

Simulation

FINAL PROJECT

第一週第九組

109704001許恒睿

109704038孫于涵

109704042陳博翰

做足準備

- 問題格式

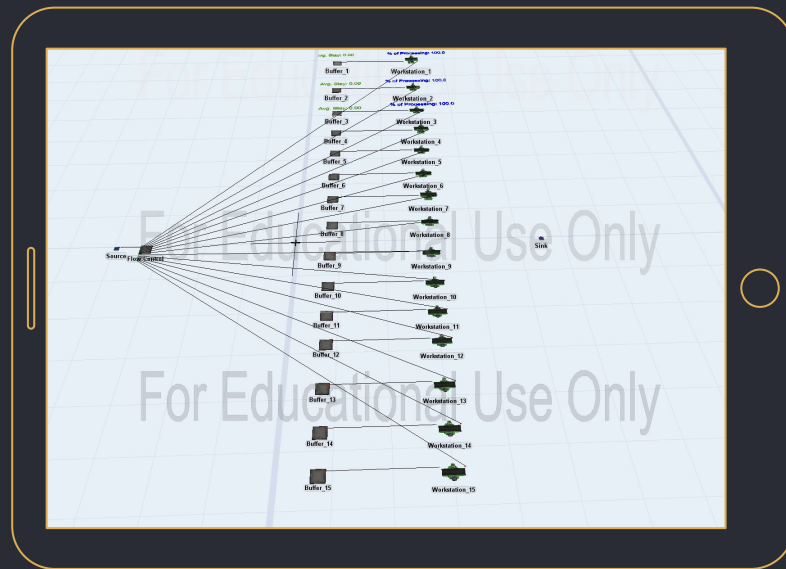
- 20 x 15 (Jobs x machines)
- All JSSP(No Flexsim)

- 問題種類

- rcmax
- csmx

- 使用工具

- Python
- Flexsim



使用元啟發式演算法來解決JSSP

基因演算法 (Genetic Algorithm)

第1步

設定GA的超參數包括
群體大小、突變率
迭代次數、交配率



第2步

`initPop()`

產生隨機的JSSP排程可行解 (基因)
基因組成染色體之個數與群體一致



第3步

`evaluatePop()`

評估每個群體中個體適應度
函數設定為對JSSP的時間長短



第4步

`crossover()`、`mutation()`

模擬生物配對
和突變 (以避免落入局部優解)



第5步

`replace()`

依照個體的適應度
適者生存, 不適者淘汰



第6步

不停迭代,
迭代時適應度會越來越好
直到迭代次數達到則停止



部份程式碼和概念源自 林春成教授 課程-基因演算法與管理科學

難道這樣就好？

The screenshot shows a Windows 10 desktop environment. In the foreground, a Jupyter Notebook is open in a web browser, displaying a table with columns 'year', 'target', and 'actual'. Below the table, there is a code cell with a dictionary definition for 'actual'. Overlaid on the Jupyter Notebook is a green Microsoft Excel 2019 splash screen. In the background, a file explorer window is open, showing a list of files with names like 'Excel_2019_1', 'Excel_2019_2', etc., and their corresponding dates.

連結: <https://www.youtube.com/watch?v=CglBegEmSRs>

本組特色

From 工人智慧 to 人工智慧



超參數優化

以貝式優化法效率調參
以高斯函數自動化
找出較優的超參數
免除人工亂猜調參



我們的GITHUB



自動化

我們只須一鍵RUN
就能自動歷遍資料夾中的
所有問題得出解
省去人工慢慢Key輸出

超參數優化

hyperparameter optimization

Algorithm 1 Sequential Model-Based Optimization

Input: $f, \mathcal{X}, S, \mathcal{M}$
 $\mathcal{D} \leftarrow \text{INITSAMPLES}(f, \mathcal{X})$
for $i \leftarrow |\mathcal{D}|$ **to** T **do**
 $p(y | \mathbf{x}, \mathcal{D}) \leftarrow \text{FITMODEL}(\mathcal{M}, \mathcal{D})$
 $\mathbf{x}_i \leftarrow \arg \max_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x}, p(y | \mathbf{x}, \mathcal{D}))$
 $y_i \leftarrow f(\mathbf{x}_i)$ ▷ Expensive step
 $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{x}_i, y_i)$
end for

```
optimizer = BayesianOptimization(  
    f = GA_solver,  
    pbounds = {  
        'group_num': (10, 400),  
        'crossover_rate': (0.00001, 0.99999),  
        'mutation_rate': (0.00001, 0.99999),  
    },  
    random_state=0,  
)  
optimizer.maximize(  
    init_points = 2,  
    n_iter = 140,  
)
```

使用貝葉斯 (Bayesian Optimization) 優化超參數
超參數包括交配率、突變率、群體個數

超參數優化

hyperparameter optimization

```
(base) yarikama@aki8a:~/Desktop/Simulation$ python GA.py
請輸入檔案路徑: /home/yarikama/Desktop/Simulation/r20_15
| iter | target | crosso... | group_num | mutati... |
|-----|-----|-----|-----|-----|
| 1 | -4.211e+0 | 0.5488 | 288.9 | 0.6028 |
| 2 | -4.31e+03 | 0.5449 | 175.2 | 0.6459 |
| 3 | -4.009e+0 | 0.6317 | 287.6 | 0.772 |
| 4 | -4.143e+0 | 0.6854 | 287.8 | 0.7582 |
| 5 | -4.228e+0 | 0.5411 | 225.7 | 0.7326 |
| 6 | -4.243e+0 | 0.7323 | 287.6 | 0.7747 |
| 7 | -4.371e+0 | 0.372 | 55.04 | 0.8571 |
| 8 | -4.188e+0 | 0.9066 | 286.0 | 0.2393 |
| 9 | -4.281e+0 | 0.9831 | 261.0 | 0.864 |
| 10 | -3.515e+0 | 0.206 | 80.61 | 0.0003831 |
| 11 | -4.414e+0 | 0.822 | 57.09 | 0.4002 |
| 12 | -4.049e+0 | 0.8175 | 389.9 | 0.1061 |
| 13 | -4.348e+0 | 0.2946 | 76.46 | 0.5298 |
| 14 | -4.202e+0 | 0.5232 | 236.6 | 0.1363 |
| 15 | -4.363e+0 | 0.793 | 45.76 | 0.3135 |
| 16 | -4.17e+03 | 0.2571 | 291.7 | 0.9751 |
| 17 | -4.276e+0 | 0.1659 | 156.4 | 0.8618 |
| 18 | -3.919e+0 | 0.8742 | 80.82 | 0.0432 |
| 19 | -4.306e+0 | 0.5161 | 364.0 | 0.4203 |
| 20 | -4.003e+0 | 0.9711 | 353.8 | 0.0997 |
| 21 | -4.201e+0 | 0.4618 | 346.8 | 0.1054 |
```

使用貝葉斯(Bayesian Optimization)優化超參數
超參數包括交配率、突變率、群體個數
利用固定隨機種子來追溯結果

自動化

1.

自動讀檔

自動將.txt的問題設定
轉換成適用在GA和
Flexsim上的格式
解決慢慢做出格式的問題

2.

