中華大學資訊工程學系

期末專題報告

應用多目標遺傳演算法於廠房內運輸機器人之避障路徑規劃

組員: 林紹驊、張懷澤、楊士練

指導老師: 陳建宏 老師

製作時間: 100 年 9 月 ~ 101 年 6 月

專題審定書

中華大學資訊工程學系學生:

林紹驊、張懷澤、楊士練

所製作之專題題目:

應用多目標遺傳演算法於廠房內運輸機器人之避障路徑規劃

符合資工系學生專題畢業水準

指導老師:

中華民國 年 月 日

目錄

摘要

未來的生活中，想必將會是智慧型機器人來為人類代勞，並為人類提供更為便利的生活。因此，近年來各大國家早已開始研究機器人的設計，思考機器人能夠為人類提供哪些服務。而在現代化的工廠中，隨著工件加工操作程序的大量化及複雜化，越來越多的工件也需要被搬送到不同的機台上進行加工操作。在這類需要高機動性搬運系統的製造工廠中，也就更需要有專業的機器人來有效率的進行工件的運輸。

本專題將針對廠房內之運輸機器人系統進行研究。假定在一個廠房中，在廠房內都建置有多部的工作機台，本專題擬在考量運輸最短路徑、路線的平滑度和保持與機台（或障礙物）足夠的安全距離下，應用多目標遺傳演算法（Genetic Algorithms）來規劃設計運輸機器人的運送路線。

本專題並將採用本系現有之樂高設備（Lego Mindstorm）來模擬工廠，並組裝一樂高機器人進行模擬測試，以驗證本專題之研究成果。我們希望藉由此研究，更深入了解人工智慧及機器人的運作原理，並加強分析環境中需要克服的問題，及選擇最佳的策略來解決問題。

簡介

過去以來，早有無數學者投入演化式運算（evolutionary computation）的研究，演化式運算在電腦科學中，屬於人工智慧的一小部份，用來解決組合最佳化的問題，其運算模型為模擬自然界的演化過程。演化式運算利用迭代的進行，就如同自然界的一群體不斷地在演化，再利用特殊設計的搜尋機制，解各種困難的組合最佳化的問題，並找到問題的最佳解。演化式計算所建立的模式通稱為演化式演算法（evolutionary Algorithms），其中最主要的演算法為演化式規劃、演化策略、與遺傳基因演算法，在本計劃中將使用遺傳基因演算法，其他演算法在此計畫中不再多做描述。

現行的製造工廠中，有些工廠可能因生產動線或設備時有調整位置的需求，因此在建造工廠時，並無配置原物料的運輸軌道，進而需要仰賴人力的搬運。另外工廠為了提高產值，必須二十四小時持續地執行任務，在配合工廠長時間的作業下，人類會有體力上的限制，並且作業中發生意外錯誤的機會較多，所花費的金錢成本也較高，此時取代人力的議題就顯得更重要。

若是能夠使用智慧型的機器人來取代人力，建立全自動化的生產流程，將可以達到避免人力疲勞問題、減少出錯可能、降低成本等優點，是為最佳選擇。除了可應用於工廠的生產流程外，未來也可應用於軍事的偵察任務、保全作業等。

智慧型機器人最基本的問題，就是“移動”。必須使機器人能夠如人類一樣有效率地規劃路徑，其中需要考慮如何取得最短距離、花費最少時間、與障礙物保持安全距離以及路徑的平滑度…等，來抵達目的地。甚至要考慮機器人如何運行在三維空間，而非只是二維平面。以上種種問題，將會是決定機器人移動的關鍵。在種種的考量之中，本專題擬針對運輸最短路徑、路線的平滑度和機台障礙物之安全距離等三種目標進行路徑規劃。

本報告書的結構如下所述，第三節說明遺傳基因演算法及其所使用的一些基本名詞及觀念，第四節說明路徑規劃之問題背景及搜尋最佳解的討論，第五章說明問題環境及數學模型，第六章說明系統的設計，第七章及第八章說明機器人的系統設計以及演算法相關程式，第九章說明實驗相關數據，第十章做結論…。

遺傳基因演算法

在所有演化式演算法中被拿來應用最多的就是遺傳基因演算法,遺傳演算法是由密西根大學的約翰·霍蘭德與他的同事於 1970 年代對細胞自動機(cellular automata）進行研究時率先提出的，經過 40 年來的發展，遺傳基因演算法已被廣泛的應該用在求解最佳化問題、資料搜尋、人工智慧等多個領域上。遺傳基因演算法是以達爾文的進化論為基礎，模擬“適者生存，不適者淘汰”（survival of the fittest)的生存演化法則,每一生物都會在其所生存的環境中互相競爭，只有適應強度較高的生物得以存活並繼續繁衍下一代,此種機制將會逐漸的淘汰不適合生存的品種，到最後將留下優良的品種。生物的演化一詞，是指生物的每一代的生物染色體中的基因排列發生了變化的情形;則適者生存是指這一代的基因排列優於上一代的基因排列，而更能適應環境存活下來。

所以遺傳基因演算法著重在基因的轉變,因此只需將問題的可能解經過編碼成基因型式，並利用遺傳運算子演化來找出最佳解。這些遺傳運算子是模擬生物界的演化程序，包括挑選（selection or reproduction)、交配（crossover)、突變（mutation)，其交配運算子為遺傳基因演算法的主要運算子，次之為突變運算子。

在遺傳基因演算法中將會使用以下名詞及觀念，群體（population）是由一堆個體（individual）所組成，其個體是由一堆基因（gene）組成，基因是構成染色體（chromosome）的基本單位。一個世代（generation）就是一次的演化過程的結果。

相關研究與探討

搜尋最佳解是日常生活中常需要用的方法，解決數學中的最大化（maximization）、極小化（minimization）的問題…等極端問題，如果函數可微分則可使用梯度法尋優（optimization by using gradient method），不可微分則有其他如模擬退火法(simulated annealing)、隨機搜尋法(random search)等方法。這些方法有一共同的特性，即找到的結果可能是最佳(global optimum)，也有可能是次佳（local optimum）。結果為最佳或次佳要看運氣，如果是一高難度的最佳化問題，那尋找到最佳結果的機會就更遙遠了。

遺傳基因演算法最大的優點就是當它在尋優的過程中，可以避免陷入局部最佳解（次佳點或局佳點），因此它找到全域最佳解的機會是較高的。這主要是因為遺傳基因演算法中擁有突變運算子的原因。另外遺傳演算法也不一定每一次都能找到最佳解，但它總能在最佳解附近出現，而不像其他方法，大部分的機會都會落在次佳點，這就是為什麼使用遺傳基因演算法的原因。

