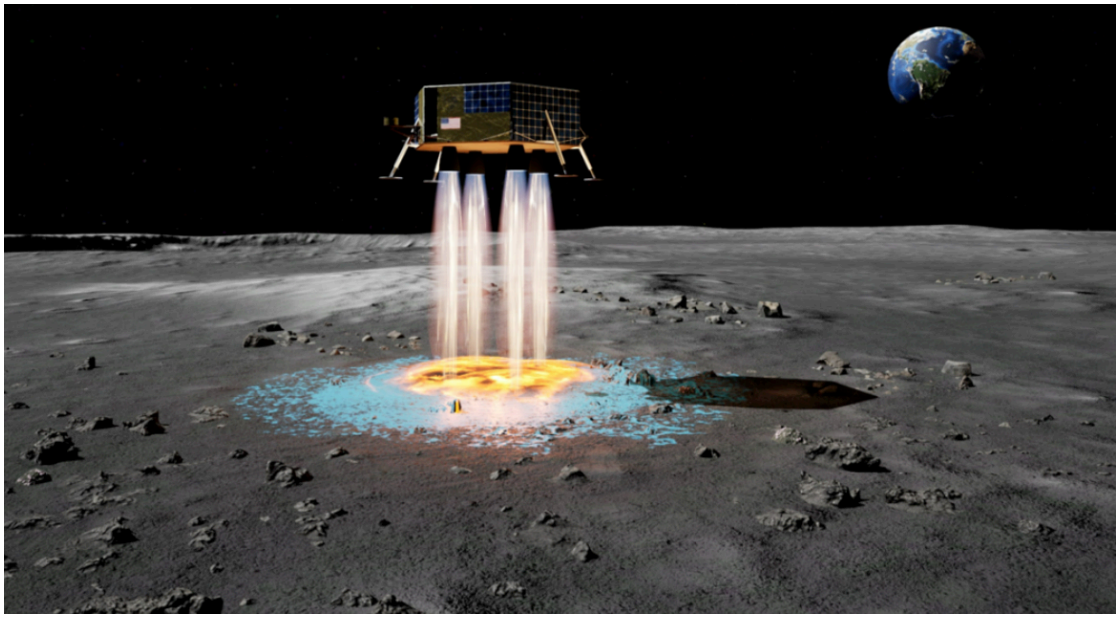
LunarLander GA

11010PME543900 Introduction to Artificial Intelligence

109033804 張軼峯 - 2021年11月17日



Pic: https://newatlas.com/space/fast-lunar-landers-build-own-landing-pads/

❖ Problem Formulation and Evaluation

LunarLander 的 observation 與 action 如作業說明中的 problem prediction 所述，一個 gene 長度為 8192 由範圍 -1 ~ 1 的實數所組成，由此 representation 可知 search space 非常龐大。Control policy (chromosome) 將 observation 與 action spaces 做 mapping，因此 LunarLander 的 evaluation 是由起始位置到降落的連續過程作評分，無法使用分段的方式將 chromosome 縮短使每次最佳化的 search space 縮小。

另外，evaluation 的方式為隨機初始化 LunarLander 的位置(固定高度)與初速，`SIMULATIONS_PER_EVALUATION` 的次數也對於 evaluation 的準確度相當重要，simulation 次數越多結果越準確，然而 simulation 的過程相當耗時，使得要獲得準確 fitness 值的成本相當昂貴。將 anytime behavior 分析的實驗組數提高也可以提升準確性，但同樣有耗時的問題。

若將所有參數固定，只調動 simulations per evaluation (SPE) 次數來重複實驗，每組各重複十次取平均可得 anytime behavior，另外可計算十次的標準差來推估 evaluation 的

