Chalmers University of Technology

FFR105 - Stochastic Optimization Algorithms

Problem set 2

Jacopo Credi
(910216-T396)

October 16, 2015

Problem 2.1 - The travelling salesman problem (TSP)

2.1(a)

In a TSP problem with N cities, there are N ways of picking the first city, $N-1$ ways to pick the second, and so on, until only 1 city is left. Therefore, a *valid* path can be constructed in $N!$ possible ways.

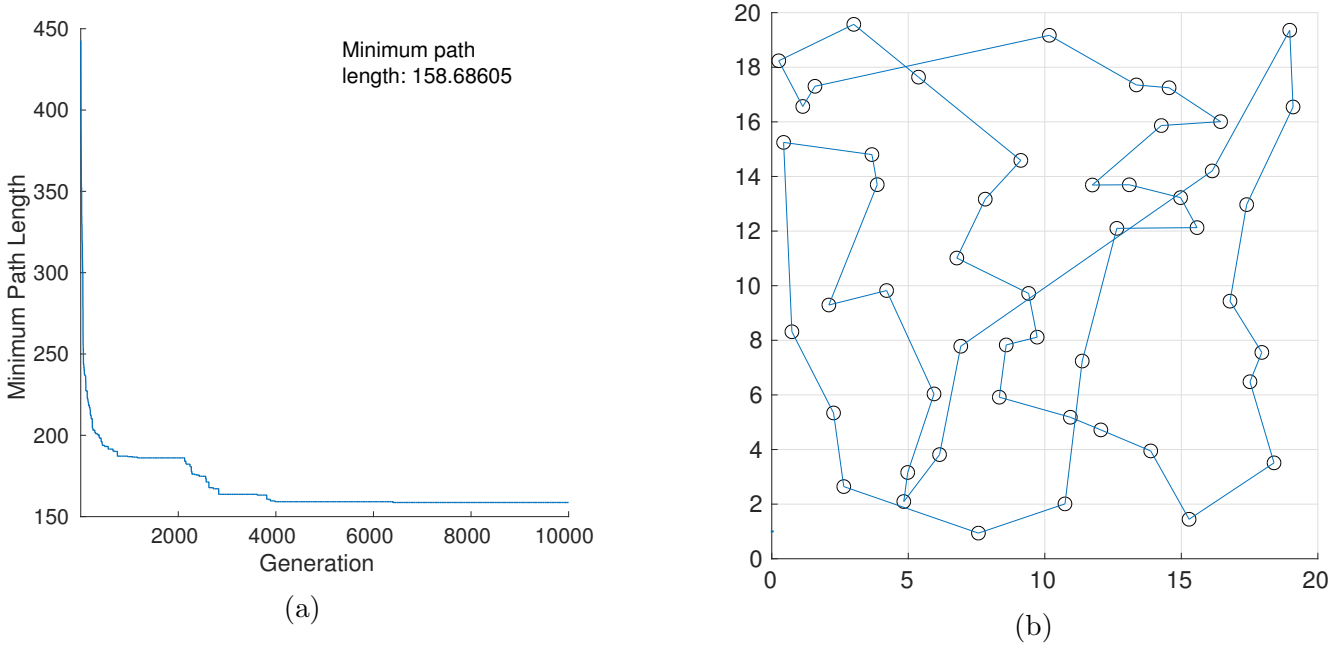
However, since paths are closed (i.e. they return to the original city), the starting node of the path does not matter, as long as the same cities are visited in the same order, thus we must divide the number of paths by a factor N to obtain the number of *distinct* paths. Furthermore, since paths going through a given sequence of cities in opposite order are also equivalent, this number must be also divided by two. Hence, the number of *distinct* paths in a TSP with N cities is

$$\frac{N!}{2N} = \frac{(N-1)!}{2} .$$

2.1(b)

A GA was implemented in MATLAB (see script `GA21b` in folder `2.1/2.1b`) to solve the TSP problem with $N = 50$ cities whose locations were provided on the course website. Paths are encoded using permutation encoding and population is initialised randomly, generating M chromosomes containing valid paths constructed at random (using MATLAB function `randperm`). The fitness is taken as the inverse of the path length, selection is carried out using tournament selection, crossover is not used and mutations are implemented as swap mutations.