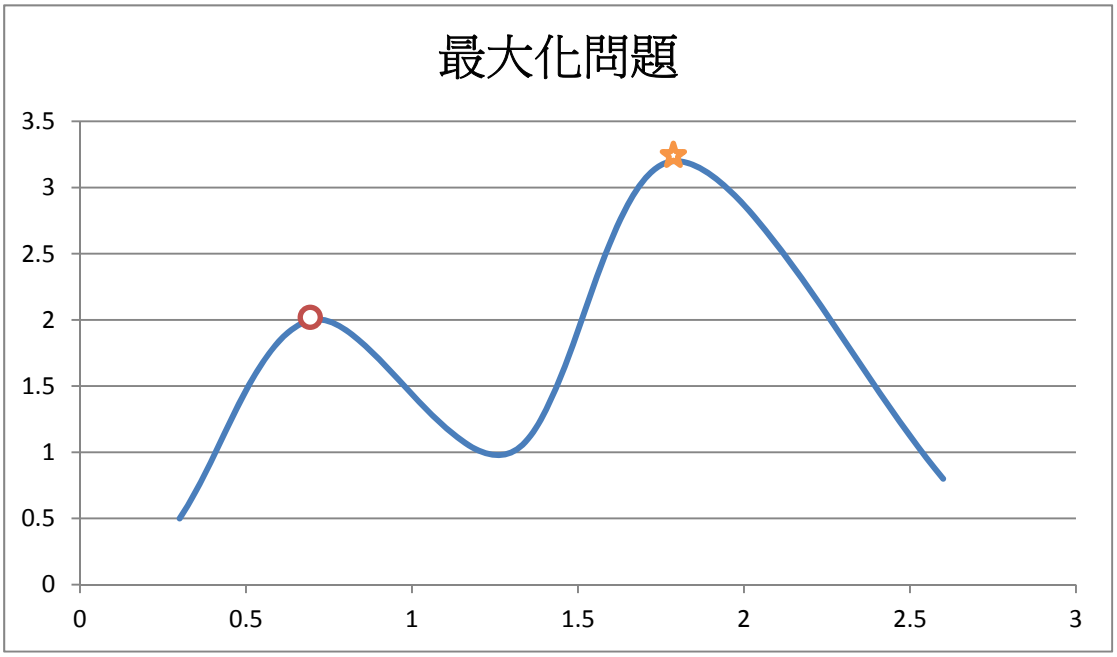


Figure 1

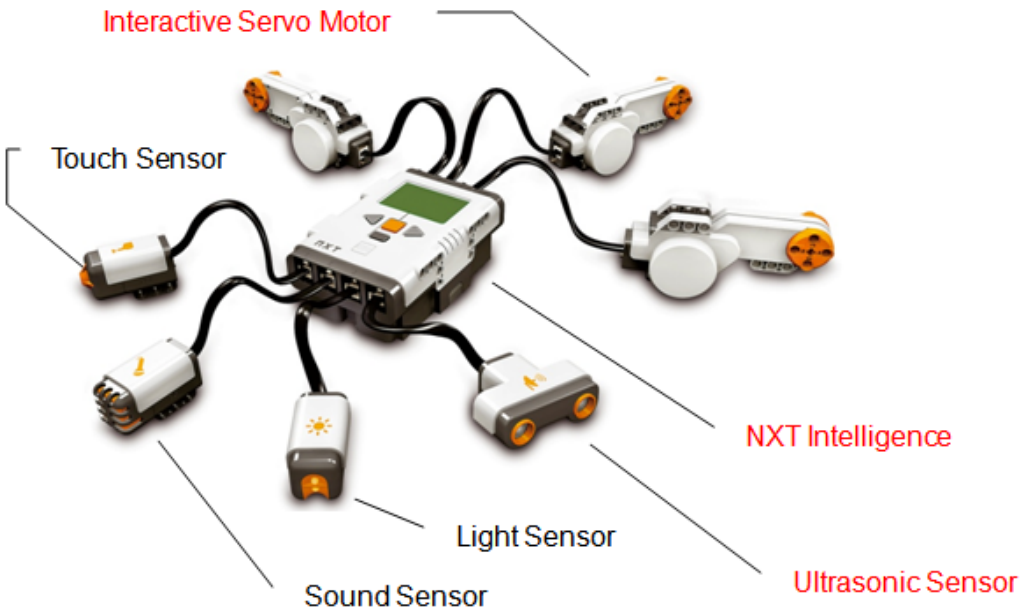
突變運算子可跳脫局部最佳解進而找到全域的最佳解

在過去已有不少學者對於路徑規劃的議題做過不少研究及探討，最著名的即為銷售員的旅程最佳化問題，但其解僅為最短路徑，不適合應用於運送工件之機器人。

為了達到多目標的路徑，因此本研究計劃將使用多目標遺傳演算法，設計出一條路徑較適合運送工件之機器人行走，其理想路徑需為較短的行走路徑、路線較為平滑，使機器人減少轉彎的機會，並且路線上能與障礙物保持足夠的安全距離。最後再利用樂高機器人進行測試，驗證本專題之研究結果的正確性。

樂高機器人套件(LEGO MINDSTORMS Robotics Invention System)為樂高集團所製造的可程式、組合的機器人玩具。針對 12 歲以上年齡，對機器人研究有興趣的一種教育器材，是丹麥樂高公司和美國麻省理工學院的 Media Lab 共同進行的一項合作案。

樂高機器人的核心是一個稱為 NXT 可程式化的積木，它具有六個 I/O 埠，分別為三個連接感應器以及三個連接使用馬達的埠。本專題使用樂高機器人作為實驗的實作工具，特色在於樂高機器人的組裝較為簡單且可以靈活應用，自行發揮創意組合出適合本專題所使用的機器人。



問題環境及數學模型

本專題假設在一個已知的廠房中，其中並無運輸軌道之設施，廠房中設置有多部的生產設備，欲製造的工件首先將由廠房中之某部機台進行組裝生產，然後再利用運輸機器人運輸至廠房中的另一部機台進行後續的操作。

在規劃路徑的表示法方面，本專題將廠房平面做格點化成整數的座標空間，每一個格點都以(x,y)表示。形成座標系統後，本專題即可以採用點線的表示方式，以點與點之間的連線代表運輸機器人之行走路徑。因此本專題之路徑規劃問題之解答即為一組座標之集合，即座標集合之連線代表運輸機器人之運輸路徑。本專題將分為多目標路徑規劃之設計和樂高系統實作兩大部分。

