-exploitation balance could be improved;
− algorithm is not capable to remove disconnected cities until the convergence;
− many parameters have to be adjusted based on the size of the problem;
− because of the diversity promotion mechanisms, it is hard task for the EA to converge;
− algorithm rarely converges for the largest benchmark problem.

The following section discusses the results, advantages and disadvantages of the proposed EA:

- as expected, a bigger dataset requires more memory, but the increase seems very scalable - 88.90625 MB are required for the smallest problem and 106.859375 MB for the largest benchmark problem.
- even though for all benchmark problems, the obtained solutions surpassed the greedy heuristic solutions, only for the smallest benchmark problem *29.csv* solution better than the optimal one was obtained. Even

though the obtained solutions for other problems were only close to being optimal ones, the proposed EA provides solutions with a relatively small standard deviation!

- from the diversity graphs, it is clear that the chosen 3-islands based approach maintains population diversity for a large number of iterations and avoids getting stuck in a local minimum even though the $2-opt$ operator is also added to the EA. This can be explained by the chosen strategy of EA - island 3 holds large mutation rate, different recombination and mutation operators. However, because of this diversity, convergence of all population to a region close to a single individual is a tough task. Application of $2-opt$ on all population can solve this issue, but even with the DLB and Fixed radius search modifications this would require too much computation time, also other parameters such as μ and λ should be modified as well.
- in a case of many disconnected cities, EA is not able to remove disconnected cities entirely from the population until the maximum number of iterations is reached. As discussed in section 4, this is because the implementation of $2-opt$, mutation and recombination operators. Thus, modifications of these stages in the EA could lead to better results in a future.
- from the convergence graphs it is clear that sub-populations quickly converge to the local minimum within each sub-population, and migration phase starts soon (often within the first $4-7$ iterations). This is because the relative improvement of the best fitness stagnates due to the application of NN in initialization phase even though sub-populations are still evolving (mean-fitness quickly decreases). Thus, soon from the start of the EA, the stopping criterion $\#1$ is satisfied and migration starts. However, it might too early and alternative convergence criterion to define the start of migration phase could lead to better results.

5.2 General comments

This project with its corresponding literature review have confirmed that prevailing research practice to tackle TSP by the evolutionary algorithm is correct. Moreover, the proposed EA has reached solutions close to the optimal ones. However, note that EA are not designed to find local minimizers cost-effectively. Instead, EA provides ability to explore the promising areas of the search domain and, in a case when local minimizer is required, local optimization methods should be used independently or added to the EA.

EA offers the ability to easily adjust it towards the problem of interest. Not only EA consists of multiple adjustable segments such as mutation or recombination, but also extensive research of EA for various applications has been conducted. Hence different problems will require different EA strategies to obtain satisfactory results and the point of EA being adjustable clearly has it's own advantages and disadvantages. For example, an adverse EA strategy can easily lead to a local minimum and should be avoided. As I have shown in this project by switching towards islands-based approach, the main strategy of EA can be easily modified and it contains many operators and parameters which can be adjusted to provide the sophisticated results. However, this requires extensive amount of experiments and literature study.

This project showed how important it is to not only think about straightforward implementation of the algorithm, which might take ages to run, but try to figure out how to improve it so that it would take only 5 minutes! I have considered evolutionary computing or genetic algorithms as math-related buzz words before starting this course. This course and project equipped me with the ability to read the literature not only about EA itself, but optimization as a subject, relate it towards many applications.

6 Other comments

No other comments.

References

[1] Don't look bits (dlb) – general idea. http://tsp-basics.blogspot.com/2017/03/using-dont-look-bits-dlb-2-opt.html. Accessed: 2020-12-10.

[2] Fixed radius search – general idea. http://tsp-basics.blogspot.com/2017/03/using-simple-fixed-radius-search-2-opt.html. Accessed: 2020-12-10.

[3] Zakir H Ahmed. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. 2015.

[4] Endre Eiben, Robert Hinterding, and Zbigniew Michalewicz. Parameter Control in Evolutionary Algorithms. 3(2):124–141, 1999.

[5] Paulo R Lafeta Ferreira. *Genetic Algorithm with Multiple Crossovers on the Traveling Salesman Problem*.

[6] Michal Gregor and Juraj Spalek. *Fitness-based Adaptive Control of Parameters in Genetic Programming*.

[7] Deep Kusum and Mebrahtu Hadush. Combined mutation operators of genetic algorithm for the travelling salesman problem. *IJCOPI*, 2, 2011.

[8] Robert Manger. Comparison of eight evolutionary crossover operators for the. 18:359–375, 2013.

[9] Goran Martinovic and Drazen Bajer. Impact of nna implementation on ga performance for the tsp. 2012.

[10] Adam Slowik and Halina Kwasnicka. Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, 32(16):12363–12379, 2020.

[11] Mingji Xu, Sheng Li, and Jian Guo. Optimization of Multiple Traveling Salesman Problem Based on Simulated Annealing Genetic Algorithm. 25, 2017.