準確性，得 Fig. 1.。下圖分為 SPE 3, 5, 20 次的 anytime behavior 標準差，因初始條件為隨機產生，所以初期 fitness 的變異較大為合理，可以觀察到三種 SPE 設定在初期的 anytime behavior 標準差差異不大；隨著演化過程，fitness 逐漸提升，在 evaluation 準確的情況下，可觀察 anytime behavior 最終表現即為此參數設定下的表現。下圖可觀察到綠色線 SPE 20 次的 anytime behavior 標準差穩定的下降，因為不管 initial state 為何，此參數設定下的(重複實驗)表現結果會逐漸收斂，evaluation 結果準確則 anytime behavior 標準差也隨之下降。SPE 3, 5 次的 anytime behavior 標準差雖然整體趨勢下降，但很明顯有不穩定的跳動，由此方法可推知 simulations per evaluation 對於 evaluation 的影響與合理設定。

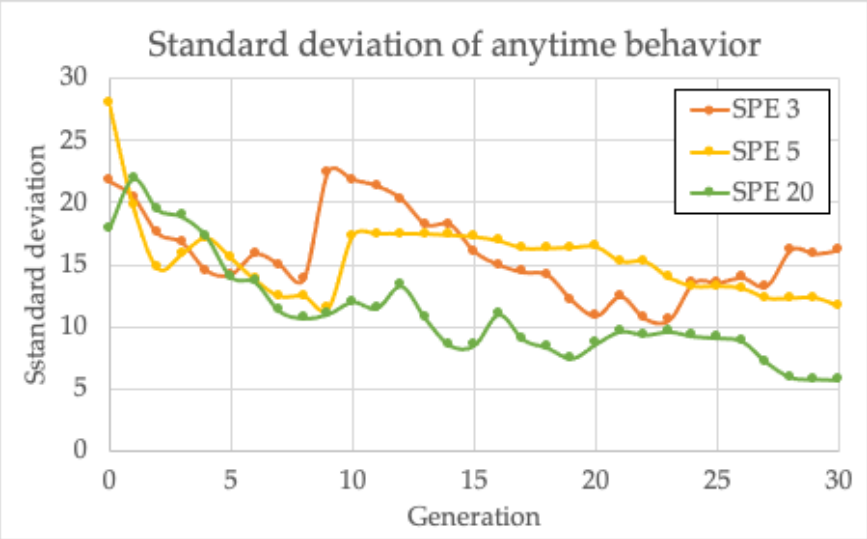


Fig. 1. Standard deviation of anytime behavior between different SPE

藉由以上各因素可預期以此種編碼與模擬方式， GA 做完最佳化的結果可能不會太理想，且過程耗時，故本報告中將 simulation 次數縮短為 5 次以節省實驗時間；同理因實驗耗時，實驗設定的 generation 也縮短，雖還沒收斂但可觀察其趨勢與收斂速度。得到最終參數後再提高 simulation 次數與演化代數作為最後結果之驗證。

❖ Parent Selection and Survivor Selection

Parent selection 設計使用 k tournament selection，理論上 k 越大 parent selection 的競爭壓力越大，且 fitness 排名倒數 k - 1 的最後幾名個體將完全沒有機會成為下一代的親代。k 值的調整也因 population size 而有所不同，需同步調整。由 Fig. 2. 可觀察到，在 population 50 的情況下，k 設定為 10 競爭壓力已經過大。雖在初期無顯著差異，但大約在 30 代過後，因為競爭壓力過大，可能導致 population 的 diversity 下降，使得 k = 10 最終結果較差。而 k = 3, 5 的兩組實驗比較雖然 k = 3 時在 anytime behavior 幾乎都贏過 k = 5，但 k = 3 在約 70 代後呈現趨近於收斂的情況，k = 5 則明顯尚未收斂，考慮實驗僅設置

generation 為 100 且 simulations per evaluation = 5，較不準確的狀況下，只能推斷 k = 3 或 5 的差異不大，但 k = 5 可能較具潛力，因此後續實驗使用 k = 5。

Survivor selection 部分使用 $(\mu + \lambda)$，每一代的親代與子代皆共同競爭篩選至下一 generation。所有實驗中皆使用此方法，這部分不再做其他改變。

