HOMEWORK

Jun Cheng

March 27, 2016

Problem 1:

Problem 2:

Using PSO algorithm, we can find the minimizer:

$$
\begin{aligned}
x_0 &= 0.000279321964182 \\
x_1 &= 0.000193196456243 \\
f(x_0, x_1) &= 2.28836604776e-07
\end{aligned}
$$

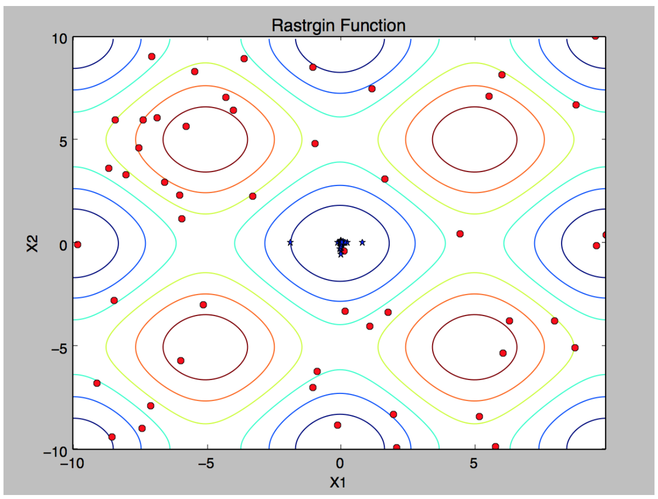

Figure 1: PSO Algorithm (problem 2): circle points are randomly generated 50 initial points. Stars indicate the positions after 50 iterations.

Problem 3:

Using PSO algorithm, we can find the maximizer:

$$
\begin{aligned}
x_0 &= -5.02482780601 \\
x_1 &= 5.02524813509 \\
f(x_0, x_1) &= -40.5025451078
\end{aligned}
$$

In fact, there are several other global maximizers. PSO method will converge to different global maximizer depending on the initial points which are randomly chosen.

Problem 4:

Population size: 50
Number of iterations: 50

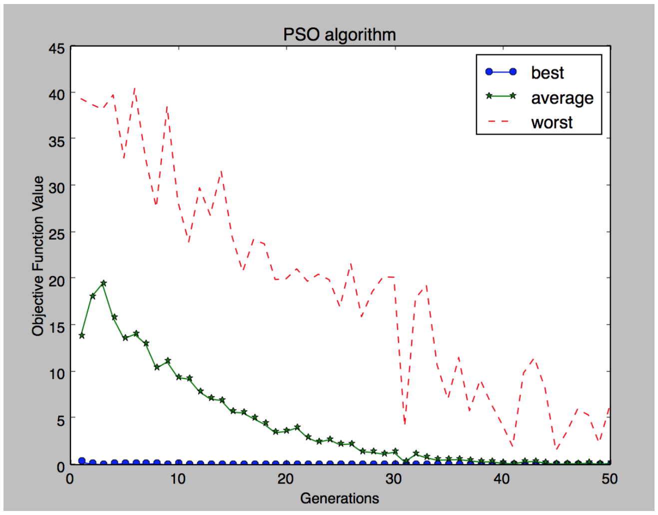

Figure 2: PSO Algorithm (problem 2): plots of the best, average, and the worst objective function values in the population for 50 generations

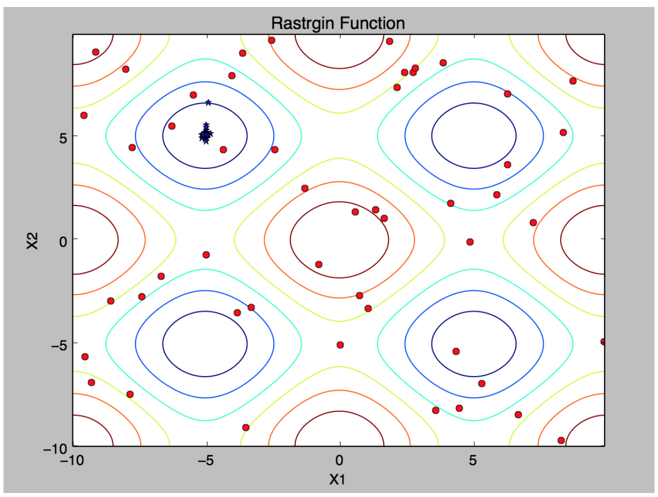

Figure 3: PSO Algorithm (problem 3): circle points are randomly generated 50 initial points. Stars indicate the positions after 50 iterations.

