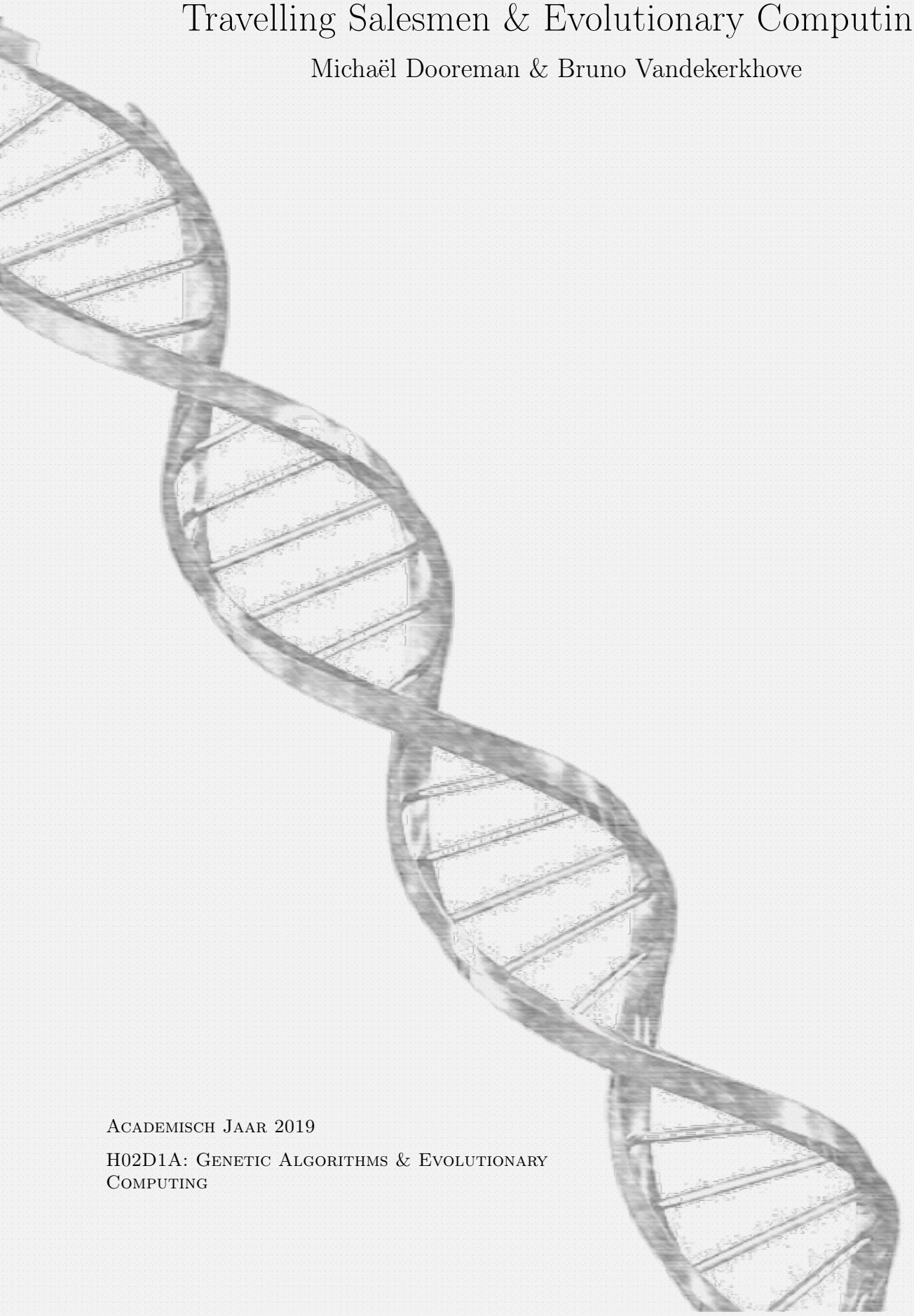
Travelling Salesmen & Evolutionary Computing

Michaël Dooreman & Bruno Vandekerkhove



Academisch Jaar 2019

H02D1A: Genetic Algorithms & Evolutionary Computing

Contents

Based on a provided toolbox for solving the travelling salesman problem (TSP), some parameter tuning with the default operators, selection methods, ... was performed. Then, new tour representations and accompanying operators were implemented. Local heuristics, other selection methods, the island model, adaptive parameter tuning and a seeding technique were also experimented with. The results are outlined below. Pointers to relevant source code files are provided next to section headings (all code is written in `MATLAB`*).*

Existing Genetic Algorithm `/experiments/*.m`

To get an initial feel for what may or may not be worth it, 375 configurations (table 1) were experimented with for a total of 15,000 runs. This made it possible to draw some initial conclusions and perform some increasingly targeted experiments afterwards. All of the experiments outlined throughout this report were automated, with occasional use of parallelism (using `MATLAB`'s parallel computing toolbox).

Population Size	*Crossover Rate (%)*	*Mutation Rate (%)*	*Elitism (%)*
[150,300,600]	[10,25,50,75,95]	[5,25,50,75,95]	[1,5,10]

Table 1: Parameter ranges for the initial experiments. Each configuration was run 10 times on 4 different datasets (with 25, 41, 70 and 127 cities). No stopping criterion was applied, and each run lasted 250 generations. For all configurations with population size set to 150 runs with 500 generations were also experimented with. Results are visualised in tables 10 to 14.

After selecting 4 datasets (with an increasing amount of cities) each of the resulting algorithms were ran 10 times. The resulting (best, average and worst) tour lengths and runtimes were recorded and averaged. The best tour found by each was also recorded, since the very point of the travelling salesman problem is to find the smallest possible tour. Diversity of solutions is only of indirect benefit as it increases the probability of reaching a global optimum. The minimum tour length found by every configuration is visualised in tables 10 to 14. It is calibrated against the minimum and maximum found by any of the 375 configurations (darker cells in the table indicate better solutions).

Aside from the obvious conclusion that the first benchmark is a tad too easy to resolve, two observations could be made ; larger population sizes improved results and lead to longer runtimes, and setting the rate of elitism too low is contra-productive. The first is not surprising since a larger population may increase diversity and the probability of finding a better local (or even global) minimum. At its worst it introduces redundancy and needlessly increases runtime. As for the second conclusion, retaining a reasonable proportion of the best individuals looks like a safe bet. Higher levels of elitism ($> 10\%$) proved to be somewhat less fruitful. While better individuals may appear attractive to the greedy, some of them probably represent second-hand solutions

(those that haven't been mutated to one of the better individuals yet, or better individuals that have been unsuccessfully mutated) that may safely be disregarded.

The effect of population size was suspected - at least partially - to be due to increased speed of convergence. This was tested by running all configurations with population size of 150 for a larger number of generations which confirmed the suspicion (table 13). Therefor, future experiments use a reasonably large population size (300), fair rate of elitism (5-20%) and an appropriate stopping criterion (or the number of generations is set reasonably high).

One slightly more surprising result is that higher mutation - and crossover rates appeared to worsen results. High rates of mutation are too disruptive and the default crossover operator (alternating edges) preserves few edges from the parent. In subsequent experiments, only rates of 5% and 95% were abandoned.

It can be noted that just 10 runs and 250 generations is hardly enough (from a statistical point of view). However, only soft conclusions were drawn and limited computing power and time was available. Subsequent experiments were usually run 30 times per configuration and tend to make use of a stopping criterion (in combination with a high maximum number of generations) to avoid prematurely stopping a run. While it would be preferable no non-parametric tests were applied. Additionally, while the number of values that were tried per parameter isn't very high, testing out more values would make for a combinatorial explosion. So only a small part of the utility landscape was explored. For a better analysis iterative methods could be used rather than this basic GENERATE-and-TEST approach.

Stopping Criterion

`/stop_criteria/*.m`

The following efficiency function E is used to calculate the algorithm's efficiency after each generation:

$$E(T) = \frac{1}{T} \sum_{t=0}^{T} F_i \tag{1}$$

where T is the index of the current generation and F_i the best fitness value of generation i. This efficiency function basically calculates the average best fitness over all generations. Since the total distance is used as the fitness of an individual and the goal is to minimize this distance, $E(t)$ is a monotonically decreasing function (assuming elitism is used).

The stopping criterion keeps track of the efficiency of the algorithm. When the relative standard deviation of the last W efficiency values (window) drops below a certain percentage Δ, the algorithm stops. Δ was chosen to be 0.5%. This stop criterion seems suitable, since the goal is to avoid useless generations where little to no improvement of the solution happens.

The window size W, being the only parameter not yet fixed, influences the quality of the solution and the convergence speed (in generations) of the algorithm. A small W will make the algorithm terminate sooner, possibly missing out on better solutions and decreasing the solution quality. A large W will make the algorithm terminate later, possibly increasing the solution quality but also allowing more useless generations.

An optimal value for W was determined. The algorithm was run 20 times on each small benchmark (`rondrit<...>, belgiumtour, xqf131`) with 500 generations (induvuduals = 50, elitism = 0.05, P_cross = 0.7, P_mut = 0.55, crossover = xalt edges, representation = adjacency, mutation = simple inversion, parent selection = linear rank, survivor selection = fitness based).

For each run, the evolution of the efficiency was observed to see what the longest streak without change greater than Δ was (excluding the last streak until termination of the run). The average streak was 30 with a standard deviation of 35.

Two window sizes were considered, namely 30 (the average) and 66 (the sum of the average and standard deviation plus one). The same setup as above was run, but with the stop criterion activated. Each time the stop criterion was met, the generation and fitness value were saved, but the run completed the full 500 generations, so the difference in quality could be observed. The results are shown in the table 2 and table 3 below.

As expected, the smaller window size ($W = 30$) results in faster termination and worse quality, the bigger window size in better solution quality and slower termination.

	No stop criterion			*Stop criterion (W = 30)*			
Route	# gen	avg best	std best	# gen	avg best	std best	quality decrease(%)
rondrit016	500	1484.877	12.547	142.35	1514.335	33.365	1.945
rondrit018	500	1202.425	39.508	199.7	1233.302	69.093	2.504
rondrit023	500	832.020	26.397	280.650	870.636	45.998	4.435
rondrit025	500	41.407	1.272	311.75	43.251	2.112	4.262
rondrit048	500	4677.841	262.778	423.45	4938.954	472.199	5.287
rondrit050	500	8.754	0.521	414.55	9.249	0.965	5.3539
rondrit051	500	625.062	21.971	373.75	672.535	57.825	7.059
rondrit067	500	4241.510	290.263	396.95	4649.488	626.395	8.775
rondrit070	500	1334.162	58.237	422.1	1420.047	121.276	6.048
rondrit100	500	21.866	0.796	356.85	24.537	1.531	10.887
rondrit127	500	317.718	10.040	251.45	371.896	29.412	14.568
belgiumtour	500	877.212	45.774	391.25	926.090	82.716	5.278
xqf131	500	1943.598	76.138	339.4	2192.587	156.261	11.356

Table 2: Each benchmark solved 20 times with a a max generation of 500. This table shows the amount of generations, average best fitness and standard deviation at the moment the stop criterion (W = 30) fired and the moment the algorithm finished the max generation. The quality is compared.

