EC_SCH_report

108062273 張晏瑄 Link to Notion for Nice Format (github link and the img is included in ./show)

Overview

本次作業為實作 Binary GA 或 Real-valued GA 去解 Schwefel function minimal solution (SCH)，並對不同的 representation 做分析，以及對 parameter 調整並討論影響。最後試著解 $N = 100$ 的SCH function

$$f_{SCH}(\vec{x}) = 418.98291N - \sum_{i=1}^{N} x_i \sin(\sqrt{|x_i|}), \quad -512 \leq x_i \leq 511 \quad and \quad N = 10$$

Binary GA and Real-valued GA implementation

Code Implement

Requirements

Fig.1. 為我的 Class Diagram，我使用 C++ 去 implement Binary GA 及 Real-valued GA (使用C++因為較快)，接著利用 python 的 `matplotlib.pyplot` 去畫出 anytime behavior chart，以下是幾個重要的細節：

1. 共有兩個 Class implement 兩個演算法，每個 Class 都有一個 `Individual` struct，表問題的解，故一個 `Individual` 裡會有10個 `bitset` 或是 `double`，值得注意的是，在binary GA 使用的是 `bitset` 而非 `vector<bool>`，因為 `bitset` 相較 `vector<bool>` 在計算上更快更方便、memory 使用較少、轉成 integer 時也更方便。
2. 每個 Class 都有 `evolution()`，他會呼叫 `intialize_population()`, `parent_selection()`, `crossover()`, `mutate()`, `survivor_selection()` 去做一個 generation的模擬（如Fig.2. 所示），而根據傳入參數 term 去決定要跑幾個 generation。
3. 在做完 `intialize_population()`, `crossover()`, `mutate()` 後，都會做一個 `evaluate_fitness()` 的操作。
4. 由於需要紀錄每個 generation 的解並畫出 anytime behavior chart，故在進到下一個 generation 時會有一個 `get_best_fitness()`，去記錄此 generation 最佳解。
5. fitness 即為將 xi 代入 SCH 的值，故 fitness 越小表現越佳。

而 Binary GA 由於要將每個 bit representation 轉成 integer，故多了 `bit2int()`, `get_xi()` function。

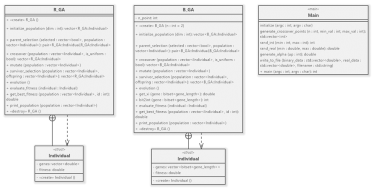

Fig.1. Class Diagram of my C++ Code

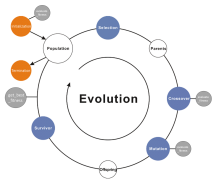

Fig.2. Evolution Diagram, Reuse from: (C.K. Ting, Design and Analysis of Multi-parent Crossover, PhD thesis, PaderbornUniversity,2005)

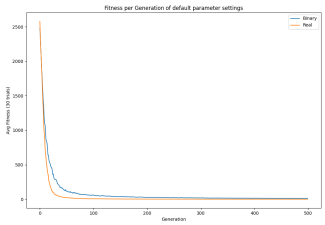

Fig.3. Fitness per Generation of default parameter settings

Results

當使用 default 參數去跑 SCH 時，最終得出的結果如下，而 Fig.3. 可以看出 Real-valued 較早收斂，最終的 fitness 也表現得較佳。
BGA fitness: 12.2329, RGA fitness: 0.156064, time cost: 20.14s

```
$ time ./main --detail 1
# ignore the ouput of here
423 422 422 419 420 420 418 419 422 422
# ignore the output of here
420.951 421.144 420.795 420.863 420.901 420.83 420.91 421.058 421.202 421.442
================================================================================
BGA in the final generation fitness: 12.2329, RGA in the final generation fitness: 0.156064
./main --detail 1  20.14s user 0.98s system 98% cpu 21.343 total
```