For canonical number genetic algorithm, the minimizer is:

$$x_1 = 0.0408935546875$$
$$x_2 = 0.0390625$$
$$f(x_1, x_2) = 0.00634456702034$$

For real number genetic algorithm, the minimizer is :

$$x_1 = 0.018313265874$$
$$x_2 = 0.0286761643909$$
$$f(x_1, x_2) = 0.00229673023909$$

Problem 5:

The shortest path is shown in Figure 9, and the shortest distance is : 37.7222579198

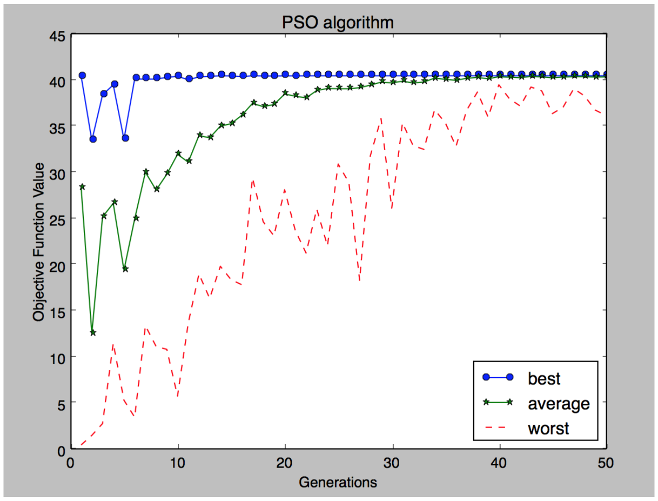

Figure 4: PSO Algorithm (problem 3): plots of the best, average, and the worst objective function values in the population for 50 generations

Problem 6:

Matlab code for Problem 6:

```
f = [7  10  14  8   7   11  12  6   5   8   15  9   ];
A = [];
b = [];
Aeq = [1    1   1   1   0   0   0   0   0   0   0   0;
       0    0   0   0    1    1   1   1    0   0   0    0 ;
       0    0   0   0    0    0   0   0    1    1   1    1;
       1    0   0   0    1    0   0   0    1    0   0    0;
       0    1   0   0    0    1   0   0    0    1   0    0 ;
       0    0   1   0    0    0   1   0    0    0   1    0 ;
       0    0   0   1    0    0   0   1    0    0   0    1  ];
beq = [30   40  30  20  20  25  35];
lb = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
ub = [] ;
x = linprog(f, A, b, Aeq, beq, lb, ub)
```

The output:

```
x =

    4.8834
    5.1166
    8.7304
   11.2696
    0.0000
    0.0000
   16.2696
   23.7304
   15.1166
   14.8834
    0.0000
    0.0000
```

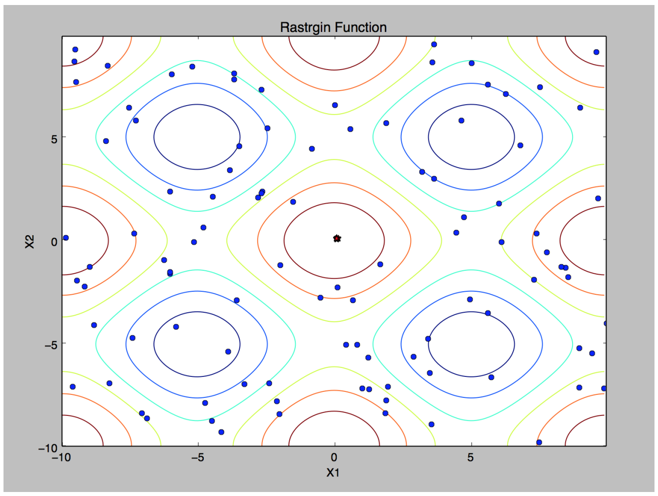

Figure 5: Canonical Genetic Algorithm (problem 4): circle points are randomly generated 50 initial points. Stars indicate the positions after 50 iterations.

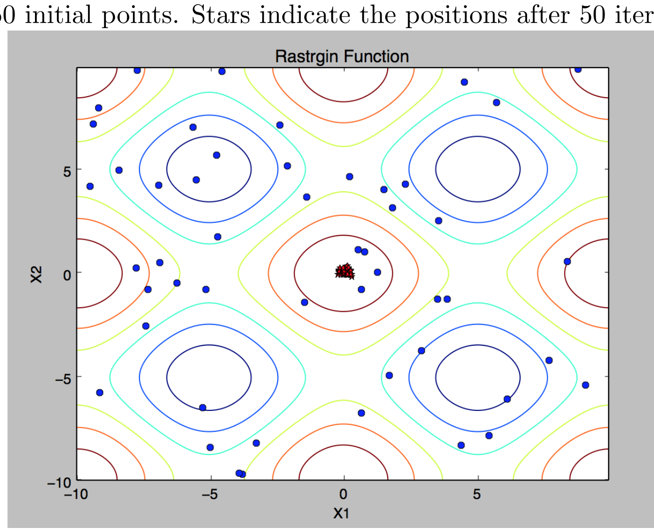

Figure 6: Real Number Genetic Algorithm (problem 4): circle points are randomly generated 50 initial points. Stars indicate the positions after 50 iterations.

