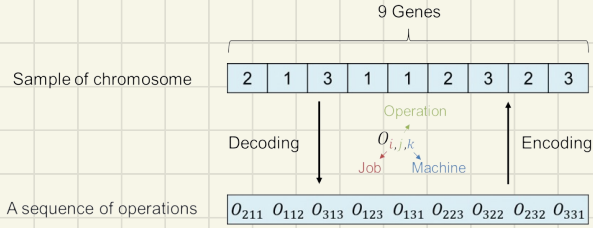
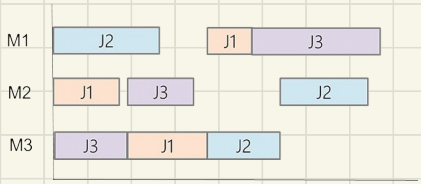


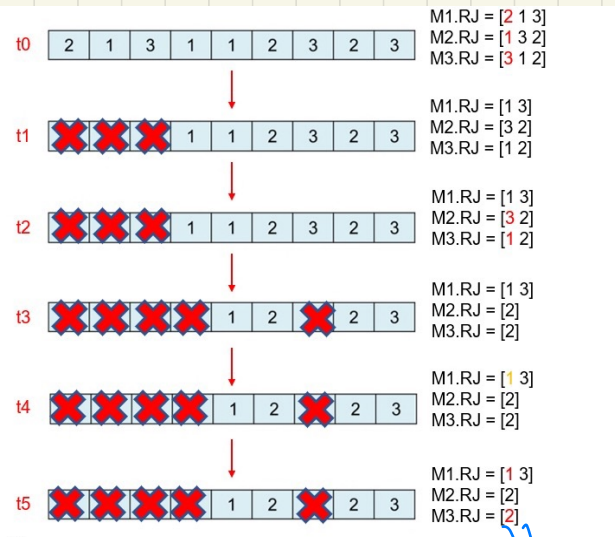


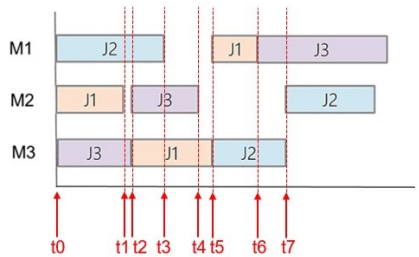
- Processing time

Operation	O1	O2	O3
Job 1	8	10	5
Job 2	15	9	12
Job 3	9	7	19

- Machine sequence

Operation	O1	O2	O3
Job 1	M2	M3	M1
Job 2	M1	M3	M2
Job 3	M3	M2	M1





Machine.ReservedJobs = […]
M1.RJ = [2 1 3]
M2.RJ = [1 3 2]
M3.RJ = [3 1 2]

Operation	O1	O2	O3
Job 1	M2 1	M3 6	M1 0
Job 2	M1 0	M3 6	M2 1
Job 3	M3 6	M2 1	M1 0

Mission List.

missionID	operation	machine	state
2	1	M1	Done.
1	1	M2	processing.
3	1	M3	processing
1	2	M3	waiting.
1	3	M1	waiting.

Task

最佳化排程 sequence.

⇓

任務，一個 operation 為單位，列表.

mission ID	operation	machine	operation state
2	1	M1	Done.
1	1	M2	processing.
3	1	M3	processing.
1	2	M3	waiting.

⇓

* 先進先出
* 最短時間
* 任務優先度.

(考慮車輛數).

若同一時間上需處理最大任務數 < 車輛數則不影響

若 > AGV 數則需先討論 AGV 運輸時間與機台加工時間是否成比例

派車優先權

* 電量
* 線動率
* 距離（距離目的地）

10/4.

* Good sequence，填進 mission List.

1. mission List 依照 FIFO. 像 decode 順序執行.

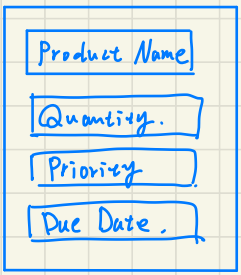
K, 探距離像光 孔平 _

工單生成

- 產品製程存DB. 需機台順序及加工時間.

