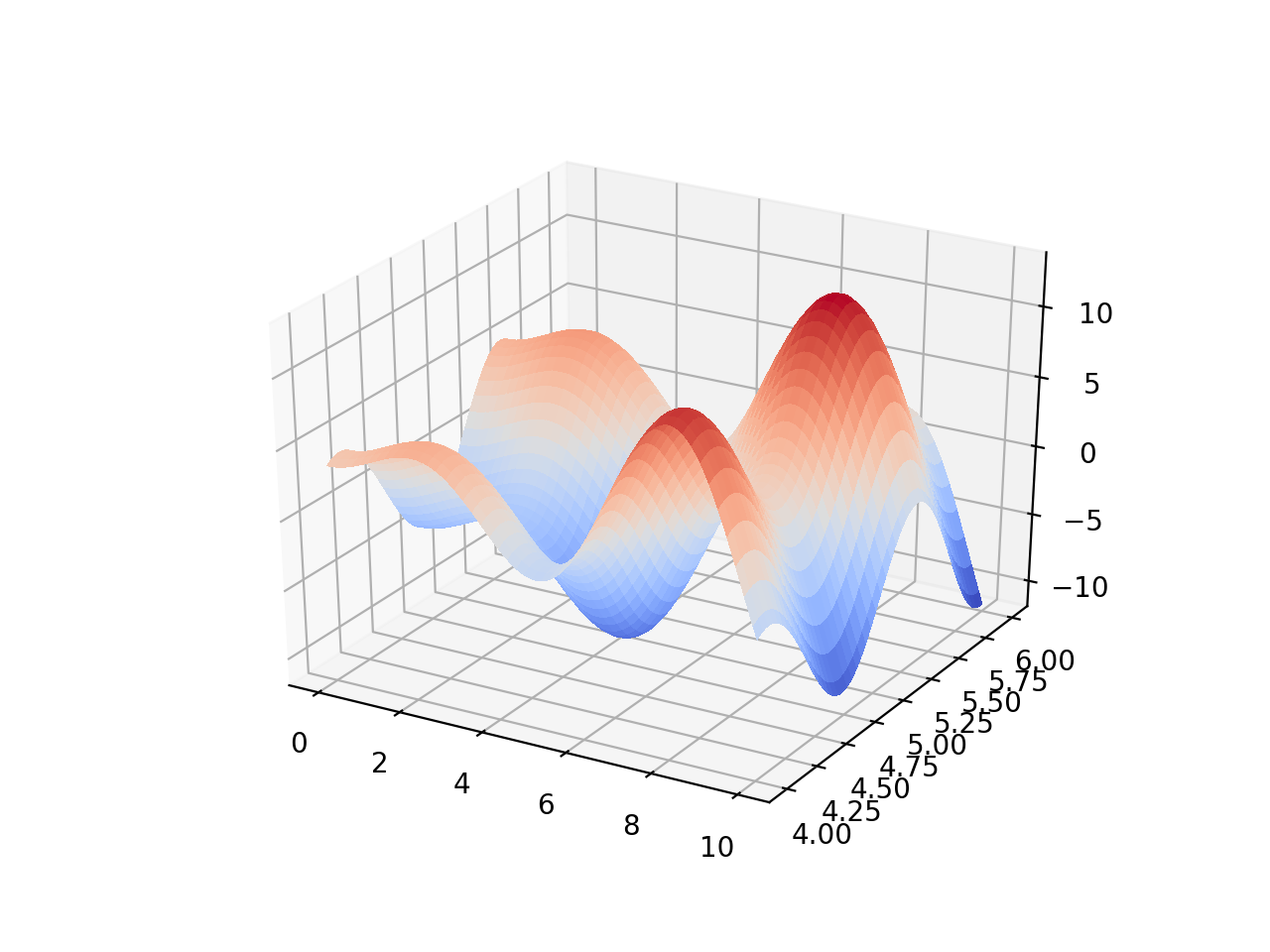
**李少琪 0860908**

**Optimization Theory and Application**

* 1. find maximum of f(x) = x1sin(x1)+x2sin(5x2), constraint set = {**x**: 0<=x1<=10, 4<=x2<=6}

1. Real-number genetic algorithm

Parameters: **population size(N)**, **mutation rate(p\_m)**, crossover rate(p\_c), generations(g\_iter), fitness function(func)

Fitness function: f(x) function we want to find out maximum

Step 1: Initial Population

任意從constraint set選出N組[x1,x2]作為P0, 用P0計算f(x), 找出最大值與其x1,x2

Step 2: Termination when population have converged.

P計算出f(x)的結果**y**, 減去其中的最小值**y** = **y**–min(**y**), 讓結果全為正數, 若結果總和為0, 則population have converged, 並停止演算法

Step 3: Selection

P計算出f(x)的結果**y**, 減去其中的最小值(全正數), 並除以結果總和**y** = **y** / sum(**y**): probability of being selected

**Roulette-Wheel selection:**

將y累加(結果:0~1), 重複N次選出**y**–random number(0~1) 中大於0的最小y, 作為mating pool(Mk)

**Tournament selection**

重複N次從y中任意取2值, 選其中較大的值, 作為mating pool(Mk)

Step 4: Crossover

如果1 random number(0~1) < crossover rate, 則從Mk任選出2

parents, alpha = random number(0~1)

offspring1 = alpha \* parent1 + (1 - alpha) \* parent2 + (random

number(0~1) - 0.5) \* (upper\_bound - lower\_bound) / 10

offspring2 = alpha \* parent2 + (1 - alpha) \* parent1 + (random

number(0~1) - 0.5) \* (upper\_bound - lower\_bound) / 10

Projection:

如果offspring超過bound, offspring = bound

將原Mk中的parent改為offspring

Step 5: Mutation

重複N次, 當random number(0~1) < mutation rate, Mk中的第n

個(Mk[n]),

Mk[n] = alpha \* Mk[n] + (1 - alpha) \* (lower\_bound + random

number(0~1) \* (upper\_bound - lower\_bound))