自動調參數

不需要依照人工調參
數, 只要迭代次數較
少, 即可得出好解

3.

自動甘特圖輸出

若是想看到機台的使用
情況GA.py會自動匯出
結果甘特圖片自資料夾

4.

自動得出最優解

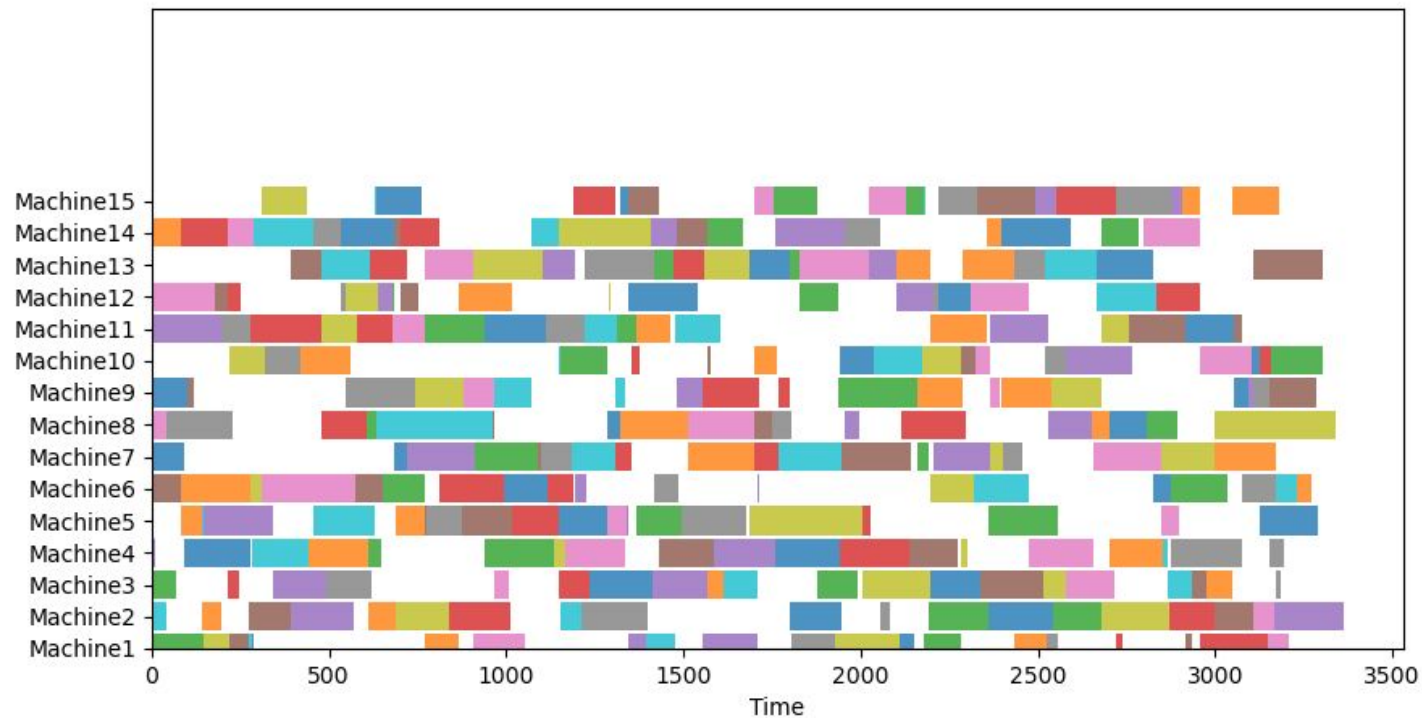
將優化參數得出的解
自動匯出成excel格式
以便匯入至Flexsim驗證

5.

自動統一檔名

統一格式讓Flexsim直接匯入
只輸出一個含有多數個工作
集的excel檔, 避免檔案雜亂

甘特圖範例



2,986

RCMAX_20_15_8 FINAL MAKESPAN
NUM_ITERATION ONLY COUNTS **400** TIMES
UB 2669

```

new_folder_path = mkdir()
pTime = [[file_list[i][j] for j in range(1, len(file_list[i]))] for i in range(1, len(file_list))]
mOrder = [[file_list[i][j] for j in range(1, len(file_list[i]))] for i in range(1, len(file_list))]
route_xlsx()

```

```

NUM_JOB = file_list[0][0]
NUM_MACHINE = file_list[0][1]
NUM_BIT = NUM_JOB * NUM_MACHINE

# ==== 參數設定 (與演算法相關) ====
GA_ITERATION = 0
NUM_ITERATION = 400

```

182	-3.081e+0	0.7973	241.9	0.005337
183	-3.304e+0	0.8989	214.2	0.009916
184	-3.373e+0	0.8823	50.57	0.005297
185	-2.986e+0	0.7941	241.8	0.003591
186	-3.257e+0	0.8297	108.5	0.005993
187	-3.282e+0	0.8334	221.8	0.00798
188	-3.195e+0	0.582	196.1	0.004742

FlexSim - C:\Users\USER\Downloads\final_test5 (1).fsm

File Edit View Build Execute Statistics Tools Debug Help





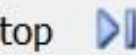
















2986.00

Library

 Model
 Time
 Cycletime

3,178

RCMAX_20_15_1 FINAL MAKESPAN
NUM_ITERATION ONLY COUNTS **400** TIMES
UB 2749

2,966

RCMAX_20_15_4 FINAL MAKESPAN
NUM_ITERATION ONLY COUNTS **400** TIMES
UB 2563

GA+BO 自動解題

統一格式並分析輸出

(符合Flexsim)

Flexsim結果驗證

我們遇到的困難！



運算量超大

BO+GA

多次改進算法再跑結果
一週的時間實在太趕
且一一保存結果導致更沒效率



自動化的程度

為了減輕組員負擔
開始了多次自動化功能新增
以及改進自動化功能
的不歸路



Flexsim嫁接

為了Flexsim程式端
Encoding的方式
以及設計如何在Flexsim
模擬JSSP來驗證結果

問題設定

將右表兩兩拆開

- **mOrder**
 - 加工機台
- **pTime**
 - 加工時間

1	20	15																
2	8	98	13	149	6	36	10	169	4	136	7	41	14	19	11	199	3	181
3	5	194	3	172	1	74	4	85	0	94	11	86	7	191	6	185	9	62
4	2	70	3	33	5	114	6	179	9	131	10	54	4	128	12	28	11	106
5	11	34	10	101	13	108	5	182	4	129	2	89	12	84	6	68	8	34
6	10	197	1	176	11	39	6	192	12	90	5	30	4	4	0	51	2	152
7	5	84	8	22	13	20	11	51	4	145	14	86	3	159	7	49	6	193
8	7	42	5	168	10	70	8	86	2	41	4	56	12	194	14	107	11	160
9	10	80	13	76	11	16	8	192	4	100	6	83	1	186	7	55	0	122
10	0	74	9	99	14	45	11	91	1	153	12	199	3	35	13	121	4	128
11	4	1	0	11	3	158	12	135	7	196	8	107	13	76	9	6	1	57
12	6	90	3	189	0	3	14	128	5	123	2	178	12	116	1	146	11	94
13	13	82	4	61	1	52	9	141	11	65	10	94	2	48	8	128	12	146
14	0	146	7	29	11	5	10	169	3	192	12	56	13	102	14	121	2	116
15	13	130	2	33	10	200	7	129	12	103	1	173	5	73	14	119	6	46
16	3	10	4	199	2	149	13	70	8	76	0	155	5	2	12	79	11	102
17	11	36	0	52	1	120	12	88	5	81	4	1	7	3	6	11	13	88
18	11	180	13	77	5	94	10	21	12	134	0	147	3	167	7	187	14	53
19	7	185	9	102	2	127	10	110	12	198	5	66	4	183	13	101	1	25
20	5	33	14	80	10	103	8	138	13	141	11	4	12	127	4	191	0	180
21	1	41	13	168	4	174	14	1	7	130	6	124	8	28	0	82	10	127

