應用:基因演算法求解排課問題 (School Timetabling Problem)

程式語言:python

Representation: 整數

將排課問題切分成兩個部分：

1. 一定要滿足的條件，例如學生每一節課只能上某一堂科目，不能衝堂

(硬限制式)

1. 不一定要滿足但滿足會加分的條件，例如某個老師偏好星期一早上第一節不排課(軟限制式)

目的: 找出適應值的最大的可行解。

EA架構

* 編碼方式 (Encoding)

一周上五天課，一天有七節課。每個班級的課表編碼方式是從星期一第一節課開始編碼到星期五的第七節課。每一節課的編碼由5個部分組成，分別是班級代碼、星期代碼、節次代碼、科目代碼和教師代碼。

* 建立初始族群 (Initial Population)

建立空白課表以後，先填入全校統一課程，接著填入連堂課程，最後再填入剩下還沒有排入的單堂課程。

* 計算適應值 (Fitness Function)

針對軟限制式設計一個懲罰函數，適應值函數是懲罰函數的倒數：

**懲罰函數**

Penalty(chromosome) = \sum_{i=1}^{N} Penalty_i(chromosome)

**適應值函數**

Fitness(chromosome) = \frac{1}{Penalty(chromosome)}

當族群當中的一條染色體違反的越多，懲罰函數的值就越高，適應值函數的值就會越小。

* 選擇 (Selection)

採用輪盤法 (Roulette Wheel Selection)，挑選父代染色體時，適應值函數的值越大的染色體被挑選為父代染色體的機會越大。

* 交配 (Crossover)

透過「單點交配」的方式進行交配，但是由於要保持染色體為可行解的特性，所以決定交配點後，父代交換基因時，若交換基因會導致子代染色體變為不可行解，則該基因位置先空白，後續再利用類似建立初始族群的方式修補染色體，使子代染色體也保持可行解。

* 突變 (Mutation)

使用「交換突變」，先針對染色體選出兩個突變點，再將突變點的基因進行交換。

* 產生新族群 (Offspring)

當染色體進行完選擇、交配和突變三種演化機制且子代染色體達到所設定之族群大小後，會將產生的子代族群取代原本的族群。在這裡採用菁英策略進行取代，會將母代族群中適應值最佳的染色體保留至子代作為下一代的族群。

* 終止條件 (Termination)

1. 適應值大於某特定目標值
2. 演算法達到最大迭代次數(60次)

做此專案的目的：

目標是找到一個符合所有硬限制條件的排課方案，同時盡量滿足軟限制條件。硬限制是絕對不能違反的規則，例如學生不能衝堂。軟限制是可選擇性的條件，但若滿足會使解更好，例如老師偏好某些時間不排課。

**程式碼解釋**

1. 資料表示 (Data Representation)
   * Course 類別：代表一堂課程的資訊，包含班級代碼、星期代碼、節次代碼、科目代碼和教師代碼等資料。
   * Timetable 類別：代表整個課表，使用二維陣列來儲存課程的安排情況。它包含以下方法：

add\_course: 將一堂課加入指定的日期和時間。

get\_course: 取得指定日期和時間的課程資訊。

\_\_str\_\_: 將課表以易讀的方式呈現出來。

1. 初始化族群 (Initial Population)

* create\_timetable 函式：隨機生成初始的課表。它隨機排列課程，確保每堂課都被安排在可行的時間段內。

1. 計算適應值 (Fitness Function)

* penalty\_function 函式：用來評估課表是否違反了軟限制條件。它目前只考慮了老師的偏好，給予不同程度的懲罰值。
* fitness\_function 函式：根據懲罰函數計算適應值，並以懲罰函數的倒數來表示適應度。適應值越高代表解越好。

1. 選擇 (Selection)

* roulette\_wheel\_selection 函式：使用輪盤法選擇父代染色體，適應值較高的染色體被選中的機率較大。

1. 交配 (Crossover)

* single\_point\_crossover 函式：進行單點交配，創造子代染色體，保持染色體為可行解的特性。

1. 突變 (Mutation)

* mutate\_timetable 函式：對染色體進行交換突變，隨機改變染色體的部分以增加多樣性。

1. 主要演算法循環

主要的演算法循環包含了初始化族群、選擇、交配、突變等步驟。在每個世代中，計算每個染色體的適應值，然後進行選擇、交配和突變操作來生成下一代族群，直到達到終止條件。

1. 最後找到最佳解

當演算法達到終止條件後，程式會找到當前最優解（適應值最高的染色體）並將其呈現出來。

**探討擴展適應值函數、選擇策略和參數調整的方式：**

1. 更複雜的適應值函數

考慮更多軟限制條件:例如 擴展 penalty\_function，考慮更多的軟限制條件。

比如：

* + 學生特定時間段的休息需求。
  + 老師的特殊時間要求。
  + 課程之間的先修要求或者不相容性。
  + 教室設備的使用限制。
* 權重調整不同條件的影響

引入權重系統來調整不同軟限制條件對適應值的影響程度。例如，某些條件的違反可能對最終適應值的影響更大，你可以根據這些情況調整懲罰函數中的權重值。

1. 不同的選擇、交配或突變策略

* 不同的選擇機制

除了輪盤法，你可以嘗試其他選擇機制，如競爭性選擇（Tournament Selection）、排名選擇（Rank Selection）等，這些方法可能會帶來不同的族群多樣性和收斂速度。

* 不同的交配方式

嘗試其他的交配方式，例如多點交配、均勻交配等，這些方式可能會對族群的探索能力和交配後染色體的品質產生影響。

* 不同的突變策略

除了交換突變，可以考慮其他的突變方式，例如插入、刪除、反轉等方式，這些策略也許會對解的搜索空間產生更廣泛的影響。

1. 參數調整以優化效能

* 族群大小和突變率

調整族群的大小和突變率，這兩個參數對演算法的收斂速度和最終解的品質都有影響。更大的族群可能帶來更多的探索，但也會增加計算成本。

* 演算法迭代次數

增加或減少演算法的迭代次數，有時可能需要更多的迭代來達到更好的解，而有時則可能在較少的迭代次數內獲得令人滿意的結果。

1. 更多註解和可視化

增加詳細的註解，讓程式碼更易理解。同時，可以利用可視化工具將課表的安排以圖形化方式呈現，這有助於觀察演算法的運作情況和最終解的表現。