多目標路徑規劃系統之設計

數學建模

執行多目標遺傳基因演算法時，我們需制定問題要求的目標函式(objective fucntion)，因此我們制定以下兩個目標函式：

1. 最短距離：

$$\textbf{Minimize dist(p)} = \sum_{\mathbf{i=1}}^{\mathbf{n-1}} \mathbf{d}(\boldsymbol{m_i}, \boldsymbol{m_{i+1}})$$

dist(p)為路徑總長，d(m_i, m_{i+1})為 m_i 點至 m_{i+1} 點的距離。

2. 路線平滑性：

$$\textbf{Maximize smooth(p)} = \sum_{\mathbf{i=2}}^{\mathbf{n-1}} \mathbf{degree}(\boldsymbol{m_{i-1}}, \boldsymbol{m_i}, \boldsymbol{m_{i+1}})$$

smooth(p)為路徑平滑性，degree(m_{i-1}, m_i, m_{i+1})為 m_{i-1} 點至 m_i 點的連線與 m_i 點至 m_{i+1} 點的連線所形成的夾角。

安全距離之目標設計

在執行初始工作時，將讀入的地圖做掃瞄，在掃瞄的工作中，將會抓取障礙物。同時並將障礙物向外澎漲一個整數單位。即可做到運行機器人與障礙物的行走路線有一定的安全距離，亦可增加機器人行走路線的誤差容許值。

多目標遺傳基因演算法之設計

在多目標遺傳基因演算法中，染色體的資訊是一件很重要的事情。我們採用不固定長度的染色體，其中每個基因代表一個格點的位置，分別代表一組整數座標(x,y)，其初始的染色體是隨機產生的。

由於必須同時滿足多個目標函式的最佳化，因此利用適應函數來表示，我們採用基於 Pareto 理論基礎，所設計出的評估函數 GPSIFF(Generalized Pareto-base Scale-Independent Fitness Function)於演算法中，數學方程式如下：

$$F(\mathrm{Y}) = \mathrm{p} - \mathrm{q} + \mathrm{c}$$

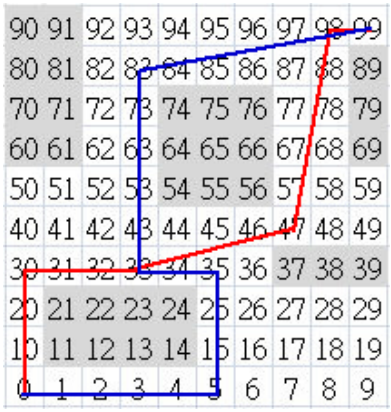
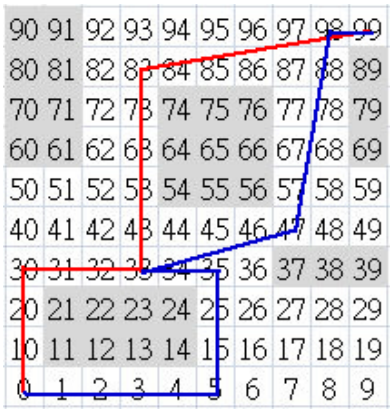
本專題使用了以下基因運算子：

1. 選擇（selection）：
   本專題採用二元競爭式選擇法（binary tournament selection）。

2. 交配（crossover）：
   隨機點交配法（one-point random crossover）可能會使結果產生迴路或不合理路徑。為了避免此情形，Li, Zhang, Yin,以及 Wang 提出及使用三種 knowledge-based 單點交配法：

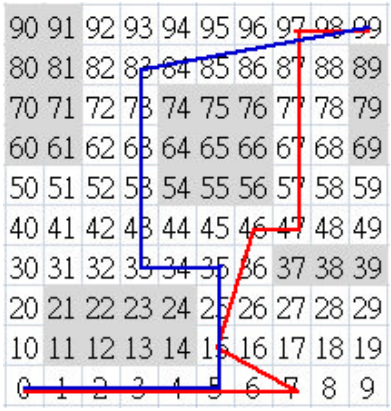
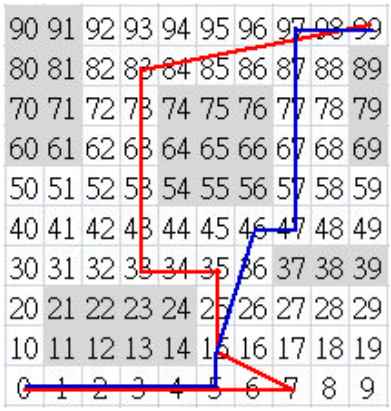
甲、相同點單點交配法（crossover at the location of the common node）：

例：母代兩條染色體分別為 V1(0,30,33,47,98,99)及 V2(0,5,35,33,83,99)，兩條染色體中有相同的點為 33。因此，還擇 33 當作其交配的點，其子代就為 V3(0,30,33,83,99)以及 V4(0,5,35,33,47,98,99)。如下圖所示：

母代染色體(V1: red, V2: blue)	子代染色體(V1: red, V2: blue)
	

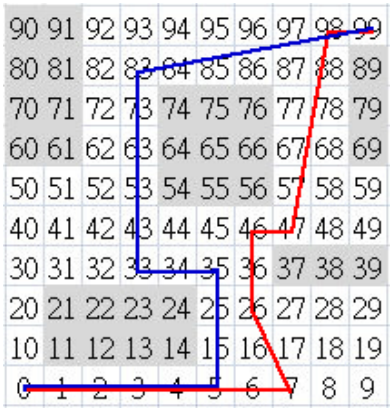
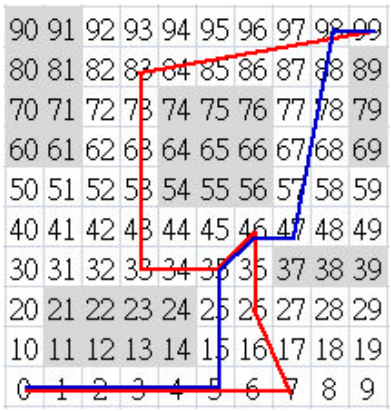
乙、相同邊單點交配法（crossover at the location of the interconnected）：

例：母代兩條染色體分別為 V1(0,7,15,46,47,97,99)及 V2(0,5,35,33,83,99)。V1 有潛在的點 15 與 V2 點 5 與點 35 所形成的邊可構成一條線。在 V2 插入點 15，V2 變成(0,5,15,35,33,83,99)，再將其做相同點的單點交配，產生的子代為 V3(0,7,15,35,33,83,99)以及 V2(0,5,15,46,47,97,99)。如下圖所示：

母代染色體(V1: red, V2: blue)	子代染色體(V1: red, V2: blue)
	