Pk+1=Mk

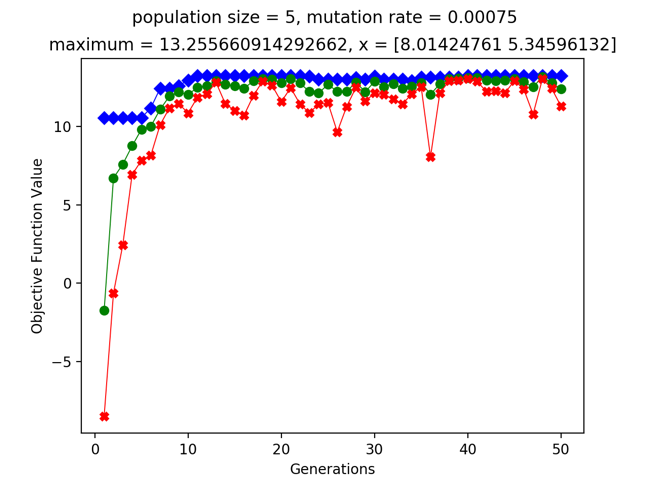
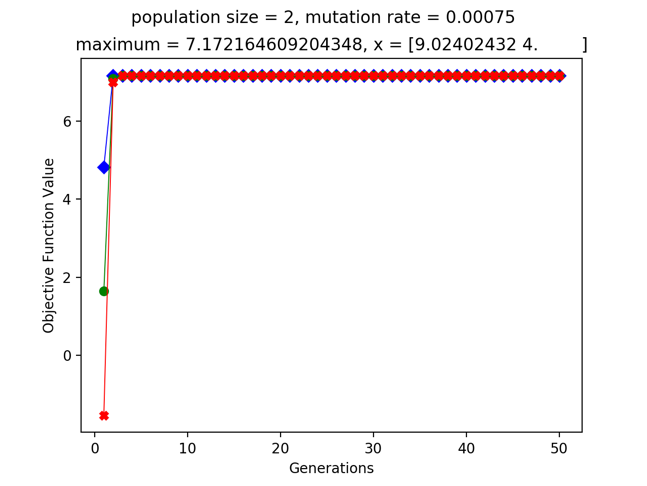
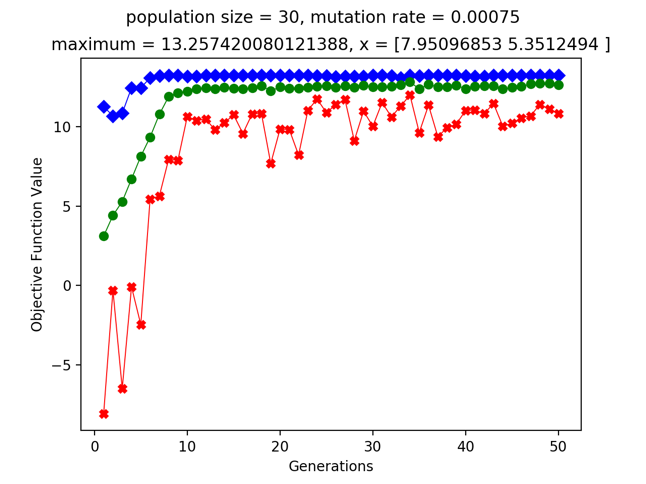
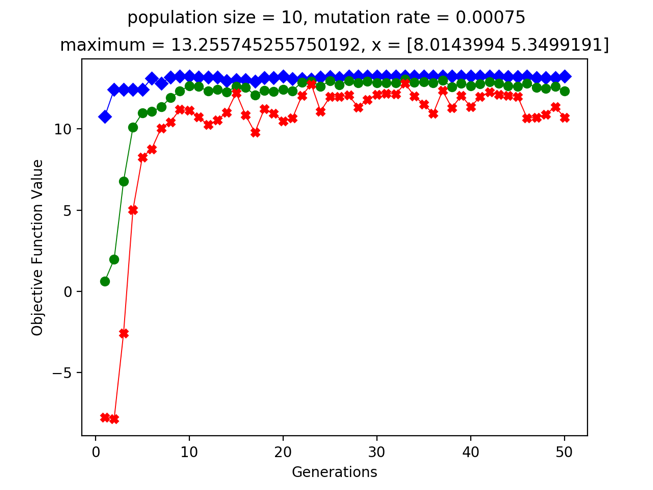
Step 6: Compute fitness function by Pk+1