Figure 1 shows the result of a long run (10000 generations) of the GA with 100 individuals, tournament size = 5, $p_{\text{tour}} = 0.8$, $p_{\text{mut}} = 2/N$, and inserting one copy of the best path in the next generation. The length of the best path found by the algorithm was 158.69.



(a) (b)

Figure 1: Results of `GA21b` for TSP with random initial population. Left panel: minimum path length vs. generations. Right panel: best path found (length $\simeq 158.69$).

2.1(c)

An Ant System (AS) algorithm was implemented by completing the provided MATLAB script `AntSystem.m` and then applied to the same TSP as above.

After experimenting with the parameters, it was found that using a larger number of ants than N (number of cities) could help avoiding premature convergence, as more pheromone is deposited in each iteration. Path in Figure 2, of length 122.44, was obtained by running the AS algorithm with 100 ants and parameters $\alpha = 2,\ \beta = 1,\ \rho = 0.5$, after 107 iterations. The AS algorithm was thus able to find a better solution than the GA, in a fraction of the time. The chromosome corresponding to this path is saved as a vector in file `bestPath.m` (in main folder `2.1`), which can be run as a script and loads the vector `bestPath` in the workspace.

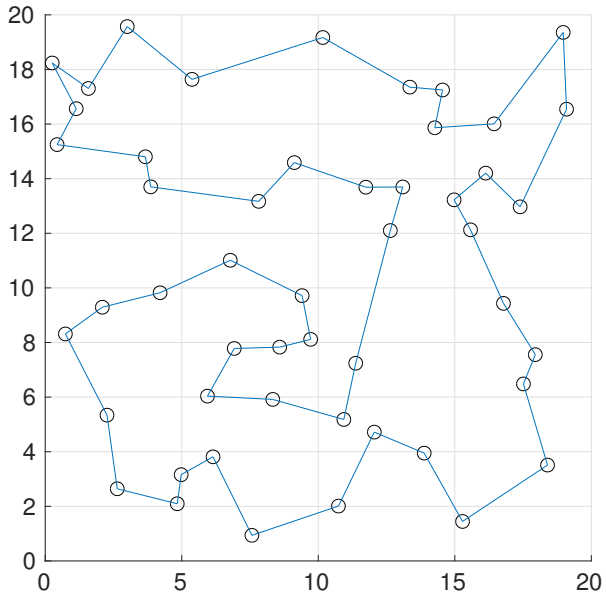


Figure 2: Best path found with Ant System algorithm. Length $\simeq 122.44$.

2.1(d)

To allow a more fair comparison between GA and AS, the initialisation of the GA population was modified, so that starting path are nearest-neighbour paths (generated by starting from a random city) with a few random swap mutations (see function `InitializePopulation21d`).

Figure 2 shows the result of a long run (10000 generations) of the GA with 100 individuals and the same parameters as in 2.1(b), starting from a population of nearest-neighbour paths with 3 swap mutations each. The length of the best path found by the GA in this case was 130.76, so this solution is much better the one in 2.1(b), as expected, but not quite good as the solution found by using AS in 2.1(c).

