



模擬學 FLEXSIM

FINAL PROJECT

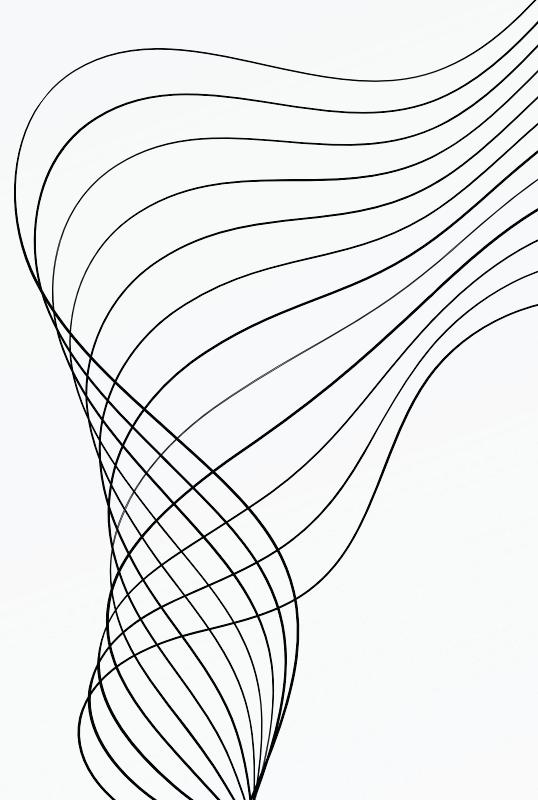
第三組

110704019 蔡宛秦

110704025 白欣怡

110704039 許甄芸

110704059 張殷祈



CONTENT

MODELING AND SOLVING JOB-SHOP SCHEDULING PROBLEMS

- 01 工作分配
- 02 文獻回顧
- 03 研究方法 - **GENETIC ALGORITHM**
- 04 研究方法 - **SHIFTING BOTTLENECK HEURISTIC**
- 05 研究結果
- 06 智慧排程
- 07 類似 JSP 的調度問題
- 08 商業模擬軟體

工作分配

蔡宛秦 25%

- 處理 DMU 4 1 資料
- 撰寫基因演算法程式
- 執行程式排序結果
- 商業模擬軟體
- 製作報告

白欣怡 25%

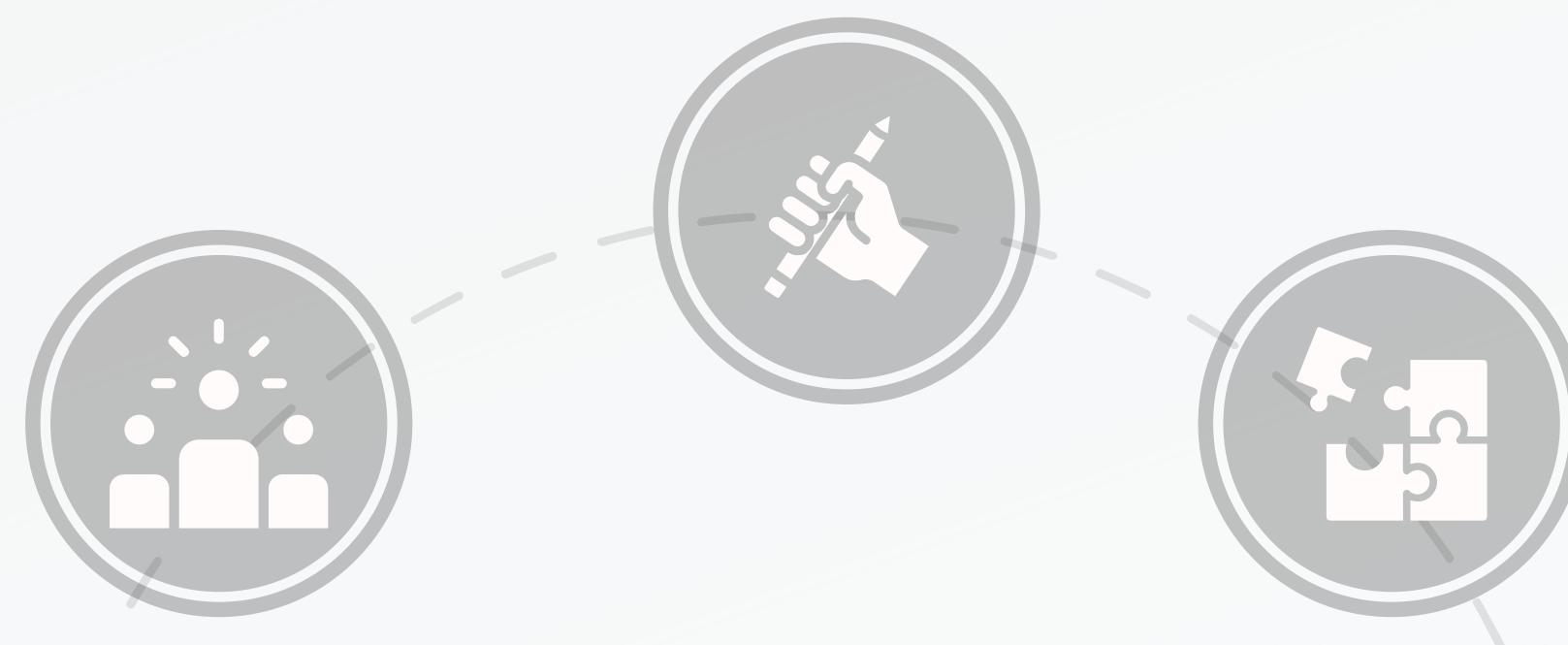
- 處理 DMU 4 5 資料
- 撰寫基因演算法程式
- 執行程式排序結果
- 製作報告