丙、相近點單點交配法（crossover at the location of the potential node）：

例：母代兩條染色體分別為 V1(0,7,26,46,47,98,99)以及 V2(0,5,35,33,83,99)。兩條染色體有相近的點為 46 以及 35。所以，有單 46 插入點 35 及 35 後面插入 46，而形成新的兩條染色體 (0,7,26,46,35,47,98,99)及(0,5,35,46,33,83,99)，再依其做相近點的交配產生的子代為 V3(0,7,26,46,35,33,83,99)以及 V4(0,5,35,46,47,98,99)。如下圖所示：

母代染色體(V1: red, V2: blue)	子代染色體(V1: red, V2: blue)
	

3. 突變（mutation）：

突變是隨機選擇一個點並且以不相同的點取代。突變的主要工作是造成族群解的差異性。因此，突變後的結果不一定會比突變前好。

4. 修補（repair）：

修補的目的是將一條不合理的路徑修正成合理的路徑。修補可分為點的修補（node repair）及線的修補（line repair）：

甲、線的修補：

若規劃的路徑會穿過障礙物，則會自動利用凸殼演算法（convex hull）運算出一條合理的路徑取代會穿過障礙物的路徑，使路徑可以完美的避開障礙物。

乙、點的修補：

行經的路徑中，若有座標落在障礙物上，則隨機產生一個新的座標點取代該點，以期能夠使路徑座標落在障礙物之外。

5. 刪除（Deletion）：

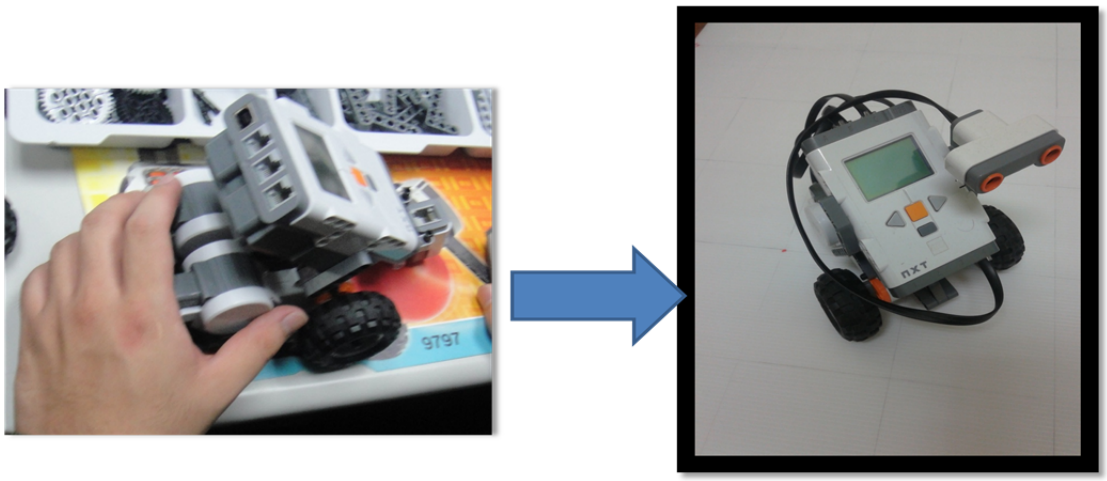
刪除應用在不合理的點，即針對不合理的點做刪除的動作，如：m 點至下一點 m_{i+1} 為同一點。

樂高系統之實作

使用硬體

- **NXT 智慧型機器人主機(NXT Intelligence)**
  I/O 介面有四個感應器輸入，三個伺服器馬達和一個 USB2.0 的連接埠。
- **馬達(Interactive Servo Motor)**
  機器人的動力來源，馬達內建有角度感應器，用來做精準的轉向控制。
- **超音波感應器(Ultrasonic Sensor)**
  距離測量，偵測範圍約在 10~150 公分之間。

實體組裝

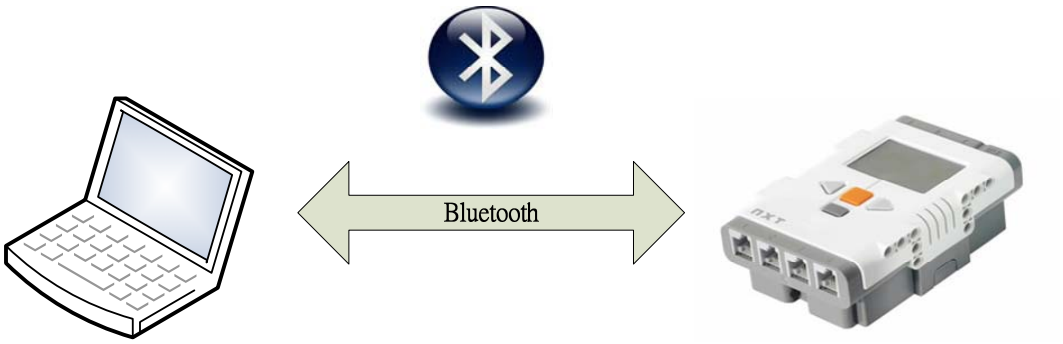


透過簡單的構造組合成一台能夠經由馬達自由行走的三輪自走車。

機器人開發工具

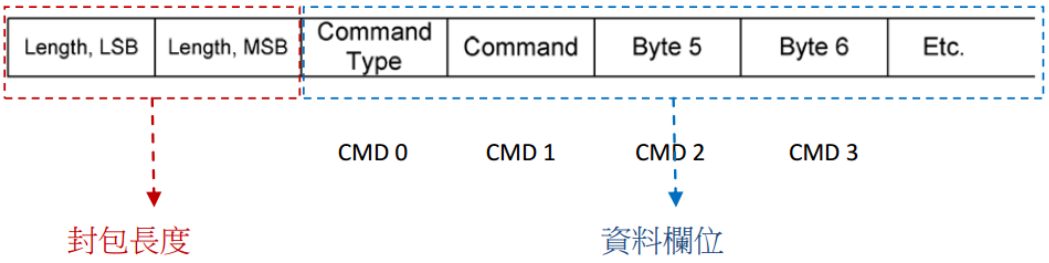
- Virtual Studio 2010
- Brick Command Center

傳輸介面



從 PC 發送資料到 NXT 主機上必須透過藍芽介面來傳送，在 PC 上偵測到 NXT 藍芽裝置後便會透過序列埠(SPP)的服務進行 PC 與 NXT 之間的通訊，當完成連線後　此時 NXT 上顯示藍芽連線的圖示(下方右圖)，也同時在 PC 上擁有一個虛擬的 COM port。

