Artificial Intelligence Homework 3　　B036060017 資管大四 謝威廷

Using Genetic Algorithm(GA), Hill Climbing(HC) and Simulated Annealing(SA)

to optimizing the function

$$f(x_1, x_2) = 21.5 + x_1\, sin(4\pi x_1) + x_2\, sin(20\pi x_2)$$

$$where\ -3.0 \le x_1 \le 12.1\ and\ 4.1 \le x_2 \le 5.8$$

required precision is six places after decimal points.

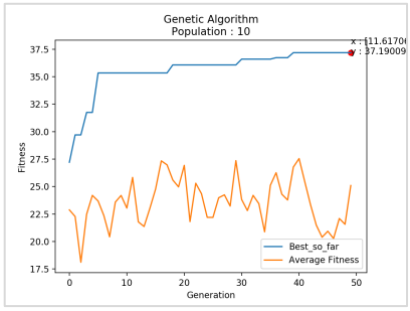
本次作業運用 python 3.6 來實作，GA 為採用 gaft 套件並加以修改（修改後檔案放置於資料夾中），HC 及 SA 則完全由自己實作，嘗試找出題目中函式的最佳解。

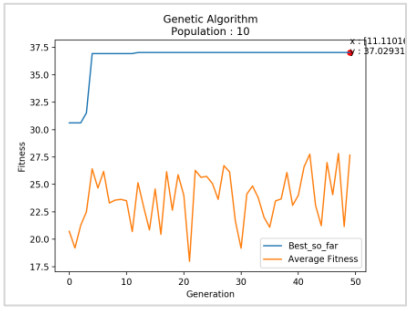
1. Result comparison among three optimizing methods

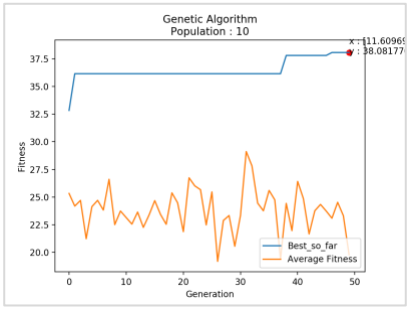
由於演算法中起始點皆為隨機，每次執行產生的最佳解皆不同，以下分別執行三次，比較各演算法運算結果。

(1) Genetic Algorithm

Population : 10, Generation : 50, Crossover : 0.8, Mutate : 0.1 (詳細參數設定於程式碼)

