Home problem 1

FFR105 Stochastic optimization algorithms

Rebecca Svensson
960427-3061
rebsven@student.chalmers.se

October 16, 2019

1 Problem 2.1

1.1 a)

Consider the general case of the travelling salesman problem with N different cities. For a problem like this, the total number of paths is $N!$. Since the paths are undirected a path is equal to itself in reverse, e.g (1, 2, 3, 4, 5) and (5, 4, 3, 2, 1) are equal. This means that only half of these $N!$ paths are distinct paths. Furthermore, if the starting node of a specific path is shifted but the order of visitation is kept the same, the new path is equal to the original one. This is the case for (1, 2, 3, 4, 5) and (2, 3, 4, 5, 1). There are N possible ways to shift the starting node, which means that for each path $\frac{N!}{2}$, N of them are equal. Therefore the total number of distinct paths is $\frac{N!}{2 \cdot N}$.

1.2 b)

When solving the travelling salesman problem with a genetic algorithm the population size was set to 40, the mutation probability to 2.5% and the tournament selection parameter was set to 75% with a tournament of 10 individuals. Each generation the five best individual was copied to the next generation. A long run, with 5000 generations and 500 reinitialisations, resulted in a shortest path with a length of 132.40822.

Matlab program information: The main file is GA21b.m. This is run with no input and the output is the length of the shortest path found for the travelling salesman problem along with a plot of the path that is updated when a better path is found.

1.3 c)

An ant system was used to solve the travelling salesman problem. When running the algorithm 50 ants was used with $\alpha = 1.0$, $\beta = 5.0$ and $\rho = 0.5$. The length of the shortest path received was 121.7534 and this was received in 12051 iterations. The program reaches below 124 very fast, about 20 to 50 iterations is enough. To get below 123 around 5000 iterations is needed, and to get below 122 about 10000 iterations is needed.

Matlab program information: The main file is AntSystem.m. This is run with no input and the output is the length of the shortest path found for the travelling salesman problem along with a plot of the path that is updated when a better path is found.

1.4 d)

By calculating the length of the nearest neighbour path 50 times, a mean value of the path length was determined to be 158.7158. The genetic algorithm found a path of length 132.40822 which is clearly better than the nearest neighbour path.

Matlab program information: The main file is NNPathLengthCalculator.m. This is run with no input and the output is the length of the nearest neighbour path and which node it started on.

1.5 e)

The shortest path found by the genetic algorithm had the length 132.40822, and this can be seen in figure 1.

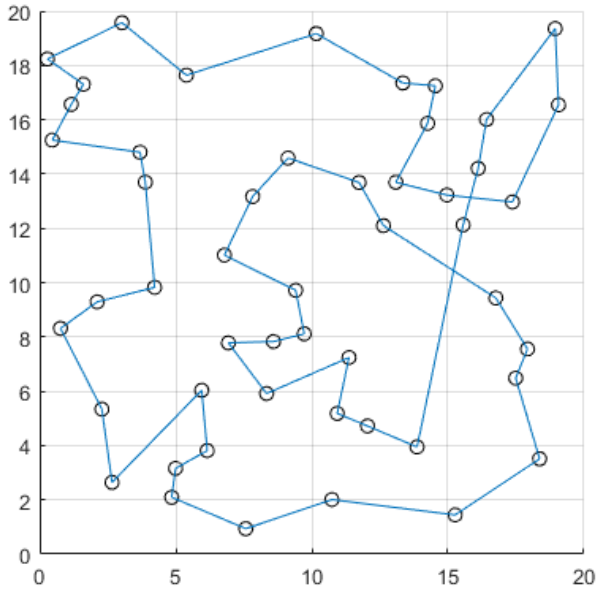

Figure 1: Path found by the genetic algorithm. The length is 132.40822.

The ant system performed better and found a path of length 121.7534, and this path can be seen in figure 2. This path can be found in the attached file BestResultFound.m.

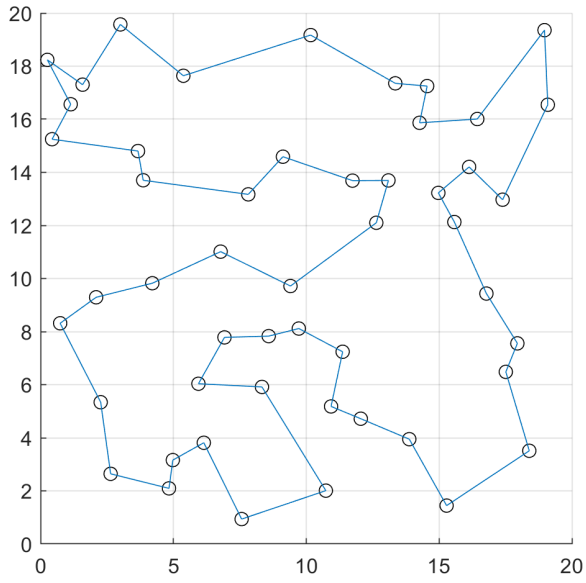

Figure 2: Path found by the ant system. The length is 121.7534.

Both of these algorithms performed better than only finding the nearest neighbour path which had the length 141.7158. However, the ant systems performance was drastically better than that of the genetic algorithm, both result wise and run time.

2 Problem 2.2

The minima of $f(x, y)$, which is given by equation 1, are to be located by a particle swarm optimisation. In figure 3 a plot of $log(0.01 + f(x, y))$ can be seen. The rough location of the minima are shown in the figure as well.