GA 編碼與解碼

	mOrder	pTime
job1	1 2 3	3 2 2
job2	1 3 2	2 1 4
job3	2 3 1	4 3 5
job4	3 1 2	3 2 1

111222333444



134434122231

將[機台,工作]的對應
順序進行隨機洗牌

最佳加工程序

GA 解碼

■ job1 ■ job2
■ job3 ■ job4

134434122231

	Order	pTime
job1	1 2 3	3 2 2
job2	1 3 2	2 1 4
job3	2 3 1	4 3 5
job4	3 1 2	3 2 1

M1

1

M2

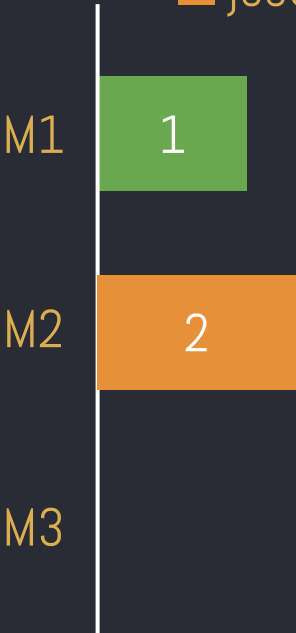
M3

GA 解碼

134434122231

job1 job2
job3 job4

	Order	pTime
job1	1 2 3	3 2 2
job2	1 3 2	2 1 4
job3	2 3 1	4 3 5
job4	3 1 2	3 2 1

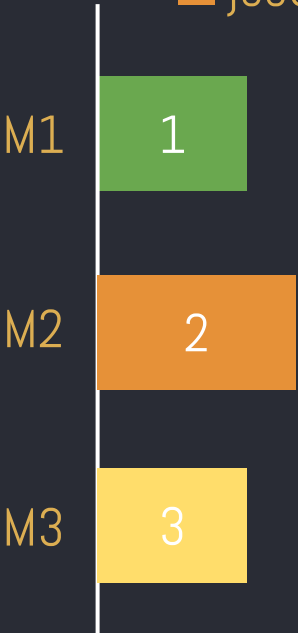


GA 解碼

134434122231

job1 job2
job3 job4

	Order	pTime
job1	1 2 3	3 2 2
job2	1 3 2	2 1 4
job3	2 3 1	4 3 5
job4	3 1 2	3 2 1

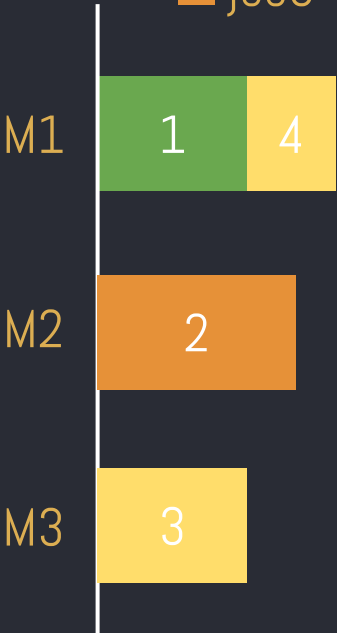


GA 解碼

134434122231

	Order	pTime
job1	1 2 3	3 2 2
job2	1 3 2	2 1 4
job3	2 3 1	4 3 5
job4	3 1 2	3 2 1

job1 job2
job3 job4

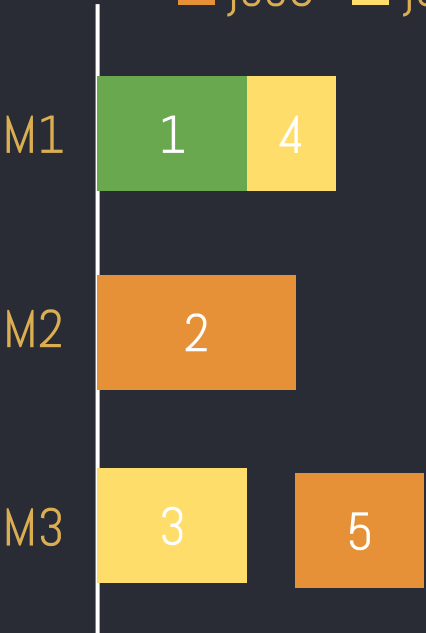


GA 解碼

134434122231

	Order	pTime
job1	1 2 3	3 2 2
job2	1 3 2	2 1 4
job3	2 3 1	4 3 5
job4	3 1 2	3 2 1

job1 job2
job3 job4

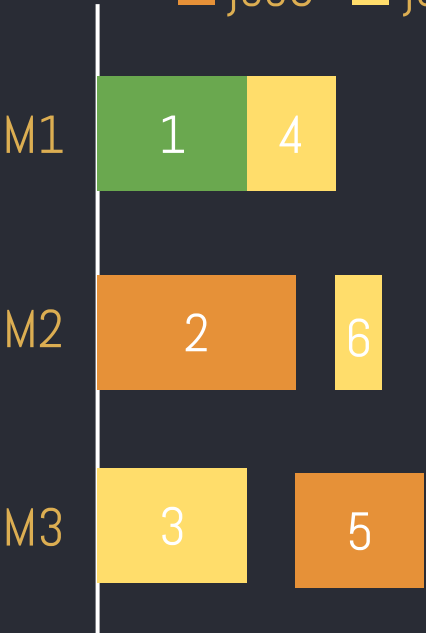


GA 解碼

134434122231