	No stop criterion			*Stop criterion (W = 66)*			
Route	# gen	avg best	std best	# gen	avg best	std best	quality decrease(%)
rondrit016	500	1497.412	20.705	237.55	1514.721	29.710	1.143
rondrit018	500	1184.392	23.386	339.75	1188.051	32.231	0.308
rondrit023	500	841.468	27.097	477.3	842.007	26.667	6.398E-2
rondrit025	500	40.990	0.719	492.8	41.273	1.328	0.687
rondrit048	500	4746.863	265.478	500	4746.863	265.478	0
rondrit050	500	8.806	0.420	500	8.806	0.420	0
rondrit051	500	608.155	29.390	500	608.155	29.390	0
rondrit067	500	4271.311	268.417	500	4271.311	268.417	0
rondrit070	500	1321.352	81.164	500	1321.352	81.164	0
rondrit100	500	21.797	0.651	500	21.797	0.651	0
rondrit127	500	315.640	11.969	500	315.640	11.969	0
belgiumtour	500	886.170	57.198	500	886.170	57.198	0
xqf131	500	1944.138	84.712	500	1944.138	84.713	0

Table 3: Each benchmark solved 20 times with a a max generation of 500. This table shows the amount of generations, average best fitness and standard deviation at the moment the stop criterion (W = 66) fired and the moment the algorithm finished the max generation. The quality is compared.

Another interesting observation is the quality decrease in function of the route size (number of cities). These larger routes are more complex and thus the algorithm converges slower. A small window size ($W = 30$) will let the algorithm terminate while the solution is still improving slowly but steadily. A larger window size ($W = 66$) will observe the slow progress and let the algorithm run longer. Taking the window size too large, might result in many generations with too little improvement, though.

These results show that the window size and this type of stop criterion in general, are problem specific. As a result, this criterion is only suitable for applications where only one problem or problems of similar complexity are to be solved.

Other Representation and Appropriate Operators `/crossovers/*.m`, `/mutations/*.m`

Aside from the adjacency representation, (at least) four alternative representations can be used. Two of those are binary which were avoided due to previous experiences in constraint programming where the use of binary representations for problems with integer domains is discouraged. They are the binary and matrix representations. The other two alternatives were both implemented. The ordinal representation because it allows for the use of the *'classic'* single-point crossover. The path representation because it is probably the most *'natural'* representation. For the latter a few crossover operators were implemented (inspired by [2] & [1]). Some of these were mentioned either in Eiben's book or in the course notes ; the *heuristic* - , *partially matched* -, *order* -, *cycle* -, *edge*

recombination - and *sequential constructive* crossovers in particular. Operators that weren't mentioned include :

- *Heuristic Edge Recombination (HERX)* : equivalent to the edge recombination crossover, but with some hybridisation in that the shortest edge is picked from the edge list of the current city, rather than the edge corresponding to the city with the shortest edge list.
- *Max Preservative (MPX)* : similar to the partially matched crossover. A subtour of the first parent is selected and remaining cities are *appended* to this sequence in the order that they appear in the other parent. The subtour's length needs to be within a specified interval.
- *Order Based (OX2)* : a few random cities are selected in a parent. These cities are removed from the other parent and replaced by the same cities, at the same locations, but in the order that they appear in the first parent.
- *Position Based (POS)* : this one also selects a few random cities but imposes their positions as well, such that the offspring is the same as the first parent with respect to the cities that were selected, after which the remaining cities are added to it in the order that they appear in the other parent.
- *Unnamed Heuristic (UHX)* : a city is selected randomly. Four edges connected to city are compared, the smallest is picked (if there are edges of equal length, one is picked at random). The city at the other side of this edge becomes the current city and the whole routine is repeated until all cities have been added to the offspring.

The following mutation operators were implemented :

- *Insertion* : a city is selected, removed and inserted at some other (random) point in the tour.
- *Displacement* : a subtour is selected, removed and inserted at some other point in the tour.
- *Inversion* : the 'simple' inversion was already implemented. It involves picking out two cities and inverting the subtour between them (which is what 2-opt does). Inversion adds to this by subsequently placing it somewhere else (like the displacement operator does).
- *Scramble* : selects a random subtour and shuffles the cities around. Clearly quite a disruptive operator.
- *Unnamed* : a hybridised operator, just like its accompanying crossover. A random city is selected and the subtour from this city to the one closest to it is reversed.

These operators were selected such that a general idea could be formed about them. Some are clearly less *'respectful'* than others, yet it wasn't entirely clear what to expect aside from what was reported in a fairly limited amount of studies which used other benchmarks. After some manual experiments (within the GUI that was extended for this purpose) further experimentation was done with all crossovers. From the configurations that were trialed before, 9 were selected and each of the crossover operators were tested in conjunction with those parameter values (table 4). The results are displayed in figures 4 to 7 (heatmaps might have been a preferable visualisation and can be generated from the `crossoverlatex` script, but they were a tad less interpretable). The average best tour length was visualised as well but not included in the report, as there was no indication that the shortest tours were a matter of luck.

As for the runtimes ; the edge recombination crossovers (both the classical and heuristic one) involve construction of an edge map which makes them the slowest in our list. The sequential crossover is a tad slower too, as is the classic single-point crossover (probably because the associated ordinal representation requires conversion to a path representation upon mutation). It's the latter, `SCX`, `HERX` and `UHX` operators who converge more rapidly. The last three are hybrid which makes this unsurprising.

Population Size	*Crossover Rate (%)*	*Mutation Rate (%)*	*Elitism (%)*	*Generations*
[300]	[25,50,70]	[25,50,70]	[5]	[1000]

Table 4: Parameter ranges for the experiments on crossover operators. Each configuration was run 10 times on 4 different datasets (with 70, 100, 127 and 131 cities). A stopping criterion was used to avoid useless iterations ; the algorithm stops if the best individual hasn't improved much in the past 100 generations. Results are visualised in figures 4 to 7.

After selection of the 4 more promising crossovers they were tested in combination with all of the 6 remaining mutation operators (all but simple inversion). The results are visualised in figures 8 and 9. To some extent an assumption is made that the performance of mutation operators can be gauged separately from the crossover operator in the sense that it wasn't tested if the other 9 crossover operators perform better than others when

combined with a particular mutation operator. Again, given the multitude of possible combinations some simplification is in order. Ideally a meta-algorithm would be used to automate the parameter tuning.

Population Size	*Crossover Rate (%)*	*Mutation Rate (%)*	*Elitism (%)*	*Generations*
[300]	[50]	[20,40,50,70]	[5]	[1000]

Table 5: Parameter ranges for the experiments on mutation operators. Each configuration was run for 4 different datasets (with 70, 100, 127 and 131 cities). A stopping criterion was used to avoid useless iterations ; the algorithm stops if the best individual hasn't improved much in the past 100 generations. Results are visualised in figures 8 to 9.

The insertion, simple inversion and reciprocal exchange mutations appeared to be a safe bet in this limited number of experiments. They're not particularly disruptive in comparison with, say, the scramble and displacement mutation operators. Performance of algorithms didn't differ much between runs ; the results for mean shortest tour length (rather than minimum shortest tour across runs) is about equivalent. Of the four mutation rates that were tested the middle 2 (40% and 50%) appeared to be appropriate for future use. The scramble mutation turned out to be the most unpredictable. Mutations can get one out of a local optimum or just improve the value of the best solution found so far, but getting towards the optimum is clearly still not guaranteed.

Local Optimisation

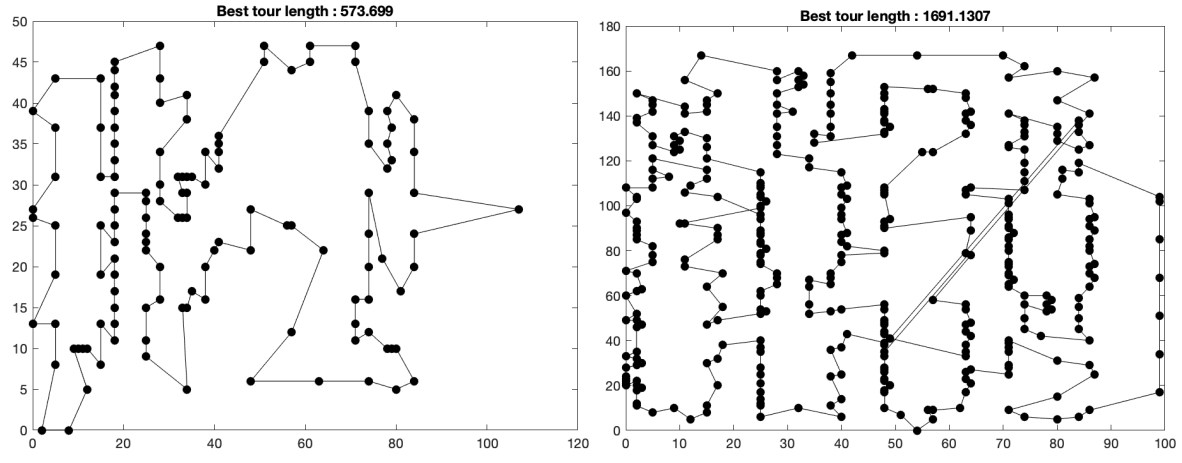
`/hybridisation/twoopt.m`, `/hybridisation/oropt.m`

In addition to the loop detection heuristic which was already part of the toolbox, two of the more simple heuristics were applied on some of the benchmarks. *Two-opt* and *Or-opt* in particular[1]. While not as powerful as the Lin-Kernighan heuristic (which even has some efficient implementations), they're easy to implement and provide good results. Their time complexity is $\mathcal{O}(N^2)$ (per tour). The experiments turned out to be fairly uninteresting in that apparently optimal results were found for each of the smaller tours. One of the better crossovers was tested in combination with these heuristics on `XQF131`. A near-optimal tour was found with just one erroneous sequence in the bottom left corner. The value of the heuristics is visualised more clearly in the experiments at the end of this report.