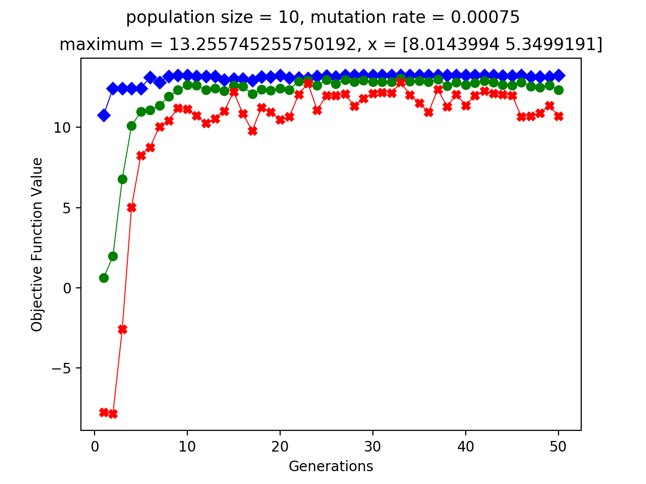
y = f(Pk+1), 找出其中的最大值和其x1,x2

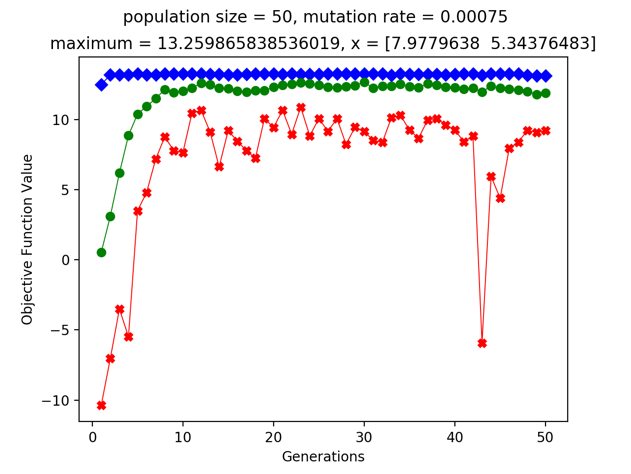
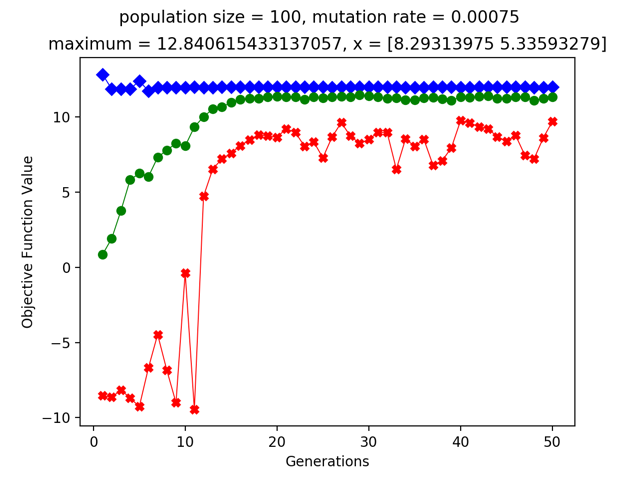
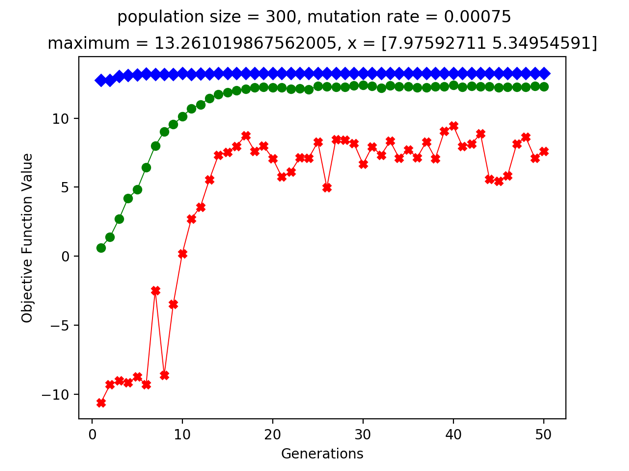
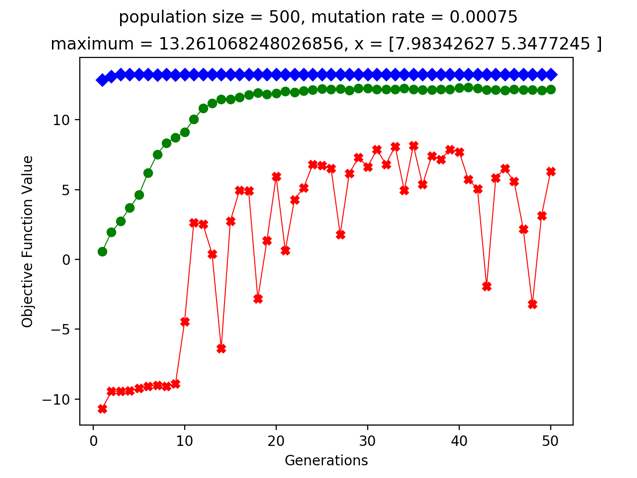
重複Step 2~Step 6

1. 實作結果
2. Population size參數設定

固定crossover rate = 0.75, mutation rate = 0.0075, generations = 50

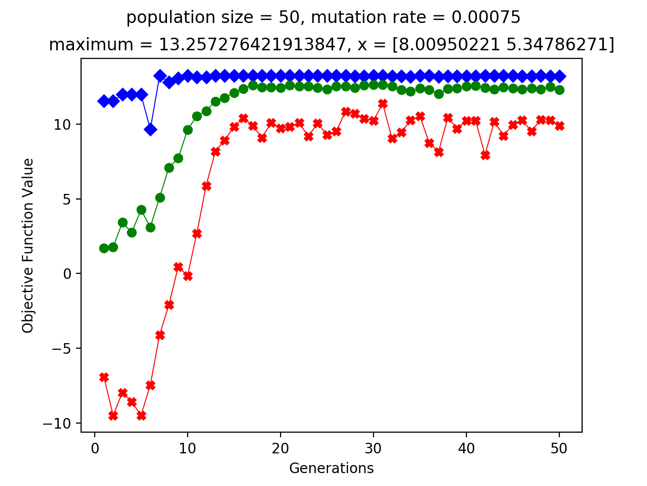
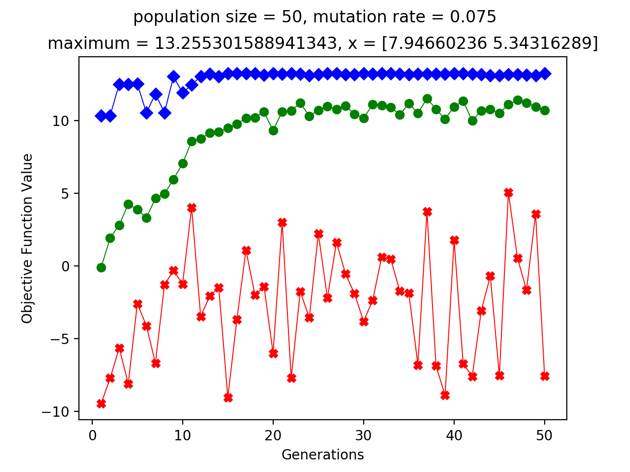
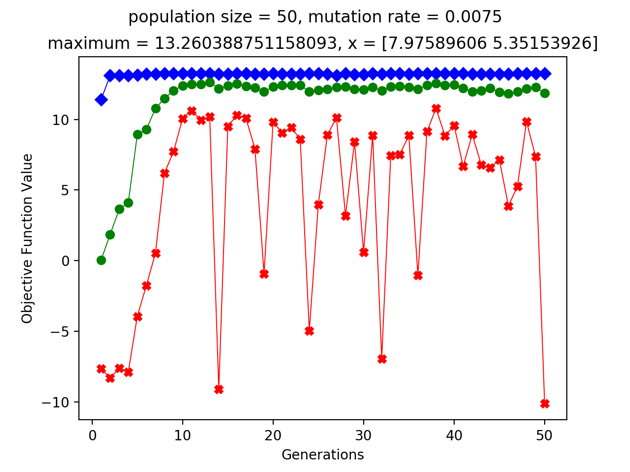
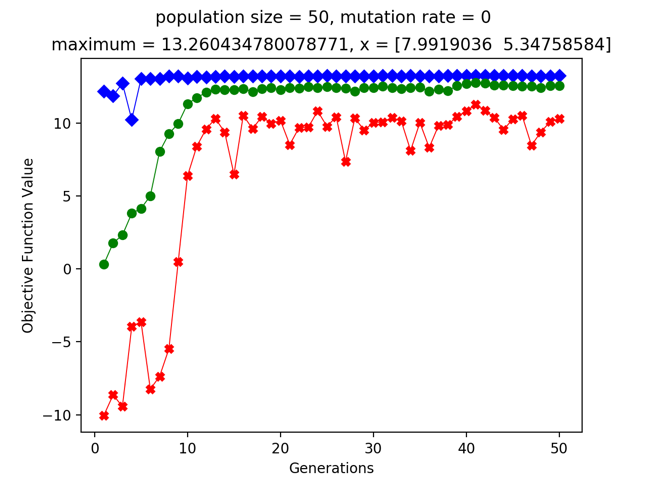
Population size = [2, 5, 10, 30, 50, 100, 300, 500]