許甄芸 25%

- 處理 DMU 4 3、4 4 資料
- 撰寫移動瓶頸演算法程式
- 執行程式排序結果
- JSP的調度例子
- 製作報告

張殷祈 25%

- 處理 DMU 4 2 資料
- 撰寫移動瓶頸演算法程式
- 執行程式排序結果
- 智慧排程
- 製作報告





基因演算法

GENETIC ALGORITHM

時間複雜度: Big O(n^2)

文獻回顧

- 過去 Job-shop Scheduling Problems 排程之比較

文獻	研究作者	研究方法	研究目標	研究問題集
Flexible Job Shop Scheduling using a Multiobjective Memetic Algorithm	Lin (2011)	基因演算法 (GA) 搭配 禁忌搜尋法 (TS)	多目標彈性零工式工廠排程 最小化完工時間、機台總工作量、最大機台工作量	Kacem data 與 BR data 共十五 個測試問題
A Three-Phase GA Algorithm with an Operation-Sequence-Based Chromosome Representation for Scheduling Flexible Job Shops	Huang (2012)	三階段的基因演算法 (GA_Sop)	最小化完工時間	Brandimarte 的 資料集(BR data) mk 01~10
A Modified Genetic Algorithm with Local Search Strategies and Multi-Crossover Operator for Job Shop Scheduling Problem	Viana et al. (2020)	Multi-Crossover Local Search Genetic Algorithm	最小化完工時間	FJSP-MR

流程說明

1. 編碼 : ($x_1, x_2 \dots$)

2. 初始化群體

3. 算適應度值 :

$\text{array_pop_fit} = f(x_1, x_2 \dots)$

$t = t + 1$

4. 選擇 :

隨機從目前 population 中選出一對父母
($x_{1\text{mom}}, x_{2\text{mom}} \dots$) 和 ($x_{1\text{dad}}, x_{2\text{dad}} \dots$)

5. 繁衍 :

$x_{1\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x1}(0, \sigma)$

$x_{2\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x2}(0, \sigma)$

6. 取代 :

$\text{Pop}(t+1) = \{\text{Pop}(t) - \text{worst}\} \cup \{\text{kid}\}$

Iteration = 100

流程說明

1. 編碼 : ($x_1, x_2 \dots$)

2. 初始群體

3. 算適應度值 :

$\text{array_pop_fit} = f(x_1, x_2 \dots)$

4. 選擇 :

隨機從目前 population 中選出一對父母
($x_{1\text{mom}}, x_{2\text{mom}} \dots$) 和 ($x_{1\text{dad}}, x_{2\text{dad}} \dots$)

5. 繁衍 :

$x_{1\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x1}(0, \sigma)$

$x_{2\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x2}(0, \sigma)$

6. 取代 :

$\text{Pop}(t+1) = \{\text{Pop}(t) - \text{worst}\} \cup \{\text{kid}\}$

方法說明

1. 染色體編碼

- 15個1 & 15個2 & ... & 個15個20的隨機排列

111111111111111111122222222222222222222...



15個

- 15個“1”分別代表JOB 1的15個加工機台
15個“2”分別代表JOB 2的15個加工機台

A	
1	Buffer_4
2	Buffer_2
3	Buffer_1
4	Buffer_5
5	Buffer_3
6	Buffer_0
7	Buffer_6
8	Buffer_11
9	Buffer_10
10	Buffer_12
11	Buffer_8
12	Buffer_7
13	Buffer_13
14	Buffer_9
15	Buffer_14

流程說明

1. 編碼 : ($x_1, x_2 \dots$)

2. 初始群體

3. 算適應度值 :

$\text{array_pop_fit} = f(x_1, x_2 \dots)$

4. 選擇 :

隨機從目前 population 中選出一對父母
($x_{1\text{mom}}, x_{2\text{mom}} \dots$) 和 ($x_{1\text{dad}}, x_{2\text{dad}} \dots$)

5. 繁衍 :

$x_{1\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x1}(0, \sigma)$

$x_{2\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x2}(0, \sigma)$

6. 取代 :

$\text{Pop}(t+1) = \{\text{Pop}(t) - \text{worst}\} \cup \{\text{kid}\}$

方法說明

2. 初始化群體

- 重要參數

```
int NUM_MACHINE = 15;
int NUM_JOB = 20;
int NUM_ITERATION = 100;
int NUM_CHROME = 40;
int NUM_BIT = NUM_JOB * NUM_MACHINE;
double Pc = 1;           //交配率
double Pm = 0.2;          //突變率
int NUM_PARENT = NUM_CHROME;
int NUM_CROSSOVER = Math.round(Pc * NUM_CHROME / 2);
int NUM_CROSSOVER_2 = Math.round(NUM_CROSSOVER*2);
int NUM_MUTATION = (Pm * NUM_CHROME * NUM_BIT);
```

流程說明

1. 編碼 : ($x_1, x_2 \dots$)