Or-opt is a restricted version of *3-opt* in which subtours of length 1, 2 or 3 are displaced. If the resulting tour is shorter it is picked for further processing. While *2-opt* makes the simple inversion mutation redundant, *Or-opt* obviously overlaps with some mutation operators as well. Results obtained with this heuristic were typically inferior to those obtained with *2-opt*.

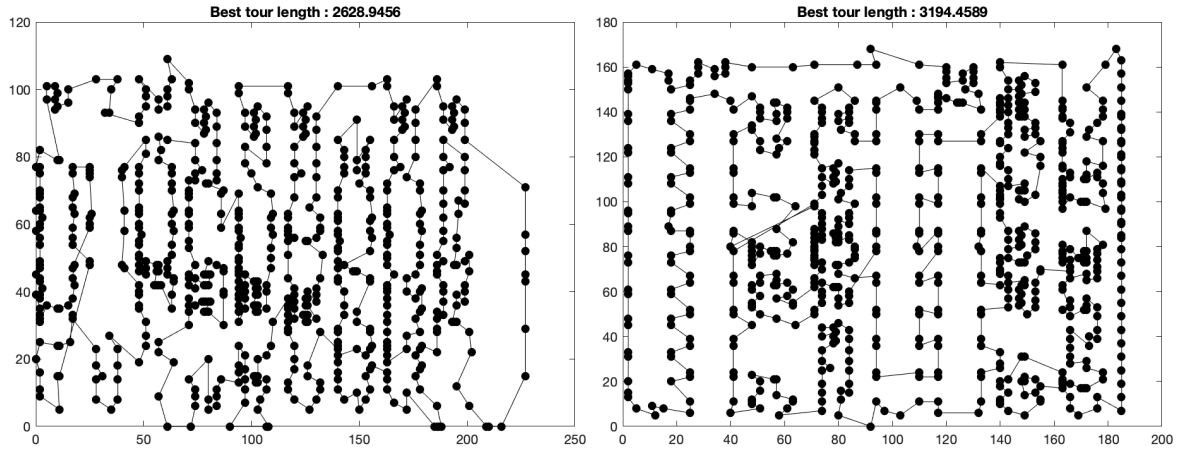
Benchmark Problems

Tests on benchmarks were run without any local heuristic (results where they *are* activated are visualised at the end of this report). Parameters used are listed in table 6. The results are displayed below.



(a) `xqf131` (b) `bcl380`

[1]There's some overlap between the *two-opt* operator and the local loop detection so it's somewhat useless to combine both.



(a) `xql662` (b) `rbx711`

Results on benchmarks. Error rates of 1.72%, 4.33%, 4.7% and 2.55% respectively.

Crossover	*Mutation*	*Population Size*	*Crossover Rate (%)*	*Mutation Rate (%)*	*Elitism (%)*	*Generations*
Unnamed	Simple Inversion	300	50%	40%	5%	1000

Table 6: Parameters used for the tests on the benchmarks. No stopping criterion was used.

Other Tasks

Until now, most of the analysis was done with the purpose of finding a good solution for every benchmark. Other facets that were neglected to some extent are selective pressure and diversity. Those can be crucial to prevent premature convergence and increase the probability of finding the better - or even the optimal - solution(s). Therefor, some attention was given to strategies like tournament selection.

Parent Selection `/selection/*.m`

The most interesting parent selection strategy that was implemented is probably tournament selection. Both tournament selection with - and without replacement were implemented. Results of some basic experiments are displayed in tabel 8.

While the rank-based method may appear preferable, in all runs where it was used the algorithm stopped early on because the stopping criterion was met (no improvement was seen for 100 generations in a row). Whereas when tournament selection with replacement ($k = 2$) was used this was not the case.

Crossover	*Mutation*	*Population Size*	*Crossover Rate*	*Mutation Rate*	*Elitism*	*Generations*
`OX1`	Reciprocal Exchange	300	50%	40%	5%	1000

Table 7: Parameters used for tests on various types of parent selection. A stopping criterion was used.

FPS	*Linear Rank*	*TNM ($k = 2$)*	*TNM ($k = 10$)*	*TNM ($k = 20$)*	*TNMWR $k = 2$ (no replacement)*
49.23/126.00	41.35/51.58	41.31/75.47	42.15/47.58	42.66/47.85	42.86/87.21
1045.12/2654.36	723.89/867.44	773.79/1358.97	793.34/867.53	811.02/878.55	767.45/1531.66

Table 8: Results of experiments with several parent selection strategies. *TNM* = Tournament selection. *TNMWR* = Tournament selection without replacement Average of best and average fitness across 30 runs is displayed. First row represents experiments on a tour of 25 cities, second row are experiments on the belgian tour (41 cities). The results are equivalent for both tours.

To elaborate on this : what's not shown is the number of generations it took for the stopping criterion to be met. In the case of the (linear) rank-based selection it was met in all runs, often fairly early on (within 400 generations). In the case of fitness-proportionate selection it was frequently met (especially in the shorter tour of 25 cities). And in the case of tournament selection it depended on the value of k and whether or not replacement was used. For $k = 10$ it was met within 300 generations in most cases, for $k = 20$ within 200 generations. Clearly, the value of k influences selective pressure and setting it too high may lead to premature convergence. When tournament selection without replacement was used, the number of generations before the stopping criterion was

met increased sharply (sometimes it wasn't even met within 1000 generations). This is not surprising since lesser individuals get a higher chance of surviving (as noted in Eiben's book).

Non-linear ranking turned out to be fairly useful as well, but tournament selection looked most likeable as a strategy as it can be run in parallel and its arguments allow one to control selective pressure quite easily (higher k values clearly increased the average versus best fitness value which is an - admittedly somewhat poor - metric for diversity).

Survivor Selection `/reinsertion/*.m`

Out of the three survivor selection strategies that were implemented - $\mu + \lambda$, round robin and uniform selection - the first two led to premature convergence and the latter was highly unpredictable. Any of the parent selection methods can be used for survivor selection as well. $\mu + \lambda$ considers the merged offspring - and parent population and has a large selective pressure. Round robin considers the merged populations as well, selecting parents more and more often (because as the algorithm converges these tend to have higher fitness values). Eventually there's a large amount of highly-fit local optima and exploration is decreased starkly[2]. Using stochastic universal selection (based on the number of wins) instead of basing selection directly on the number of wins themselves is only of limited benefit.

Diversity Preservation

An island model was implemented in which the population is divided into islands if it is large enough. Every island hosts at least 25 individuals and the best ones are transferred from one to the other every so often (leading to periodic drops in average and worst fitness values throughout the generations). It obviously helped with diversity but not in the way tournament selection does ; if there was premature convergence in some algorithm, using the island model tends to transfer that problem to most of the islands themselves (as in, most islands experience a similar degree of convergence). *After* migration the degree of convergence drops (this is based on visual analysis of the graphs in the GUI).

One way to measure diversity is by counting the number of unique fitness values in the population (admittedly not a perfect measure since different genotypes may have the same fitness value). The median number of uniques across generations typically turned out to be higher when using the island model. Even the mean was nearly higher despite the fact that during migration the worst individuals in islands are replaced by the best individuals in other islands, which leads to a stark but short-lived drop in the number of unique fitness values, followed by an increase (until it reaches a value that's typically higher than what it was before migration). For example, the average of the median of the number of unique fitness values across 250 generations (for 30 runs on the belgian tour) was calculated to be 75 and 63 (with and without island model).

Adaptive Parameter Tuning

It would seem that, as several good solutions have been found after quite a few generations, increased frequency of mutation and reduced frequency of local optimisation (if hybridisation is activated) would prove useful by allowing for a more liberal exploration of the search space. Therefor, an adaptive parameter strategy was implemented which does just that. Instead of seeing interesting new solutions emerge, the average fitness just kept on diverging. Probably because the search became equivalent to a rather random search. A more interesting approach would be to increase the mutation rate if diversity sinks too low.

Further Hybridisation `/hybridisation/nearest_neighbor.m`

Seeding is a frequently used technique to give a head start. There's several possible approximate solutions for the TSP that can be used for this purpose. One of them is a nearest neighbour solution where a random city is picked and a tour is formed by repeatedly selecting the city nearest to the current one until all of them have been selected. This works well until at the end the remaining edges have to be added, often at high cost. It provides a decent initial solution which is retained through elitism and may provide inspiration for subsequent tours (or lead the algorithm towards a local optimum). In most experiments the resulting initial tour turned out to have a length that's within 15% of the optimal one, and the genetic algorithm caused further improvements until (in a typical run) the resulting tour ended up within about 6% of the optimum. No *kd*-tree was used as a seed is only

[2] In all experiments a termination condition was used ; if at least 95% of individuals are equal, the algorithm stops. It was never met when round-robin was used, despite the fact that probably 50%-90% individuals had the same fitness value in most generations.