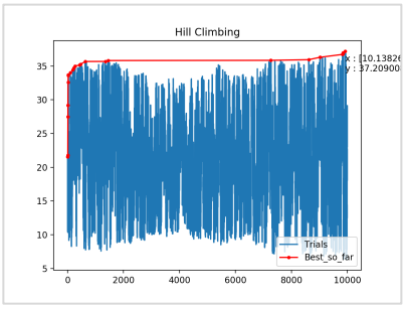


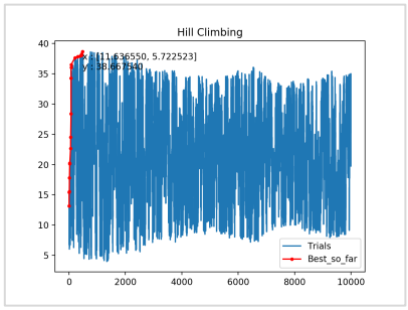


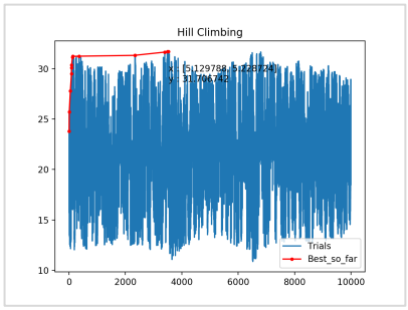


(2) Hill Climbing

Iteration : 1000

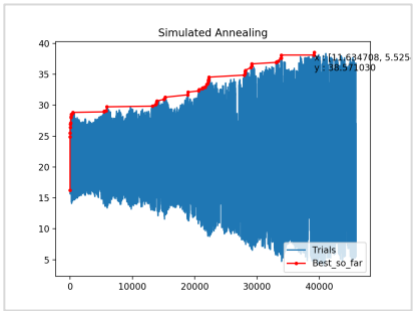


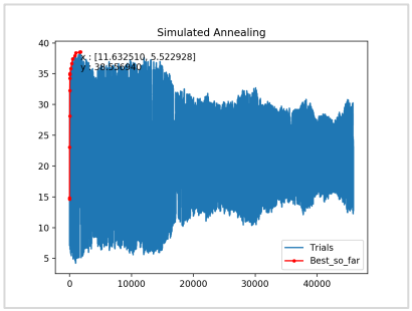


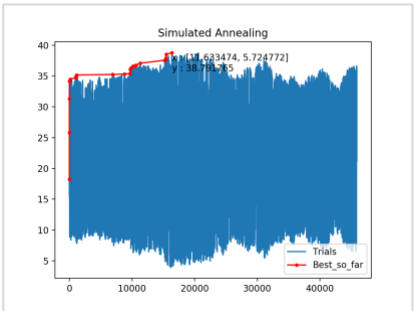


(3) Simulated Annealing

Temp : 100, Min Temp : 1, Cooling Rate : 0.99, length : 100 (詳細參數設定於程式碼







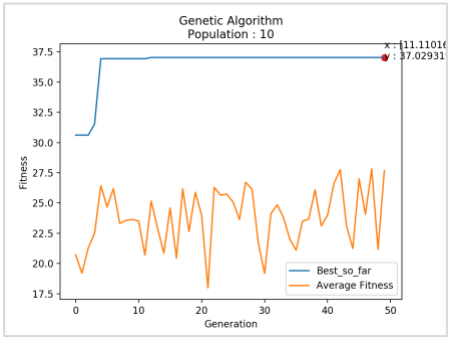
Algorithm_index	Optimal x	Optimal y	Optimal value	Average
GA_1	11.62	4.52	37.190094	
GA_2	11.11	4.63	37.029319	37.433727
GA_3	11.61	5.43	38.081769	
HC_1	10.14	5.72	37.209008	
HC_2	11.64	5.72	38.667539	35.861096
HC_3	5.13	5.23	31.706742	
SA_1	11.63	5.53	38.571029	
SA_2	11.63	5.52	38.556939	38.639911
SA_3	11.63	5.72	38.791764	

以表格彙整結果，得知函式最佳解大致落在 38.8 附近。三種演算法分別執行三次，以 SA 平均最佳解最高，GA 次之，HC 表現最差，其中 SA 及 HC 由於本身演算法特性，有相當的機率可能陷於局部最佳解，HC_3（紅字）即是其中一個例子，進而造成平均解低於最佳解。而 SA 在此次試驗算幸運，三次運算皆很幸運的落在最佳解附近，若是再執行幾次，可能就會發現陷於局部最佳解的情況。GA 借鏡生物演化學，透過族群、交配、突變及選取等流程，同時執行多點搜尋，相較於 HC 及 SA 的單點搜尋，擁有極低的機率會陷於局部最佳解，理論上若執行數次，平均表現會優於其他兩種演算法。

2. For Genetic Algorithm, need to provide

   a. The fitness curves (best and average of each generation and best-so-far)

      以其中一次執行結果為例。大約在第五代(Generation=5)時就已經收斂，每代的平均 fitness 趨勢呈現上升的，這是由於 GA 有物競天擇的 Selection 機制（不同的選擇機制於下題中深入討論），每代皆會篩選出 fitness 較高的物種，再進行後續的演化，優秀的物種產生更優秀的物種，子代擁有更高的機會會達到最佳解，迭代到最後，整體族群往更佳的方向演進，平均 fitness 逐漸上升。



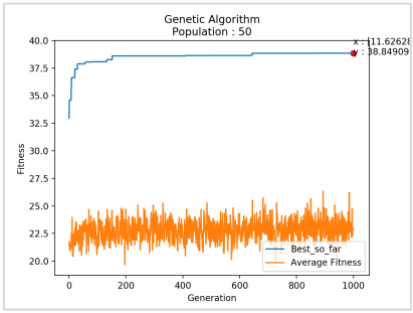
   b. The effects of population size, generation number, selection scheme, and genetic operator

      - Population Size 族群

        每代族群數量越多，代表變異性越高，通常找到最佳解的機會和速度也越高，不過計算量也隨之增加，因此族群數量並非越高越好。族群數量少但是迭代次數多，有機會在更短時間內找到最佳解。

- Generation Number 迭代次數

在物競天擇、優勝劣汰的架構下，每代都會經過選擇、交配和突變等流程，每代的表現會優於或等於上一代，設定的迭代次數越多，也會距離最佳解越靠近。下圖為迭代 1000 次，大約在第 180 到 650 代時，最佳表現已經趨於穩定，不過持續讓族群迭代下去，在第 650 代後又有些微的突破，到達 38.849097，結果高於第一題中 9 種試驗。GA 較 SA 和 HC 的優就在於此，GA 不會陷於局部最佳解，只要有足夠多的迭代，就會逐漸逼近最佳解。