2. 初始群體

3. 算適應度值 :

$\text{array_pop_fit} = f(x_1, x_2 \dots)$

4. 選擇 :

隨機從目前 population 中選出一對父母
($x_{1\text{mom}}, x_{2\text{mom}} \dots$) 和 ($x_{1\text{dad}}, x_{2\text{dad}} \dots$)

5. 繁衍 :

$$x_{1\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x1}(0, \sigma)$$

$$x_{2\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x2}(0, \sigma)$$

6. 取代 :

$$\text{Pop}(t+1) = \{\text{Pop}(t) - \text{worst}\} \cup \{\text{kid}\}$$

方法說明

3. 算適應度值：

- $S[i][j] = \text{job } i \text{ 在機台 } j \text{ 的開始時間}$
- $P[i][j] = \text{job } i \text{ 在機台 } j \text{ 的執行時間}$
- $C[i][j] = \text{job } i \text{ 在機台 } j \text{ 的完成時間}$
- $S[i][j] = \max\{C[i-1][j], C[i][j-1]\}$
- $C[i][j] = S[i][j] + P[i][j]$

目標：最小化makespan

適應度函式： $\text{minimize } C[i][j]$

流程說明

1. 編碼 : ($x_1, x_2 \dots$)

2. 初始化群體

3. 算適應度值 :

$\text{array_pop_fit} = f(x_1, x_2 \dots)$

4. 選擇 :

隨機從目前 population 中選出一對父母
($x_{1\text{mom}}, x_{2\text{mom}} \dots$) 和 ($x_{1\text{dad}}, x_{2\text{dad}} \dots$)

5. 繁衍 :

$x_{1\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x1}(0, \sigma)$

$x_{2\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x2}(0, \sigma)$

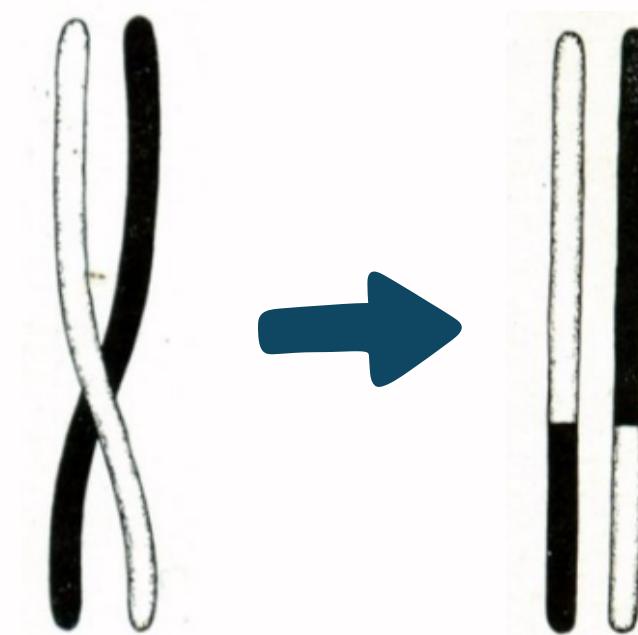
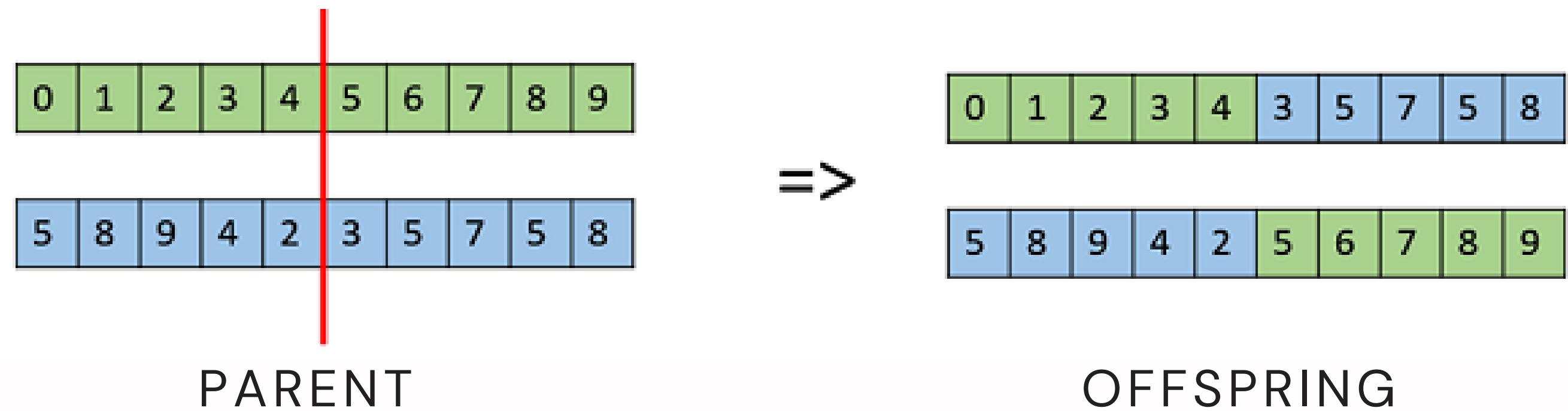
6. 取代 :