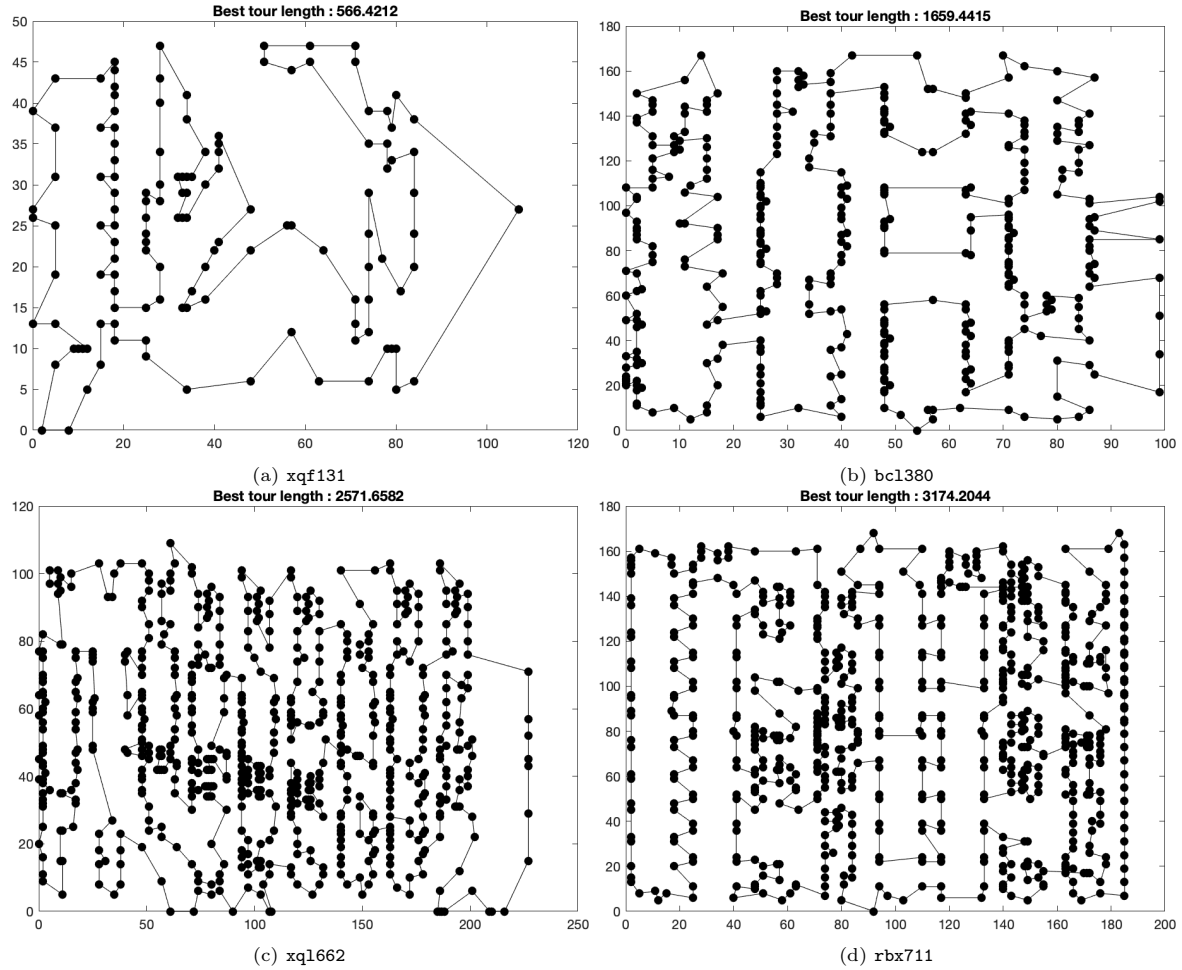
constructed once for every run which (in comparison to a whole run) takes little time. Unsurprisingly, however, it barely helped with finding a *truly* good tour. Instead it improved the average best tour found across runs.

Benchmarks with Heuristics

A final test on the benchmarks was done, this time with *2-opt* enabled. It was not applied in each generation, only every so often because it is still a fairly costly operation (as the number of generations increases it was applied less and less to increase the chances of getting out of a local minimum).

Crossover	*Mutation*	*Population Size*	*Crossover Rate (%)*	*Mutation Rate (%)*	*Elitism (%)*	*Generations*
Unnamed	Reciprocal Exchange	300	50%	40%	5%	1000

Table 9: Parameters used for the final tests on the benchmarks. No stopping criterion was used. Tournament selection and the island model were enabled.



(a) `xqf131` (b) `bcl380`

(c) `xql662` (d) `rbx711`

Figure 3: Results on benchmarks, this time *with* heuristics like *2-opt*. Error rates of 0.43%, 2.37%, 2.33% and 1.9% respectively.

Crossover Rate	*Mutation Rate*	*Elitism*	`rondrit025`	`rondrit070`	`rondrit0127`	`belgiumtour`	*Total Time (s)*
5	10	1	39.88	1129.53	301.73	705.33	3.16
5	10	5	39.85	1111.11	295.07	704.37	2.92
5	10	10	39.88	1128.86	292.94	714.93	3.07
25	10	1	39.85	1089.23	277.36	717.66	6.33
25	10	5	39.95	1078.36	271.84	701.80	5.30
25	10	10	39.81	1060.66	269.12	701.97	5.54
50	10	1	39.81	1252.28	333.35	733.01	10.01
50	10	5	39.81	1047.86	272.19	701.33	9.16
50	10	10	39.81	1052.85	267.24	689.64	8.89
75	10	1	50.24	2044.69	443.77	1285.26	15.39
75	10	5	39.81	1398.13	325.15	784.33	15.31
75	10	10	39.81	1139.36	291.46	701.02	13.24
95	10	1	57.56	2332.09	462.01	1465.70	18.95
95	10	5	45.43	1888.66	398.31	1198.29	17.31
95	10	10	39.81	1444.34	354.56	790.83	16.69
5	25	1	39.85	984.46	275.61	684.16	3.79
5	25	5	39.81	982.26	262.45	685.49	3.48
5	25	10	39.81	939.78	259.67	677.71	3.46
25	25	1	39.81	1107.00	274.10	691.25	7.05
25	25	5	39.81	1005.26	260.25	690.62	6.43
25	25	10	39.88	1023.40	239.45	678.65	5.99
50	25	1	39.95	1474.90	333.94	883.15	11.77
50	25	5	39.85	1089.82	282.48	680.63	10.57
50	25	10	39.81	992.49	270.82	677.26	9.88
75	25	1	53.05	2013.87	422.98	1273.33	16.31
75	25	5	40.26	1388.90	328.45	881.81	15.09
75	25	10	39.81	1196.01	296.75	684.32	13.83
95	25	1	58.29	2324.97	469.66	1495.37	20.09
95	25	5	43.54	1733.56	400.03	1129.13	18.53
95	25	10	39.81	1458.97	350.66	772.29	17.59
5	50	1	39.81	1037.46	263.77	706.61	4.92
5	50	5	39.81	941.95	249.01	677.22	4.48
5	50	10	39.85	957.15	246.60	690.40	4.75
25	50	1	39.81	1129.27	287.48	794.02	8.40
25	50	5	39.81	1004.02	255.89	684.30	7.99
25	50	10	39.85	989.45	252.58	681.72	7.07
50	50	1	42.25	1518.22	342.51	916.12	12.75
50	50	5	39.81	1162.22	278.63	691.92	11.67
50	50	10	39.81	1056.73	270.87	687.67	10.79
75	50	1	45.89	2040.46	417.27	1160.69	17.57
75	50	5	40.23	1381.74	323.16	830.28	16.54
75	50	10	39.81	1274.47	306.12	746.85	15.28
95	50	1	56.11	2190.59	463.83	1393.32	20.43
95	50	5	43.70	1730.25	387.40	1015.05	19.47
95	50	10	40.05	1516.67	339.17	892.06	18.64
5	75	1	39.81	1163.46	269.17	812.60	6.57
5	75	5	39.81	945.76	238.76	700.27	6.26
5	75	10	39.85	930.06	240.42	682.05	6.05
25	75	1	40.26	1276.73	287.03	859.56	10.01
25	75	5	39.81	1008.69	248.93	714.51	9.28
25	75	10	39.85	1003.08	243.49	713.66	8.09
50	75	1	41.63	1520.14	338.83	929.25	13.45
50	75	5	39.81	1136.45	279.42	740.43	12.74
50	75	10	39.81	1071.56	281.17	689.91	12.13
75	75	1	47.22	1958.61	406.44	1221.32	17.49
75	75	5	40.04	1388.59	312.78	860.72	17.59
75	75	10	39.85	1292.43	308.98	780.71	18.28
95	75	1	59.41	2290.11	469.52	1480.32	23.01
95	75	5	41.28	1760.24	384.56	1146.44	21.89
95	75	10	40.63	1539.35	352.51	930.64	19.27
5	95	1	40.35	1215.18	263.81	821.78	7.13
5	95	5	39.85	1021.89	244.37	709.13	6.58
5	95	10	39.81	1009.70	236.50	695.87	6.51
25	95	1	42.21	1354.51	297.81	878.39	10.48
25	95	5	39.88	1082.92	262.25	719.63	10.23
25	95	10	39.81	1060.29	263.16	703.41	9.34
50	95	1	46.21	1435.87	339.08	985.20	15.25
50	95	5	39.81	1198.67	290.15	823.71	13.31
50	95	10	39.88	1152.10	274.71	766.47	12.85
75	95	1	47.77	1908.15	404.15	1178.48	20.32
75	95	5	40.49	1384.93	336.02	913.31	18.70
75	95	10	40.49	1398.93	316.32	814.54	17.73
95	95	1	57.91	2228.48	460.84	1418.74	21.39
95	95	5	42.92	1731.17	390.41	1120.34	21.23
95	95	10	41.07	1649.37	362.66	1025.49	19.49

Table 10: Results of 75 configurations (population size of 150). The grayscale values of the cells are calibrated against the minimum and maximum for all tests (from table 10 to 14). Darker cells represent better solutions. No stopping criterion was used, maximum number of generations was set to 250. Each configuration was run 10 times.