NXT 的通訊協定封包如下圖所示:

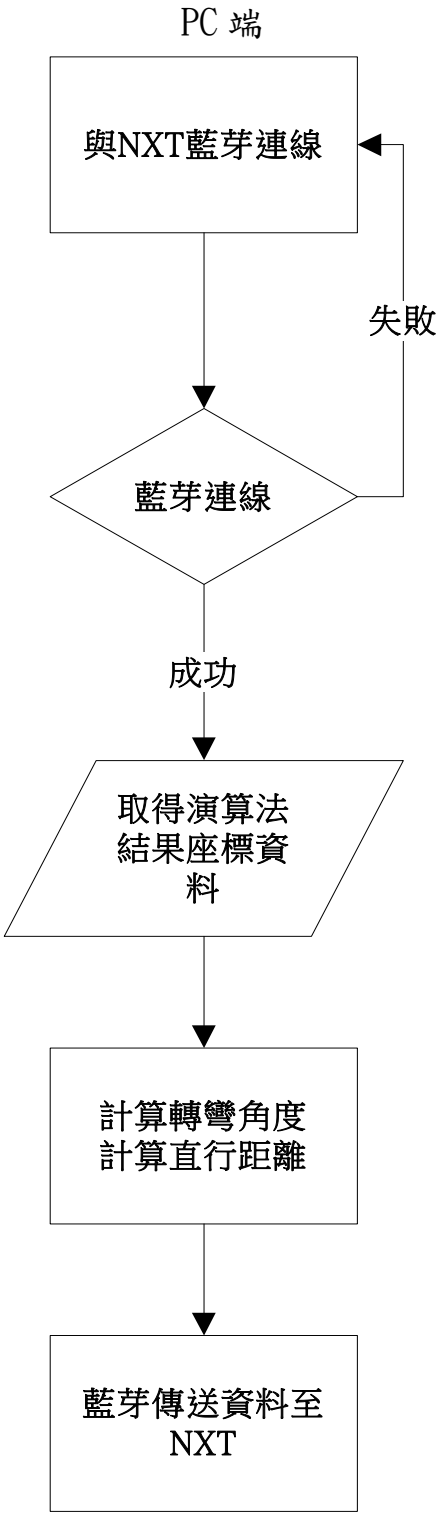


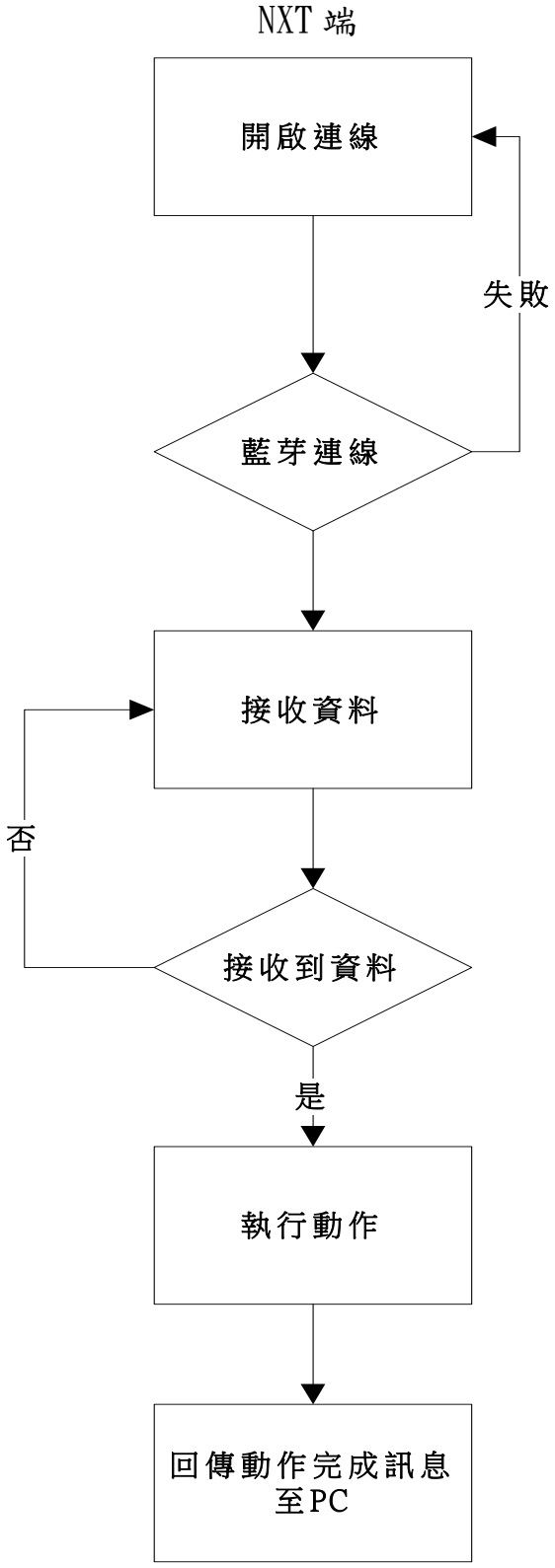
- 封包長度: 告知封包總長度。
- Command Type: Type 種類如下所示。

內容	命令型態	是否需要回應
0x00	直接命令	需要回應
0x01	系統命令	需要回應
0x02	回應命令	NXT 回應
0x80	直接命令	不須回應
0x81	系統命令	不須回應