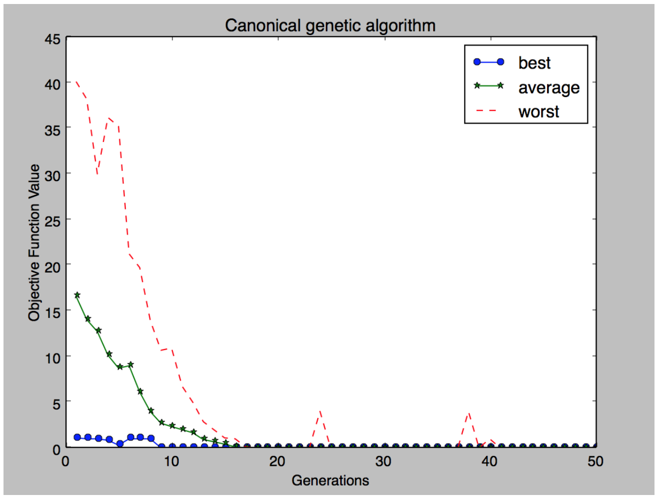

Figure 7: Canonical Genetic Algorithm (problem 4): plots of the best, average, and the worst objective function values in the population for 50 generations

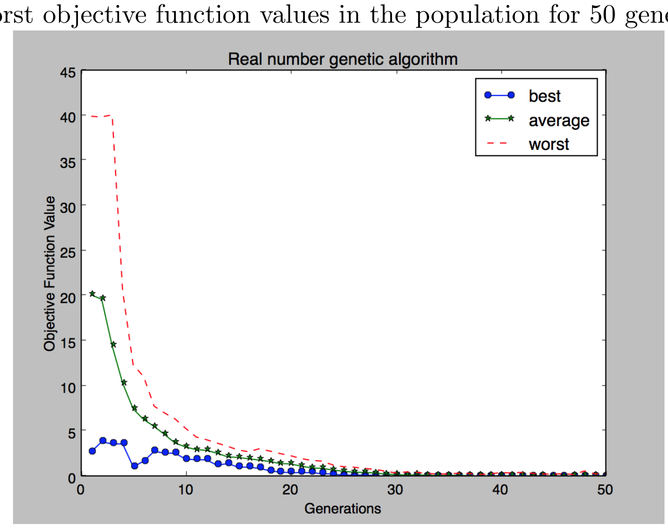

Figure 8: Real Number Genetic Algorithm (problem 4): plots of the best, average, and the worst objective function values in the population for 50 generations

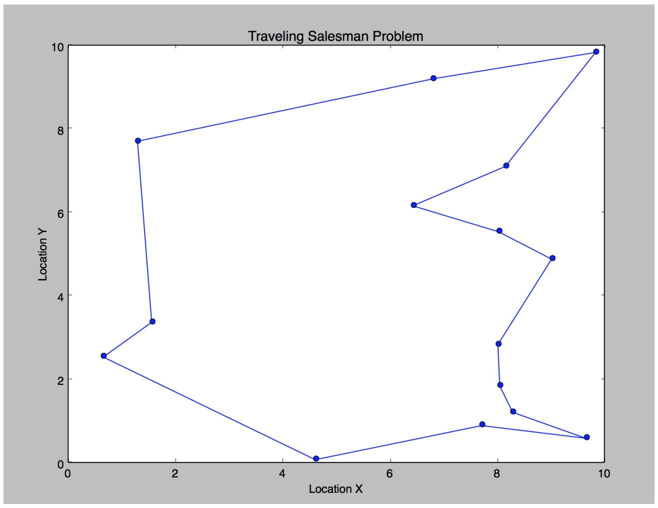

Figure 9: Traveling salesman problem (problem 5): plots of the shortest distance path

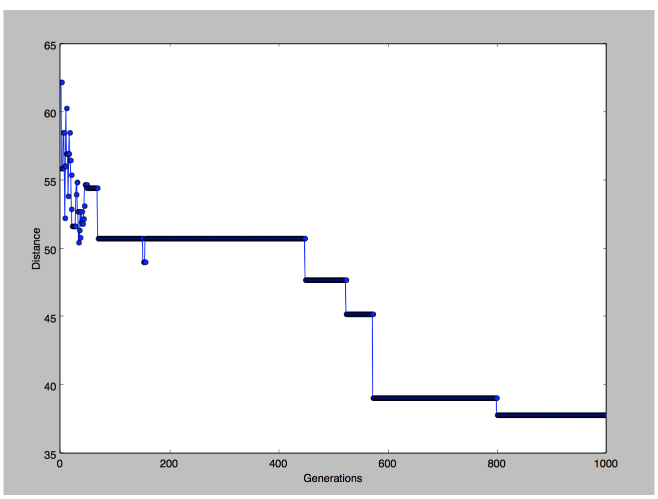

Figure 10: Traveling salesman problem (problem 5): plots of the shortest distance for different combinations of the population for 1000 generations