	Order	pTime
job1	1 2 3	3 2 2
job2	1 3 2	2 1 4
job3	2 3 1	4 3 5
job4	3 1 2	3 2 1

job1 job2
job3 job4

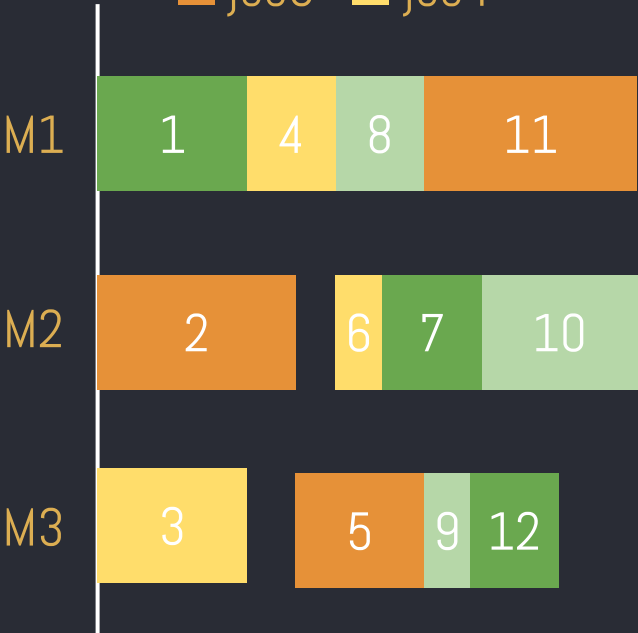


GA 解碼

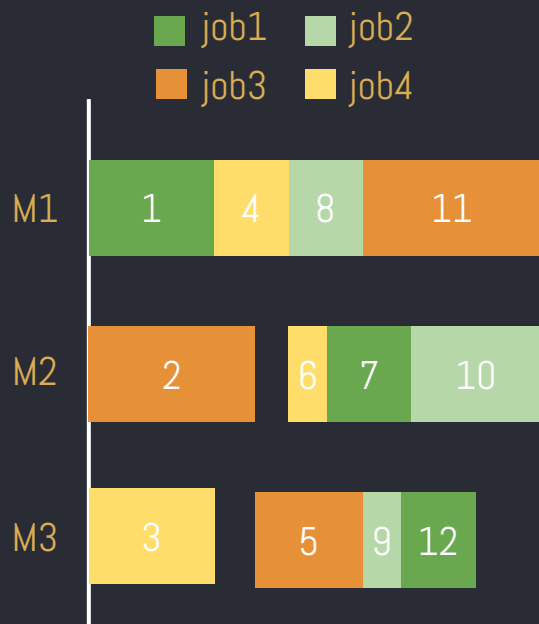
134434122231

	Order	pTime
job1	1 2 3	3 2 2
job2	1 3 2	2 1 4
job3	2 3 1	4 3 5
job4	3 1 2	3 2 1

job1 job2
job3 job4



GA 解碼



Rank Label

每次 Operation 更新
用於 Buffer 內排序

Simulation 設計

State

	num
Machine1	1
Machine2	1
Machine3	3
Machine4	2
Machine5	4
Machine6	2
Machine7	1
Machine8	1

當前已輸出數量

Sequence

	Seq1	Seq2	Seq3	Seq4	...
Machine1	5	8	4	1	
Machine2	5	6	1	3	
Machine3	3	7	5	4	
Machine4	2	4	3	5	
Machine5	4	3	7	6	
Machine6	7	2	1	8	
Machine7	6	1	2	7	
Machine8	8	6	2	9	

運作於 Machine 之 job 排序

Simulation 設計

State	
	num
Machine1	1
Machine2	1
Machine3	3
Machine4	2
Machine5	4
Machine6	2
Machine7	1
Machine8	1

當前已輸出數量

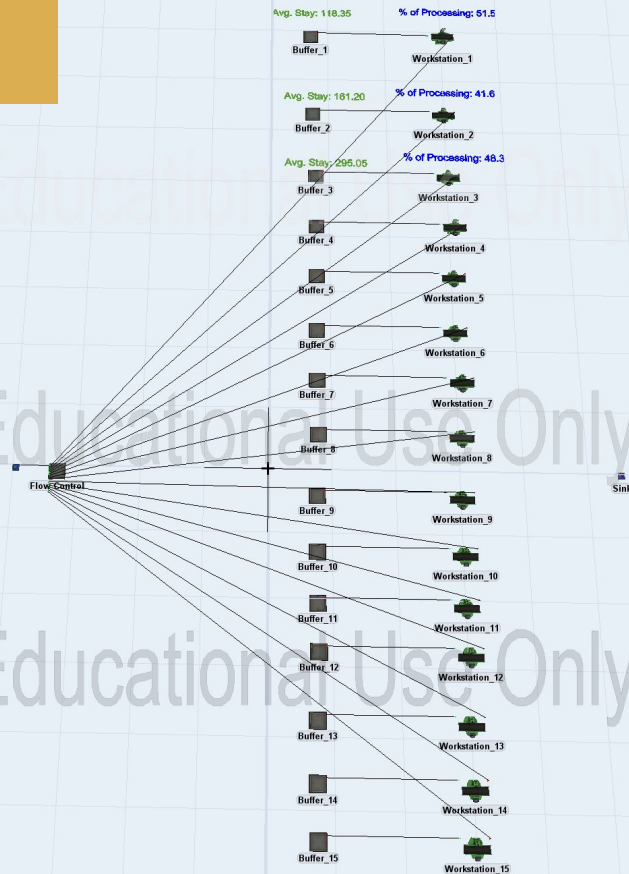
Sequence				
	Seq1	Seq2	Seq3	Seq4
Machine1	5	8	4	1
Machine2	5	6	1	3
Machine3	3	7	5	4
Machine4	2	4	1	5
Machine5	4	3	7	6
Machine6	7	2	3	8
Machine7	6	1	2	7
Machine8	8	6	2	9