- Command: 所使用的 NXT Direct Command。

系統流程圖

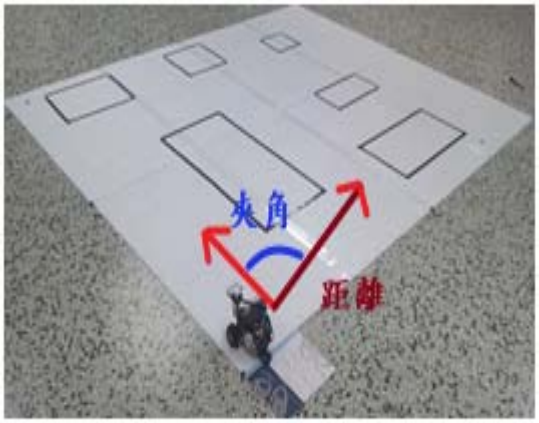

PC 端
與NXT藍芽連線
失敗
藍芽連線
成功
取得演算法
結果座標資
料
計算轉彎角度
計算直行距離
藍芽傳送資料至
NXT

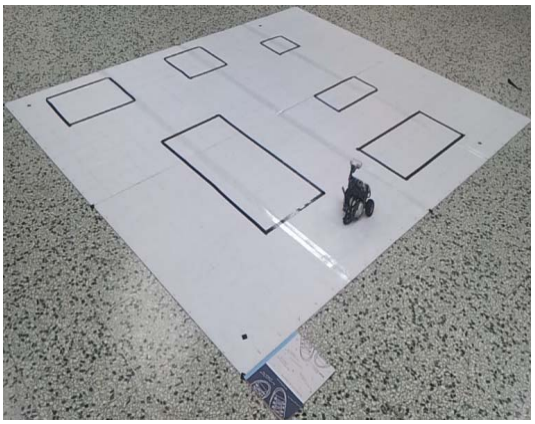

NXT 端
開啟連線
失敗
藍芽連線
接收資料
否
接收到資料
是
執行動作
回傳動作完成訊息
至PC

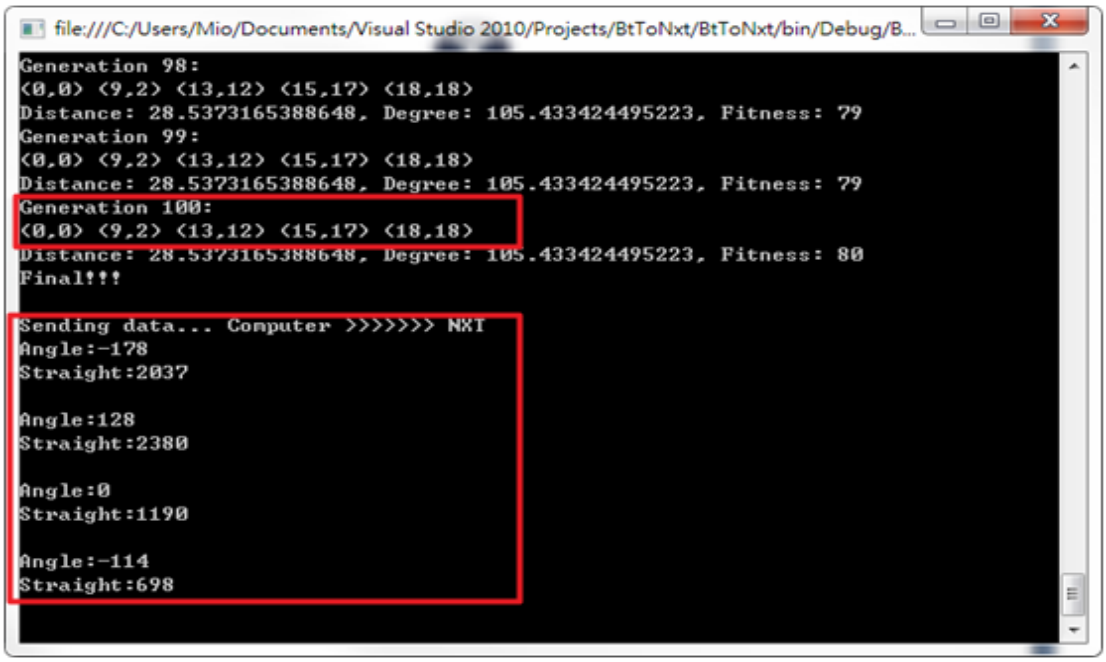
計算轉彎角度、直行距離方法:

機器人由虛擬地圖座標經過 PC 演算法產生的座標結果加以運算後，會得到機器人輪胎需要的旋轉角度以及直線行駛的距離。計算角度的方式利用向量夾角的原理計算，而直線的行走距離則是利用座標兩點距離計算；當兩個主要數據計算出後再將實際地圖上與虛擬地圖上的角度和距離單位做數據轉換，轉換完成後再依序的將這些數據傳送至 NXT。

- 向量內積（dot product）
  A dot B = |A| |B| cos(θ)
- 取夾角
  θ = Acos((A dot B) / (|A| |B|))







PC 程式碼

```
/*
*與 NXT 的藍芽連線
*/
bool InternalConnect()
{
    if (!_BTConn.IsOpen)
    {
        _BTConn.Open();
        _BTConn.ReadTimeout = 1500;
        return true;
    }
    else
    {
        return false;
    }
}

/*
*傳送資料 tmp:數據、mailbox:傳遞給 nxt 上的 mailbox 號碼
*/
void SendMsg(int tmp, int mailbox)
{
    Byte[] MessageLength = { 0x00, 0x00 };
    byte[] Command = { 0x00, 0x09, (byte)mailbox, 0x05, 0x00, 0x00, 0x00,
    0x00, 0x00 };
    for (int ByteCtr = 0; ByteCtr <= 3; ByteCtr++){
        Command[4 + ByteCtr] = (byte)tmp;
        tmp >>= 8;
    }
    MessageLength[0] = (byte)Command.Length;
    _BTConn.Write(MessageLength, 0, MessageLength.Length);
    _BTConn.Write(Command, 0, Command.Length);
}
```

```
/*
*計算轉彎角度、直走距離。參數 v1,v2 為兩個向量
*/
const int transAngle = 4; //1:4(地圖轉彎角度的單位轉換)
const int transLength = 200; //1:200(地圖行走距離的單位轉換)
move(Vector v1, Vector v2)
{
    int angle = (int)Math.Round(Vector.AngleBetween(v1, v2) * transAngle,
    MidpointRounding.ToEven);
    int straight = (int)(v2.Length * transLength);
    if (angle > 0)
    {
        SendMsg(angle, 0);
        SendMsg(straight, 2);
    }
    else if (angle == 0)
        SendMsg(straight, 2);
    Else
    {
        SendMsg(angle, 1);
        SendMsg(straight, 2);
    }
    return 0;
}
```

NXC 程式碼

```
/*
* 連線確認
*/
bool BTCheck(int conn)
{
    //偵測 NXT 的藍芽連線狀態
    if(BluetoothStatus(conn)!=NO_ERR)
    {
        return true;
    }
    else if(BluetoothStatus(conn)==NO_ERR)
    {
        return false;
    }
}

/*
* 資料取得
*/
Void GetData()
{
    //NTX 接收來自電腦傳送的資料
    ReceiveRemoteNumber(0,true,mail1);
    Receive*R*emoteNumber(1,true,mail2);
    ReceiveRemoteNumber(2,true,straight);
}
```

```
/*
* 判斷接收到的資料及執行 NXT 的動作
*/
void NXTmove()
{
    //左轉判斷
    if(mail1!=0){
        RotateMotorEx(OUT_BC,75,abs(mail1),-100,true,true);
        RotateMotor(OUT_BC,75,straight);
        SendMessage(3,"f");  //資料接收完成回傳 f 至電腦
    }
    //右轉判斷
    else if(mail2!=0){
        RotateMotorEx(OUT_BC,75,abs(mail2),100,true,true);
        RotateMotor(OUT_BC,75,straight);
        SendMessage(3,"f");  //資料接收完成回傳 f 至電腦
    }
    //直走判斷
    else if(straight!=0){
        RotateMotor(OUT_BC,75,straight);
        SendMessage(3,"f");  //資料接收完成回傳 f 至電腦
    }
}

main()
{
    while(BTCheck(0));    //檢查藍芽連線
    while(dist())
    {
        NXTmove();
        Wait(2000);
        ClearScreen();    //清除 NXT 上 Screen 資訊
    }
}
```