Other Works

為了更方便去調參數及做測試，我有寫 Makefile 去做自動化 script。以及使用 C++ 去做 argument parser, demo 如下，其他詳細資訊可以參考資料夾的 `README.md`。

```
$ make
$ ./main --help
# ignore the ouput of here
Options:
  --algorithm <value>  Set the algorithm to use. Only binary, only real or both (default: both)
  --detail <value>     Set the number of printing detail or not (default: 0)
  --cross_prob <value> Set the crossover probability (default: 0.9)
  --mut_prob <value>   Set the mutation probability (default: 0.1)
  --n_point <value>    Set the number of crossover points (only for binary GA, and need to set uniform to 0) (default: 2)
  --p_select <value>   Set the number of tournament selection when parent selection period (default: 2)
  --p_size <value>     Set the population size (default: 100)
  --term <value>       Set the termination criterion (default: 500)
  --trial <value>      Set the number of trials (default: 30)
  --uniform <value>    Use uniform crossover (1)
                       or 2-point for binary GA, whole arithmetic for real-valued GA (0) (default: 1)
```

Analysis

Binary GA v.s. Real-valued GA

在 dim = 10 的情況下，RGA 都比 BGA 表現得更好，達到了較好的 fitness。我們可知 RGA 在解決 SCH function 最小值問題時更具優勢。

Representation: BGA 使用 binary bit 去表示解，而 RGA 則使用 real-valued。對於 continuous function 優化問題，如 SCH function，real-valued GA 的精度可能更適合用來作為 solution space 的 representation。

Parameter Setting

- crossover probability (Fig.4.)
    1. 隨著 crossover probability 的增加，time cost 也增加，若 crossover probability 越高，通過 crossover 產生的後代也越多，這需要額外的計算資源
    2. 隨著 crossover probability 的增加，BGA 和 RGA 的 fitness 也會較佳，因 crossover 能提供 exploration，使 GA 能夠找到好的 solution
    3. RGA 在所有測試的 crossover probability 中始終表現的比 BGA 佳。 這與前面的分析一致，表明 RGA 的 representation 和 genetic operator 更適合此優化問題。

prob	0.1	0.4	0.7	1
time cost	5.53s	7.84s	10.57s	12.59s
fitness	BGA: 35.058, RGA: 0.378918	BGA: 18.3954, RGA: 0.0927711	BGA: 14.3794, RGA: 0.0762096	BGA: 10.8697, RGA: 0.0540917

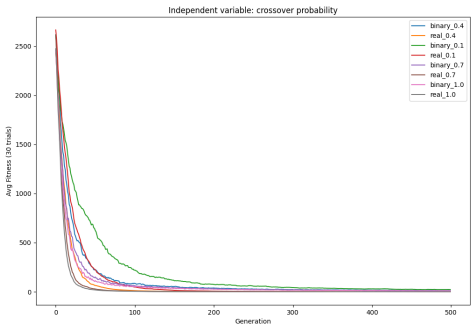

Fig.4. crossover probability

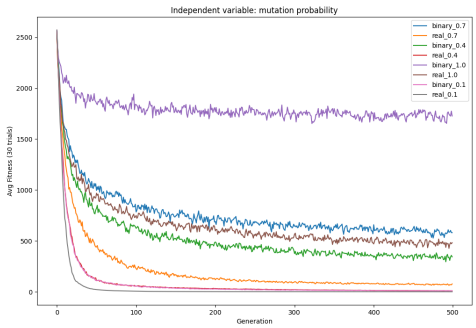

Fig.5. mutate probability

- mutate probability (Fig.5.)
    1. 隨著機率的增加，time cost 沒有顯示出明顯的趨勢。因為 code implement 是對每個 genes 去看是否需要 mutate。

2. 隨著機率的增加，BGA 和 RGA 的 fitness 表現都會降低。 較高的突變機率可能導致過度 exploration 並破壞 GA 的 convergence。 突變機率較低時，GA 可以更好地平衡 exploration 和 exploitation，從而提高 performance。

prob	0.1	0.4	0.7	1
time cost	11.91s	12.90s	12.93s	12.39s
fitness	BGA: 8.67412, RGA: 0.132756	BGA: 421.171, RGA: 10.0505	BGA: 505.855, RGA: 124.681	BGA: 2053.09, RGA: 465.32