Crossover Rate	*Mutation Rate*	*Elitism*	`rondrit025`	`rondrit070`	`rondrit0127`	`belgiumtour`	*Total Time (s)*
5	10	1	39.81	1026.32	277.45	689.66	5.42
5	10	5	39.85	995.24	267.05	684.67	4.70
5	10	10	39.85	1034.40	271.66	677.14	4.97
25	10	1	39.81	967.10	267.26	680.93	12.03
25	10	5	39.85	942.67	248.75	679.28	11.62
25	10	10	39.81	942.36	252.09	679.51	10.81
50	10	1	39.85	1181.41	308.58	699.21	20.95
50	10	5	39.85	995.10	260.14	677.24	19.37
50	10	10	39.85	932.83	251.33	681.64	17.94
75	10	1	49.77	1909.07	413.22	1189.79	34.17
75	10	5	39.81	1331.43	312.04	764.27	28.72
75	10	10	39.81	1110.10	301.25	679.07	25.53
95	10	1	52.69	2099.32	436.22	1388.94	41.01
95	10	5	43.11	1699.08	384.74	1015.50	37.93
95	10	10	39.81	1435.37	347.33	826.52	36.89
5	25	1	39.81	921.40	255.56	660.77	8.42
5	25	5	39.81	871.97	255.94	681.10	7.11
5	25	10	39.81	915.70	252.50	677.05	6.94
25	25	1	39.81	1021.10	263.19	677.28	12.73
25	25	5	39.81	923.51	244.58	672.53	12.60
25	25	10	39.85	904.66	237.40	664.40	12.21
50	25	1	39.81	1303.49	317.24	765.23	21.33
50	25	5	39.81	1002.86	267.11	666.79	19.68
50	25	10	39.81	990.84	254.46	680.47	18.79
75	25	1	44.17	1799.59	397.71	1053.97	32.16
75	25	5	39.81	1263.23	303.54	792.28	30.09
75	25	10	39.81	1127.83	288.70	697.76	29.91
95	25	1	52.78	2121.78	449.08	1309.36	39.69
95	25	5	41.60	1663.95	376.11	993.88	40.59
95	25	10	39.81	1476.09	325.93	842.27	36.03
5	50	1	39.81	956.59	238.68	677.69	10.07
5	50	5	39.81	898.48	239.26	665.35	10.19
5	50	10	39.81	858.82	227.84	678.52	8.44
25	50	1	39.81	1063.69	254.09	686.64	15.23
25	50	5	39.81	933.68	247.84	674.69	14.41
25	50	10	39.81	918.11	239.30	670.85	13.57
50	50	1	39.88	1277.97	296.90	787.59	23.64
50	50	5	39.81	1009.26	266.49	703.43	24.47
50	50	10	39.81	1005.42	254.37	689.34	23.99
75	50	1	41.75	1633.11	374.51	995.25	36.07
75	50	5	39.81	1230.24	315.37	773.48	33.90
75	50	10	39.81	1118.49	296.97	753.74	28.70
95	50	1	50.10	2044.78	418.96	1323.22	39.23
95	50	5	41.54	1650.17	373.91	950.69	37.83
95	50	10	40.39	1478.49	344.82	850.07	39.25
5	75	1	39.85	978.26	251.08	709.15	12.48
5	75	5	39.81	950.27	232.39	659.82	13.75
5	75	10	39.81	906.79	226.98	674.53	13.02
25	75	1	39.85	1171.11	272.02	737.26	20.39
25	75	5	39.81	938.22	247.26	689.44	19.30
25	75	10	39.81	965.60	241.18	676.45	17.34
50	75	1	40.43	1308.53	302.33	819.90	28.25
50	75	5	39.81	1083.16	261.68	712.63	26.59
50	75	10	39.81	1057.10	259.96	689.85	25.56
75	75	1	41.99	1472.81	365.27	1002.86	34.86
75	75	5	39.81	1334.12	310.97	809.11	35.57
75	75	10	39.85	1204.97	301.66	728.16	36.40
95	75	1	49.62	1958.84	437.49	1228.65	44.36
95	75	5	40.83	1672.70	375.14	993.54	41.09
95	75	10	40.11	1561.75	345.73	916.72	39.94
5	95	1	39.81	1101.82	259.08	751.20	14.21
5	95	5	39.81	937.43	239.62	682.55	13.48
5	95	10	39.81	910.88	237.34	674.67	13.08
25	95	1	39.81	1180.11	267.77	757.28	20.92
25	95	5	39.81	1054.81	253.74	682.71	20.09
25	95	10	39.85	1001.84	247.52	693.13	19.87
50	95	1	39.95	1321.32	295.87	810.30	29.16
50	95	5	39.81	1081.48	272.77	716.21	29.37
50	95	10	39.81	1085.92	275.22	736.31	27.83
75	95	1	41.47	1562.49	347.35	1038.87	40.31
75	95	5	40.11	1369.60	326.74	834.21	42.06
75	95	10	39.85	1272.90	302.24	832.11	36.91
95	95	1	51.78	2012.85	421.27	1265.05	48.92
95	95	5	41.43	1692.14	373.30	1030.19	48.13
95	95	10	41.12	1575.49	360.56	985.45	44.60

Table 11: Results of 75 configurations (population size of 300). The grayscale values of the cells are calibrated against the minimum and maximum for all tests (from table 10 to 14). Darker cells represent better solutions. No stopping criterion was used, maximum number of generations was set to 250. Each configuration was run 10 times.

Crossover Rate	*Mutation Rate*	*Elitism*	`rondrit025`	`rondrit070`	`rondrit0127`	`belgiumtour`	*Total Time (s)*
5	10	1	39.85	930.43	256.78	680.21	13.03
5	10	5	39.81	942.02	259.12	674.14	13.41
5	10	10	39.81	951.70	261.30	669.99	11.99
25	10	1	39.81	923.12	248.02	675.16	26.50
25	10	5	39.81	890.55	236.60	676.53	23.47
25	10	10	39.88	873.56	234.77	659.82	23.67
50	10	1	39.85	1167.88	297.34	686.65	42.95
50	10	5	39.85	952.54	251.20	669.99	37.99
50	10	10	39.81	884.58	240.65	680.78	35.27
75	10	1	47.98	1727.69	388.89	1005.35	62.23
75	10	5	39.81	1205.74	304.29	768.49	58.18
75	10	10	39.85	1130.15	279.38	659.82	54.04
95	10	1	55.65	2149.21	442.85	1285.62	74.09
95	10	5	39.81	1677.19	369.89	1108.41	76.86
95	10	10	39.85	1420.02	337.04	833.56	73.09
5	25	1	39.81	891.62	244.70	673.74	20.15
5	25	5	39.81	855.17	236.98	682.29	18.65
5	25	10	39.81	865.42	235.36	678.32	18.50
25	25	1	39.85	973.69	244.89	684.03	35.46
25	25	5	39.81	860.88	230.74	679.11	33.15
25	25	10	39.81	896.96	227.69	659.82	31.72
50	25	1	39.81	1143.05	294.28	703.27	51.45
50	25	5	39.81	969.63	260.42	664.53	50.82
50	25	10	39.81	934.17	241.03	659.82	42.09
75	25	1	41.33	1611.60	357.94	1038.32	66.56
75	25	5	39.92	1206.36	303.92	745.79	63.17
75	25	10	39.81	1123.31	288.50	680.52	62.24
95	25	1	50.13	2096.39	423.11	1355.53	83.54
95	25	5	40.46	1614.63	365.97	976.52	76.27
95	25	10	39.95	1520.48	343.53	866.19	66.51
5	50	1	39.81	922.21	240.11	664.32	19.89
5	50	5	39.85	878.41	228.43	674.69	21.16
5	50	10	39.81	858.99	222.21	669.45	19.55
25	50	1	39.81	1001.98	251.38	707.75	39.39
25	50	5	39.81	901.20	231.07	682.90	35.02
25	50	10	39.81	881.74	233.71	673.95	29.01
50	50	1	39.85	1151.30	293.41	774.45	48.71
50	50	5	39.85	1005.83	246.74	685.03	49.16
50	50	10	39.81	912.96	258.98	669.43	48.11
75	50	1	40.16	1536.11	348.50	944.13	66.21
75	50	5	39.81	1218.18	310.51	785.96	74.57
75	50	10	39.81	1170.13	280.64	705.90	68.51
95	50	1	49.69	2052.34	437.40	1244.72	89.73
95	50	5	41.26	1607.94	372.48	938.75	90.36
95	50	10	39.81	1525.31	349.08	903.00	79.17
5	75	1	39.81	954.07	243.90	687.42	29.31
5	75	5	39.81	881.50	226.49	666.27	29.93
5	75	10	39.81	878.71	232.14	675.99	27.87
25	75	1	39.88	1084.91	247.89	723.22	47.19
25	75	5	39.81	894.75	238.77	677.30	42.40
25	75	10	39.81	923.96	229.14	681.82	38.25
50	75	1	39.81	1177.05	275.39	823.70	68.42
50	75	5	39.81	1073.21	260.73	715.15	61.51
50	75	10	39.81	991.54	252.15	706.53	50.47
75	75	1	40.38	1455.75	364.65	917.46	81.53
75	75	5	39.81	1305.71	285.74	789.07	78.83
75	75	10	39.88	1249.36	290.64	723.96	75.63
95	75	1	48.52	2013.42	426.76	1216.01	102.22
95	75	5	41.53	1589.63	363.68	964.11	100.81
95	75	10	39.85	1544.52	346.05	925.74	93.16
5	95	1	39.85	1028.84	233.52	717.65	27.73
5	95	5	39.81	942.56	237.00	678.32	26.74
5	95	10	39.81	955.08	231.25	681.05	25.91
25	95	1	39.85	1104.89	259.69	727.78	40.92
25	95	5	39.81	996.10	242.36	700.38	40.91
25	95	10	39.85	964.75	249.32	703.11	38.52
50	95	1	39.92	1188.30	292.16	816.51	57.02
50	95	5	39.85	1136.67	272.94	725.23	53.99
50	95	10	39.81	1094.00	259.76	757.17	50.85
75	95	1	40.76	1385.94	344.12	882.17	73.13
75	95	5	39.95	1314.50	314.43	820.45	69.95
75	95	10	39.96	1364.09	306.11	795.39	65.77
95	95	1	47.15	2067.66	429.05	1222.32	87.39
95	95	5	42.18	1698.60	368.14	1091.35	83.09
95	95	10	41.59	1560.07	368.22	982.94	79.12

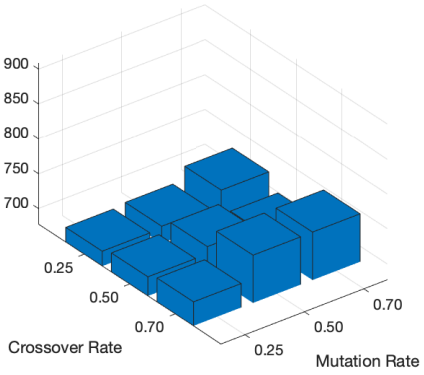
Table 12: Results of 75 configurations (population size of 600). The grayscale values of the cells are calibrated against the minimum and maximum for all tests (from table 10 to 14). Darker cells represent better solutions. No stopping criterion was used, maximum number of generations was set to 250. Each configuration was run 10 times.