$\text{Pop}(t+1) = \{\text{Pop}(t) - \text{worst}\} \cup \{\text{kid}\}$

方法說明

4. 選擇 & 5. 繁衍

- 單點交配(one point crossover)：
隨機選擇父母與交配點
- 突變(mutation)：一條基因的任兩點互換

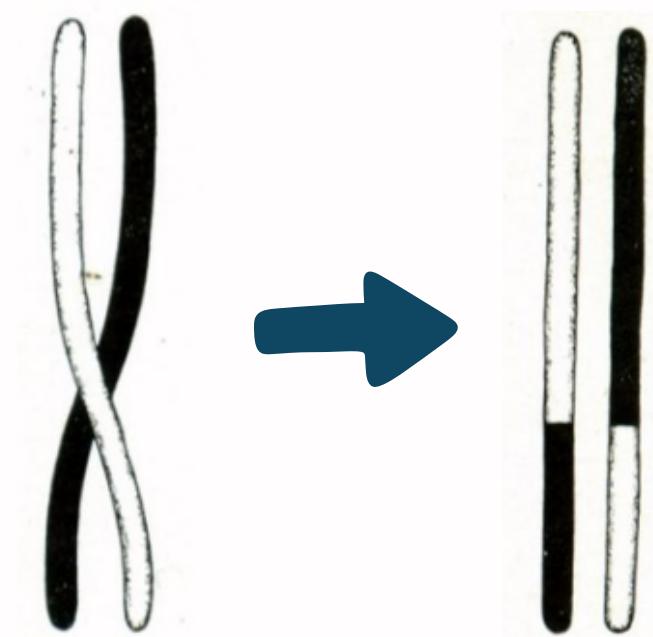
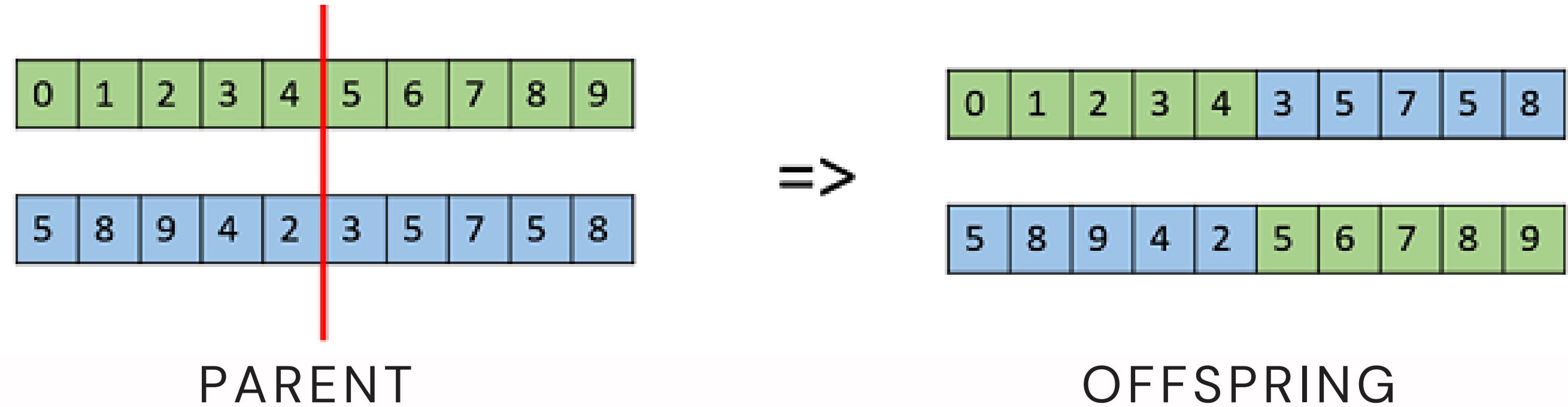


* 圖片取自網路

方法說明

4. 選擇 & 5. 繁衍

- 單點交配(one point crossover)：
隨機選擇父母與交配點
- 突變(mutation)：一條基因的任兩點互換



* 圖片取自網路

流程說明

1. 編碼 : ($x_1, x_2 \dots$)

2. 初始群體

3. 算適應度值 :

$\text{array_pop_fit} = f(x_1, x_2 \dots)$

4. 選擇 :

隨機從目前 population 中選出一對父母
($x_{1\text{mom}}, x_{2\text{mom}} \dots$) 和 ($x_{1\text{dad}}, x_{2\text{dad}} \dots$)

5. 繁衍 :

$x_{1\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x1}(0, \sigma)$

$x_{2\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x2}(0, \sigma)$

6. 取代 :