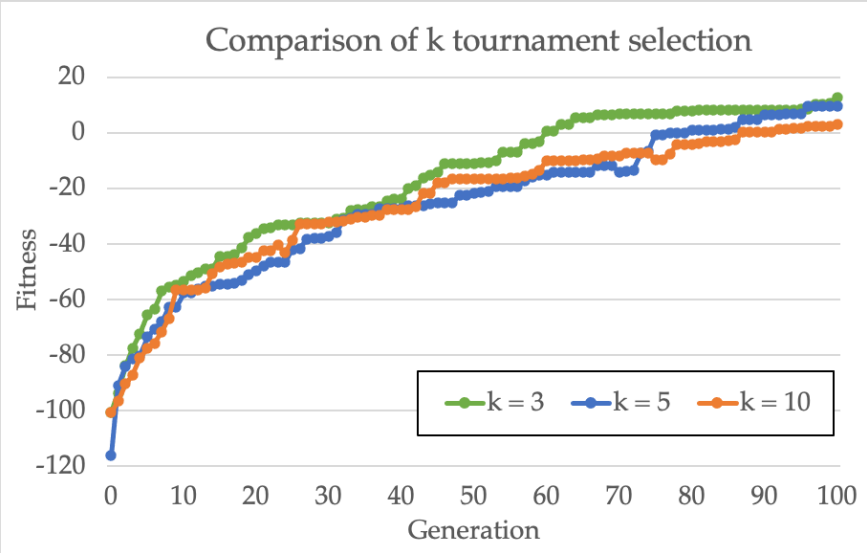


Fig. 2. Comparison of k tournament selection

❖ Variational Operator Design

• Crossover

因作業提供的 uniform crossover 原始程式有誤，在修正之後，比較了一下差異，結果如下圖 Fig. 3.。Crossover rate 的正確用法為決定該次 crossover 要不要執行的機率，而不是選擇 p1 & p2 的機率。

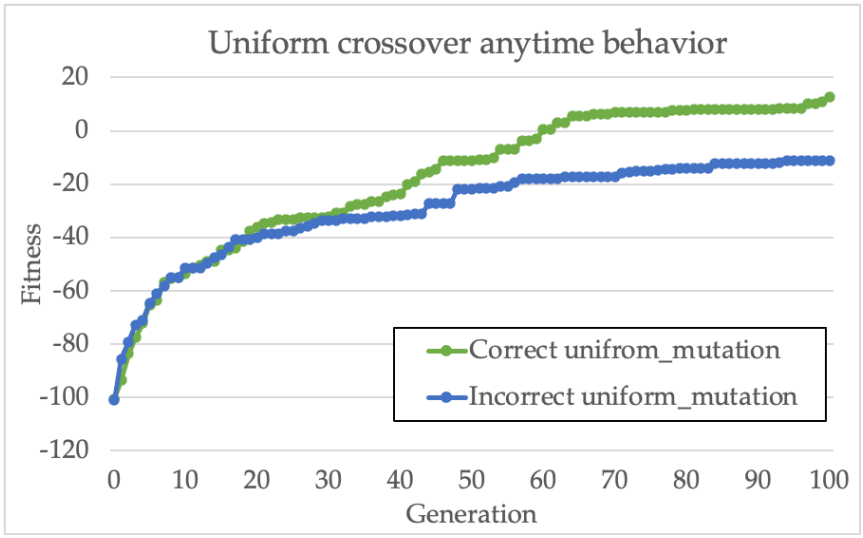


Fig. 3. Comparison of correct and incorrect crossover rate usage

除作業 source code 提供的 uniform crossover，另外實作了 n-point crossover，並可分為固定 n 與每次隨機產生不同 n 個 crossover point，實作原則與課堂上的作法相同，此處不贅述。以下分別對實驗結果對不同 crossover 方法做分析與比較。