運作於 Machine 之 job 排序

檢查

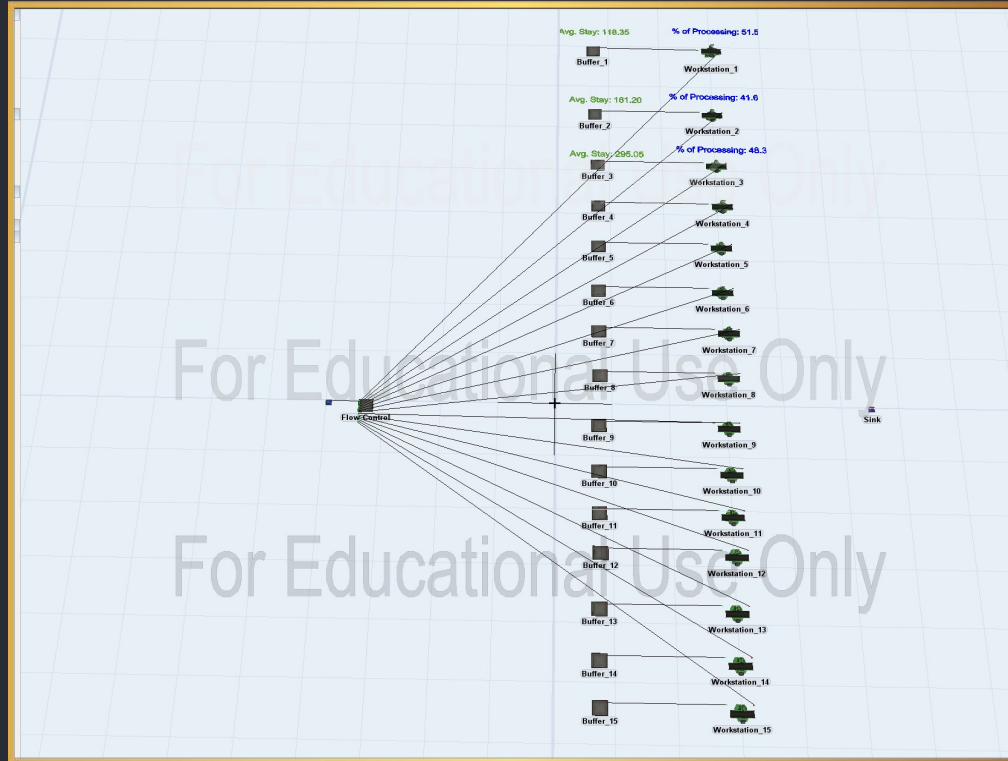
是否為下一執行 Job

Flexsim

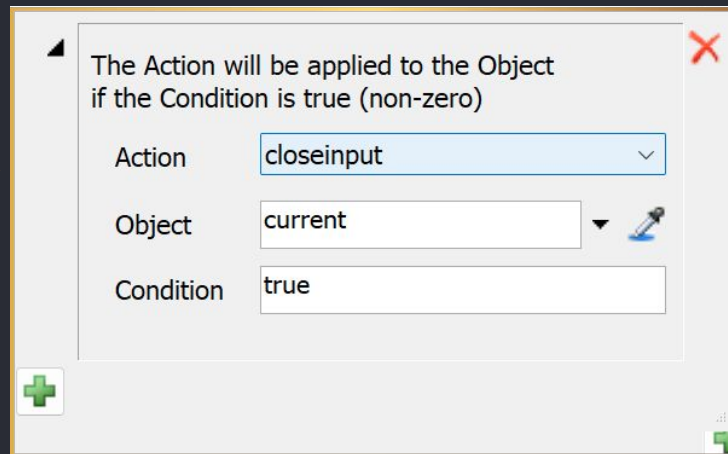
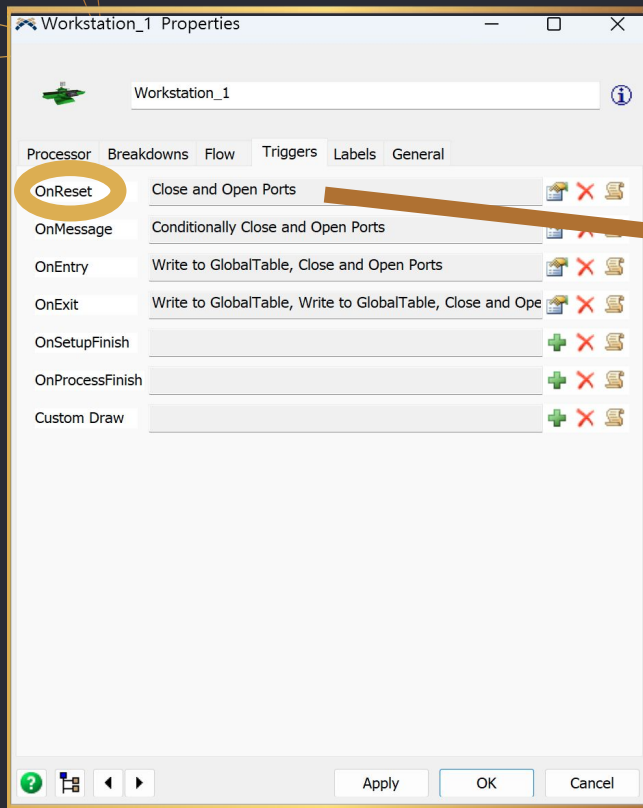


