1. 我做到了 3 顆星，加 Dashboard 跟 GUI

2. how to build my model

首先建立一個 Globaltable，叫 demand_location 跟 server_location 來計錄村落地址、醫院地址、跟各個 lambda。

各拉一個 Source、Queue、Processor、Sink、Operator。

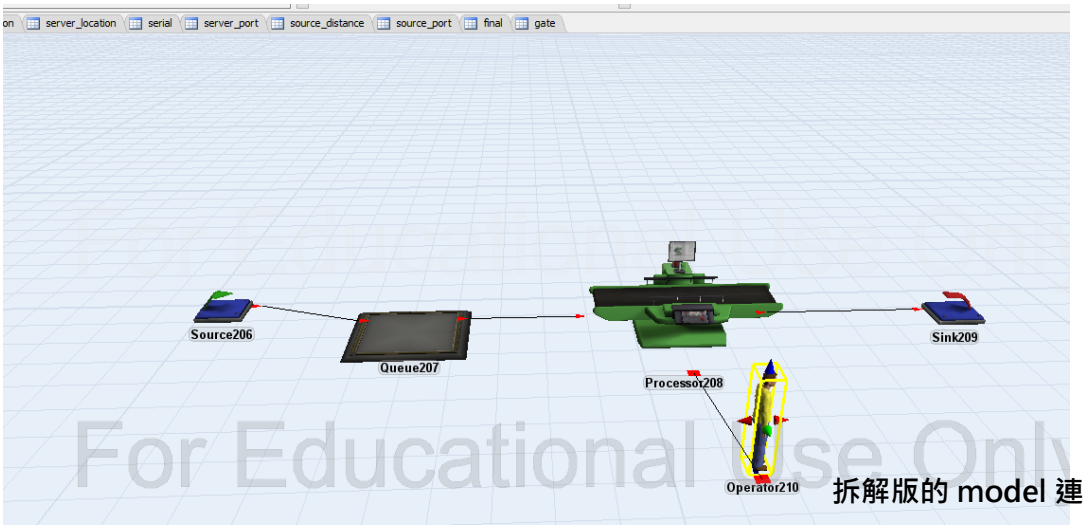
在 Source 內的 triger 寫一個 script 使每個由 Source(NUM)出來的物件都叫做(NUM)，此用意是未來方便做判斷。Source 的 expontential 輸入連線時再弄。
再來是要用最短路徑的設定，把 Source 的 flow 按下去，並打開 send to port 的 script 打開在裡面打關於最短路徑的 code 在此不多做說明(主要是給最近的 available port，如果都沒有 available port 就給最近的 port)。

Queue 的聯動頗多先在 triggers 的 Onentrys 內打 script 設定關於" serial " 這個 globaltable 的值，使 serial((各個 Source) , 1)代表需求 item 的數量，serial((各個 Source) , 2) 代表此 Source 被解決的 item 數量，serial((各個 Queue) , 3)代表此 Source 失敗的次數。並把全部 MAP 的輸出總數、失敗總數寫進 final 這個 globaltable 的 column1、column2，column3 代表失敗率。