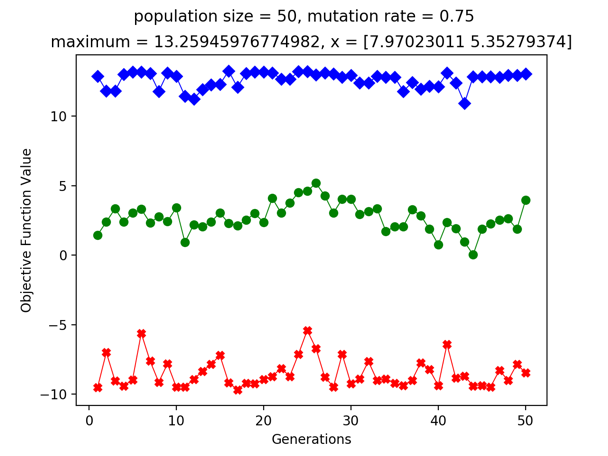
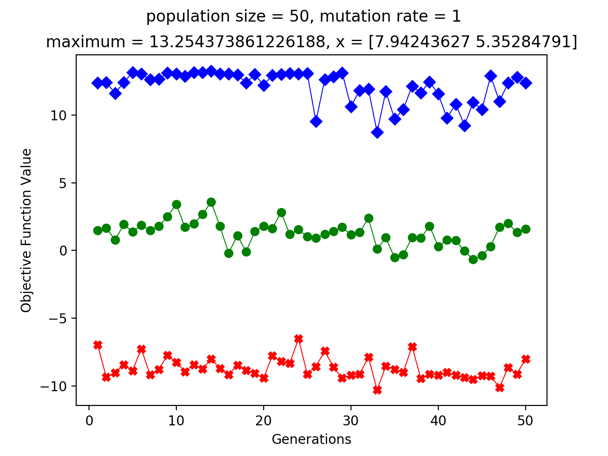




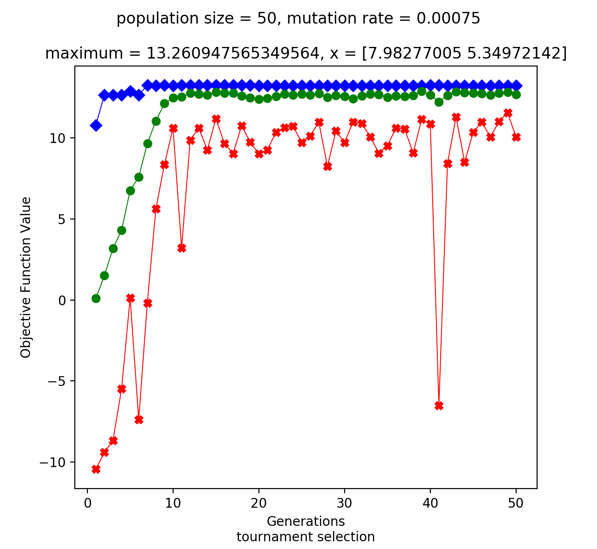
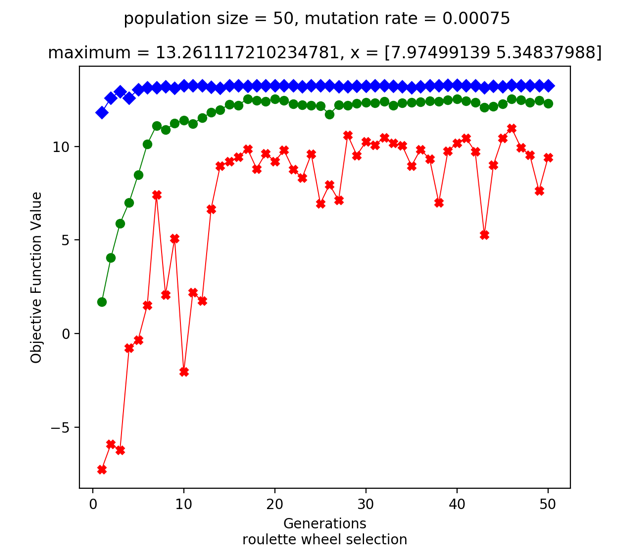
2. Mutation rate參數設定