Crossover Rate	*Mutation Rate*	*Elitism*	`rondrit025`	`rondrit070`	`rondrit0127`	`belgiumtour`	*Total Time (s)*
5	10	10	39.81	998.91	278.60	684.21	4.35
5	10	20	39.88	1021.32	283.56	714.14	3.97
5	10	40	39.81	1197.49	297.72	701.67	3.27
25	10	10	39.81	963.74	255.61	683.17	9.30
25	10	20	39.85	958.17	267.42	688.87	8.37
25	10	40	39.81	1125.63	281.72	721.75	6.55
50	10	10	39.81	985.41	256.67	692.25	15.75
50	10	20	39.81	1026.09	252.21	687.57	14.24
50	10	40	39.81	1077.53	274.45	690.53	10.89
75	10	10	39.88	1048.69	296.26	693.85	23.21
75	10	20	39.81	1101.46	281.61	673.74	20.24
75	10	40	39.81	1149.65	300.44	690.24	15.45
95	10	10	39.85	1489.54	360.88	786.25	30.25
95	10	20	39.81	1262.33	310.76	768.36	26.09
95	10	40	39.85	1384.64	329.65	768.77	19.83
5	25	10	39.85	930.60	244.39	687.91	5.62
5	25	20	39.81	981.64	257.45	677.25	5.13
5	25	40	39.85	1065.91	293.39	684.94	4.08
25	25	10	39.85	895.79	249.12	665.48	10.67
25	25	20	39.81	907.49	238.69	678.20	9.60
25	25	40	40.08	987.49	279.64	682.90	7.68
50	25	10	39.81	930.06	259.44	679.09	17.27
50	25	20	39.81	983.54	262.35	673.49	15.57
50	25	40	39.81	1014.45	273.68	686.19	11.89
75	25	10	39.81	1175.26	298.26	707.98	24.84
75	25	20	39.81	1139.01	269.62	691.57	21.67
75	25	40	39.81	1166.39	290.27	684.86	16.46
95	25	10	39.81	1501.04	344.14	928.82	31.65
95	25	20	39.81	1331.36	320.25	752.12	27.59
95	25	40	39.88	1317.27	320.03	782.09	20.84
5	50	10	39.81	888.57	242.11	684.69	7.71
5	50	20	39.81	905.60	245.25	675.28	7.00
5	50	40	39.88	1009.15	259.02	674.71	5.50
25	50	10	39.81	938.81	243.48	664.40	12.92
25	50	20	39.81	923.20	246.14	677.71	11.58
25	50	40	39.85	1051.47	256.28	702.04	8.99
50	50	10	39.81	1000.19	259.62	689.94	19.76
50	50	20	39.85	1034.11	261.99	660.77	17.62
50	50	40	39.85	1081.69	286.20	698.26	13.34
75	50	10	39.81	1188.37	311.46	738.11	27.33
75	50	20	39.81	1172.89	269.57	740.21	23.88
75	50	40	39.81	1170.34	301.70	763.01	18.17
95	50	10	39.81	1493.67	358.62	829.74	34.27
95	50	20	39.81	1475.72	337.97	843.31	29.58
95	50	40	39.81	1435.66	346.59	860.19	22.30
5	75	10	39.85	840.14	236.99	681.11	9.66
5	75	20	39.81	912.07	240.76	679.31	8.73
5	75	40	39.81	1057.58	261.20	688.88	6.86
25	75	10	39.85	977.63	240.95	673.74	15.03
25	75	20	39.81	1013.75	248.55	690.24	13.45
25	75	40	39.81	1010.13	270.87	691.52	10.41
50	75	10	39.81	1076.18	265.19	697.80	22.01
50	75	20	39.85	1105.90	274.20	695.24	19.52
50	75	40	39.81	1166.18	282.47	681.93	15.03
75	75	10	39.81	1280.02	300.91	781.15	29.49
75	75	20	39.81	1183.88	293.88	742.12	25.93
75	75	40	39.81	1269.35	308.44	806.14	19.67
95	75	10	40.41	1566.22	356.75	993.65	35.70
95	75	20	40.28	1454.29	350.82	909.13	31.58
95	75	40	39.81	1465.02	339.11	921.52	23.89
5	95	10	39.81	978.23	235.28	690.17	11.25
5	95	20	39.81	964.56	243.53	692.42	10.19
5	95	40	39.81	1066.87	253.69	709.25	7.99
25	95	10	39.81	1023.99	241.74	701.62	16.68
25	95	20	39.85	1012.17	256.76	698.19	15.01
25	95	40	39.81	1102.06	274.07	727.95	11.65
50	95	10	39.85	1102.65	275.30	753.05	23.72
50	95	20	39.85	1156.65	277.78	762.74	21.20
50	95	40	39.81	1224.12	284.71	782.02	16.16
75	95	10	39.81	1344.22	319.51	822.96	31.30
75	95	20	39.81	1273.87	307.99	849.56	27.75
75	95	40	39.88	1409.01	313.02	846.51	21.07
95	95	10	40.98	1637.14	356.96	1018.36	37.32
95	95	20	40.35	1536.69	353.30	957.62	33.14
95	95	40	40.56	1544.99	350.25	983.28	25.08

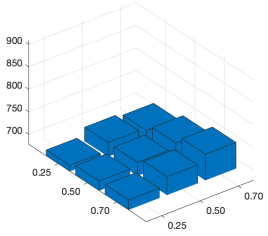
Table 13: Investigation of higher rates of elitism (higher rates of elitism don't appear to improve results). Results of 75 configurations (population size of 150). The grayscale values of the cells are calibrated against the minimum and maximum for all tests (from table 10 to 14). Darker cells represent better solutions. No stopping criterion was used, maximum number of generations was set to 250. Each configuration was run 10 times.

Crossover Rate	*Mutation Rate*	*Elitism*	`rondrit025`	`rondrit070`	`rondrit0127`	`belgiumtour`	*Total Time (s)*
5	10	1	39.81	798.23	221.93	667.22	5.51
5	10	5	39.81	800.32	214.99	677.22	6.03
5	10	10	40.35	855.03	212.22	659.82	5.75
25	10	1	39.81	824.85	220.84	679.12	11.80
25	10	5	39.81	780.99	216.14	708.19	10.48
25	10	10	39.81	792.45	206.12	677.22	10.00
50	10	1	39.81	998.84	262.47	677.32	18.84
50	10	5	39.81	763.28	206.57	683.67	17.24
50	10	10	39.81	759.71	207.28	674.14	16.23
75	10	1	48.51	1936.12	410.20	1241.86	28.98
75	10	5	39.85	1030.30	292.26	674.53	25.95
75	10	10	39.81	842.67	220.41	673.16	23.14
95	10	1	53.91	2277.86	469.51	1453.19	35.41
95	10	5	40.57	1604.17	357.54	1058.75	33.84
95	10	10	39.95	1091.10	297.81	674.14	30.28
5	25	1	39.85	774.41	200.65	681.68	6.99
5	25	5	39.81	729.42	187.68	669.45	6.72
5	25	10	39.81	748.27	185.29	682.78	6.44
25	25	1	39.81	839.93	215.04	675.16	12.67
25	25	5	39.92	738.24	181.63	677.32	12.00
25	25	10	40.11	748.23	182.84	679.31	11.42
50	25	1	39.88	1186.09	286.35	773.08	21.24
50	25	5	39.81	778.69	219.01	683.87	19.04
50	25	10	39.88	758.84	199.88	659.82	17.69
75	25	1	48.16	1795.08	396.64	1150.86	30.24
75	25	5	39.85	1093.56	280.48	691.31	28.01
75	25	10	39.81	869.67	236.77	684.99	24.96
95	25	1	55.68	2306.38	460.72	1412.22	36.68
95	25	5	40.38	1514.12	352.94	943.33	35.13
95	25	10	39.81	1194.99	283.63	692.60	31.99
5	50	1	39.81	859.93	204.75	687.18	9.39
5	50	5	39.85	726.00	177.49	678.52	9.01
5	50	10	39.81	730.74	180.55	678.20	8.59
25	50	1	39.92	989.30	239.87	722.17	15.45
25	50	5	39.81	770.96	196.54	688.00	14.54
25	50	10	39.81	731.32	185.71	677.24	13.72
50	50	1	39.81	1206.34	300.53	863.52	24.01
50	50	5	39.81	874.65	212.70	700.99	21.94
50	50	10	39.81	791.85	195.57	677.10	20.36
75	50	1	42.74	1718.14	379.48	1072.10	32.67
75	50	5	39.81	1069.07	268.12	725.72	30.66
75	50	10	39.81	966.67	238.25	683.39	27.84
95	50	1	52.66	2194.85	454.80	1381.65	38.81
95	50	5	40.66	1463.04	344.99	915.54	37.26
95	50	10	39.85	1178.59	291.26	731.80	34.53
5	75	1	39.81	922.42	222.04	744.06	11.60
5	75	5	39.81	760.67	180.52	659.82	11.18
5	75	10	39.81	729.88	176.65	678.50	10.61
25	75	1	40.08	1054.43	253.36	720.25	17.77
25	75	5	39.81	818.27	196.65	670.28	16.86
25	75	10	39.81	759.18	184.73	678.32	15.87
50	75	1	39.92	1340.11	289.93	835.43	26.12
50	75	5	39.88	905.59	208.14	702.32	24.40
50	75	10	39.85	813.66	215.77	678.65	22.90
75	75	1	40.35	1578.21	356.97	1018.76	34.47
75	75	5	40.11	1063.12	270.97	746.47	32.75
75	75	10	39.88	998.40	240.25	687.51	30.21
95	75	1	58.31	2132.52	434.13	1372.71	40.90
95	75	5	40.08	1431.33	344.71	887.70	39.29
95	75	10	39.81	1246.93	307.08	746.63	36.66
5	95	1	39.85	1030.89	236.36	733.34	13.32
5	95	5	39.81	775.10	193.08	677.12	12.82
5	95	10	39.85	775.18	187.48	674.57	12.28
25	95	1	39.95	1139.96	268.00	812.21	19.68
25	95	5	39.81	825.21	198.97	691.17	18.66
25	95	10	39.81	817.35	198.17	678.54	17.71
50	95	1	40.08	1281.54	275.92	869.53	27.97
50	95	5	39.85	915.83	232.84	722.80	26.25
50	95	10	39.88	873.46	224.81	679.28	24.59
75	95	1	42.88	1588.25	370.08	1096.23	36.62
75	95	5	39.81	1156.23	271.29	703.57	34.44
75	95	10	39.85	1055.12	257.24	711.60	32.14
95	95	1	51.71	2161.43	438.43	1367.28	42.66
95	95	5	40.41	1478.49	324.10	931.82	40.92
95	95	10	40.79	1347.96	309.01	778.48	38.29

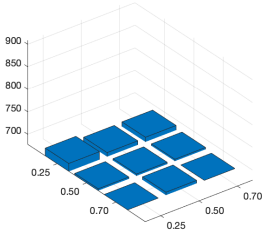
Table 14: Investigation of effect of the number of generations (the results are much better than when 250 generations were run). Results of 75 configurations (population size of 150). The grayscale values of the cells are calibrated against the minimum and maximum for all tests (from table 10 to 14). Darker cells represent better solutions. No stopping criterion was used, maximum number of generations was set to 500. Each configuration was run 10 times.