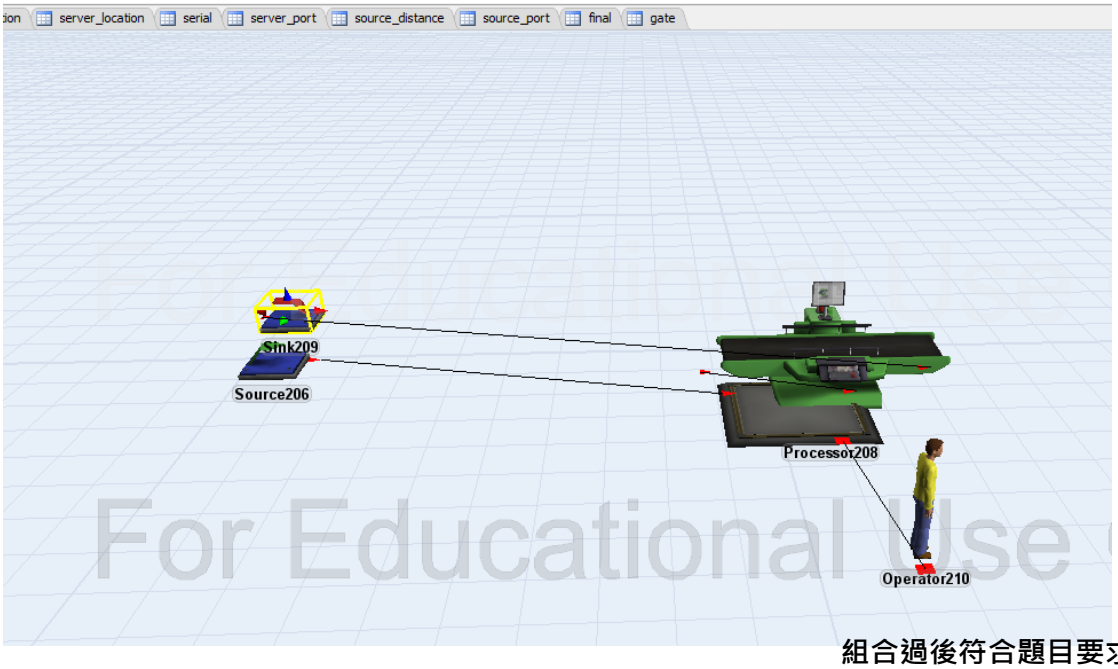
Processor 要先在 process time 的地方輸入根據物件名稱對應 Globaltable 的 expontential lambda，把 use operator for process 勾起來(之後解釋)。然後把 flow 打開按 send to 的 script，在裡面寫這個物件要送到哪一個 SINK。把 Processor 的 Use Transport 打開使 Operator 可以運送物件算路程時間，也要把 Use operator for processes 打開使加工不會趁 Operator 外出送貨時偷跑。

Operator 要設定的比較少，只需在 Trigers 的 unload 寫 script，叫 Operator 送完貨要記得回醫院。

Sink 沒什麼太特別的設定



拆解版的 model 連線大致長這樣



組合過後符合題目要求的排列

此是為了方便看，實際上 Source 跟 Sink 會貼合再一起，processor 跟 Queue 也是

接下來重頭戲是寫一個用來創造各個物件的 Script 叫 demand_server_map，內容主要是把物件創造複製，並且設定地點。其中，創造 Source 的同時把各個需求點的 expontential 輸入。在創造 Operator 時也把各個 operator 設成 reset position。
我的 Model 主要有 80 個 Sink 跟 Source， 13 個 Processor、Operator、Queue，要把 Source 當成某個村落，Queue 當成醫院的等候室，Processor 當成急救室，最後再由 Operator 把 item 由 Processor 送回位在 Source 同地點的 Sink。

建立完物件，我建了一個叫 connection 的 script，主要把所有物件連結起來。首先是直接把同地點的 Queue 與 Processor 用" A"連線(因為一個等候室只屬於一間醫院)，再用" S " 把 Operator 與 Processor 連線 。
根據距離判斷是否<=250 判斷，(Source , Queue)，(Processor , Sink)是否要連線。
然後段同時也順便把 80 個村莊到 13 間醫院的距離表建到 source_distance 的 globaltable(再用 shortest server 的時候會用到，主要是給 source_port 做一個判斷依據)，然後再根據 source_distance 建一個 globaltable 叫 source_port 用來給 Source 判斷最近的 port。

我額外又再建立一個 script 叫 disconnect 是用來方便解連結除錯的，其 code 跟 connect 很像只是做反向操作。

我又有加寫一個 Sript 叫 destroy_all_object 是用來清除場上的地圖用的。

最後就是 Optimier 找最佳解的地方，先創一個(13 X 1) 的 table 叫 gate，並且把此 13 個變數輸入到 Experiment control 的 scenario 內。Performance measure 以 globaltable final(1 , 2)總失敗次數為量測值。到 Optimizer design 內把 13 個變數設為 Binary，並且在旁邊限制式設定 13 個變數相加等於 7(代表一次挑 7 個 server)，最後把 Run Time 設成 10000 跟 replicate 設成 5 就可以跑結果了。
我是把 MAX Solution 設成 400，取到的結果後面第四段會提到~~~

3. Instructions to use your model

首先把 demand_server_map 打開並 run，再打開 connection 按 run 此時基本上已經大功告成，剩下要做的就是自己去按 opermizer run 求出可行解。