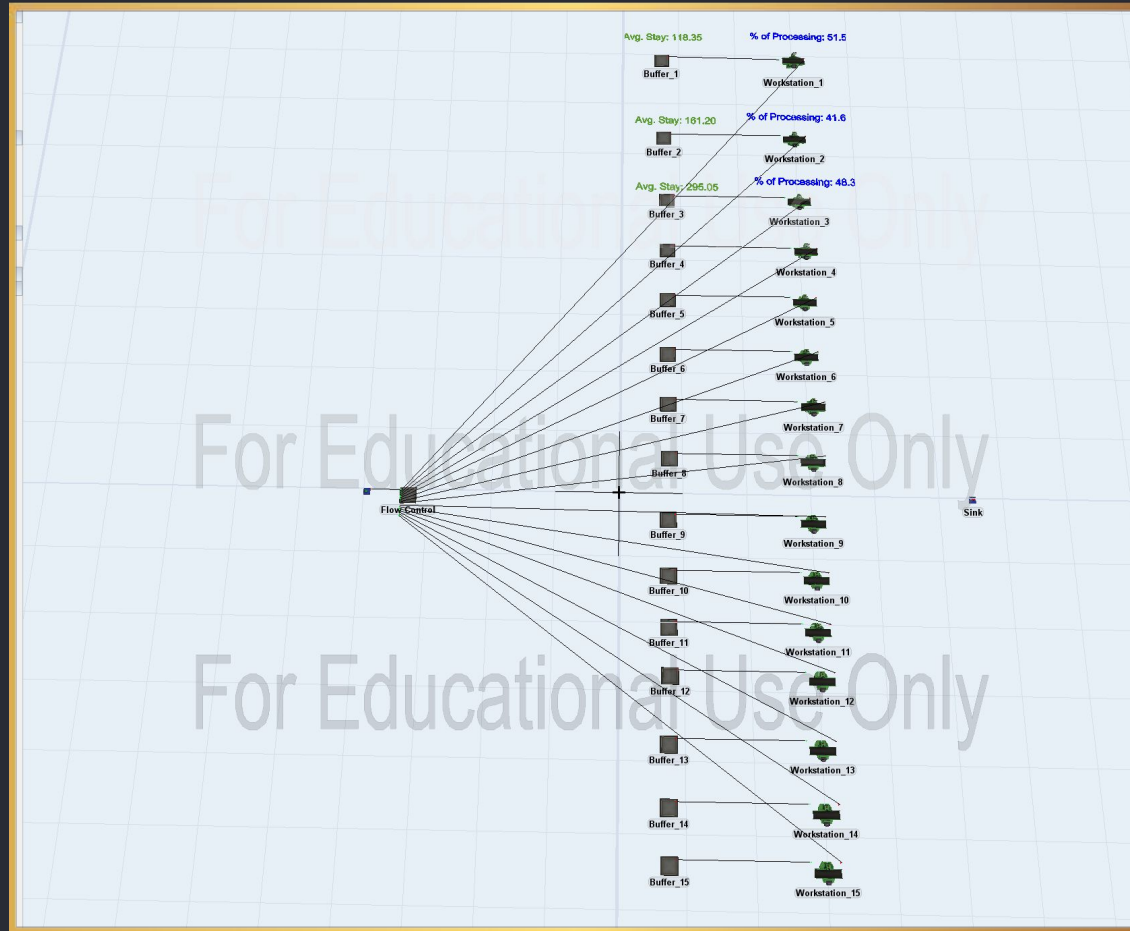
Step 1: 建立基本Model

Step 2: 確認自動化資料輸入成功

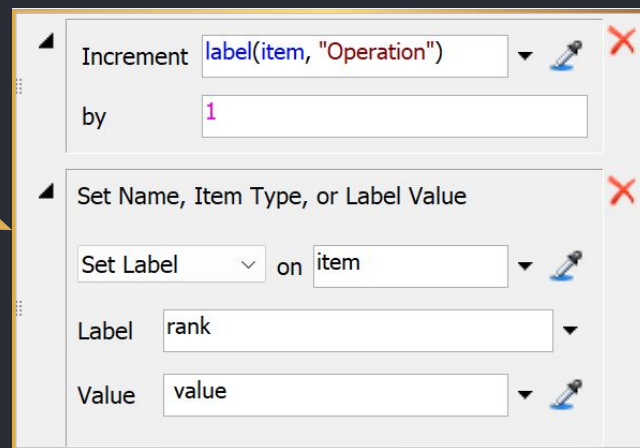
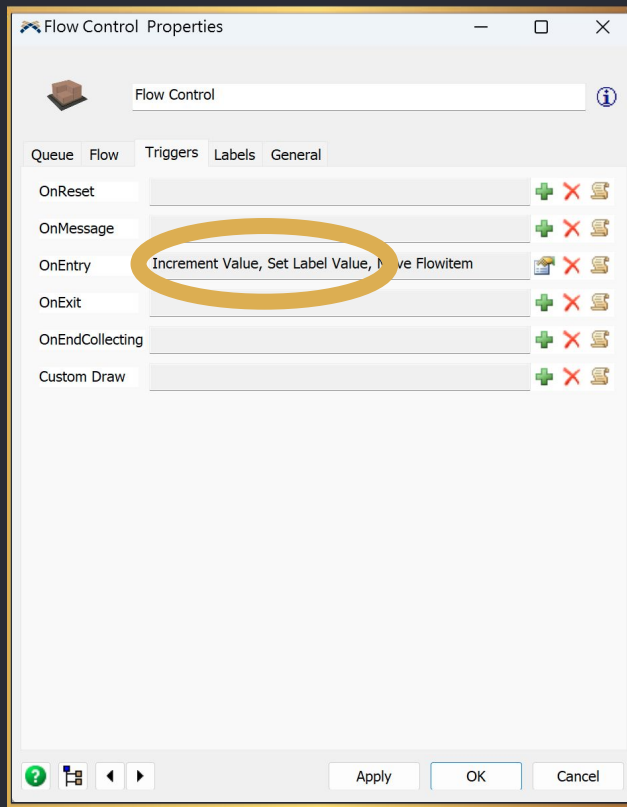


Step 3:關閉所有機台的入口

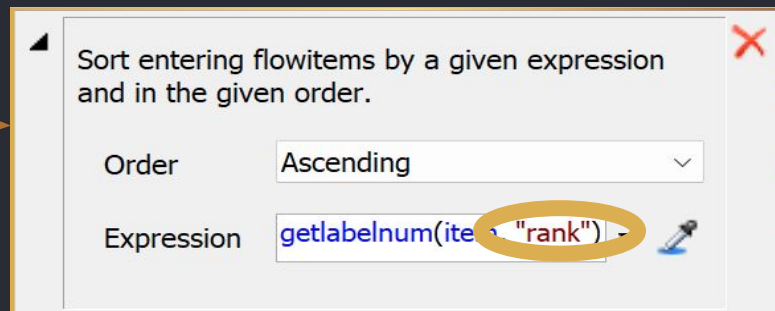
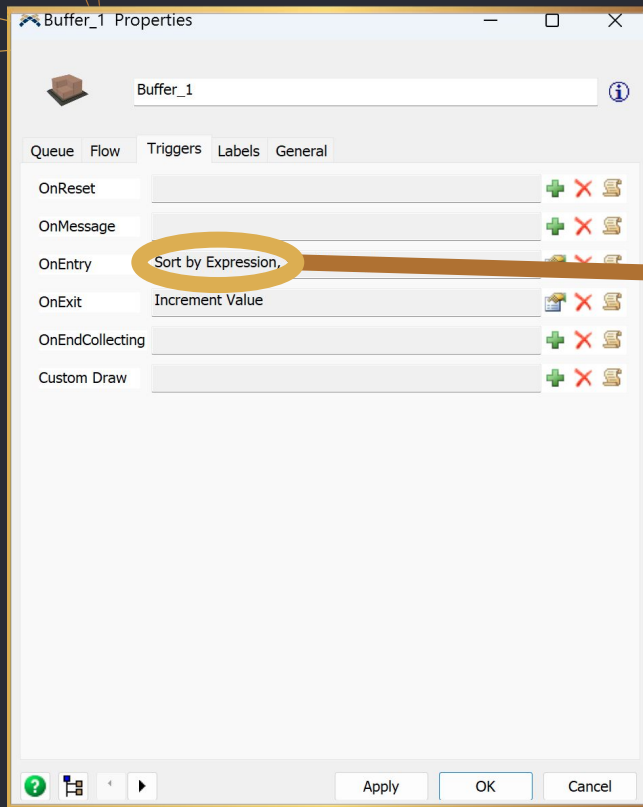




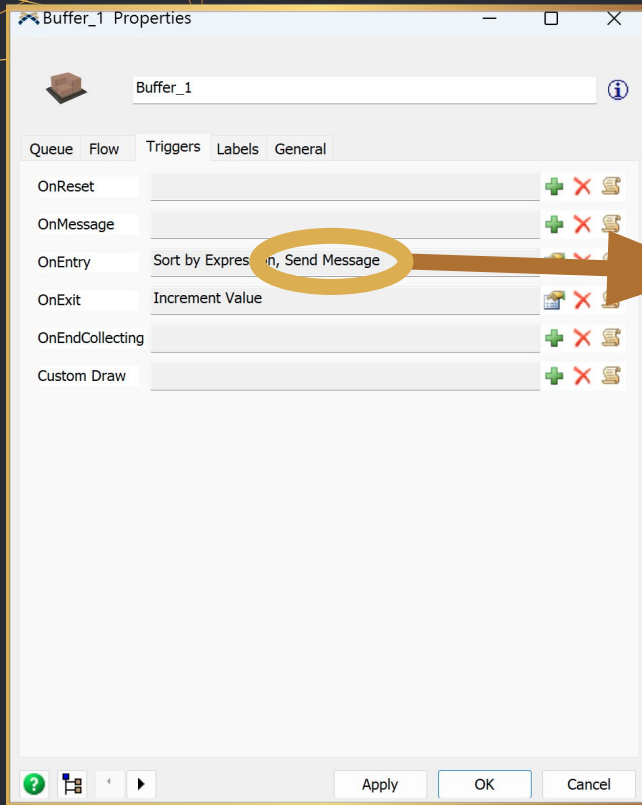
Step 4: Job到Flow Control領取該次的Rank



Step 5: 對進入Buffer的Job進行排序 (決定若開門第一個出去的Job是誰)