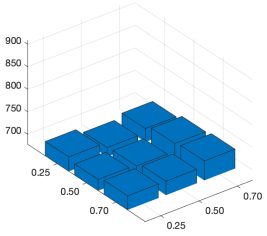
(a) Alternating Edges Crossover (adjacency representation)



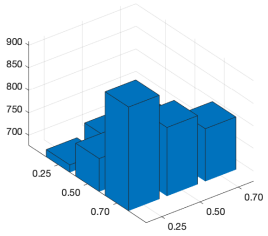
(b) `CX`



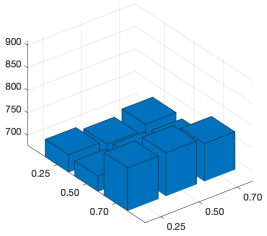
(c) `HERX`



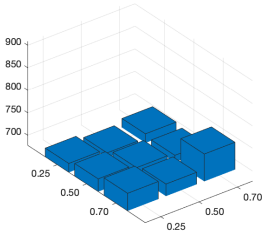
(d) `ERX`



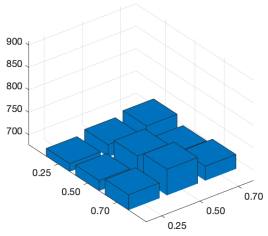
(e) `HEUR`



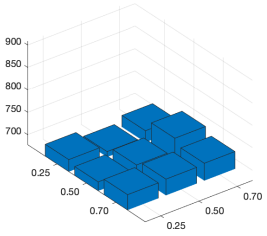
(f) `MPX`



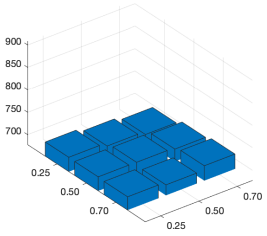
(g) `O1`



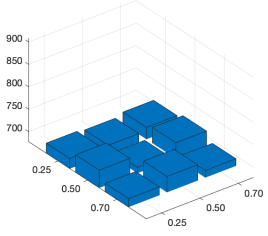
(h) `O2`



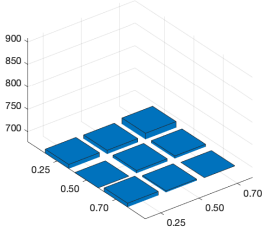
(i) `XOVSP`



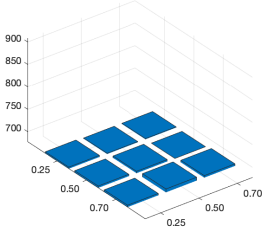
(j) `PMX`



(k) `POS`

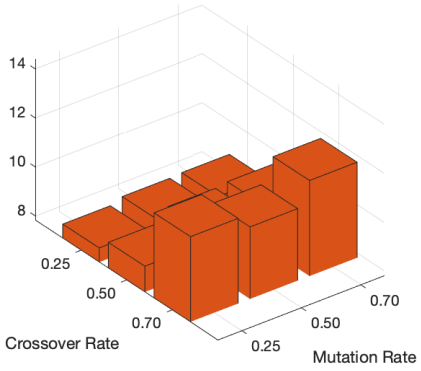


(l) `SCX`

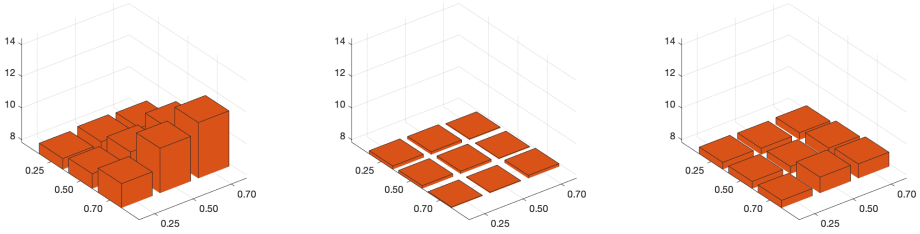


(m) `UHX`

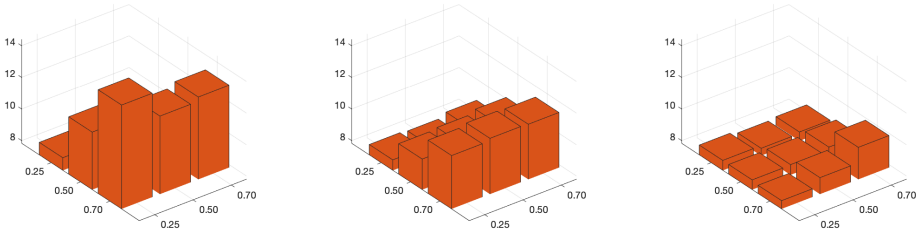
Figure 4: Shortest tour found by each crossover operator (tour of 70 cities). Maximum of 1000 generations and basic stopping criterion (the algorithm stops when there has been no improvement in 100 generations which is quite strict, such that late and surprising improvements can be ruled out). The stop condition was met in most cases.



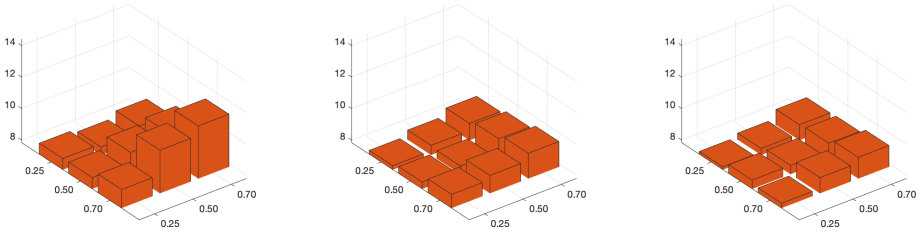
(a) Alternating Edges Crossover (adjacency representation)



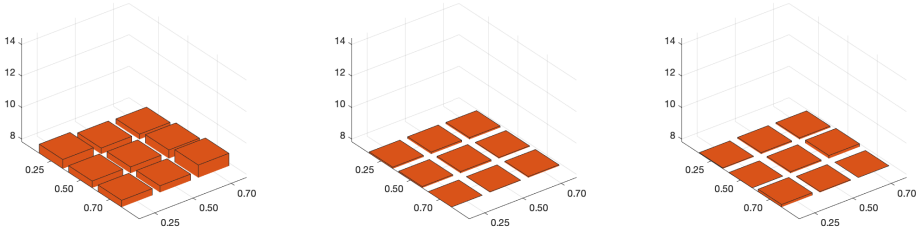
(b) `CX` (c) `HERX` (d) `ERX`



(e) `HEUR` (f) `MPX` (g) `O1`

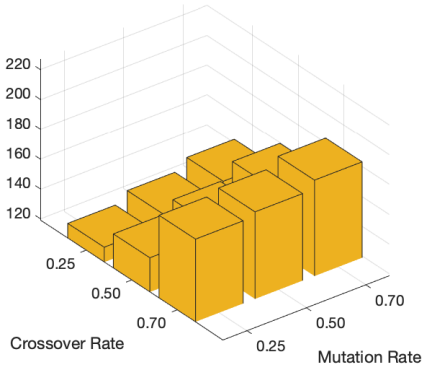


(h) `O2` (i) `XOVSP` (j) `PMX`

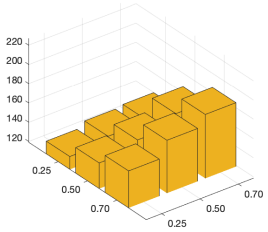


(k) `POS` (l) `SCX` (m) `UHX`

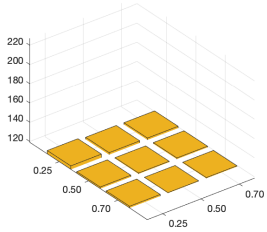
Figure 5: Shortest tour found by each crossover operator (tour of 100 cities). Maximum of 1000 generations and basic stopping criterion (the algorithm stops when there has been no improvement in 100 generations which is quite strict, such that late and surprising improvements can be ruled out). The stop condition was met in most cases.



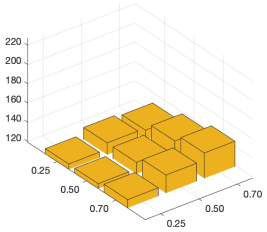
(a) Alternating Edges Crossover (adjacency representation)



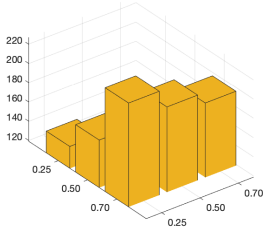
(b) `CX`



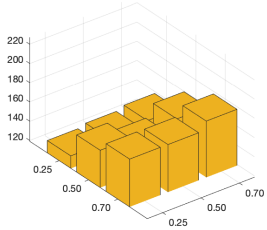
(c) `HERX`



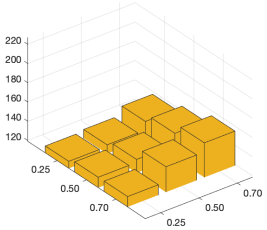
(d) `ERX`



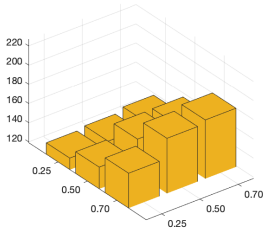
(e) `HEUR`



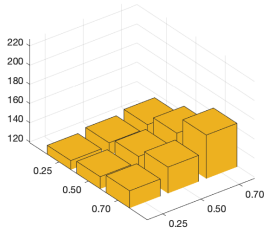
(f) `MPX`



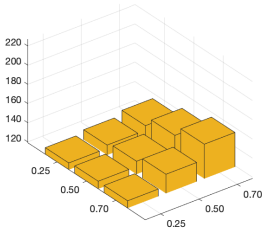
(g) `O1`



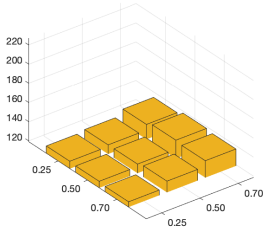
(h) `O2`



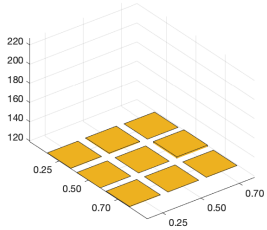
(i) `XOVSP`



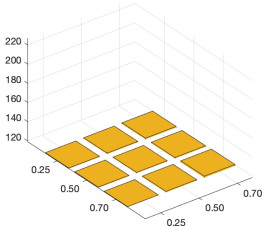
(j) `PMX`



(k) `POS`

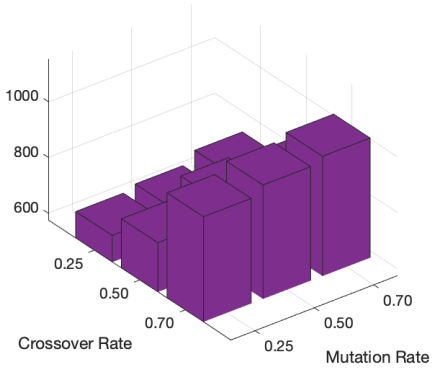


(l) `SCX`

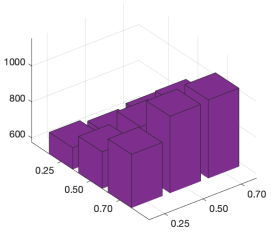


(m) `UHX`

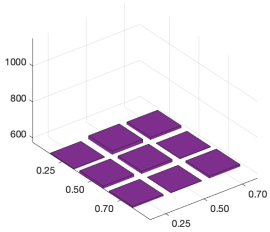
Figure 6: Shortest tour found by each crossover operator (tour of 127 cities). Maximum of 1000 generations and basic stopping criterion (the algorithm stops when there has been no improvement in 100 generations which is quite strict, such that late and surprising improvements can be ruled out). The stop condition was met in most cases.