Fig. 4. 設定 n-point crossover 的 n 分別為 2, 50, 100, 200, 900 與隨機產生，mutation 固定設為 uniform mutation。觀察 Fig. 4. 的結果，因為 chromosome 長度很長，可以想像若 crossover point 太少，效果不甚理想。但此時將 n-point crossover 的 n 每次隨機產生 (範圍 1 ~ 8000)，效果也不佳，如 Fig. 4. 中藍色線。(此組實驗我重複了兩次，結果相仿) 推測可能因為隨機產生的 n 若很小，則效果如前面所述；但通常會產生 n 很大的情況(> 1000)，子代 chromosome 會分別由許多親代的小片段所組成，而 evaluation 的過程為一連續動作的結果，需要 chromosome 內連續指令來達成，因此將兩親代的基因平均分散地截取不見得會有更好的子代產生。同理 n = 900 的結果比 n = 50, 100 要來得差。

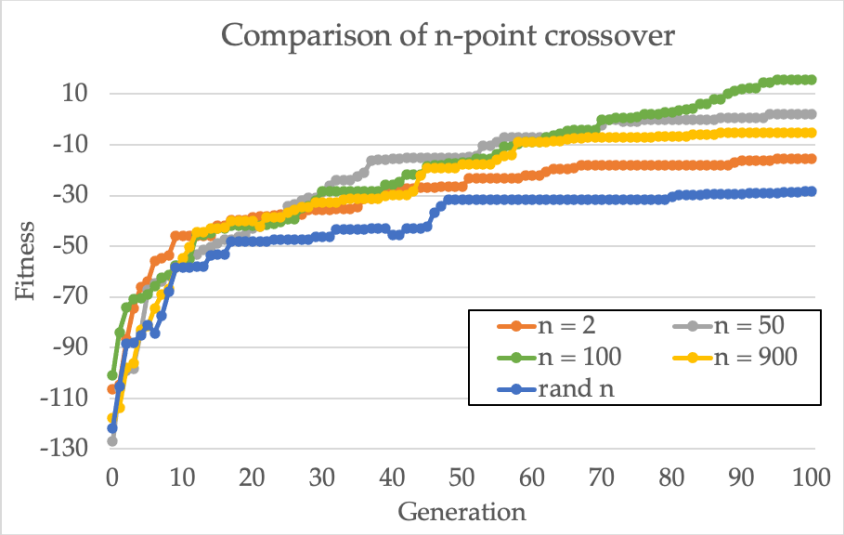


Fig. 4. Comparison of n-point crossover anytime behavior

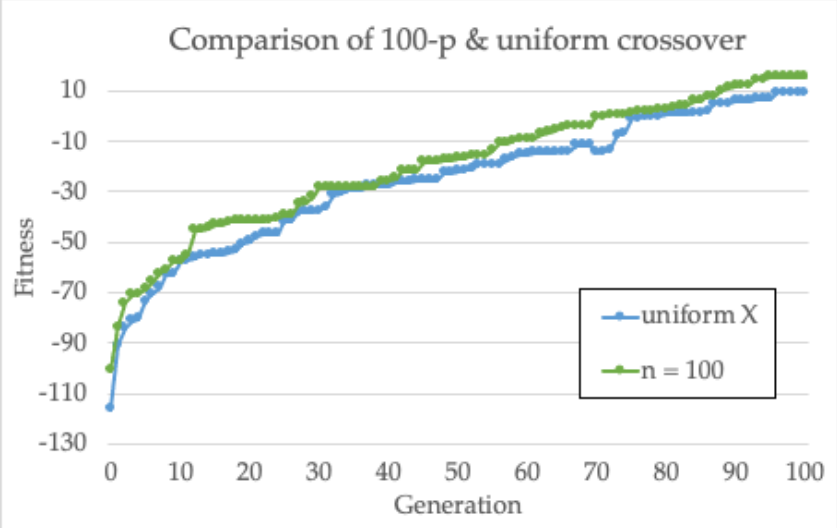


Fig. 5. Comparison between 100-point crossover and uniform crossover

Fig. 5. 比較 uniform crossover 與前一組實驗表現較佳的 100-point crossover，100-point crossover 同樣有較好的結果。