固定crossover rate = 0.75, population size = 50, generations = 50

Mutation rate = [0, 0.00075, 0.0075, 0.075, 0.75, 1]



1. 比較roulette wheel和tournament selection

固定crossover rate = 0.75, population size = 50, generations = 50, mutation rate = 0.0075

1. 分析
2. Population size參數設定

由上述結果可以看出，當population size越小時，越容易在還沒找到最大值以前就已經收斂了，但當population size越大時，容易找到最大值，不過population也容易包含到其他較小的值，在運算中，可能會浪費許多空間。

1. Mutation rate 參數設定

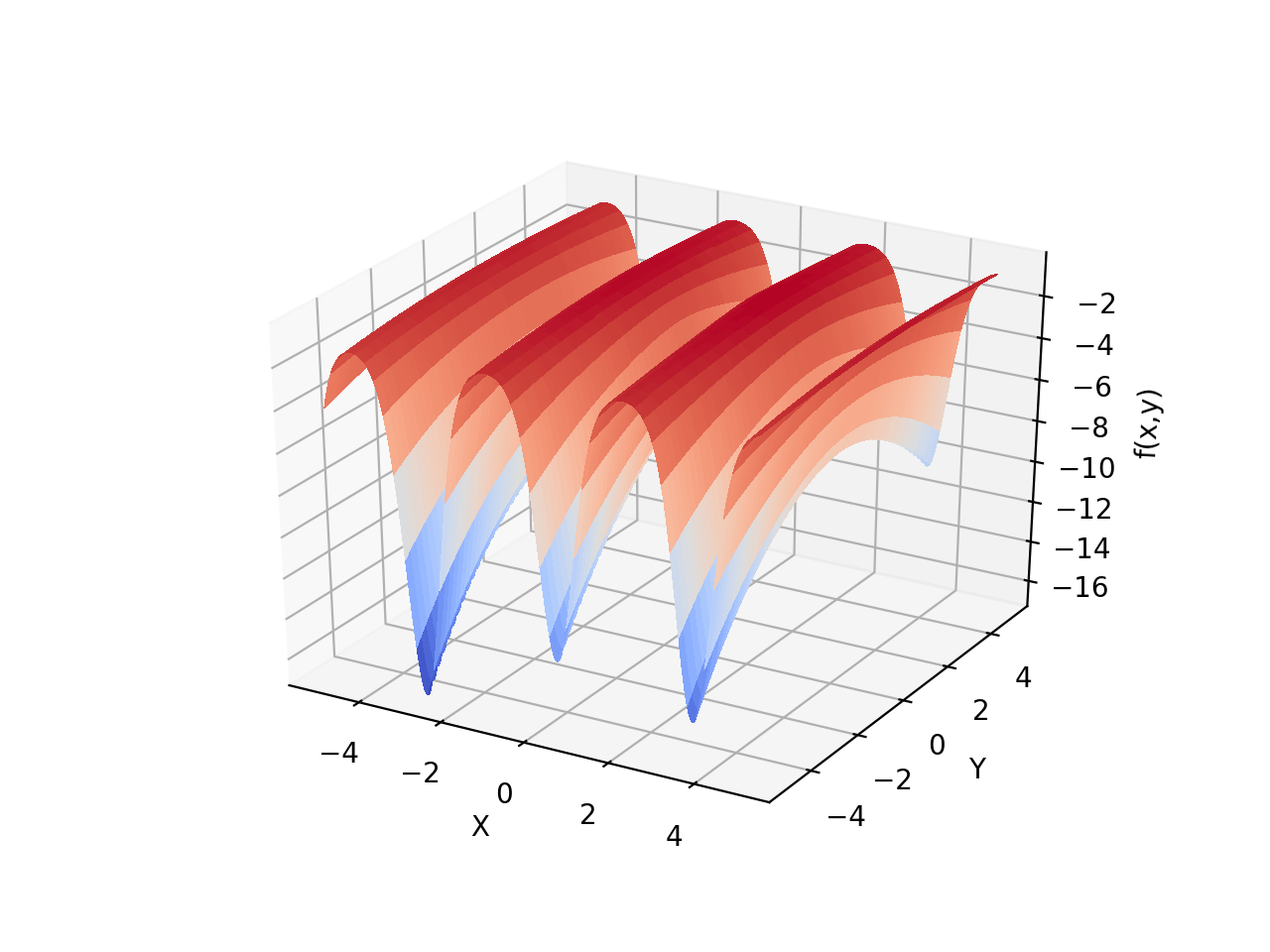
Mutation rate越大時，表示對原本資料產生的變異越大。由上述結果可以得知，當Mutation rate越大時，與前一population差異過大，會導致每一generation之間毫無意義，但如果Mutation rate為零時，可能會難以跳脫出前一population，而找不到global maximum。

1. 比較roulette wheel和tournament selection

roulette wheel是按chromosome對應的函數值給予被選取的機率之後做選擇，tournament則是任選二chromosome後選出較大的，就以定義上可以發現tournament是不會選到最小值的，但roulette wheel有機會選到最小值，tournament 整體選出的chromosome會比roulette wheel來得大，這個結果也可以在上述的實驗結果中發現，平均值tournament會比roulette wheel更接近最大值。