現在檔案內已經將地圖建起來並已成功連線，但如果助教您要把 excel 的測資改掉的話，就要先把 destroy_all_object 這個 script 執行(清除場上物件)，再來做前面提到的步驟。

至於績效的部分，把 statistics 的 Experimenter 打開我基本上已經設定完了，剩下就是設定 Run Time 跟 Max solution ，我自己 Run Time 設 10000，Max solution 設 400，所以系統取到 400 組解時就會停下來了。(結果在後面第四段呈現)

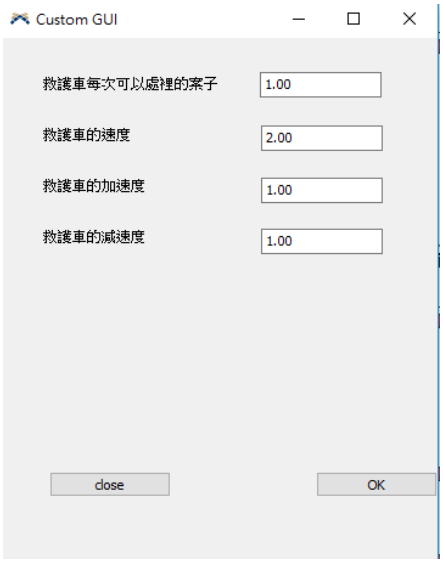
EXTRA POINT

Dash board:

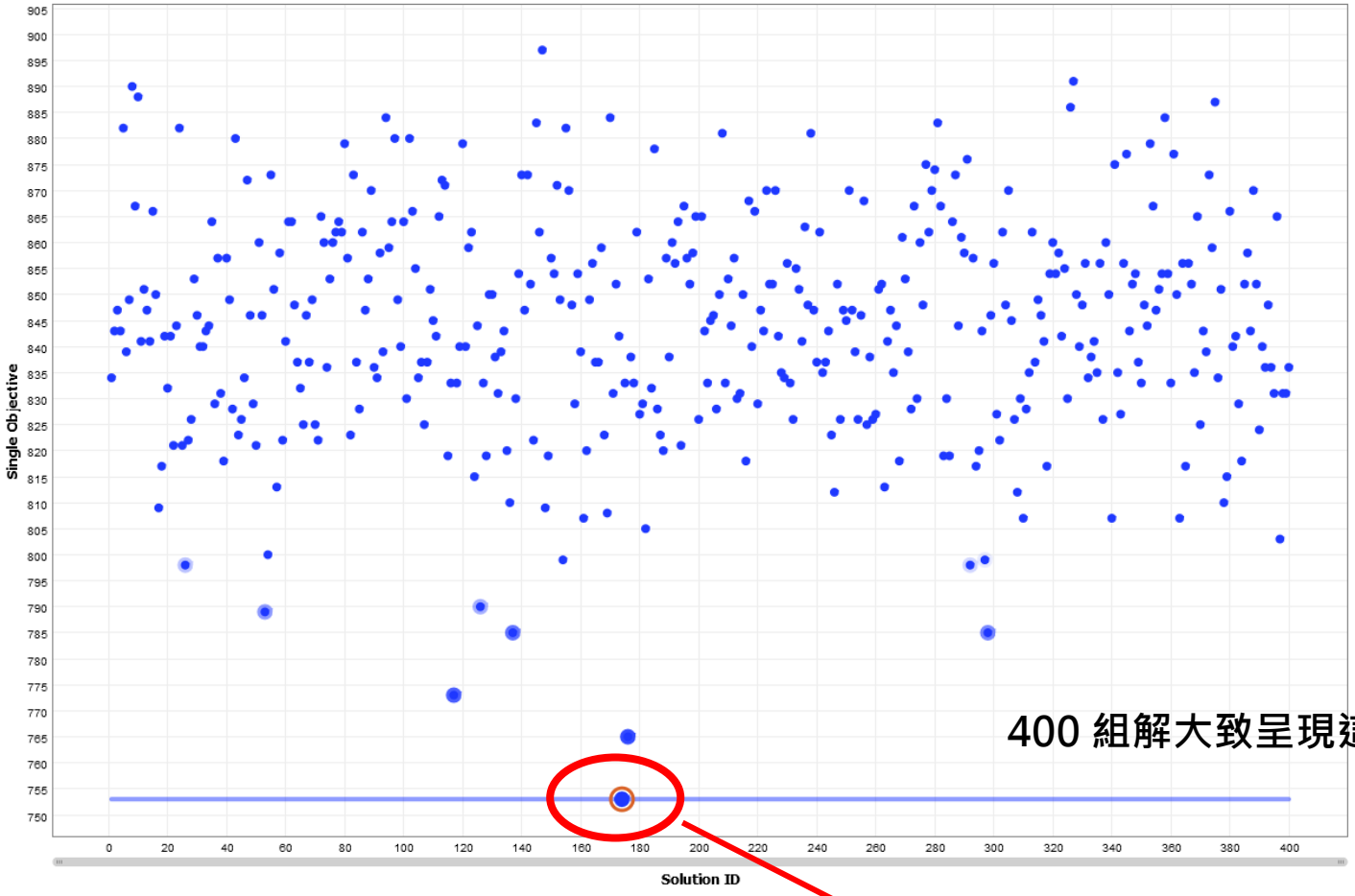
案 statistics 內的 Dashboard 我有建一個叫 Dashboard 的表，內有
Processor status(醫院的使用狀態)、
current waiting patient(現在該醫院正在等候的病患數)、
Source Output Per Hour(每個村庄的輸出數率)、
which ambulance is working(救護車是否忙線)、
total finish by hospital(該醫院看完診的病患)