- **Mutation**

題目中提示的 creep mutation 實際上屬於 Integer representation 的 mutation 方法之一。

Integer representation 的 mutation principle form 主要可分為兩類：Random Resetting 與 Creep Mutation；Random resetting 中每一 gene value 有 p_m 機率執行 mutation，隨機產生一 permissible value 取代原本的值，且生成的值在允許範圍中個數值機率相同。Creep mutation 則為在一固定 distribution 中隨機 sample 出一個值(可正或負)作為一 mutation step 加回原本的 gene value，且此 distribution 的 mean 為 0。依照 step size 的大小可分為 “big creep” 與 “little creep”，step size 還需另外做參數的調整與設定，但原則上因 random resetting 可能造成的影響較劇烈，random resetting 的 mutation rate 會設得較低。

此作業為 Real-Valued 的 representation，mutation 產生的變異範圍固定於 lower bound 與 upper bound 之間。依照 probability distribution 也可分成兩類：Uniform Mutation 與 Nonuniform Mutation。Uniform mutation 與 random resetting 相仿，單純轉變為實數域並符合上下界規範；Nonuniform mutation 則類似於 integer representation 的 creep mutation，為較常用的 mutation 方法，若 sampling 使用的 distribution 為 Gaussian distribution 則通稱為 Gaussian mutation，可自行設定 σ ， mean 原則上皆設定為 0。

$$p\left(\Delta x_i\right) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{\left(\Delta x_i - \xi\right)^2}{2\sigma^2}}$$

另外也可使用不同 distribution 如 Beta, Cauchy 等等，對於 permissible value 較邊緣的範圍有不同的生成機率。此作業實作中僅針對 Gaussian distribution 的不同的 σ 值做設定與比較。

Fig. 6. 為 Gaussian mutation 設定不同 sigma 值之比較，可看出 sigma 0.6 表現最佳，且 fitness 略高於 uniform mutation，sigma 0.8 最差。Fig. 7. 比較不同 mutation rate，mutation rate 設定為 0.001 明顯較佳，因此固定 mutation rate = 0.001。

