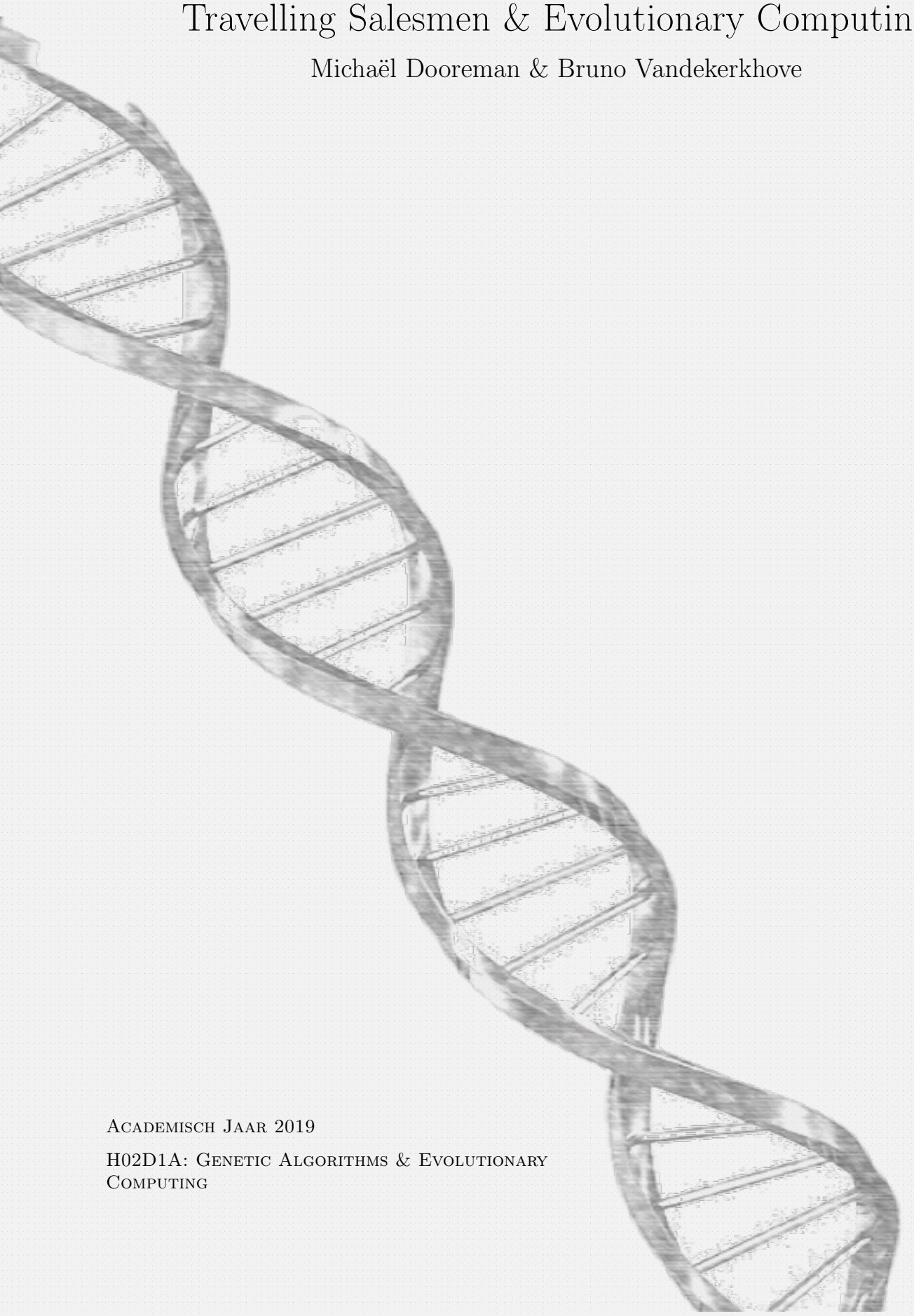
Travelling Salesmen & Evolutionary Computing

Michaël Dooreman & Bruno Vandekerkhove



Academisch Jaar 2019

H02D1A: Genetic Algorithms & Evolutionary Computing

Inhoudsopgave

A toolbox for solving the traveling salesman problem (TSP) was provided. Some parameter tuning with the default operators, selection methods, ... was performed. New tour representation were implemented with accompanying operators. Local heuristics, other selection methods, the island model, adaptive parameter tuning and a seeding technique were experimented with. The results are outlined below. Footnotes can be disregarded ; they provide the name of relevant source code files (all written in `MATLAB`*).*
Please use this template to structure your report to facilitate the reading and evaluation.
Do not repeat information from the textbook (e.g. if you use a crossover operator described in the book: don't copy the explanation or algorithm, but refer to the relevant page). The report should be ≈ 10 pages long; if useful you can add extra details in appendices (not included in the ≈ 10 pages).

Existing Genetic Algorithm

To get an initial feel for what may or may not be worth it, 225 configurations (table 1) were experimented with for a total of 13,500 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 (%)*
[150,300,1000]	[5,25,50,75,95]	[5,25,50,75,95]	[1,5,10]

Tabel 1: Parameter ranges for the initial experiments. Each configuration was run 10 times on 5 different datasets (with 25, 41, 70 and 127 cities). No stopping criterion was applied, and each run lasted 250 generations.