- n_point crossover (Fig.6.)
  1. 隨著 crossover 切的點越多，time cost 和 fitness 皆沒有顯示出明顯的趨勢。
  2. 改變 point of crossover 可以透過用不同方式組合 parent 的子代來改變 exploration 和 exploitation 的平衡，但其影響可能還不足以影響平衡

n value	2	3	4	5	6	7	8
time cost	18.95s	16.67s	16.44s	19.55s	23.80s	23.00s	27.75s
fitness	BGA: 20.301	BGA: 12.8473	BGA: 20.2379	BGA: 13.6951	BGA: 24.521	BGA: 14.468	BGA: 11

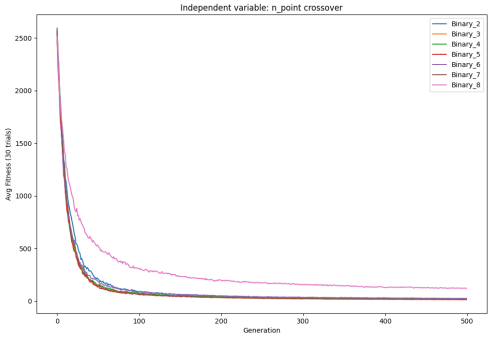

Fig.6. n_point crossover

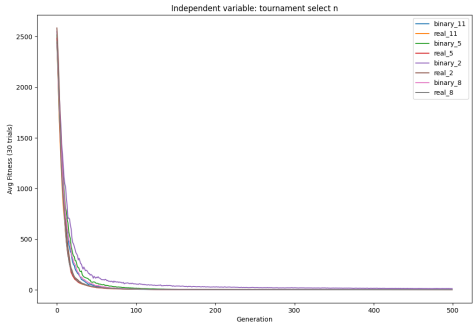

Fig.7. tournament n

- p_select (Fig.7.)
  1. 隨著 tournament n 的增加，time cost 也會增加。因為較大的 n 需要在parent selection 時進行更多比較，導致 time cost 較高
  2. BGA fitness: 隨著 tournament n 的增加，fitness 照理來說應該要提高，因較多的候選人會傾向於選擇更合適的 parent 去繁衍，這促進了 solution space 的 exploitation。然而，這也可能會減少 diversity，導致太早收斂。而在提供的結果中，BGA 在 n 為 5 時表現最佳
  3. RGA fitness: RGA 中較難看出 n 和fitness的關係。在 n 為 2 時 RGA 表現最佳，但 fitness 沒有隨著 n 的增加而不斷提高或降低

value	2	5	8	11
time cost	10.6s	19.2s	22.2s	26.59s
fitness	BGA: 22.9161, RGA: 0.0620879	BGA: 0.135577, RGA: 0.281693	BGA: 0.37207, RGA: 0.226822	BGA: 0.506187, RGA: 0.487685

- p_size (Fig.8.)
  1. BGA: 隨著 population size 的增加，BGA 的 fitness 普遍提升。因為更多的 population 會友更多的 diversity 並能更徹底地 explore solution space。它降低了過早收斂到 local optimal 的風險，代價則是 time cost 較高
  2. RGA: 與 BGA 類似，RGA 的 fitness 也會隨著 population size 的增加而提高。但 RGA 獲得的 benefit 比 BGA 高 (能更增加 diversity 及 explore solution space 更多)。而與 BGA 相比，RGA 收斂到更低的 fitness，與一開始所說的，RGA 在解決此問題更有效

value	100	400	700	1000
time cost	11.83s	2:2.88	5:34.68	11:3.69
fitness	BGA: 14.0489, RGA: 0.242471	BGA: 7.92516, RGA: 0.00679581	BGA: 5.55451, RGA: 0.00224302	BGA: 4.91188, RGA: 0.00173924