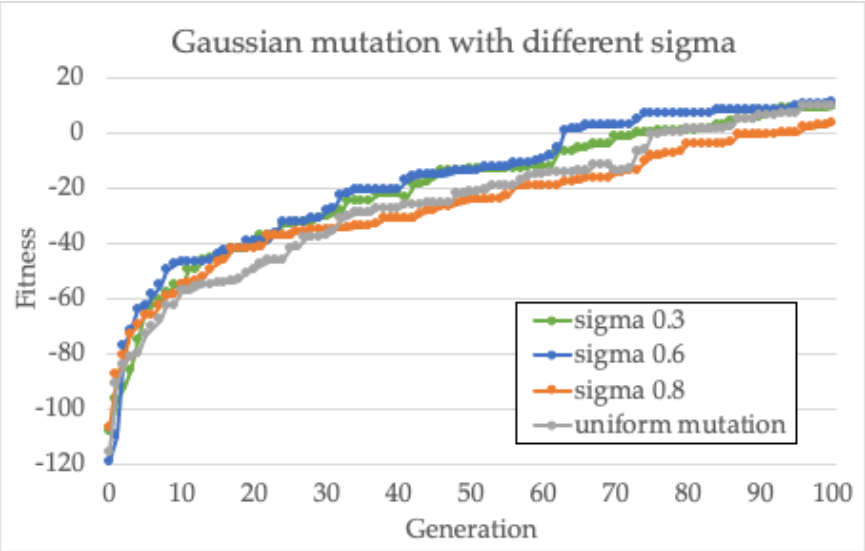


Fig. 6. Comparison of Gaussian mutation with different sigma

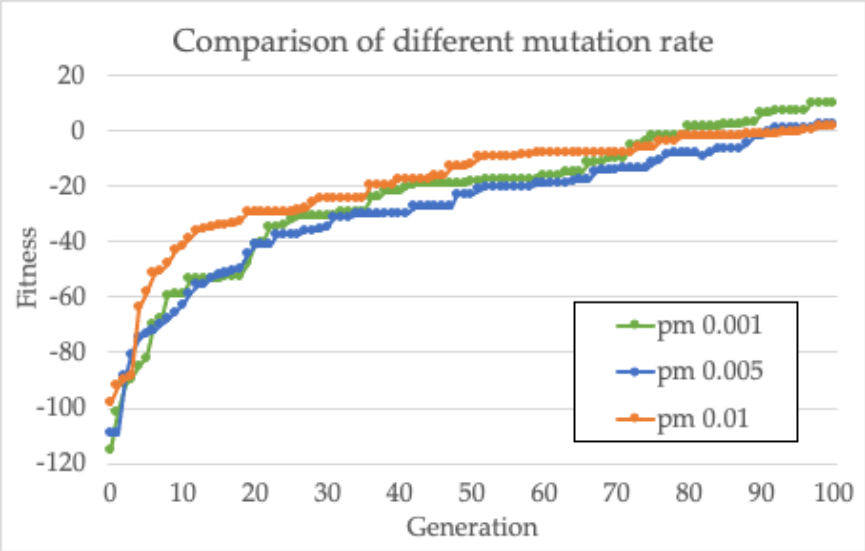


Fig. 7. Comparison of different mutation rate

❖ Deterministic Parameter Control

若參數的調整是在演化進行之前，稱為 Parameter tuning；若是在演化的過程中調整參數，則稱為 Parameter control，Parameter control 又可分為 Deterministic, Adaptive, Self-adaptive。Deterministic 為 user-specified 的方法，人為控制何時進行何種 parameter control 的行為；Adaptive 則必須回傳如 quality of solution 來決定時機進行 parameter control，且 control mechanism 為事先定義好的；Self-adaptive 則是將要 control 的參數 encode 成 chromosome 的一部分同時進行 recombination 與 mutation。

Parameter setting 的分類簡圖如 Fig. 8.。

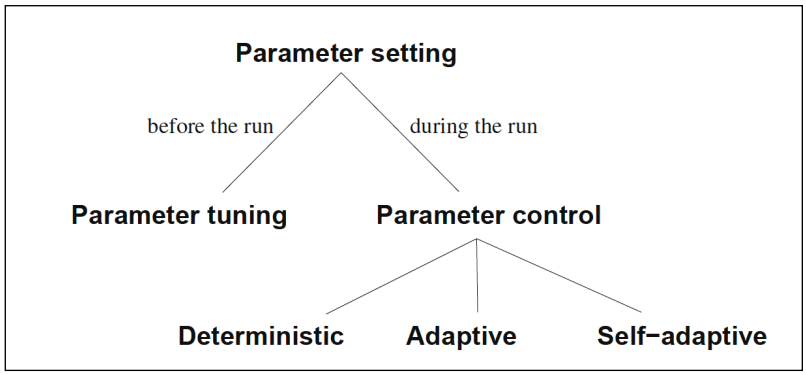


Fig. 8. Global taxonomy of parameter setting in EAs
(ref: Intro to EC, A.E. Eiben & J. E. Smith)

本次練習實作了兩種 deterministic parameter control policy，在 200 代的演化中，Policy 1 的設定為: 0 ~ 69 代 Gaussian mutation 的 sigma 設為 0.6；70 ~ 139 代 sigma 設定為 0.45；140 代之後 sigma 設為 0.3。Policy 2 只設定兩段：100 代以前 sigma = 0.6，100 代以後 sigma = 0.3。概念皆為隨演化結果逐漸收斂，mutation step size 隨之縮小，並與固定 sigma = 0.6 和 uniform mutation 做比較。Anytime behavior 作圖如 Fig. 9.所示。