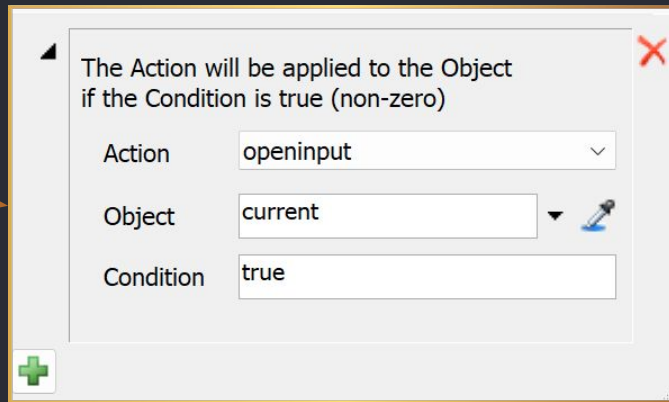
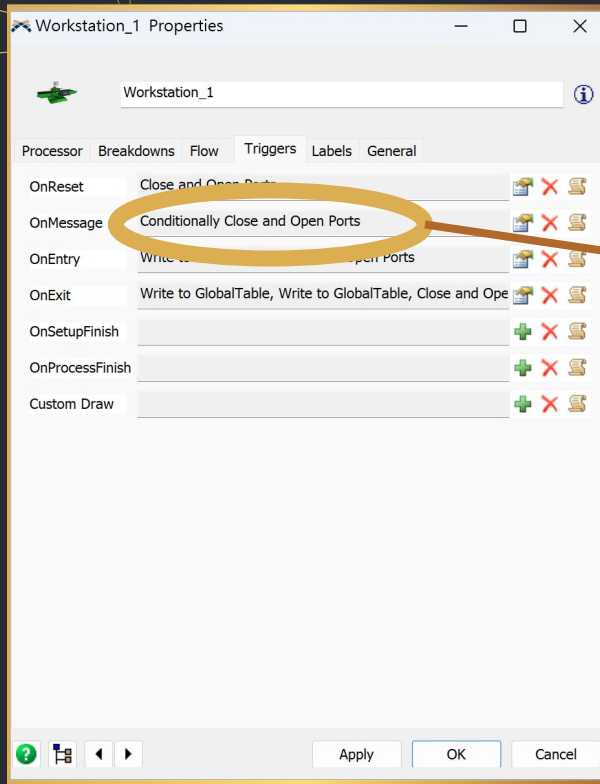


Step 6: 進入Buffer的Job和機台順序第一的Job配對 (決定機台是否開門)

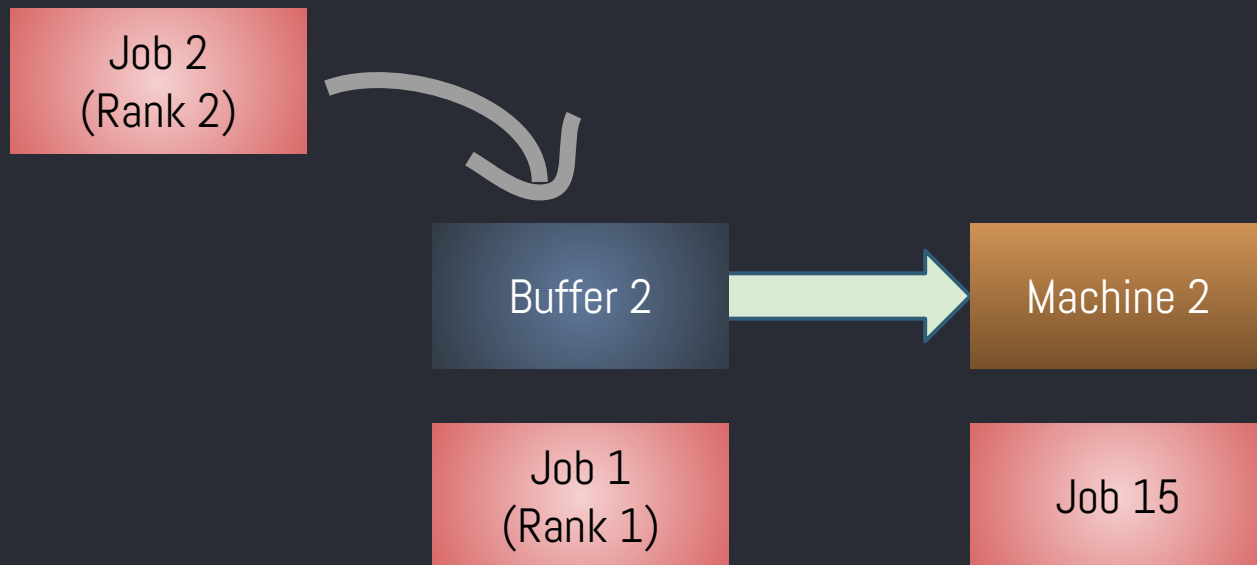


```
{ //***** PickOption Start *****\n/**popUp:SendMessage*/\n/**Send Message*/\nint NoDelay = -1;\ndouble delaytime = /** \nDelay Time:\n**/**tag:delaytime**/**/0/**list:NoDelay~0~10~getlabelnum(current,\n"messagedelay")*/;\ntreenode tobject = /** \nTo: **/**tag:to**/**/node("Workstation_1",\nmodel())**/**;\ntreenode fromobject = /** \nFrom: **/**tag:from**/**/current**/**;\ndouble param1 = /** \nParam1: **/**tag:par1**/**/0**/**;\ndouble param2 = /** \nParam2: **/**tag:par2**/**/0**/**;\ndouble param3 = /** \nParam3: **/**tag:par3**/**/0**/**;\nint condition ;\n/**\n\nDelay Time:\n\nNoDelay: message sent immediately within trigger context\n\n0: delayed message sent in 0 time*/\nif ( getitemtype(item) == gettablenum("Sequence", 1, gettablenum("State", 1, 1)+1\n) ) {\n    if (delaytime == NoDelay)\n        sendmessage(toobject,fromobject,param1,param2,param3);\n    else senddelayedmessage(toobject, max(0,delaytime),\n        fromobject,param1,param2,param3);\n}\n} //***** PickOption End *****\n}
```

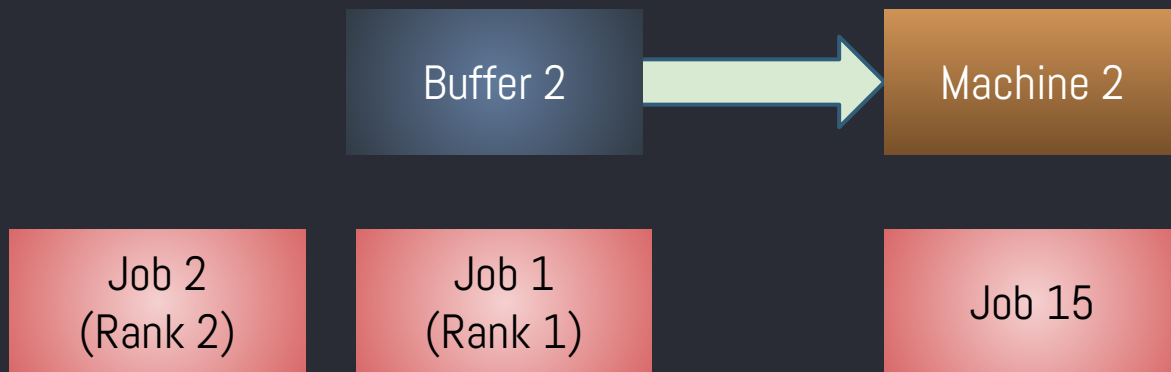
Step 6: 進入Buffer的Job和機台順序第一的Job配對 (決定機台是否開門)