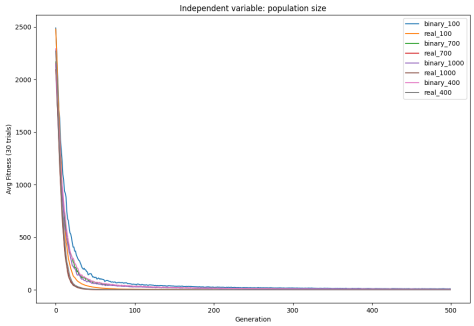

Fig.8. population size

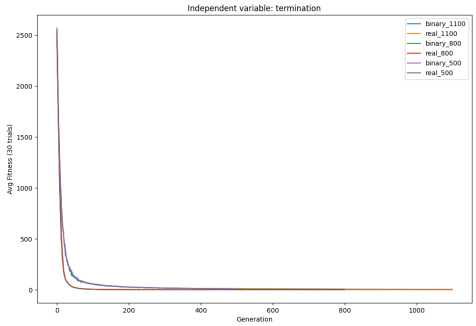

Fig.9. termination

- termination (generation) (Fig.9)
  1. exploration: 有了更多generation，就有更多機會explore 不同的 solution space
  2. exploitation: 有了更多generation，就有更多時間能透過 crossover 和 mutate 去微調和提升現在 solution 的 performance，可以幫助找到 global optimal
  3. convergence: 更多的 generation 能有更多機會收斂
  4. diversity: 更多的 generation 可以幫助維持 diversity，避免太早收斂到 local optimal

value	500	800	1100
time cost	11.81s	18.73s	26.09s
fitness	BGA: 3.78584, RGA: 0.254383	BGA: 5.49853, RGA: 0.115951	BGA: 3.51734, RGA: 0.0506551

- uniform
  1. 在 BGA 使用 uniform crossover 或是 n-point crossover 沒有明顯的改變，但在 RGA 中，使用 whole arithmetic 的結果卻變很糟
  2. RGA：使用 uniform crossover 時，fitness 為 0.178，而使用 whole arithmetic 時，fitness 則明顯更差 (987.698)。 可能是因為 exploration 過強 (offspring 繼承 parent 的 feature，容易導致陷入 local optimal)、缺乏 diversity、SCH function 問題的特性 (多個 local optimal，whole arithmetic對此問題可能不適合)

is using	1	0
time cost	11.96s	11.95s
fitness	BGA: 12.6458, RGA: 0.178	BGA: 15.5105 RGA: 987.698