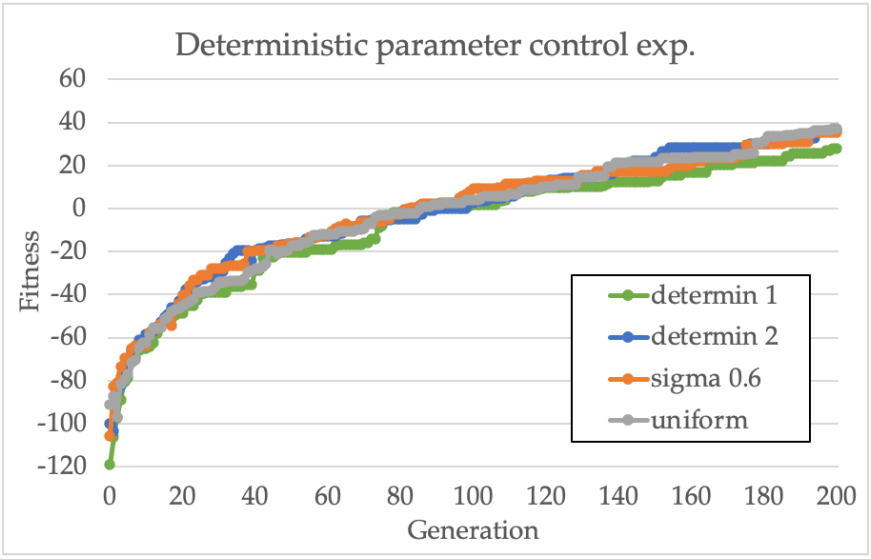


Fig. 9. Comparison of deterministic parameter control policy

Table. 1. Setting of sigma of Gaussian distribution in experiment

	Deterministic 1	Deterministic 2	Gaussian mutation	Uniform mutation
Sigma of gaussian distribution	0 - 69 gen: 0.6 70 - 139 gen: 0.45 140 - 200gen: 0.3	0 - 100 gen: 0.6 101 - 200 gen: 0.3	0.6	NA
Final fitness	27.748	36.178	35.44	37.163

雖然 policy 2 的結果比 uniform mutation 的結果差了一點，但比固定 sigma 的實驗結果略好，因此將此方法作為最終設定。

❖ Results and Discussion

實驗中不難發現，若 `SIMULATIONS_PER_EVALUATION` 的次數設得低，容易出現 fitness 偏高的現象 (雖然初始狀態為隨機，但只要 fitness 高估的都會被留下來，而低估的將被淘汰)。因此調參數實驗中的結果都僅能視為參考，不能當作可靠結果。在最終結果的實驗中，我將`SIMULATIONS_PER_EVALUATION` 設為 20，確保能得到相對可靠的結果。Table. 2. 為最終參數設定。

Table. 2. Final setting

Items	Setting
EXP_Times	10
Population size	50
Generation	300
Parent selection	K Tournament (k = 5)
Survivor selection	$(\mu + \lambda)$
Crossover rate	0.9
Mutation rate	0.001
Crossover mechanism	N-point (n = 100)
Mutation mechanism	Gaussian mutation 0 - 100 gen: sigma = 0.6 101 - 200 gen: sigma = 0.3

Fig. 10. 為 Anytime behavior 做圖，平均 fitness 為 26.392，從圖上可以觀察到不管是這次實驗或前面提及的實驗都還沒收斂，因為 search space 實在過大，設計的 operator 與 GA 架構還不能有效、快速的找到收斂解。Fig. 11. 為 10 次實驗中找到最佳解的一次，final fitness 為 101.48。