1.2

a. find minimum of , constraint set={}



1. 將Real-time genetic algorithm的maximize selection轉為minimize

selection

1. Roulette wheel selection

y = -(y – max(y))/sum(y)讓y中越小的值, 被選取的機率越大

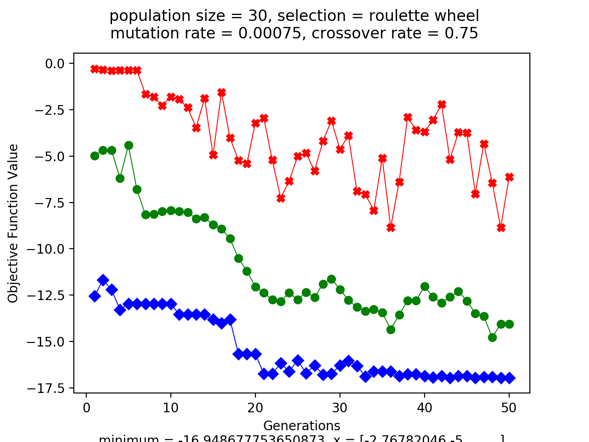
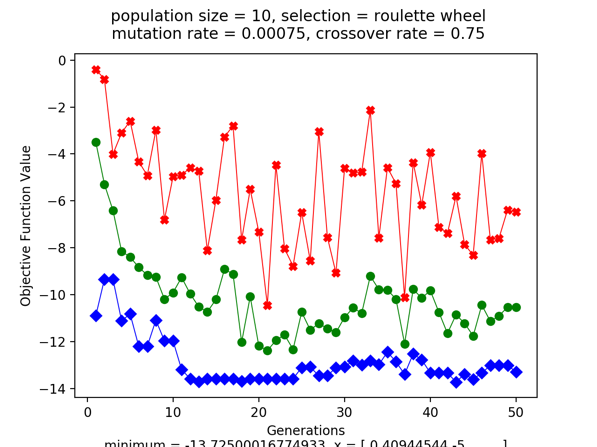
1. Tournament selection

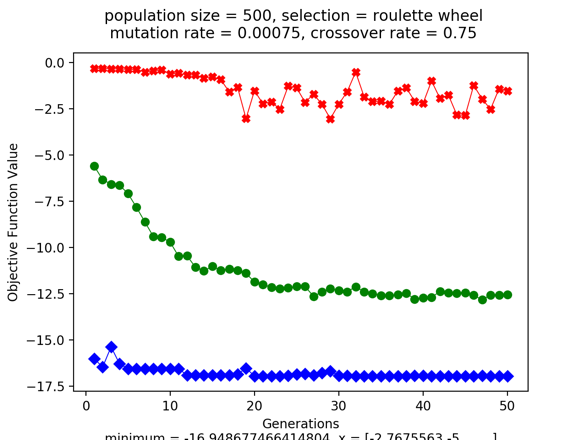
重複N次從y中任意選取2值, 以其中較小的值, 作為mating pool

1. 實作結果
2. Population size參數設定

固定crossover rate = 0.75, mutation rate = 0.0075, generations = 50

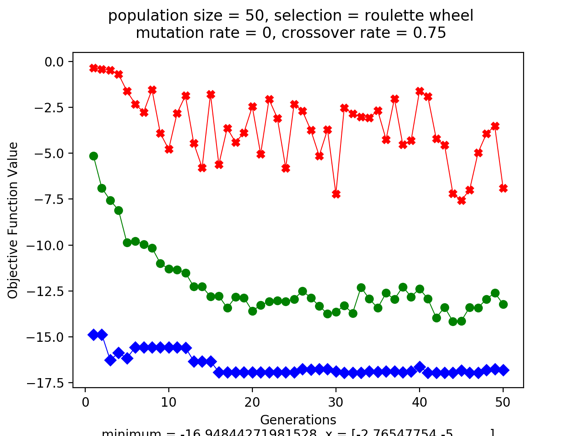
Population size = [10, 30, 100, 500]

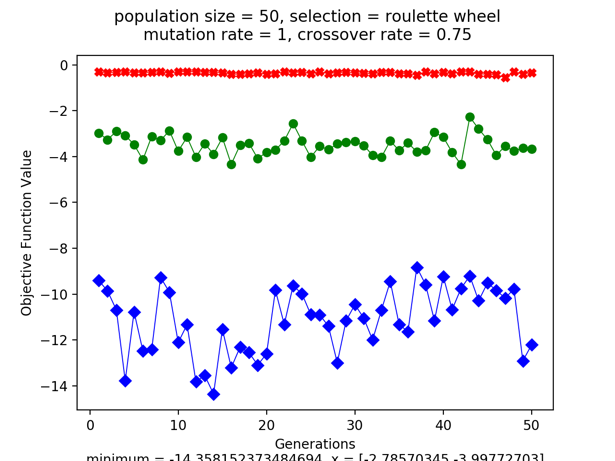
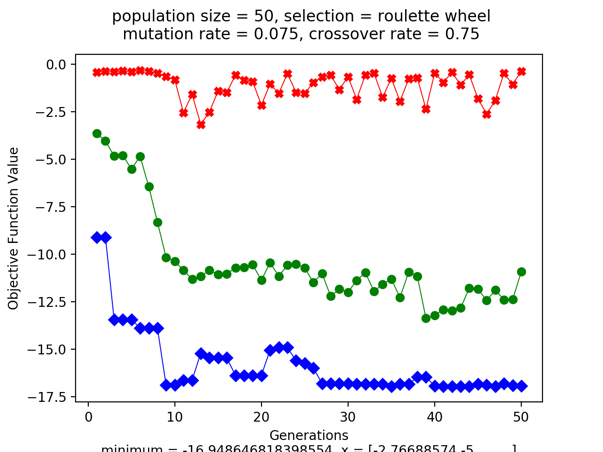




1. Mutation rate參數設定

固定crossover rate = 0.75, population size = 50, generations = 50

Mutation rate = [0, 0.0075, 0.075, 1]