演算法及虛擬碼

遺傳基因演算法主體

以下為遺傳基因演算法之虛擬碼：

```
Procedure GeneticAlgorithm
{
    Randomly generate initial population P(t);
    Evaluate P to obtain fitness;
    while (termination is not met)
    {
        Select two parents p1 and p2 from P(t);
        Perform crossover for p1 and p2 to produce c1 and c2;
        Mutate c1 and c2;
        Check path of c1 and c2 are not cross over obstacle;
        Evaluate C to obtain fitness;
        Put C and P into P(t+1);
    }
}
```

遺傳基因演算法開始運行之第一步，須先產生第一代群體(群體數量假設為N)，接下來再替第一代群體計算其適應值。第二步，根據群體的適應值使用二元競爭式選擇法，挑選出兩個母代。第三步，將兩個母代依據機率做交配運算，並產生兩個子代。第四步，依據機率將產生的兩個子代做突變運算。最後，計算子代的適應值並將子代與母代混合，重新選出較優秀的前N個群體。

挑選基因運算子

以下為二元競爭式選擇法之虛擬碼：

```
Procedure BinaryTournamentSelection
{
    int i;
    while (i is not count of population P)
    {
        if fitness of P(i) is greater than P(i+1) then choose P(i);
        else choose P(i+1);
        i = i + 2;
    }
}
```

將群體兩兩根據適應值做比較，選出較高適應值的個體。

突變基因運算子

```
Procedure Mutation
{
    if (p is greater than MP)
    random a new point and replace one point of individual;
}
```

相較交配而言，突變基因運算子為一個較簡單的基因運算子。只需根據機率 p 的值決定是否執行突配運算（p 必須大過 MP，通常 MP 為 0.2）。若條件成立，則只需產生一個新的點，其點不可以落在障礙物上（需為合法點），並取代個體原有的其一點即可。

交配基因運算子

```
Procedure Corssover
{
    if (p is greater than CP )
{
    if two parents have same point
do Crossover1;
    else if two parents which of one, have one point in the other line
        do Crossover2;
    else
        do Crossover3;
}
else put two parents into new P
}
```

根據機率 p 的值決定是否執行交配運算（p 必須大過 CP，通常 CP 為 0.8）。

若兩個母代擁有相同點，則執行第一型態的交配運算-相同點單點交配法；若兩個母代其一母代的點，在另一母代的點集合連線中，則執行第二型態的交配運算-相同邊單點交配法；最後的一種情況就執行第三型態的交配運算-相近點單點交配法。

修補基因運算子

```
Procedure CheckPath
{
    if line(k to k+1) is cross over the Obstacle(k)
    {
        do convex hull then get two line;
        if line1 and line2 are not on edge of map
        {
            if distance of line 1 is shortest
                choose line1 replace origin line which is point k to k+1;
            else
                choose line2 replace origin line which is point k to k+1
        }
        else if line1 is on edge of map
            line2 replace origin line which is point k to k+1
        else if line2 is on edge of map
            line1 replace origin line which is point k to k+1
    }
}
```

修補基因運算子的目的是將不合法的路徑合法化，這此只修補線段；修補點的工作在產生點（初始群體、突變）的同時，已做過檢查，因此可保證點一定是合法之點。若行走的路線為點 k 到點 k+1，與其一障礙物有交點，就執行凸殼演算法（凸殼演算法在此不再贅述），結果可以大致分為兩條線，點 k 至點 k+1 的向右方向及向左方向。此時只需再確認線 1 和線 2 是不是與地圖的邊界相臨，若是則表示為不合法的路徑；若線 1 和線 2 皆為合法路徑，則挑選較短的路線。

適應函式

```
Procedure Fitness
{
    get D which is summation distance of Individual;
    get d which is summation degree of Individual;
    int p, q;
    while (i is not greater than count of P(k))
    {
        while (j is not greater than count of P(k))
        {
            if (d of P(i) is smaller or equals d of P(j) and D of P(i) is smaller or
    equals D of P(j))
                p = p + 1;
            else (d of P(i) is greater than d of P(j) and d of P(j) and D of P(i) is
greater than D of P(j))
                q = q + 1;
        }
    }
}
```

適應函式為計算出個體的總路徑長及總路徑的總轉彎角度。為了完成 Pareto 理論基礎（F(Y) = p - q + c），因此後續再計算其中的 p 值與 q 值，分別為 P(i)贏過的個體數及輸過的個體數，c 值為一常數。得到的 F(Y)即代表此個體的適應值。