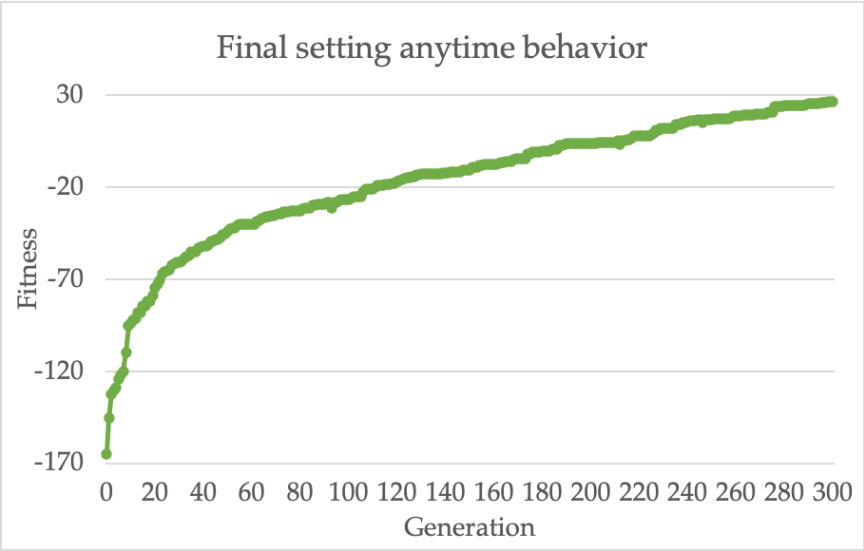


Fig. 10. Final setting anytime behavior (10 runs avg.)

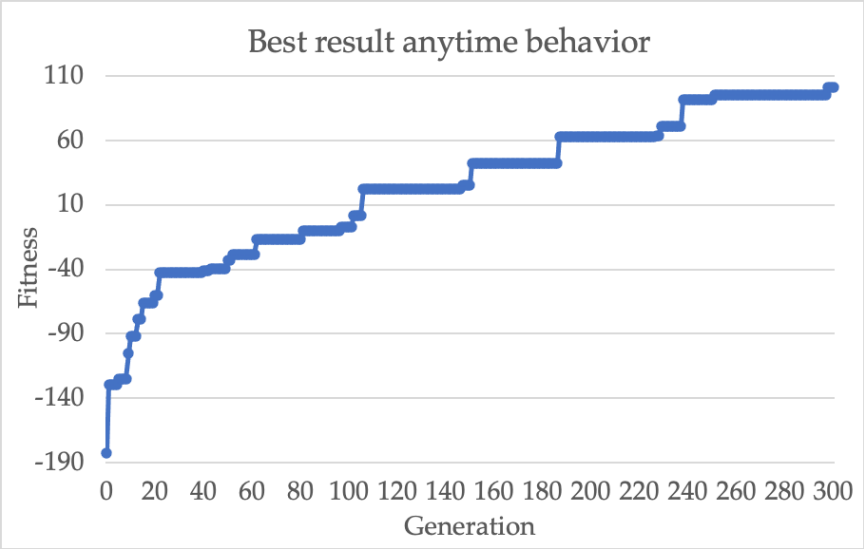


Fig. 11. Best result anytime behavior (single run)

這個作業中 GA 的 evaluation 相當昂貴，雖然麻煩，但也與現實世界中的問題相似，許多問題的 True function evaluation 成本也相當高，因此有 surrogate model 的概念產生，EC 領域也有許多相關演算法與研究，因此若花心思這個作業可以繼續鑽研，實作各種 surrogate assist method。

實驗過程中為了省時，降低了 `SIMULATIONS_PER_EVALUATION` 的次數，除了 GA 本身具有隨機性，再加上 evaluation 也具有隨機性，使得整個實驗組的偏差更大，為解決這個問題，未來可研究 low- and high-fidelity 相關的 paper，對於 simulation-based 的 evaluation 流程控制，可能有方法使演化方向不至於 low-fidelity 的 simulation 越走越偏，同時降低整體 evaluation 的 cost。

若要確保實驗收斂，拉長 generation至足夠長度，相當耗時。因此我的實作中使用 generation 100 的實驗結果來調控參數。如果未來時間足夠應以足夠長的 generation 進行測試；另外因為 GA elitism 的概念，保留了較高 fitness 的個體在 population 中，若 evaluation 不準確，產生了高估的個體，不斷在 parent selection 中被選出成為親代，也將導致 GA 的演化越走越偏。以上種種因素都可能導致 GA 耗時又效果不佳，但也點出了許多值得研究的方向，許多現實問題中也都隱含這些性質需要方法解決。