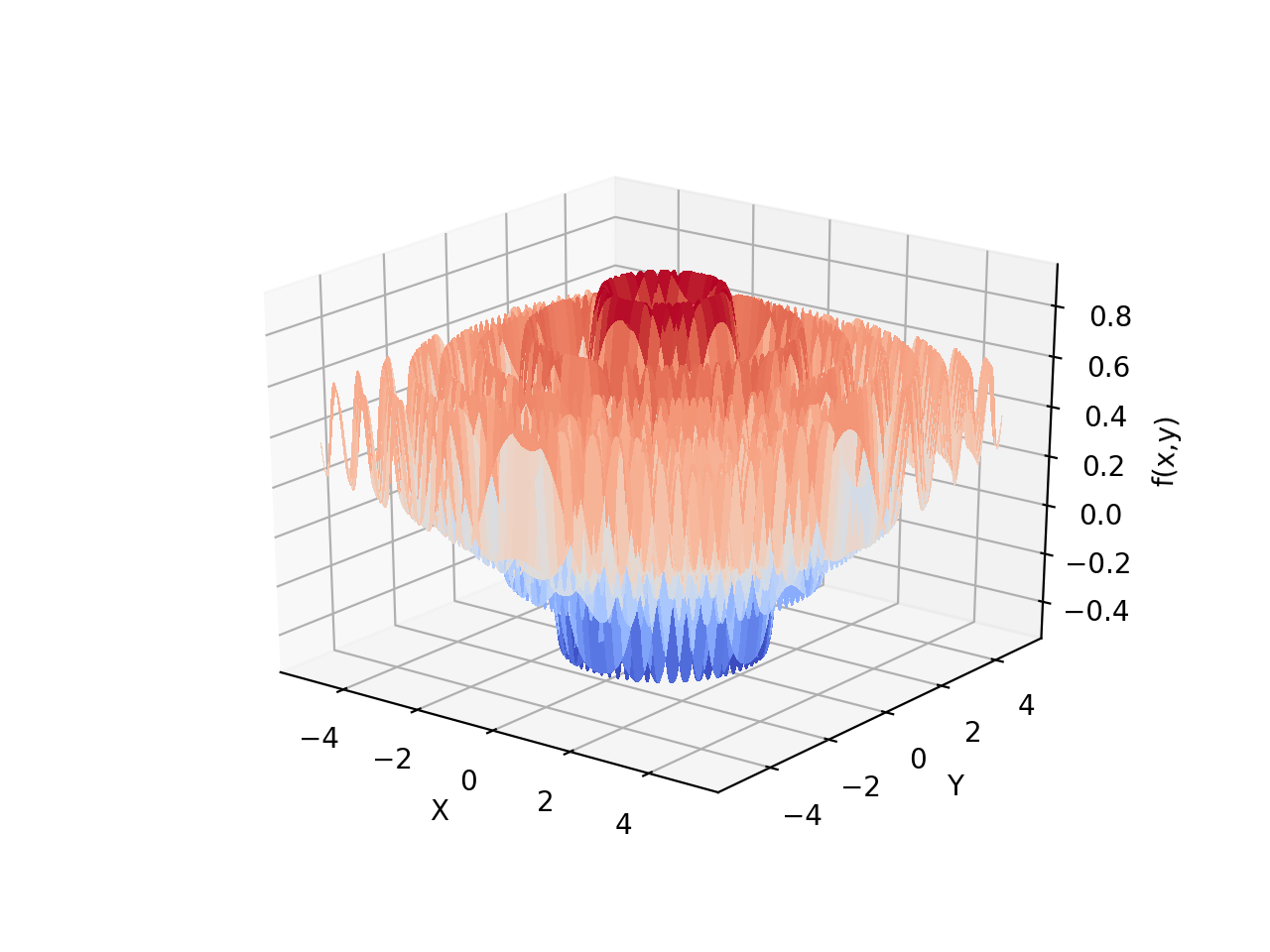


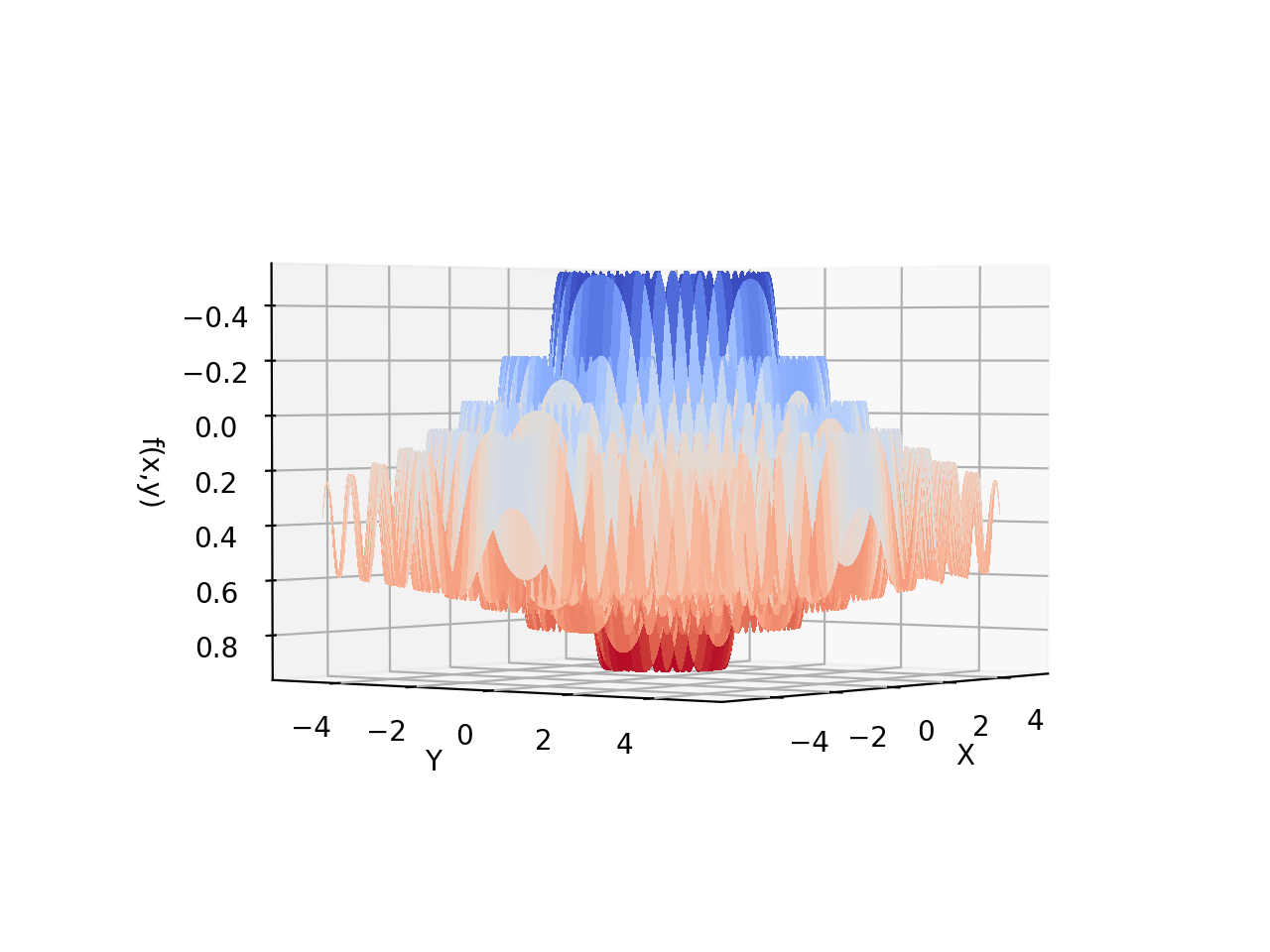
1. 分析
2. Population size參數

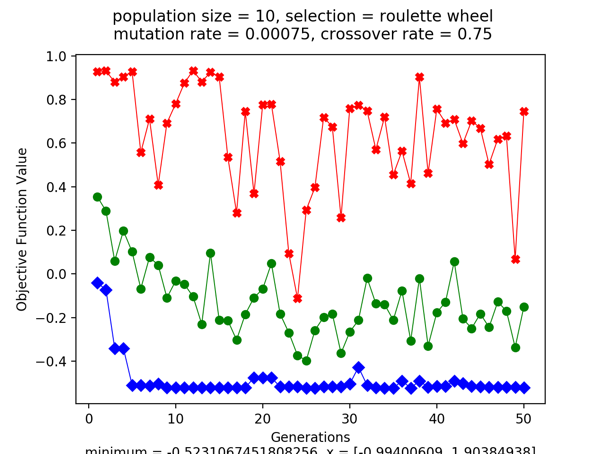
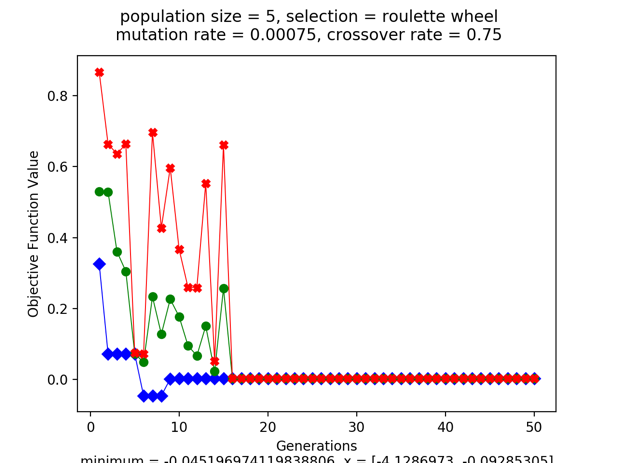
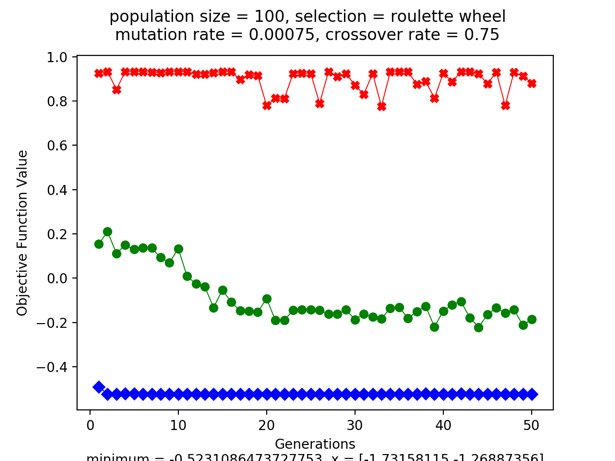
這個函數有較多的極大值，由上述結果可以發現，當population size越大時，generation的走勢趨於平緩，無較大的震盪。

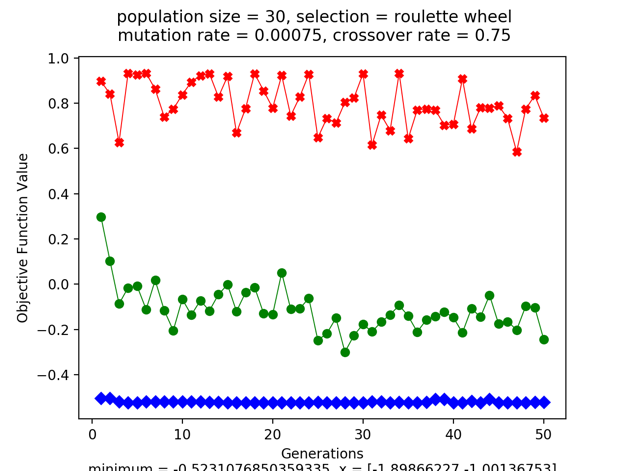
1. Mutation rate參數

由上述結果可以得知，當Mutation rate越大時，與前一population差異過大，會導致每一generation之間毫無意義，但如果Mutation rate為零時，可能會難以跳脫出前一population，而找不到global maximum。

b. find minimum of , constraint set={-}



1. 實驗結果



(2)分析

Genetic algorithm是用來找出global maximum/minimum，可以和之前學習 找local maximum/minimum比較發現，找local maximum/minimum的方

法主要是沿著可行方向找，容易停在local maximum/minimum跳不出來，而genetic algorithm是一直不斷有機率地隨機抽樣，並且嘗試做一些變異，在計算是容易跳脫local maximum/minimum，更能找到我們真正需要的global maximum/minimum。