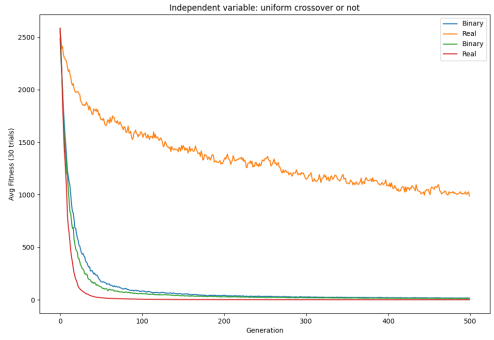

Fig. 10. uniform using or not

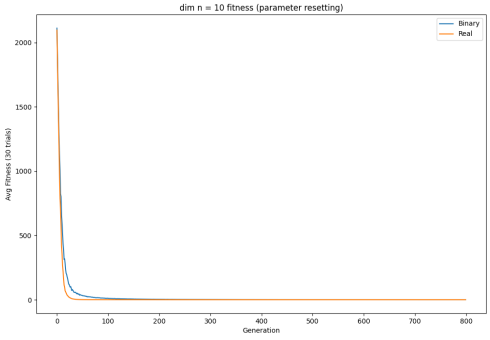

Fig.11. N=10 parameter resetting

Parameter Settings for N=10 Problem

根據上方觀察，我綜合評估重新設置了新的測試參數，也成功獲得了較佳的 performance (Fig.11.)：
BGA fitness: 0.00145984
RGA fitness: 0.000997442

```
---------------------------------
|Parameter      |Value          |
---------------------------------
|algorithm      |both           |
```

```
|cross_prob     |1              |
|detail         |1              |
|mut_prob       |0.1            |
|n_point        |4              |
|p_select       |5              |
|p_size         |1000           |
|term           |800            |
|trial          |30             |
|uniform        |1              |
-----------------------------------
```

```
# ignore the ouput of here
421 421 421 421 421 421 421 421 421 421
# ignore the output of here
420.977 420.973 420.964 421.004 420.968 420.961 420.974 420.972 420.985 420.926
================================================================================
BGA in the final generation fitness: 0.00145984, RGA in the final generation fitness: 0.000997442
./main --cross_prob 1 --detail 1 --n_point 4 --p_select 5 --p_size 1000 --ter  838.39s user 11.28s system 98% cpu 14:19.10 total
```

Large-scale Problem: N = 100

若我們使用 default parameter 去做 N=100 的SCH function，performance會相當的差。

```
-----------------------------------
|Parameter      |Value          |
-----------------------------------
|algorithm      |both           |
|cross_prob     |0.9            |
|detail         |1              |
|mut_prob       |0.1            |
|n_point        |2              |
|p_select       |2              |
|p_size         |100            |
|term           |500            |
|trial          |30             |
|uniform        |1              |
-----------------------------------
```

BGA the final generation fitness: 13478.9

RGA the final generation fitness: 2144

time cost: 47.99s

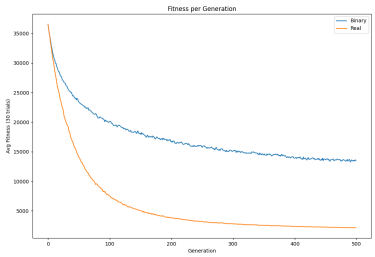

Fig.12. default parameter for N=100

Trial and Error

首先，我先以 trial 為 1 的情況下，去對每種參數做測試，並且觀察到以下特點：

1. population size 要夠大，real-valued fitness才會高 (維度高的影響)
2. parent selection 不能太多，超過10後 performance 下降很多
3. BGA 的表現：對於高維問題，BGA 可能能表現較強的能力。binary representation 可以在solution space 中產生大量變化，使得 BGA 在高維問題中更容易找到較佳的解。
4. RGA 的收斂：在高維問題中，real-valued GA 可能容易陷入 local optimal solution。由於 solution space 變得更加複雜，RGA 可能在搜尋最佳解中遇到困難，從而導致收斂速度變慢。

---

parameter setting 觀察: 只要讓 binary GA 的termination夠大，parent_selection 夠，就算population size 較小也能有較高的performance。

BGA the final generation fitness: 1.00976
RGA the final generation fitness: 30.5084
time cost: 2:42:02.97

```
-----------------------------------
|Parameter      |Value          |
```

中間和右邊 parameter setting 觀察: 增加 1000 termination fitness 可以少一半。

BGA the final generation fitness: 34.1177
RGA the final generation fitness: 6.42585
time cost: 13:55.29 total

BGA the final generation fitness: 11.6868
RGA the final generation fitness: 3.36684
time cost: 16:43.47

```
-----------------------------------
|Parameter      |Value          |
-----------------------------------
|algorithm      |both           |
|cross_prob     |1              |
|detail         |1              |
|mut_prob       |0.1            |
|n_point        |2              |
|p_select       |5              |
```

```
---------------------------------
|algorithm      |both           |
|cross_prob     |0.9            |
|detail         |1              |
|mut_prob       |0.1            |
|n_point        |2              |
|p_select       |20             |
|p_size         |1000           |
|term           |5000           |
|trial          |30             |
|uniform        |1              |
---------------------------------
```

```
---------------------------------
|Parameter      |Value          |
---------------------------------
|algorithm      |both           |
|cross_prob     |1              |
|detail         |1              |
|mut_prob       |0.1            |
|n_point        |2              |
|p_select       |5              |
|p_size         |3800           |
|term           |5000           |
|trial          |1              |
|uniform        |1              |
---------------------------------
```

```
|p_size         |3800           |
|term           |6000           |
|trial          |1              |
|uniform        |1              |
---------------------------------
```

Final Result!

