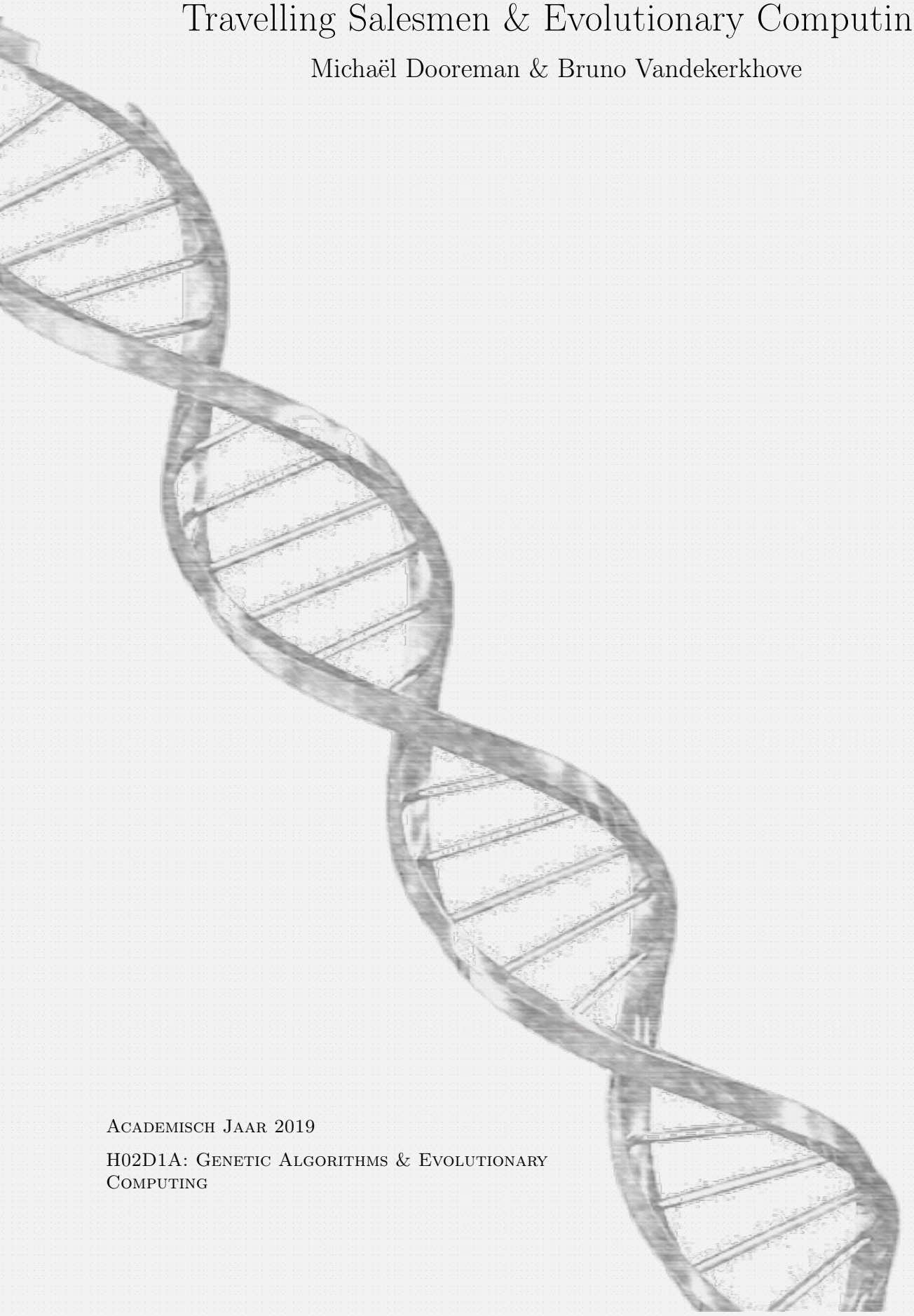
Travelling Salesmen & Evolutionary Computing

Michaël Dooreman & Bruno Vandekerkhove



Academisch Jaar 2019

H02D1A: Genetic Algorithms & Evolutionary Computing

Inhoudsopgave

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, 270 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).

After selecting 4 datasets (with an increasing amount of cities) each of the resulting algorithms were ran 10 times, after which 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.

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.

It can be noted that just 10 runs is hardly enough (from a statistical point of view). The limited computing power and time that was available for running the experimented may legitimate this. Subsequent experiments were run 30 times each.

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

Next to the adjacency representation, (at least) four alternatives 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, 11 crossover operators were implemented. We

1. *Operator* :
2. *Operator* :
3. *Operator* :
4. *Operator* :
5. *Operator* :
6. *Operator* :
7. *Operator* :
8. *Operator* :
9. *Operator* :
10. *Operator* :
11. *Operator* :

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

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

Population Size	*Crossover Rate*	*Mutation Rate*	*Elitism*	*A*	*B*	*C*	*D*	*Time*

Tabel 2: 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

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.