實驗分析與數據

環境 1 的路徑規劃

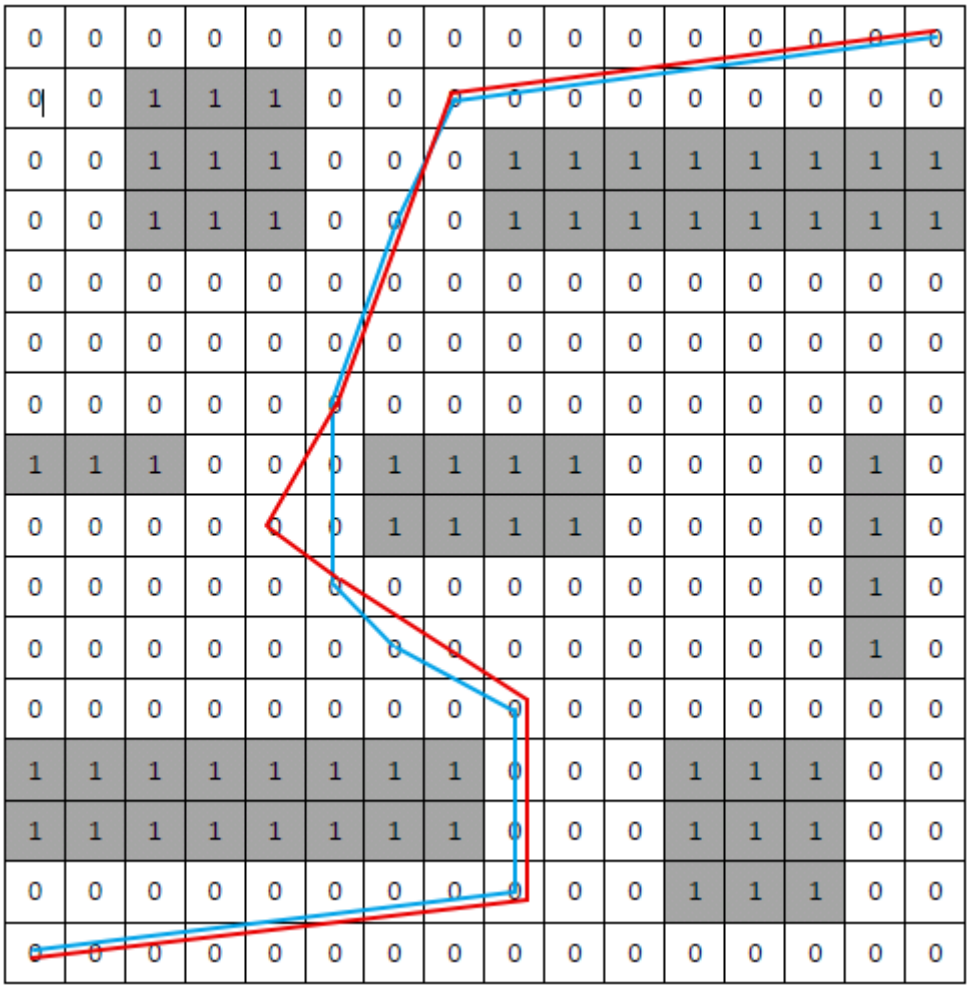


Figure 2

藍線為最短路線，紅線為最小角度路線

最短路徑：(0,0) -> (8,1) -> (8,4) -> (6,5) -> (5,6) -> (5,9) -> (6,12) -> (7,14) -> (15,15)
dist(p)：31.17314
smooth(p)：292.61986

最小角度路徑：(0,0) -> (8,1) -> (8,4) -> (5,6) -> (4,7) -> (5,9) -> (7,14) -> (15,15)
dist(p)：31.76551
smooth(p)：287.89711

環境 2 的路徑規劃

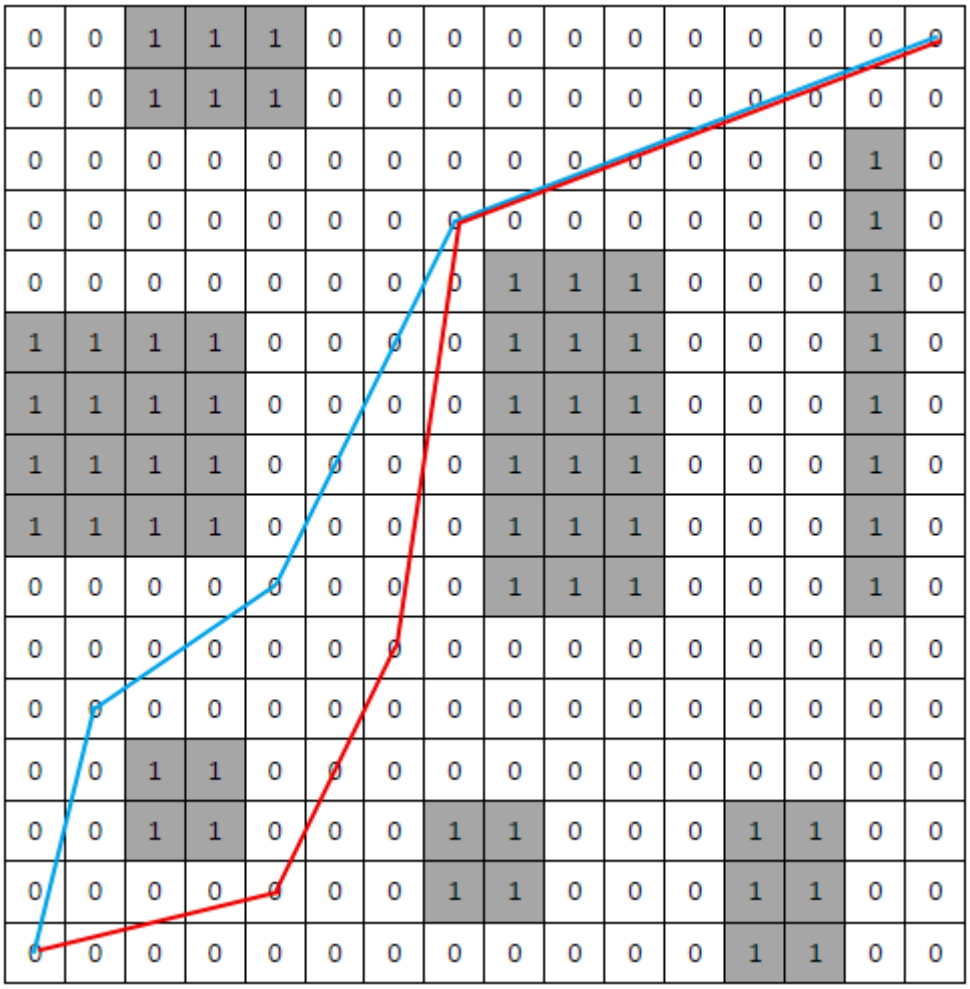


Figure 3

藍線為最短路線，紅線為最小角度路線

最短路徑：(0,0) -> (1,4) -> (4,6) -> (7,12) -> (15,15)
dist(p)：22.98086
smooth(p)：114.89747

最小角度路徑：(0,0) -> (4,1) -> (6,5) -> (7,12) -> (15,15)
dist(p)：24.21031
smooth(p)：129.14750

結論

對於無配置原物料運輸軌道之工廠，運輸機器人就非常的重要。但機器人的行走路線更是一個重要的議題，需要達到一條理想、或是一條較適合廠房的行走路線，本專題利用多目標遺傳基因演算法完成。本專題中使用了三種特殊的交配基因運算子、凸殼演算法尋找出合法路線取代不合法之路線及其他運算，尋找出兩條合理的目標路線，分別為最短路線及最小角度路線兩個目標。為了測試本專題的正確性，計畫中亦使用了樂高機器人來驗證。與樂高機器人的通訊協定，本專題使用藍牙技術將伺服器運算出來的路線傳遞給機器人，機器人收到資訊後，即開始行走。經實體環境實驗及數據分析後，證明本專題的方法及正確性是可行的。