$\text{Pop}(t+1) = \{\text{Pop}(t) - \text{worst}\} \cup \{\text{kid}\}$

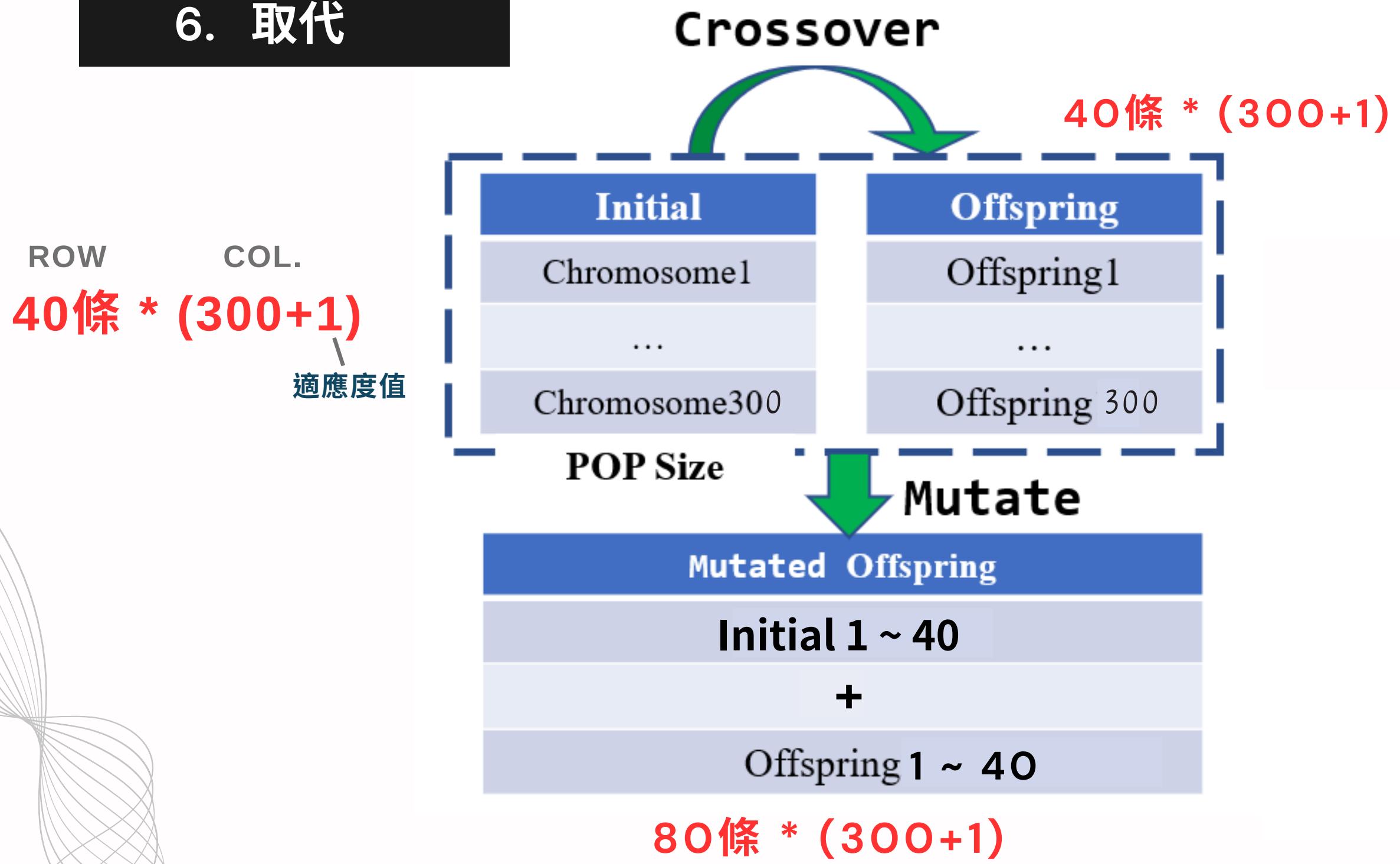
方法說明

6. 取代

- 從新一代染色體中選擇一些個體來取代上一代中的個體
 1. 計算子代的適應度
 2. 子代與母代結合，選出適應度值最佳的40條染色體

方法說明

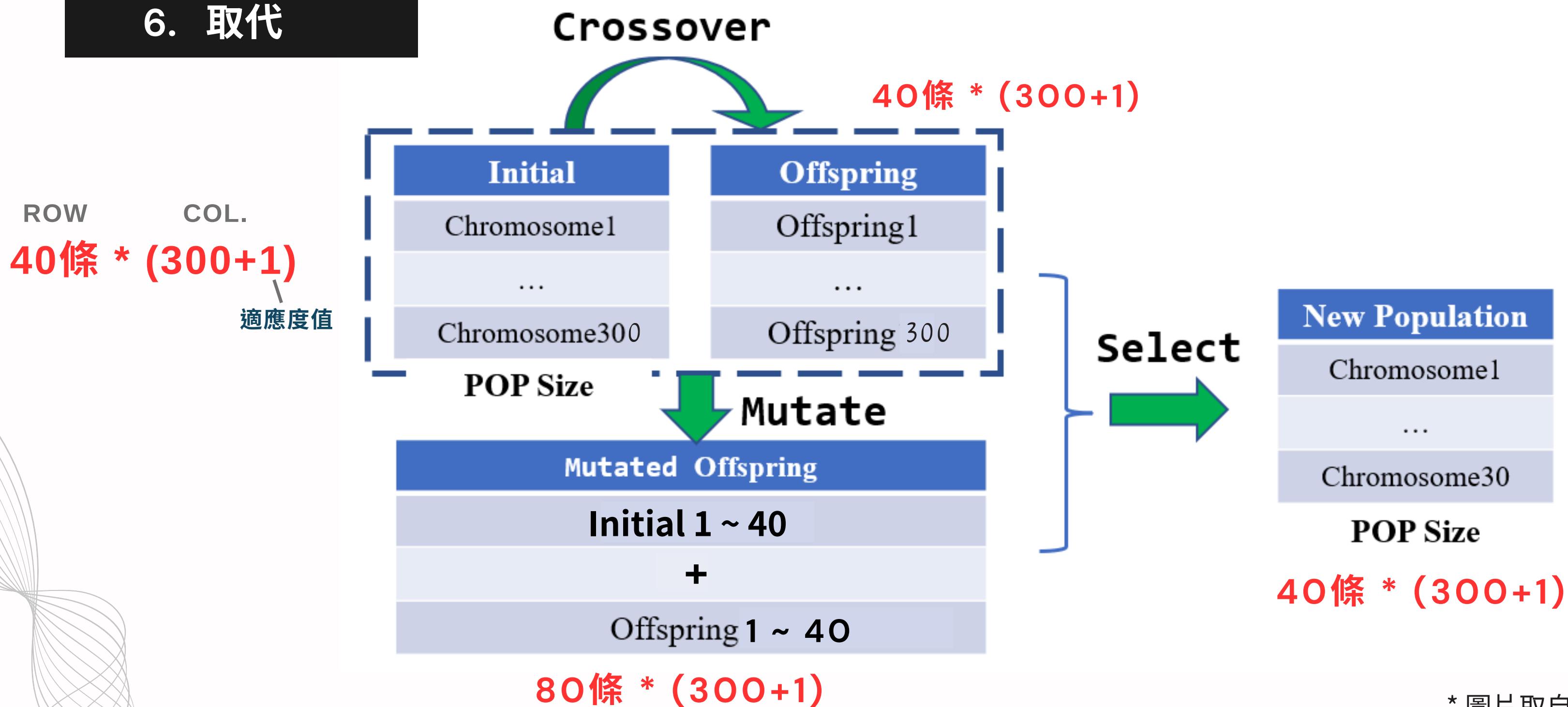
6. 取代



* 圖片取自網路

方法說明

6. 取代



* 圖片取自網路

流程說明

1. 編碼 : ($x_1, x_2 \dots$)

2. 初始群體

3. 算適應度值 :

$\text{array_pop_fit} = f(x_1, x_2 \dots)$

$t = t + 1$

4. 選擇 :

隨機從目前 population 中選出一對父母
($x_{1\text{mom}}, x_{2\text{mom}} \dots$) 和 ($x_{1\text{dad}}, x_{2\text{dad}} \dots$)

5. 繁衍 :

$x_{1\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x1}(0, \sigma)$

$x_{2\text{kid}} = \text{rnd}(x_{1\text{mom}}, x_{2\text{dad}}) + N_{x2}(0, \sigma)$

6. 取代 :

$\text{Pop}(t+1) = \{\text{Pop}(t) - \text{worst}\} \cup \{\text{kid}\}$

Iteration = 100

FLEXSIM

展示

GA BEST SOLUTION

時間複雜度: Big O(n^2)

Instance	DMU41	DMU42	DMU43	DMU44	DMU45
Cmax	4002	4468	4543	4573	4252

移動瓶頸法

SHIFTING BOTTLE NECK HEURISTIC

時間複查雜度: Big O(n^5)

文獻回顧

- Shifting Bottleneck Heuristic Algorithm 相關論文

文獻	研究作者	研究方法	研究目標
A distributed shifting bottleneck heuristic for complex job shops	Lars Mönch, René Drießel(2004)	a modified shifting bottleneck heuristic	minimize the makespan
A hybrid shifting bottleneck-tabu search heuristic for the job shop total weighted tardiness problem	Kerem Bala, Ulku(2010)	hybrid shifting bottleneck-tabu search (SB-TS) algorithm	minimizing the total weighted tardiness
Job Shop Scheduling Using Shifting Bottleneck Heuristic Algorithm	Bobi Tri Tamtomo, Dwi Agustina Kurniawati, Noordin Mohd. Yusof, Kuan Yew Wong (2021)	Shifting Bottleneck Heuristic algorithm	minimize the makespan