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 averaged. The best tour found by each was also recorded, since the very point of the traveling 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.

Two observations could be made ; larger population sizes improve 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 proved to be fruitless. While better individuals may appear attractive to the greedy, some of them represent local optima that may safely be disregarded. Though they may prove useful if the crossover operator can make full use of the information they carry. The default (alternating edge) crossover is not particularly fit for this purpose.

It can be noted that just 10 runs and 250 generations is hardly enough (from a statistical point of view). Only soft conclusions were drawn and limited computing power and time was available. Subsequent experiments were

run 30 times per configuration. 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.

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

1. Data set(s) used & explain why you selected these data set(s)
2. Parameters considered and intervals/values
3. Performance criteria used
4. Test results
5. Discussion of test results

Stopping Criterion

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

1. Stopping criterion & explain why you selected this criterion
2. Test results (incl. performance criteria and parameter settings)
3. Discussion of test results

Other Representation and Appropriate Operators

Aside from the adjacency representation, (at least) four alternative representations are available. Two of those are binary which were avoided due to previous experiences in constraint programming where the use of binary representation 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 [**?**] & [**?**]). 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. The others are :

- *Heuristic Edge Recombination (HERX)* :
- *Max Preservative (MPX)* :
- *Order Based (OX)* :
- *Position Based (PX)* :
- *Unnamed Heuristic (UHX)* :

The following mutation operators were implemented :

- *Displacement* :
- *Inversion* : the 'simple' inversion was already implemented, this one generalises it by ???.
- *Scramble* :
- *Unnamed* : accompanying the *UHX* crossover, this one's an example of hybridisation

As nothing was known about these operators aside from some experimental results on other benchmarks, they were all experimented with manually (in the GUI which was extended for this purpose). The more interesting ones were selected for further experimentation. From the 225 configurations that were trialed before, 20 were selected and each of the possible operators were tested in conjunction with those parameter values.

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

1. Representation
2. Crossover operators & explain why you selected the operators
3. Mutation operators & explain why you selected the operators
4. Parameter tuning: parameters considered and intervals/values
5. Data set(s) used & explain why you selected these data set(s)
6. Test results (incl. performance criteria and parameter settings)
7. Discussion of test results

Local Optimisation

Aside from the loop detection that 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. While not as powerful as the Lin-Kernighan heuristic (which even has some efficient implementations), they're simple to implement and provide good results. The time complexity of improving a tour with those is $\mathcal{O}(N^2)$. The experiments turned out to be fairly uninteresting in that apparently optimal results were found for each of the smaller tours. While a bit preliminary, we tested out one of the better crossovers in combination with these heuristics on `XQF131`. A near-optimal tour was found with just one erroneous sequence in the bottom left corner.

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

1. Local optimisation heuristic & explain why you selected this heuristic
2. Test results (incl. performance criteria and parameter settings)
3. Discussion of test results

Benchmark Problems

Test the performance of your algorithm using *some* benchmark problems (available on Toledo) and critically evaluate the achieved performance.
Keep in mind that for a large number of cities the search space is extremely large! If your algorithm doesn't perform well for a rather small number of cities, it doesn't make sense to use it for a benchmark problem with a large number of cities ...
Note: For most of the benchmark problems the length of the optimal tour is known. However, the Matlab template program scales the data. Therefore this scaling must be switched off to be able to compare your result with the optimal tour length.

1. List of benchmark problems
2. Test results (incl. performance criteria and parameter settings)
3. Discussion of test results

Other Tasks

Parent Selection

aaa

Survivor Selection

aaa

Diversity Preservation

aaa

Adaptive Parameter Tuning

aaa

Further Hybridisation

aaa

You should select *at least one* task from the list below:

1. Implement and use two other parent selection methods, i.e. fitness proportional selection and tournament selection. Compare the results with those obtained using the default rank-based selection.
2. Implement one survivor selection strategy (besides the already implemented elitism). Perform experiments and evaluate the results.
3. Implement and use one of the techniques aimed at preserving population diversity (e.g. subpopulations/islands, crowding, ...). Perform experiments and evaluate the results.
4. Incorporate an adaptive or self-adaptive parameter control strategy (e.g. parameters that depend on the state of the population, parameters that co-evolve with the population, ...). Perform experiments and evaluate the results.

For each task:

1. Description of implementation
2. Description of the experiments
3. Test results
4. Discussion of test results

Time Spent on the Project

1. For each student of the team: estimate how many hours spent on the project (NOT including studying textbook and other reading material).
   (a)
   (b)
2. Briefly discuss how the work was distributed among the team members.

Crossover Rate	*Mutation Rate*	*Elitism*	*A*	*B*	*C*	*D*	*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

Tabel 2: Results for 75 configurations (population size of 150). The grayscale values of the cells are calibrated against minima and maxima of all tests (for population size of 150, 300 and 600). No stopping criterion was used, maximum number of generations was set to 250.