Stochastic optimization algorithms 2020
Home problems, set 2

David Tonderski (davton)
199812101773

Problem 2.1, The traveling salesman problem (TSP)

a Overall, there are $N!$ possible paths (the number of permutations of N objects). Each path runs through the cities in the same order as $N-1$ other paths, which can be understood by "shifting" the path N times to generate the original path, e.g. (1,2,3)->(3,1,2)->(2,3,1)->(1,2,3). This brings down the number of distinct paths to $\frac{N!}{N} = (N-1)!$. However, each of the paths that are left is also equivalent to one other by reversal, e.g. (1,2,3) is equivalent to (3,2,1). Therefore, the final number of distinct paths is $\frac{(N-1)!}{2}$.

b,c The GA program is *GA21b.m*, while the ACO program is *AntSystem.m*.

d,e The figures below show the shortest obtained path and the corresponding path length for the GA (figure 1) and ACO (figure 2, as well as one nearest-neighbor path (figure 3).

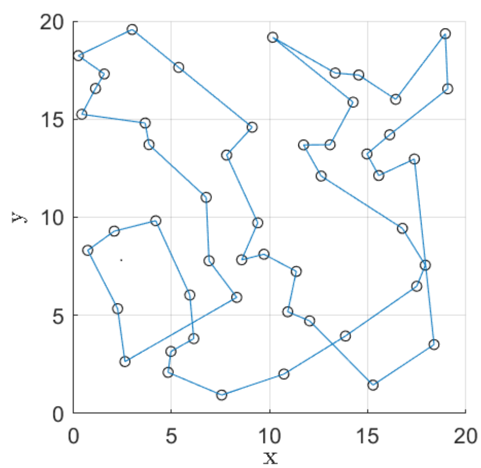

Figure 1: The shortest path obtained with the GA, with path length 142.4501.

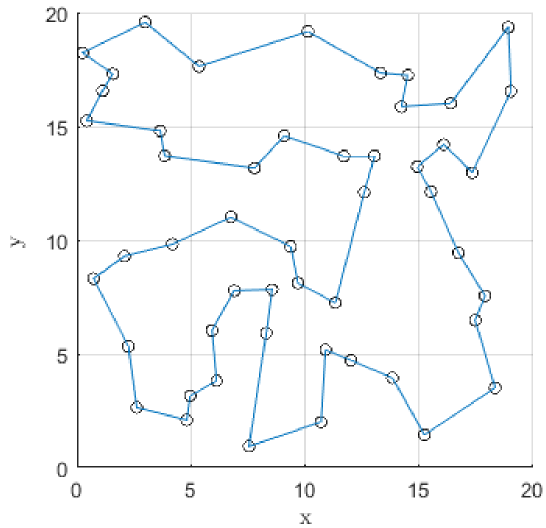

Figure 2: The shortest path obtained with the ACO, with path length 122.6576.

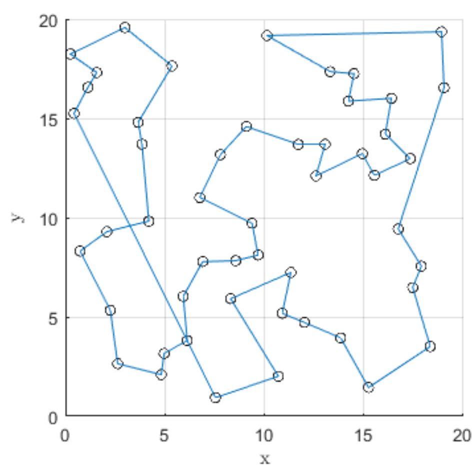

Figure 3: The nearest neighbor paths starting in city 2, with path length 137.3522.

As expected, the GA performs worse than the NN, and the ACO is the best approach.

Problem 2.2, Particle swarm optimization

The main program is *PSO22.m*. The fact that the PSO finds different minima in different runs was solved by running the algorithm 30 times. For each run, the minimum found was saved if it hadn't been found previously. This resulted in the four minima seen in figure 4 and table 1. Visual inspection of the contour plot confirms that all minima have been found.

Upon running, the main program *PSO22.m* uses the above procedure to find the minima, and then plots the contour plot.