方法說明

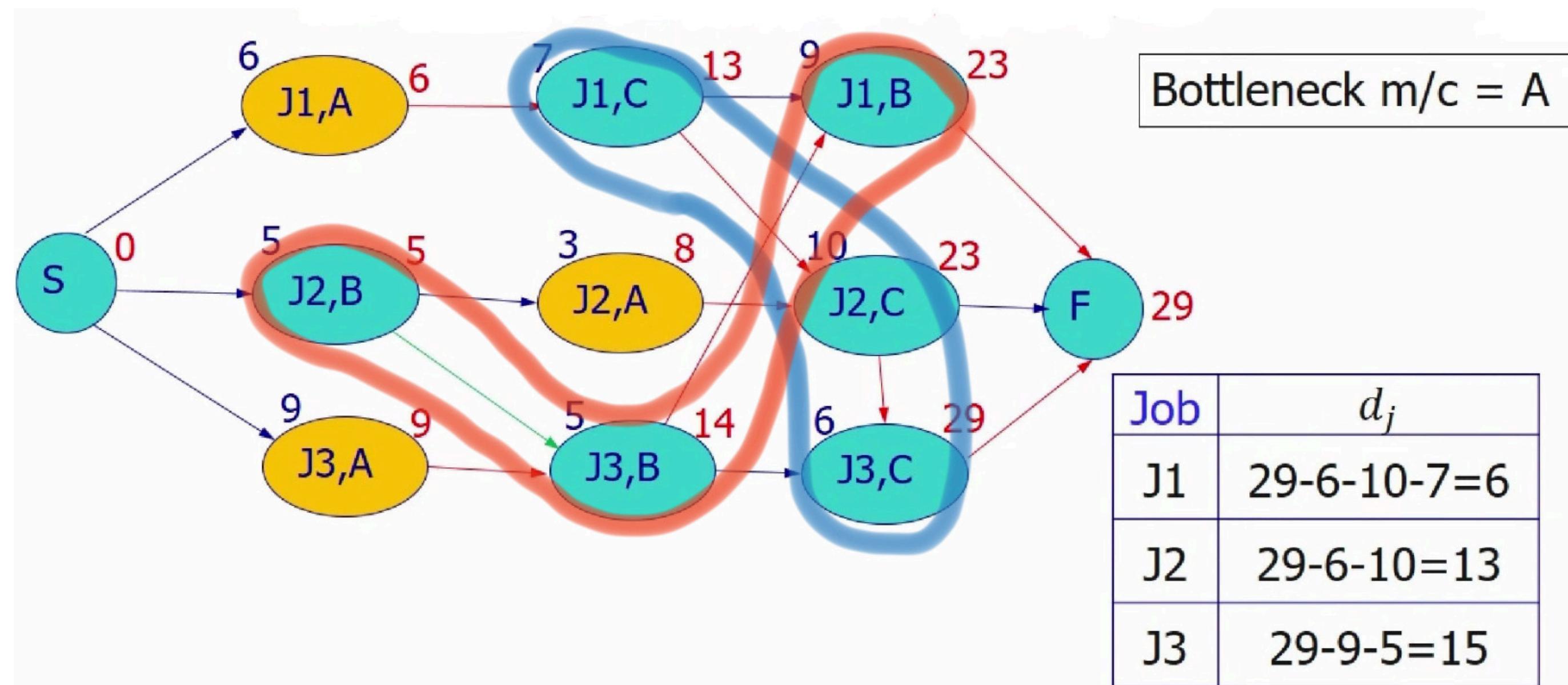
1. 計算各machine的 Σ work load並取max值當作makespan LB
2. 找出切換瓶頸機台的順序
3. 利用 p_j, r_j, d_j, C_j 計算出 T_j ，並選出每個job的 $T_{j\max}$
4. 使用branch and bound找出 $T_{j\max}$ 中最小的job sequence
5. 將新的順序update至network，找出longest path(新的makespan)
6. 從新的network中更新下個瓶頸機台的 d_j
7. 切換至下個瓶頸機台，每個機台都要重複步驟3~6

FLEXSIM

展示

CODE

從新的NETWORK計算下個瓶頸機台的 D_j



* 圖片取自網路

6.從新的network中更新下個瓶頸機台的d_j

- 將新路線的機台PROCESSING TIME更新，之後的機台清0
- 若瓶頸機台在新路線機台之後則不變

```
//填update_route表格
int update_count = 1;
string bufferName_next = "Buffer_" + gettablenum("MachineSum", i+1, 1);
for(int j = 1; j <= 60; j+=3)
{
    int next_buffer_position = 0;
    int new_route_count = 0;
    for(int k = 1;k <= 15; k++)
    {
        if (gettablestr("UpdateRoute", k, j) == bufferName_next)
        {
            next_buffer_position = k;
        }
    }
    for(int k = 1;k <= 15; k++)
```

```
{  
    if (gettablestr("UpdateRoute", k, j) == bufferName)  
    {  
        if(next_buffer_position < k)  
        {  
            for(int l = 1;l <= 20;l++)  
            {  
                if(gettablenum("dispatch_pj", l, 1) == update_count)  
                {  
                    for(int m = l;m <= 20;m++)  
                    {  
                        new_route_count += gettablenum("dispatch_pj", m, 2);  
                    }  
                    break;  
                }  
            }  
            settablenum("UpdateRoute", k, j+2, new_route_count);  
            for(int l = k+1;l <= 15; l++)  
            {  
                settablenum("UpdateRoute", l, j+2, 0);  
            }  
            break;  
        }  
        else  
        {  
            break;  
        }  
    }  
    update_count += 1;
```

SBN BEST SOLUTION

	DMU41	DMU42	DMU43	DMU44	DMU45
1st run	4422	4841	4807	4993	4577
2nd run	4134	4738	5105	4838	4513
3rd run	4524	4712	4753	4993	4917
4th run	4320				

BEST SOLUTION

Instance	DMU41	DMU42	DMU43	DMU44	DMU45
Cmax	4002	4468	4543	4573	4252

智慧排程

- What is intelligent scheduling?