- Selection Scheme

「選擇」策略決定哪些子代會被挑選出來，用來繁殖下一代，主要的原則就是表現較佳的子代，會有更高的機會成為下一代的父母。選擇策略會將 fitness 衡量進來，但並非完全挑選最佳的 fitness 子代個體，因為最優的子代個體不一定落在全域最佳解附近，因此衍生出不同的選擇策略。其中 gaft 套件中提供四種不同的選擇策略，以下分別介紹：

(1) Proportionate Roulette Wheel Selection 輪盤選擇

子代中個體被選中的機率與 fitness 值成正比。將所有個體的 fitness 加總做標準化，最後隨機落在哪個區域，就踢選對應的個體，類似於賭場裡面的輪盤因此得名。

$$P_{select}(a_i) = \frac{f(a_i)}{\sum_{i=1}^{n} f(a_i)}$$

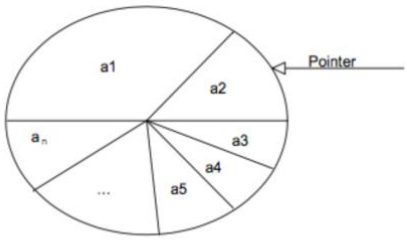


Fig. 2. Roulette wheel selection [22]

(2) Tournament Selection 錦標賽選擇

從子代中隨機抽出 n 個個體，挑選其中表現最佳的個體作為父母，由於此算法的時間複雜度為 O(n)且容易實現，是目前基因演算法的主流選擇策略。下圖為 n=3 的 Tournament Selection 範例。

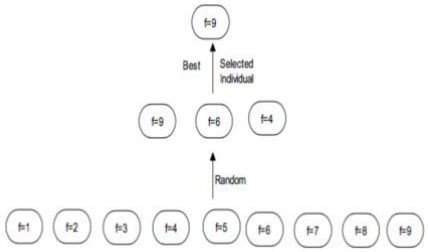


Fig. 3. Tournament Selection Mechanism [15]

(3) Linear Ranking Selection 線性排序選擇

這選擇策略是建立在輪盤選擇的基礎上，由於在輪盤選擇中，若是子代個體的 fitness 為零的話，該個體便完全沒有機會產生後代，因此在線性排序選擇中，依照各子代個體的 fitness 由大至小排序，依序賦予對應的機率。

$$P_i = P_{min} + (P_{max} - P_{min})\frac{i-1}{N-1}$$

(4) Exponential Ranking Selection

類似於上述的線性排序選擇，不過是賦予的機率是依循指數函式。

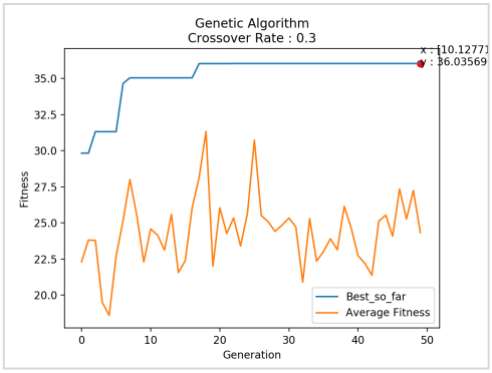
$$P_i = \frac{c^{N-i}}{\sum_{j=1}^{N} c^{N-j}}$$

- Genetic Operator

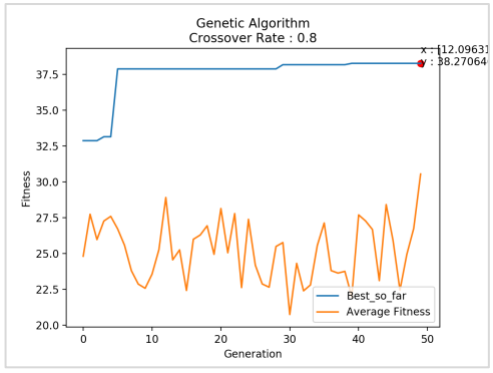
基因演算法核心概念由選擇、交配和突變建構成，選擇策略於上述已描述過，下面針對交配和突變詳細討論。

(1) Crossover 交配

模擬自然界交配現象，兩兩優秀個體重新組合，產生更新更優秀的子代，這樣的設定屬於有方向性的，朝向更好的方向前進。個體間是否交配和交配的位置皆可以透過參數來設定。 在位元中實現，就是對兩個不同的個體中相同的位置進行交換，繁殖新的位元序。