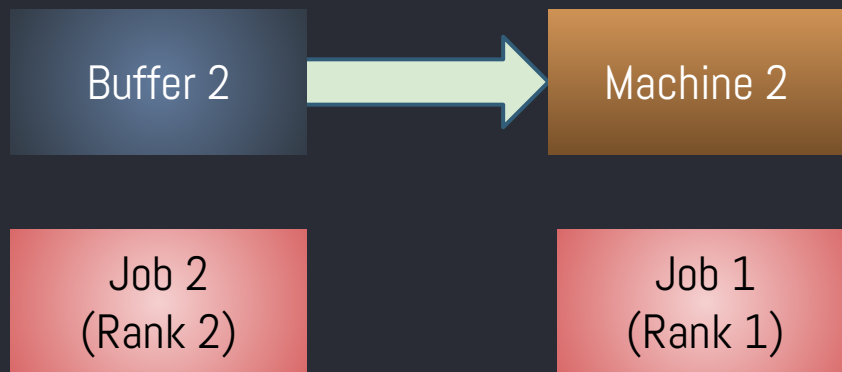
Step 7: 加強Buffer找到內部Job的能力



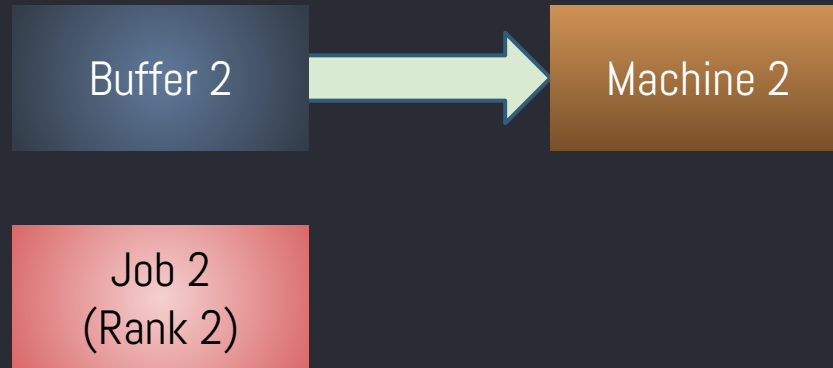
Step 7: 加強Buffer找到內部Job的能力



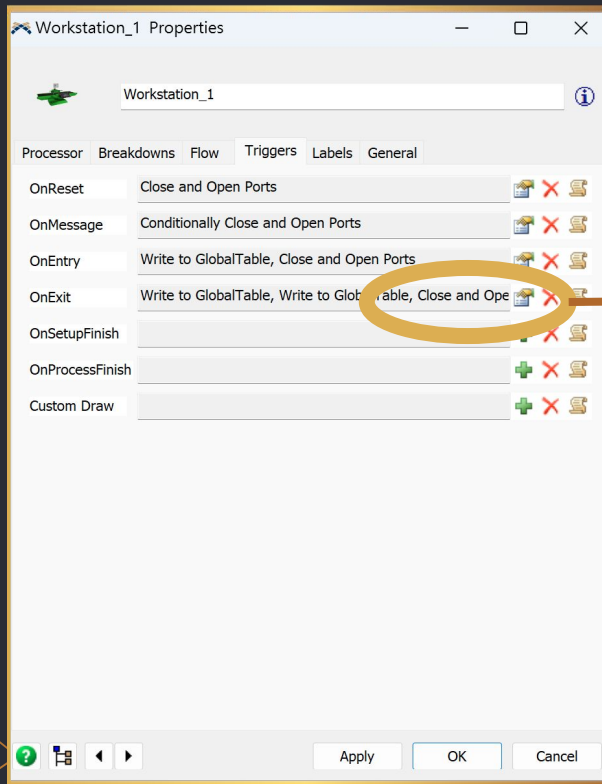
Step 7: 加強Buffer找到內部Job的能力



Step 7: 加強Buffer找到內部Job的能力

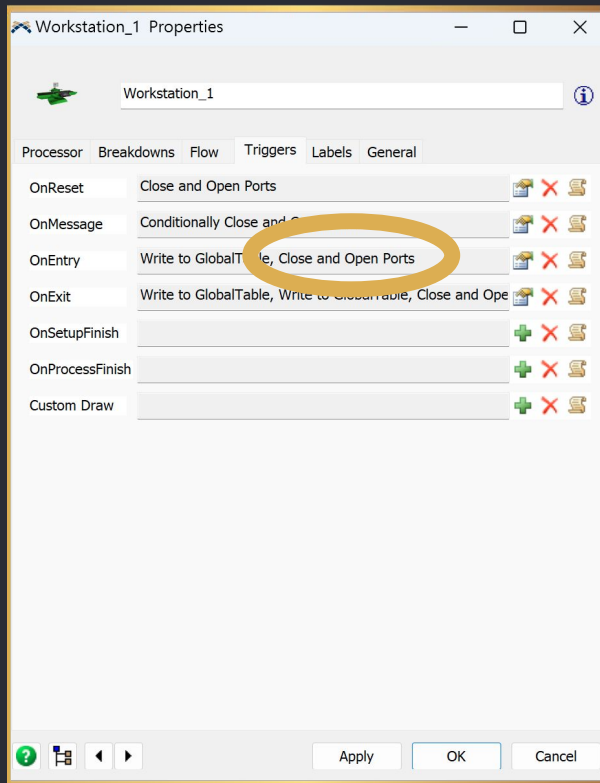


Step 7: 加強Buffer找到內部Job的能力



```
{ //***** PickOption Start *****\\
/**popup:CloseOpenPorts*/
/**Close and Open Ports*/
/** \nThe action will be performed if some condition is true (equal to 1).*/
treenode involved = /** \nObject: */ /**tag:object*//**current*/;
int condition = /** \nCondition: */ /**tag:condition*//**true*/;
treenode Buffer = node("Buffer_1", model() );
treenode FirstItem = first(Buffer);
if (getitemtype(FirstItem) == gettablenum("Sequence", 1, gettablenum("State", 1, 1)+1
) ) {
    /** \nAction:
    /**tag:action*//**openinput*/list:closeinput~openinput~stopinput~resumeinput~closeoutput~openoutput~stopoutput~resumeoutput*/
    (involved);
}
/**\n\nWarning: It is better to send a delayed message to open or resume the ports of
the current object!*/
} //***** PickOption End *****\\
```

Step 8: Job進入機台後機台入口關閉 &再開啟條件



Step 9: 紀錄Job進入、離開機台的時間



STEP 1
基本設置：
建立整體 Model

STEP 3
關閉所有機台的入口

STEP 5
Job進入Buffer時，
利用已知的Rank進行排序

STEP 2
確認自動化的相關
資料皆已成功匯入

STEP 4
Job回到Flow Control
並改變其Rank

STEP 6
進入Buffer的Job和
機台當時順序比對

STEP 8
決定機台
再度打開的時機

STEP 10
重複動作，
直到所有作業完成

STEP 7
加強Buffer找Job條件

STEP 9
紀錄每個Job進入、
離開機台的時間

分工

Python

Flexsim

許恒睿

基因演算法實作、貝氏優化
實作、自動化格式設計

孫于涵

基因演算法與flexsim間
的连接設計、優化程式碼

陳博翰

flexsim功能設計與優化、偵
錯、驗證GA結果

THANKS!