利用AI技術來自動化和優化排程，提高利用效率和整體生產力。

1. 資料收集範圍擴大到不同環節
2. 接獲訂單後，智慧排程系統就會即時查詢產線產能、倉儲系統內的原物料存貨、近期訂單等資訊，並在短時間顯示排程結果
3. 掌握生產設備狀態，準確預測、調度、規劃未來產能。

智慧排程

- How it is different from the traditional scheduling?
 1. 使用方便、後續資料調整和功能擴充也較容易
 2. 操作介面簡單直觀，比傳統排程系統易上手
 3. AI排程引擎讓產線更有彈性，可因應全球多變的供應鏈，使產能最佳化
 4. 成本較傳統排程系統低

類似JSP的調度問題

- **醫院手術室排程：**

最大化手術室和醫療人員的利用率，同時最小化病人的等待時間和手術延遲。

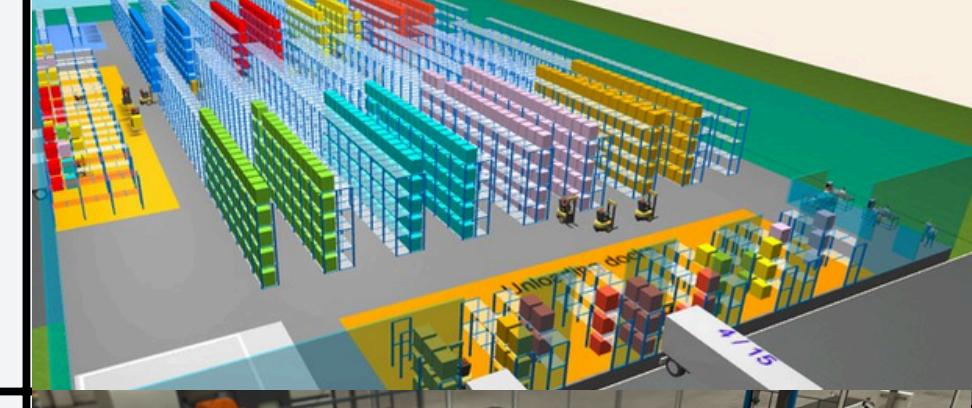
- **機場地面操作：**

管理機場多架飛機的地面操作排程（例如加油、行李處理、維修）。每項操作需要特定的設備和人員，必須在特定的時間窗口內完成，以確保航班按時起飛。

- **車輛維修排程：**

為一車隊（例如巴士、卡車）安排維修和保養任務。以最小化車輛停機時間並最大化維修資源的利用率為目標。

商業模擬軟體

軟體	主要功能	費用	介面
 FlexSim	離散事件模擬軟體，可建立3D 模擬環境、具備流程分析工具，並提供統計報告、數據分析和即時數據形成	\$5,000 USD/ year	
 AnyLogic	支持三種建模方法：系統動力學、離散事件模擬、代理基模擬，提供可視化與動畫，允許用戶在模擬過程中觀察模型行為和變化	\$4000 USD (researcher version)	
 Factory I/O	提供工廠仿真建模、場景編輯器、控制系統整合、即時互動、故障排除、配備真實感的物理引擎，模擬真實世界中的物理特性	17 €/month	

*圖片來源：

https://www.google.com/search?sca_esv=7848dc2536e4d400&rlz=1C1CHZN_zh-TWTW995TWTW995&q=flexsim&udm=2&fbs=AEONm0AaBOazvTRM_Uafu9eNJJzC3QMRKTS5UleA1ZwBo3sfl5tRK2wzmpOoTr82Uvr9kDU5R00xZEAgPH8kQ6uw4YLujEAiXbmg8cQIkVdEzftlSVnJamesF_YaiMzeimFnugbDbOnfirnOPJlamty4Q_DLvYS8qdzbheUpREWKyQj6swPQiguXGYLqZeZBmG6k_bgGxd&sa=X&ved=2ahUKEwi9LSpuqGAXuVm68BHcSwJ4QQtKgLegQIDRAB&biw=1422&bih=612&dpr=1.35#vhid=CopZLuMs4hBTmM&vssid=mosaic

https://www.google.com/search?sca_esv=7848dc2536e4d400&rlz=1C1CHZN_zh-TWTW995TWTW995&q=AnyLogic&udm=2&fbs=AEONm0AaBOazvTRM_Uafu9eNJJzC3QMRKTS5UleA1ZwBo3sfl5tRK2wzmpOoTr82Uvr9kDU5R00xZEAgPH8kQ6uw4YLujEAiXbmg8cQIkVdEzftlSVnJamesF_YaiMzeimFnugbDbOnfirnOPJlamty4Q_DLvYS8qdzbheUpREWKyQj6swPQiguXGYLqZeZBmG6k_bgGxd&sa=X&ved=2ahUKEwiH1oSYOOqGAXudxPUHHTRmAOOQtKgLegQIDRAB&cshid=1718902022103506&biw=1422&bih=612&dpr=1.35#vhid=LzPSC5tL5gmBeM&vssid=mosaic

https://www.google.com/search?q=Factory+I%2FO&sca_esv=7848dc2536e4d400&rlz=1C1CHZN_zh-TWTW995TWTW995&udm=2&biw=1422&bih=612&ei=CFIOZviGlvg1e8PkjiUsAI&ved=OahUKEwj3hsaEOeqGAXuLdfuUHHRAMBSYQ4dUDCBA&uact=5&oq=Factory+I%2FO&gs_ip=Egxnd3Mtd2i6LXNlcniCOZhY3RvcnkgsS9PMgcQAbiABBgTMgcQAbiABBgTSPEKUjsFWJsFcAJ4AJABAJgBN6ABN6oBATG4AQPIAQD4AQL4AQGYAqQgAkbcAgUQAbiABMICBBAAAG7CAGYQABgHGB6YAwCIBgGSBwEzoArdt&sclient=gws-wiz-serp#vhid=IC_kmwIQNxPwtM&vssid=mosaic

THANK'S FOR WATCHING