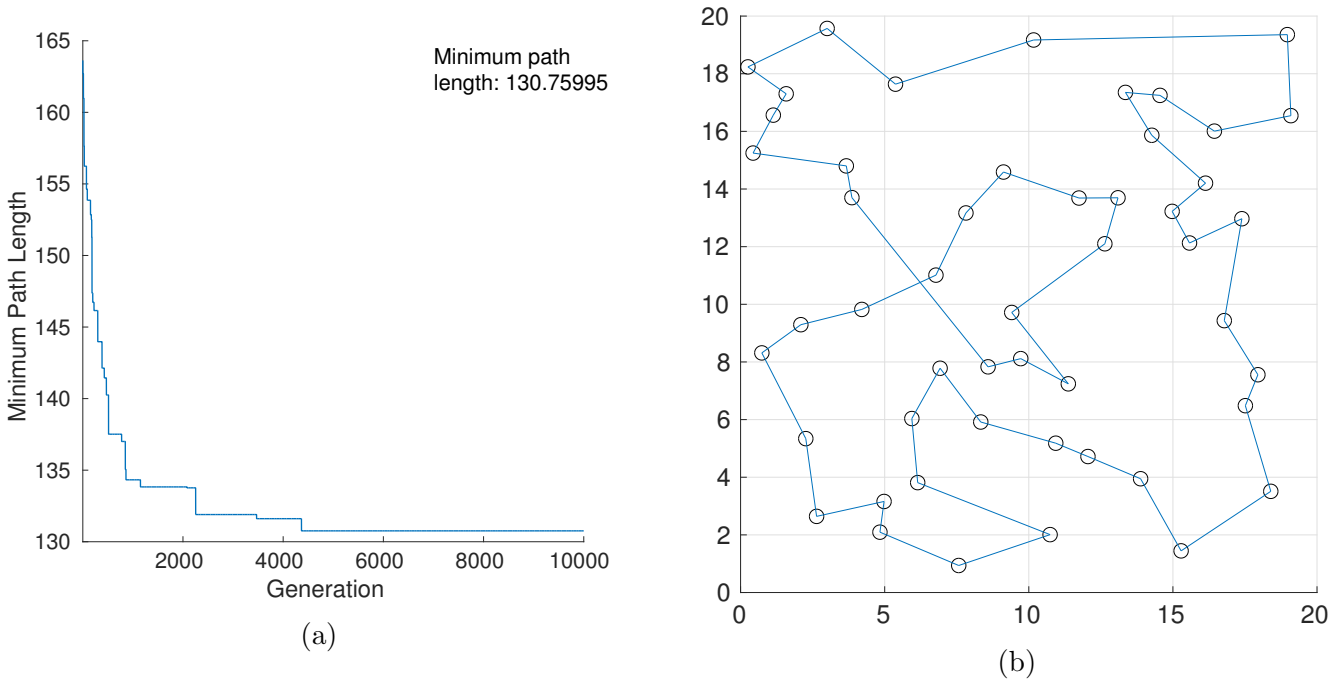


Figure 3: Results of `GA21d` for TSP with initial population of nearest-neighbour paths with 3 swap mutations. Left panel: minimum path length vs. generations. Right panel: best path found (length $\simeq 130.76$).

Problem 2.2 - Particle swarm optimisation

2.2(a)

MATLAB script `PSO22a` in folder `2.2/2.2a` contains an implementation of a standard PSO algorithm for function minimisation, with inertia weights and no craziness operator.

The algorithm was used to find the minimum of the function

$$f(x,y) = 1 + (13 + xy^3 + 5y^2 2y)^2 + (29 + x + y^3 + y^2 14y)^2 \ ,$$

with x and y real values in $[-10, 10]$.

After a few runs to explore the (rather small) parameter space, the settings in Table 1 were used to find the minimum

$$(x^*, y^*)^T = (5.00, 4.00)^T \ , \qquad \text{with} \qquad f(x^*, y^*) = 1 \ .$$

Swarm size	Number of iterations	α	δt	c_1	c_2	Maximum speed	Starting inertia weight	Inertia weight decrease rate	inertia weight inertia weight
20	1000	1	1	2	2	20 ($\alpha/\delta t$)	1.4	0.99	0.3

Table 1: PSO settings and parameters used in `PSO22a`.

2.2(b)

The PSO was then modified to handle integer programming (see script `PSO22b` in folder `2.2/2.2b`). That is, when the objective function is evaluated, particle positions are temporarily rounded to the nearest integer values, whereas floating point values are using in all other stages of the computation.
The script was then used to find the minimum of the function

$$f(\boldsymbol{x}) = -(15\ 27\ 36\ 18\ 12)\ \boldsymbol{x} + \boldsymbol{x}^T \begin{pmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{pmatrix} \boldsymbol{x}\ ,$$

where $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5)^T$, and $x_i \in \{30, 29, \ldots, 29, 30\} \in \mathbf{Z}$.
Using parameters in Table 2, two minima (claimed to be global) were found:

$$\boldsymbol{x}^*_{(1)} = (0, 11, 22, 16, 6)^T\ , \quad \text{and} \quad \boldsymbol{x}^*_{(2)} = (0, 12, 23, 17, 6)^T\ , \qquad \text{both with } f(\boldsymbol{x}^*_{(i)}) = -737\ .$$

Swarm size	Number of iterations	α	δt	c_1	c_2	Maximum speed	Starting inertia weight	Inertia weight decrease rate	inertia weight inertia weight
50	2000	1	1	2	2	20 ($\alpha/\delta t$)	1.4	0.99	0.3

Table 2: PSO settings and parameters used in `PSO22b` (integer programming).

Problem 2.3 - Optimisation of braking systems