Crossover Rate : 0.3



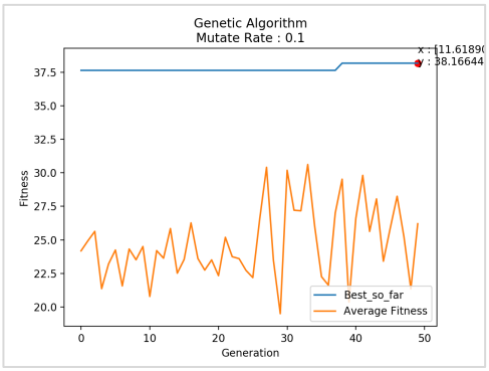
Crossover Rate : 0.8

理論上 Crossover rate 越大，每次迭代收斂的速度也會越快，卻有機會陷於局部最佳解。上圖試驗比較，交配率為操控變因，其他參數設定為控制變因，fitness 為應變變因，觀察到交配率較低，需要較多的迭代來往上提升，而交配率較高，每次提升的值較顯著，也較早收斂。

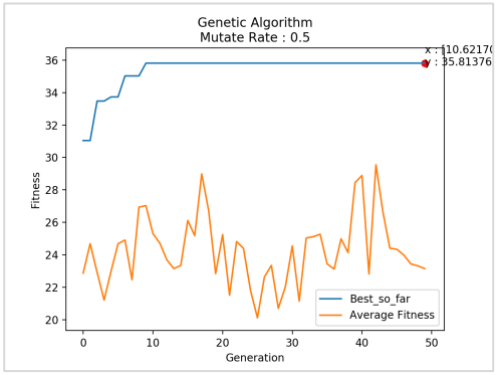
(2) Mutate 突變

模擬自然界染色體突變的可能，在既有的基因上產生突變，創造出變異性，這樣的設定屬於無方向性的，有可能會變好，也有可能變壞，不過正因為如此，可以避免在搜尋過程中陷於局部最佳解。在位元中實現，就是對某個位置相反轉換，突變新的位元序。在

gaft 套件中除了提供一般的突變，還提供每隔個週期進行大突變的機制，來預防早熟（premature）和陷於局部最佳解的情況。



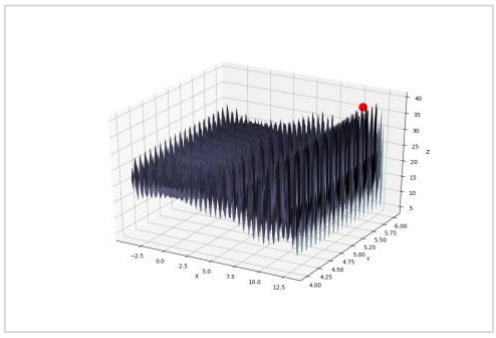
Mutate Rate : 0.1



Mutate Rate : 0.5

理論上 Mutate rate 越大，每次迭代的變異數也越大，可以避免早熟和收斂在局部最佳解，不過會需要較多的迭代收斂到全域最佳解。上圖試驗比較，突變率為操控變因，其他參數設定為控制變因，fitness 為應變變因，觀察到突變率較低的表現較穩定，突變率較高的試驗結果雖然目前不如預期，但是若是迭代次數越多，其表現很有機會會高於突變率較低的試驗。

3. Conclusion



函式的解空間

上圖立體圖為題目給予的函式解空間，觀察到擁有很多的山峰和山谷，也就是說有相當高的機率會收斂至局部最佳解，面對這樣的情況，關鍵就在於採用適當的演算法來求最佳解了。HC 爬山法，隨機從其中一點可行解出發，隨機試驗鄰近的解，若是鄰近解屬於往上即移動到該位置，依此重複不斷往上爬，這樣的缺點在於，要是起始出發的點不是全域最佳解的山，最後僅會收斂到相對高的局部最佳解之中。SA 模擬退火法，屬於爬山法的改良版，差異點在於借鏡溫度越高變動越高的金屬退火特性，相較於 HC 有一定的機會往下走，脫離局部最佳解，不過依然是單點式搜尋，仍然有陷於局部最佳解的疑慮。最後 GA 基因演算法，採用與上述演算法完全不同的思維，屬於多點搜尋法，配合創造變異的機制（選擇、交配和突變），避免掉過早收斂和局部最佳解的問題。雖然以上三種演算法，在這次題目中皆有計算出逼近最佳解的 38 多，推測是因為這次的解空間還不算太複雜，多試驗幾次就有不錯的結果，未來面對更複雜的解空間問題時，採用 GA 會是比較理想的演算法。這也解釋 GA 成為當今流行的搜尋演算法之一，也衍生出 Genetic Programming 的概念，僅需要告訴電腦「完成目標」（optimizing goal）和「適應函數」（Fitness），而不必告訴電腦「如何完成」，由電腦自行演算找出最佳可行解，實現像是有人工智慧的機器一樣。