GUI:

這個我不太確定裡面要放甚麼參數
先在 View 打開 model control GUI 裡面有一個我設定好的使用者介面，可以設定救護車的速度、加速度、減速度、每次搬運輛，(初始值已經根據題目要求設定好了)。



4. If you did optimization or any other data analysis of running result, put on your insights.



400 組解大致呈現這樣

最佳 scenerio174

這組解挑選了醫院 1、5、8、9、10、12、13(7 間醫院)

	Variable	Scenario 1	Solution 17
Variable 1	MODEL:/Tools/GlobalTables/gat		1.00
Variable 2	MODEL:/Tools/GlobalTables/gat		0.00
Variable 3	MODEL:/Tools/GlobalTables/gat		0.00
Variable 4	MODEL:/Tools/GlobalTables/gat		0.00
Variable 5	MODEL:/Tools/GlobalTables/gat		1.00
Variable 6	MODEL:/Tools/GlobalTables/gat		0.00
Variable 7	MODEL:/Tools/GlobalTables/gat		0.00
Variable 8	MODEL:/Tools/GlobalTables/gat		1.00
Variable 9	MODEL:/Tools/GlobalTables/gat		1.00
Variable 10	MODEL:/Tools/GlobalTables/gat		1.00
Variable 11	MODEL:/Tools/GlobalTables/gat		0.00
Variable 12	MODEL:/Tools/GlobalTables/gat		1.00
Variable 13	MODEL:/Tools/GlobalTables/gat		1.00

總失敗次數為 769 次

這組解挑選了醫院 1、5、8、9、10、12、13(7 間醫院)

但是最佳解 scenerio174 有個缺點，之所以總失敗次數這麼低是因為犧牲了 Queue1(醫院一)的能力負荷(這也跟我的操作策略有關)

我的操作策略: 如果某村莊發出需求針對距離該村狀最近的醫院排序給此病患派送，但，如果此時每間醫院都忙線，就把這個病患放在最近的醫院給他排隊(如此一來犧牲這一類病患，就不會因為要給這些病患長途跋涉到其他很遠地區的醫院，而延誤了其他地區的救護需求)，如上圖可知醫院一附近的需求超多，其他地區雖然排隊的人很多，但相對醫院一真的是相形失色。

我對此次參數的見解，我發現 80 個村落當中每個村落的 lambda 都頗大(意味著單位時間內需求大)，一開始 7 間醫院幾乎都可以正常服務病患，但一有排隊的情況發生後面幾乎都會屬於 fail 的情況發生了，所以越到後面 fail rate 就會一直往上飆升，這次我做的成果 fail rate 為 0.88 相對於其他逼近 0.99 算非常好的結果了，也因此代表著這 80 個村落只有 7 間醫院 7 台救護車，是很吃力的!!