Product Name	01	02	03
Product 1	M1/4	M2/8	M/6
Product 2			
Product 3			

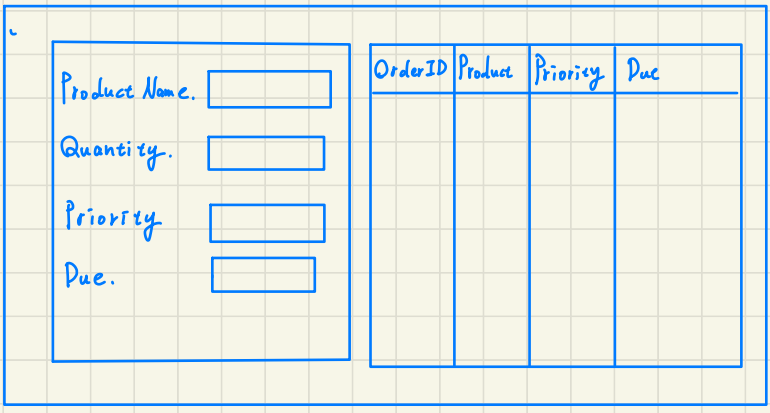
- 下單介面.



⇒ 下拉選單, 選擇.
⇒ 手動 key 入.
⇒ 下拉 1~5
⇒ 手動 key 入.

- 工單生程.

orderID	Product Name	Quantity	Priority	Due Date



最佳化演算法.

輸入. ⇒ 對應工單之加工時間及機台順序表格.

加工時間

Order ID	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10
1	29	78	9	36	49	11	62	56	44	21
2	43	90	75	11	69	28	46	46	72	30
3	91	85	39	74	90	10	12	89	45	33
4	81	95	71	99	9	52	85	98	22	43
5	14	6	22	61	26	69	21	49	72	53
6	84	2	52	95	48	72	47	65	6	25
7	46	37	61	13	32	21	32	89	30	55
8	31	86	46	74	32	88	19	48	36	79
9	76	69	76	51	85	11	40	89	26	74
10	85	13	61	7	64	76	47	52	90	45

對應機台.

Order ID	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10
1	1	2	3	4	5	6	7	8	9	10
2	1	3	5	10	4	2	7	6	8	9
3	2	1	4	3	9	6	8	7	10	5
4	2	3	1	5	7	9	8	4	10	6
5	3	1	2	6	4	5	9	8	10	7
6	3	2	6	4	9	10	1	7	5	8
7	2	1	4	3	7	6	10	9	8	5
8	3	1	2	6	5	7	9	10	8	4
9	1	2	4	6	3	10	7	8	5	9
10	2	1	3	7	9	10	6	4	5	8

輸出⇒. 最加化加工序列(optimal sequence).

[2, 1, 3, 1, 1, 2, 3, 2, 3]

SFC system.

輸入 ⇒ optimal sequence.

[2 1 3 | 1 2 3 2 3] ⇒ mission List、⇒ machine.ReservationJobs

mission List.

Mission ID	Order ID	Operation	Machine	State.
1	2	1	1	
2	1	1	2	
3	3	1	3	
4	1	2	3	
5	1	3	1	
6	2	2	3	
7	3	2	2	
8	2	3	2	
9	3	3	1	

* mission state: { waiting, processing, process done, finished.

* machine state: { idle, loading, processing, unloading, processdone,

* machine. ReservationJob.
  { M1.RJ = [2 1 3]
  M2.RJ = [1 3 2]
  M3.RJ = [3 1 2]