經過上述觀察測試後，我使用了以下參數產生最終結果，而由於 real-valued GA 需要較大的 population size，但 binary GA population size 不需要太大，只需要足夠的 termination 就可以有較好的 performance。因此最後才折衷使用下面的參數：

BGA the final generation fitness: 3.75244
RGA the final generation fitness: 1.91158
time cost: 18:31.98

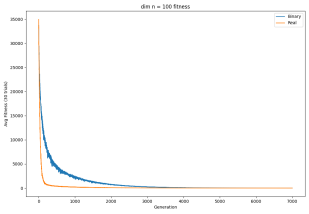

Fig.13. Final Result

```
---------------------------------
|Parameter      |Value          |
---------------------------------
|algorithm      |both           |
|cross_prob     |1              |
|detail         |1              |
|mut_prob       |0.1            |
|n_point        |2              |
|p_select       |5              |
|p_size         |3800           |
|term           |7000           |
|trial          |1              |
|uniform        |1              |
---------------------------------
```

```
# ignore the output of here
In 7000th generation, the best fitness is: 3.75244, When xi are,
421 421 421 422 421 421 421 422 421 421 421 421 421 422 420 421 421 421 421 421 421 421 421 421 421 421 421 420 421 421 421 421 420 421
421 420 421 421 421 420 420 420 421 420 420 422 422 421 421 421 421 421 421 421 421 421 420 420 420 421 421 421 421 421 422 421 420 421
421 420 421 421 422 422 421 421 421 421 421 420 421 421 420 421 421 421 421 422 421 421 421 421 421 422 422 421 421 420 421 422 420 421
421
# ignore the output of here
In 7000th generation, the best fitness is: 1.91158, When xi are,
420.941 421.627 420.619 420.68 420.728 420.307 421.427 421.4 421.345 421.548 420.436 421.016 421.129 420.25 420.668 420.318 421.101
420.322 421.011 420.464 420.291 420.565 420.695 421.74 420.593 421.234 421.46 420.147 420.692 420.502 420.063 421.92 421.308 421.393
421.168 421.497 421.201 421.506 420.417 420.742 420.732 421.032 420.484 421.122 420.322 421.029 421.228 421.007 421.37 421.137 421.776
421.092 420.829 421.169 420.636 421.187 420.63 420.884 421.179 420.662 420.813 421.174 421.199 421.13 421.287 420.493 420.66 420.727
420.601 420.975 420.581 421.225 421.143 421.118 421.109 421.646 420.906 420.475 421.091 420.655 421.387 421.299 420.879 420.966 420.863
420.971 421.527 420.954 421.156 420.71 421.365 420.934 421.285 421.19 420.885 421.188 420.988 420.894 421.008 420.488
================================================================================
BGA in the final generation fitness: 4.38413, RGA in the final generation fitness: 2.63381
./main --cross_prob 1 --p_select 5 --trial 1 --p_size 3800 --term 7000 --trial 1 --detail 1  1107.35s user 4.60s system 99% cpu 18:31.98 t
```

Problem Encounter and Future Works

Problem Encounter

- 由於 `bitset` 的長度需在 run time 決定，故 dim_n 要手動設置，除非使用 `dynamic_bitset`，但因為需要灌額外的 library package，因此最後就用手動設置
- 自動化做得不夠多，還是需要手動調 anytime behavior 的 chart name

Future Works

- 將 code 利用 C++ template 變得更簡潔
- 傳入參數若非可控參數則跳出 error