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{1}$$

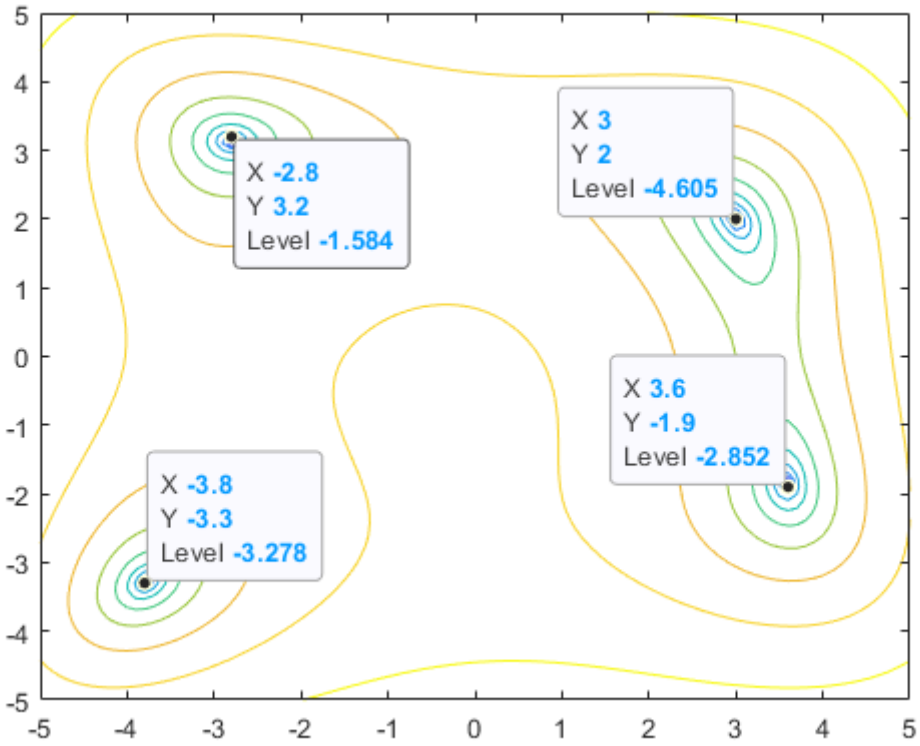

Figure 3: Plot of $log(0.01 + f(x, y))$.

By running the particle swarm optimisation a number of times, the minima are determined. The results are presented in table 1.

x	y	$f(x, y)$
-2.805118	3.131313	$7.888609 \cdot 10^{-31}$
-3.779310	-3.283185	$7.888609 \cdot 10^{-31}$
3	2	0
3.584428	-1.848127	0

Table 1: The four minima for equation 1.

Matlab program information: The main file is PSO22.m. This is run with no input and the output the x and y values of the minimum along with the function value for these x and y.

3 Problem 2.4

The Matlab program was able to find a chromosome that resulted in an error less than $2.84840 \cdot 10^{-9}$. This chromosome corresponded the function presented in equation 2. During the run three variable registers were used and three constant registers set to 1, -1, -3. In figure 4 the datapoints from evaulating the chromosome can be seen together with $g(x)$.

$$g(x) = \frac{1 - x^2 + x^3}{1 - x^2 + x^4} \tag{2}$$

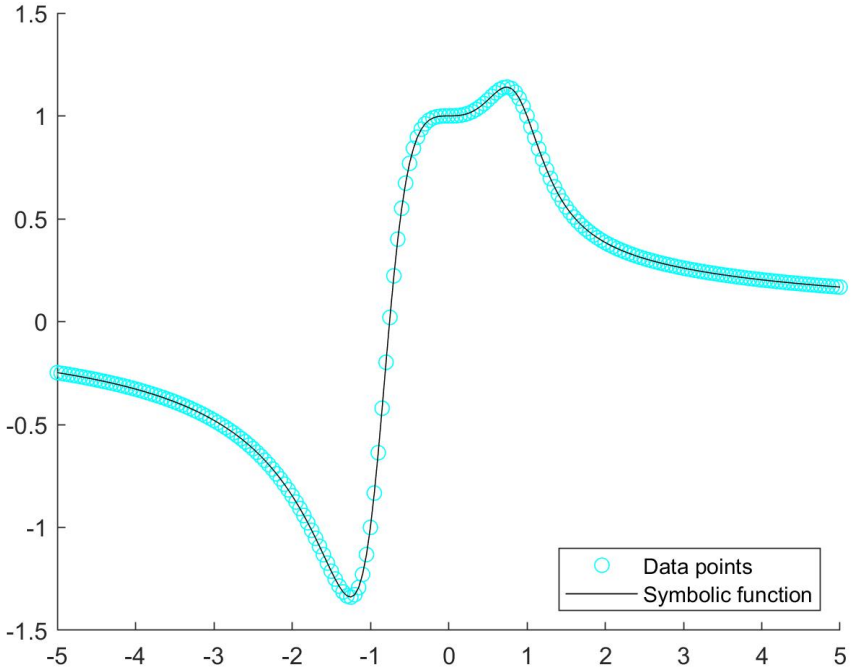

Figure 4: Plot of the found chromosome and symbolic function.

To increase the efficiency of the program, the mutation rate is varied in two different ways. A general mutation rate is used throughout the program, and when mutating a chromosome an individual mutation rate is used. The individual mutation rate is set to be the general mutation rate divided by the length of that specific chromosome. In addition to this, the general mutation rate is adjusted depending on the diversity in the population.

Matlab program information: The main file is LGP24.m. This is run with no input and the output is a plot of the wanted function as well as the plot of the best chromosome so far in this run. The generation number will be printed every 100th time, and the fitness value of a chromosome will be printed each time a new best for the run has been found. If the maximum fitness hasn't been updated for 3000 generations, then the population is reinitialised. The chromosome will be saved to a file if the error reaches below 0.01. If the fitness continues to increase after this, the new best chromosome is saved as well. When the fitness isn't increasing anymore, the best found chromosome will be decoded and the symbolic function will be printed.