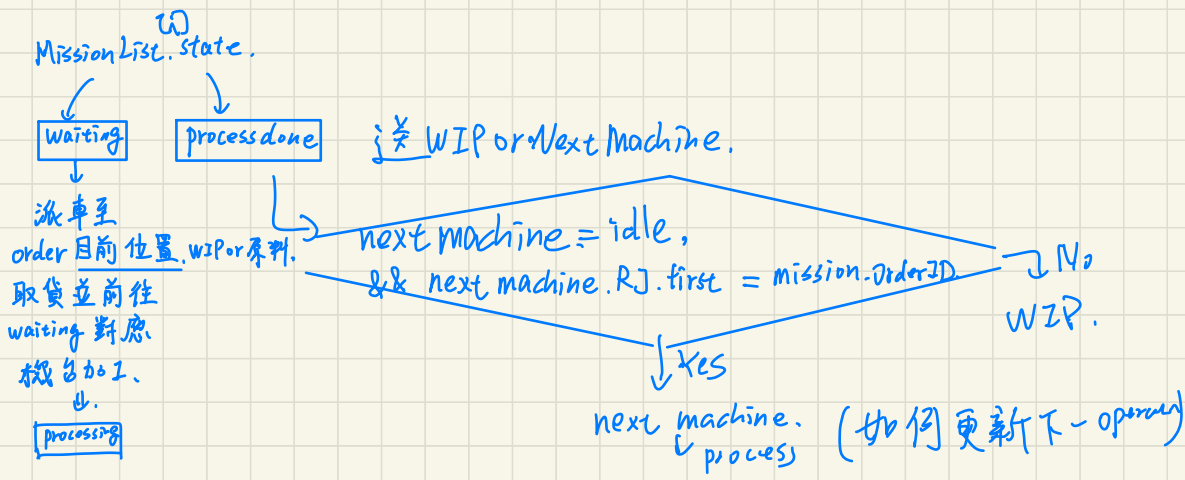
* AGV State: { idle, moving, loading, unloading, disconnected, error,

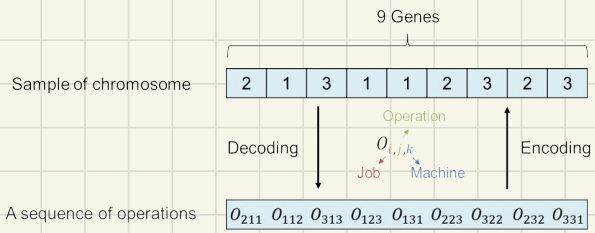
* order
  { position.
  default. = 原料台.

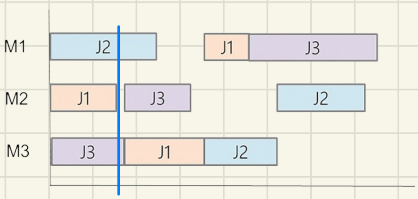
任務指派方法.

① 遵守 FIFO，但需依照機台預約順序.

```
while (true).
{ for (i=1, i≤missionList.count, i++)
  { if( missionList[i].state == mission.waiting &&
    missionList[i].machine.ReservationList.First().ID == missionList[i].orderID
    && mission[i].machine.state == machine.idle).
    { var AGV = FindFitnessAGV ( missionList[i].machine.referNode)
      if (AGV != null)
      { Send AGV to (
```







- Processing time

Operation	O1	O2	O3
Job 1	8	10	5
Job 2	15	9	12
Job 3	9	7	19

- Machine sequence

Operation	O1	O2	O3
Job 1	M2	M3	M1
Job 2	M1	M3	M2
Job 3	M3	M2	M1

Mission ID	Order ID	Operation	Machine	State.
1	2	1	1	processing
2	1	1	2	prodone
3	3	1	3	proing.
4	1	2	3	Wait
5	1	3	1	;
6	2	2	3	;
7	3	2	2	;
8	2	3	2	;
9	3	3	1	;

```
while (true)
{  foreach (MissionList)
   { if ( Mission.state = waiting & Mission.Machine.state = idle
       & Mission.orderID = Mission.Machine.RJ.first.ID )
     {取得(Mission.order.position), 派車取貨至 Mission.Machine
      進行加工. (5項狀態改變), Mission.做更新.
     }
     if ( Mission[i].state = processdone )
     { foreach ( MissionList[j] ). Mission 完成時, 執行用 order 下一 Missio
       { if ( Mission[i].orderID = Mission[j].orderID &
              Mission[j].Operation - Mission[i] operation).
         { var nextmission = Mission[j].
           if ( nextmission.machine.state = idle &
                nextmission.machine.RJ.firstID = nextmission.orderID)
           { 派車至 MissionList[i].order.possition 取貨,
             送至 nextmission.machine 加工. 5項狀態更新.
           }
           else { 派車至 MissionList[i].order.position 取貨,
                  送至 WIP } 5項狀態更新.
       } else {continue;}
     } else {break}
   else { continue}
   }
}.
```