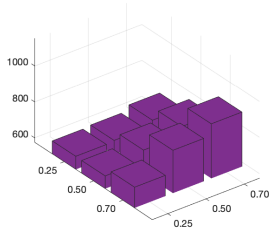
(a) Alternating Edges Crossover (adjacency representation)



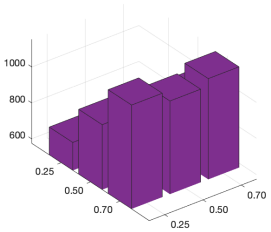
(b) `CX`



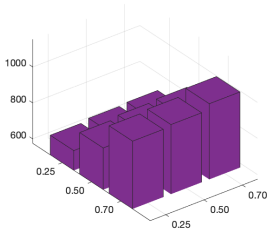
(c) `HERX`



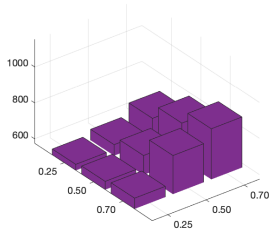
(d) `ERX`



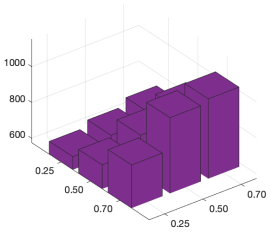
(e) `HEUR`



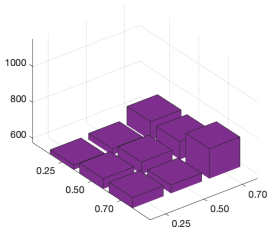
(f) `MPX`



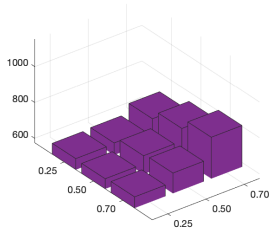
(g) `O1`



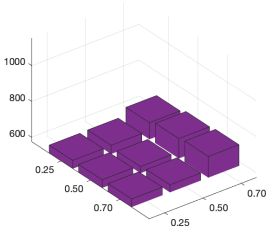
(h) `O2`



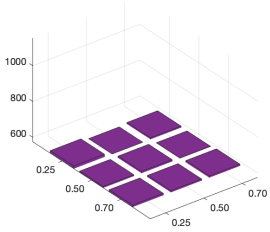
(i) `XOVSP`



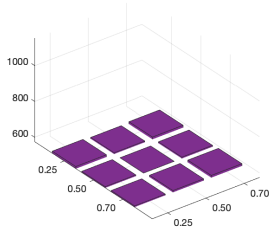
(j) `PMX`



(k) `POS`

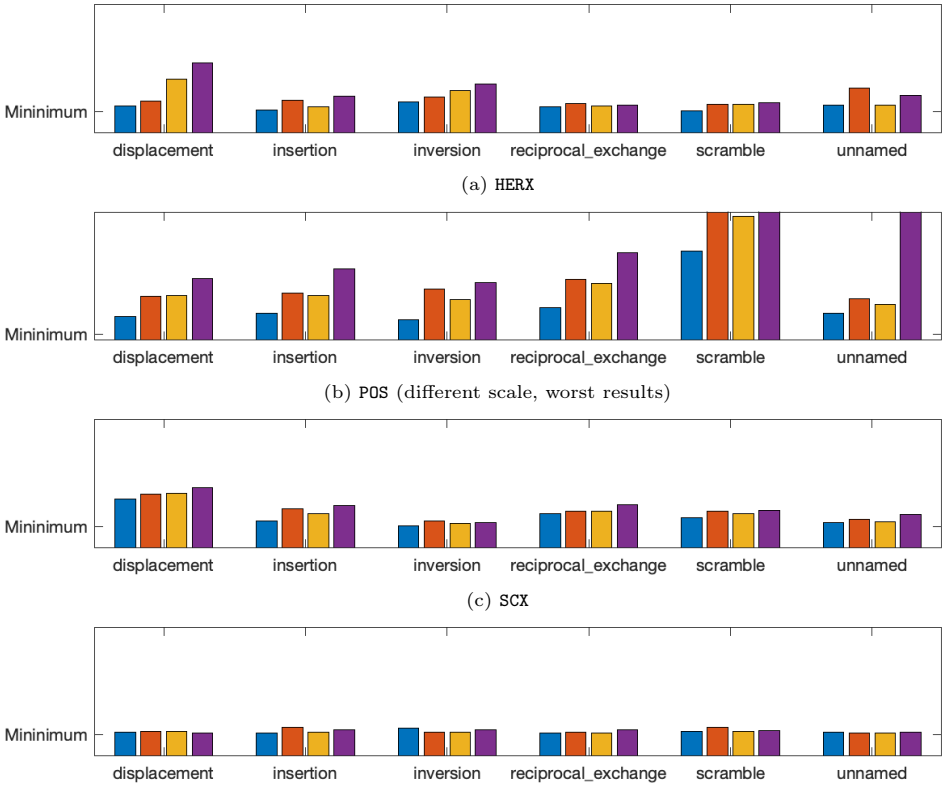


(l) `SCX`



(m) `UHX`

Figure 7: Shortest tour found by each crossover operator (tour of 131 cities). Maximum of 1000 generations and basic stopping criterion (the algorithm stops when there has been no improvement in 100 generations which is quite strict, such that late and surprising improvements can be ruled out). The stop condition was met in most cases.



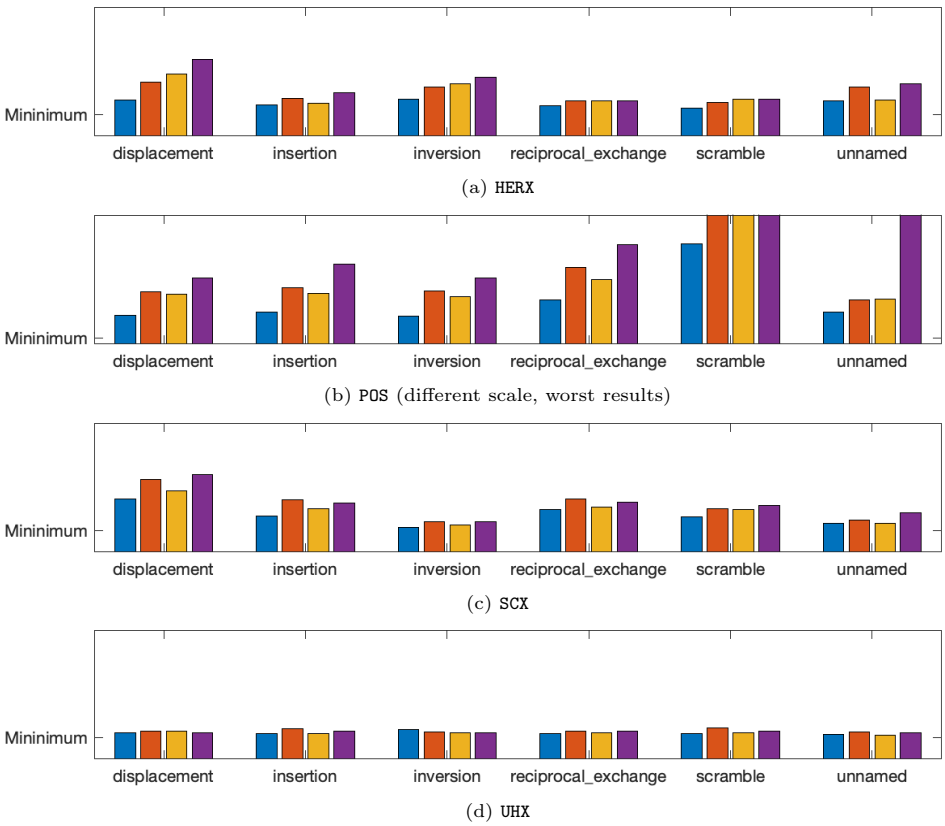
(a) HERX

(b) POS (different scale, worst results)

(c) SCX

(d) UHX

Figure 8: Shortest tour found by each mutation operator, one row per selected crossover operator. Colors refer to one of 4 tours (the same as in figures 4 to 7, i.e. those with 70, 100, 127 and 131 cities). The y-axis represents the (percentage) deviation from the best tour length that was found for each of those tours. 1000 generations, same stopping criterion, 4 different rates of mutation of which one is pictured (50%). Results don't vary much across the different rates, though a rate of 40-50% appears to be preferable.



(a) HERX

(b) POS (different scale, worst results)

(c) SCX

(d) UHX

Figure 9: Same as picture 8, looking at average shortest tours rather than the shortest tour of all runs for a given configuration.

References

[1] Hassan Ismkhan and Kamran Zamanifar. Study of some recent crossovers effects on speed and accuracy of genetic algorithm, using symmetric travelling salesman problem. *CoRR*, abs/1504.02590, 2015.

[2] P. Larrañaga, C.M.H. Kuijpers, R.H. Murga, I. Inza, and S. Dizdarevic. *Artificial Intelligence Review*, 13(2):129–170, 1999.

Time Spent on the Project

We worked together on the project and worked more than 35 hours each, without keeping track of time much. Most of the effort was spent on writing the code, the experiments were run in the background. The report took us a good deal of hours to write as well.