Table 1: The minima of the function $f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$ in the interval $(x, y) \in [-5, 5]$ and the corresponding function values.

x	y	f
3	2	0
3.584428	-1.848127	0
-3.779310	-3.283186	0
-2.805118	3.131313	0

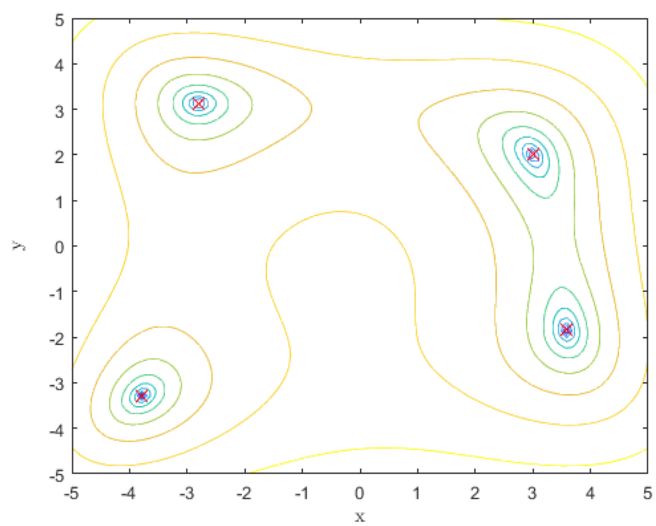

Figure 4: Contour plot of the function $\ln(0.01 + f(x, y))$, where $f(x, y) = (x^2+y-11)^2+(x+y^2-7)^2$, in the interval $(x, y) \in [-5, 5]$. Minima are marked out with red crosses.

Problem 2.3, Optimization of braking systems

The program *GA23.m* is used to optimize a network with 8 neurons in the hidden layer. It implements a standard genetic algorithm. The fitness is defined as d when $d < L$, where d is the horizontal distance driven by the truck before termination and L is the horizontal slope length. If the network does not terminate prematurely, the fitness is instead defined as $L\bar{v}$, where $\bar{v}$ is the average speed of the truck. This leads to a big jump in fitness when the network learns to drive the whole distance, and ensures it never unlearns that, which is exactly what we are after. Upon running *GA23.m*, the optimization is carried out and a plot showing the training and fitness validations as a function of the number of generations is drawn.

The program *TestProgram.m* allows for testing the best trained neural network on any slope. By default, the testing is done on the first training slope. This can be changed by editing the GetSlopeAngleFunction on line 7 of the file. The program draws the slope angle, pedal pressure, gear, speed and brake temperature as functions of time. If the truck does not complete the slope (i.e. it terminates before $x = L$), a message is printed.

For the best found network, the training and validation fitness as a function of the generation number is shown in figure 5.

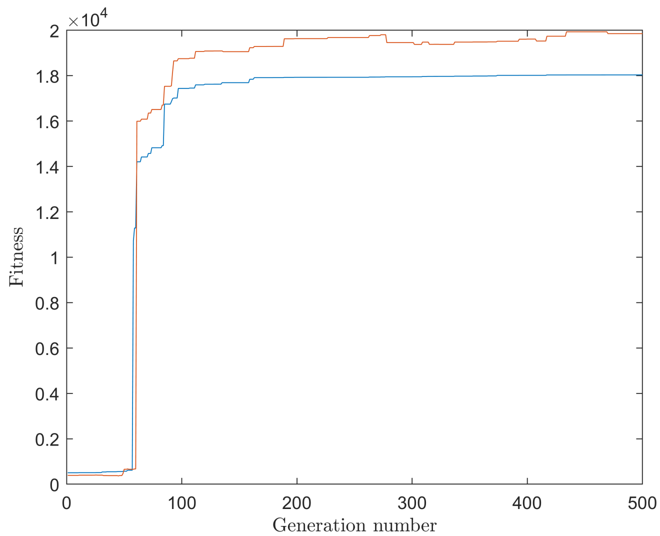

Figure 5: Training and evaluation fitness as a function of the number of generations.

1 2.4, Function fitting using LGP

The program *LGP24.m* is used to find the best chromosome. With 4 variable registers and 3 constant registers, using the constants $\{1, 3, 10\}$, the best found function was

$$g(x) = \frac{12 + 45x + 117x^2 + 108x^3}{16 + 120x^2 + 180x^3 + 108x^4}. \tag{1}$$

The program *TestLGPChromosome.m* plots the above function against the known function data and displays the function equation as well as the corresponding error.