Genetic Algorithms Homework #2

資訊所 黃子睿 P76054088

問題描述

Minimize the function

where:

本次作業為利用爬山演算法和模擬退火法來計算最佳解並且和基因演算法做比較，以下將進行爬山演算法和模擬退火法的介紹及步驟參數設定，結尾將用此三種演算法分析效果優劣。

爬山演算法

先選定隨機的位置，反轉該位置附近的位元值得到不同組的最佳解，比較不同組的最佳解和初始位置，找出最佳的解並且更新值，一直重複這樣的步驟直到收斂。

1. 初始條件設定

Generation = 200

Genetic representation: Binary string

bits

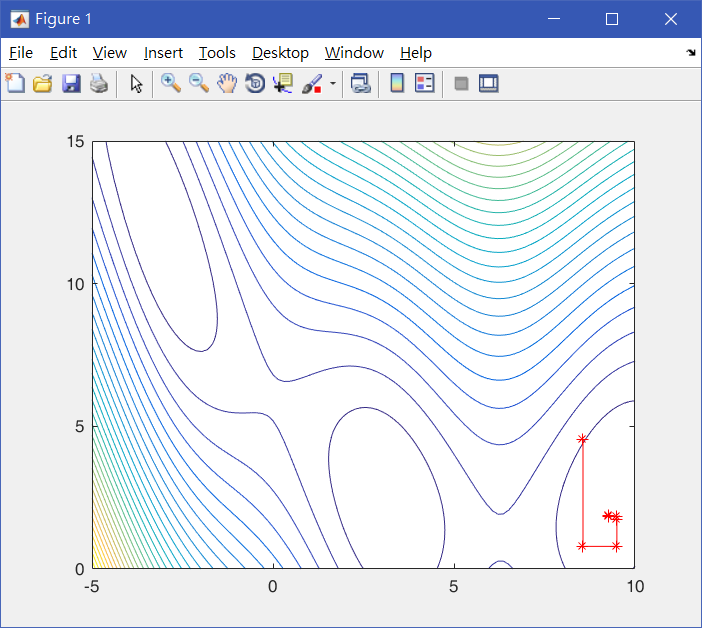
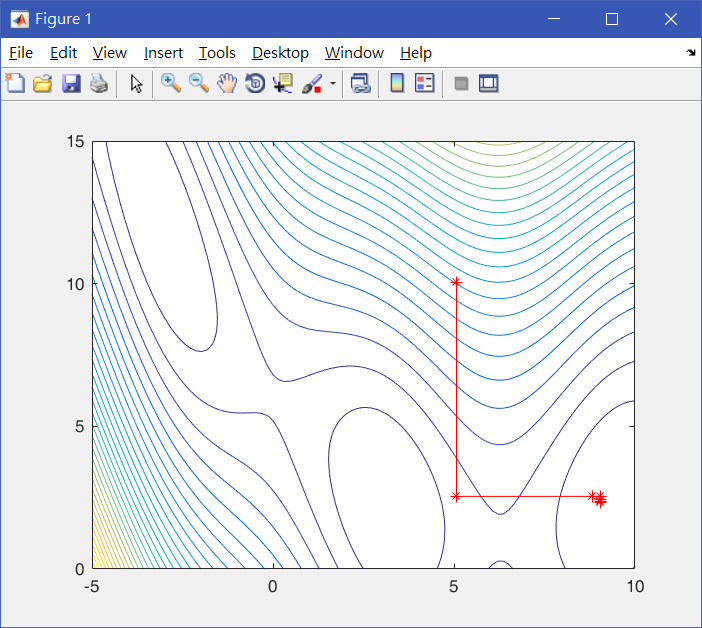
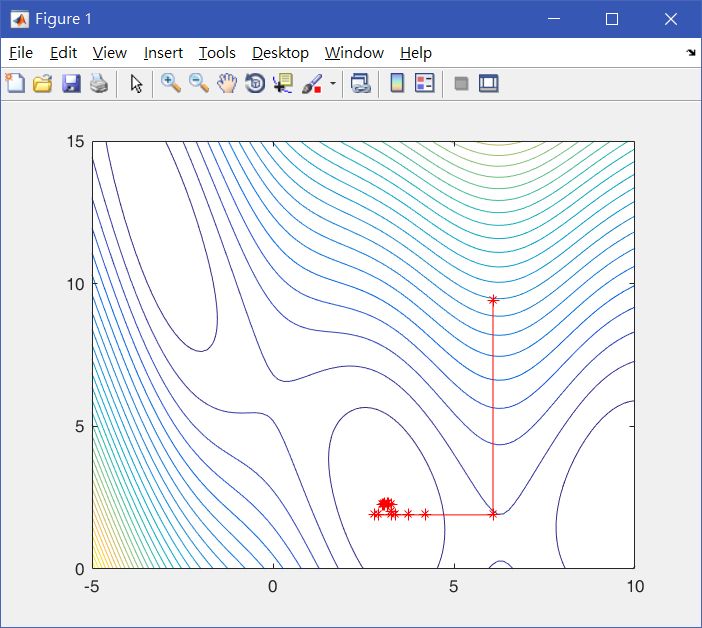
Decoding:

鄰居定義:

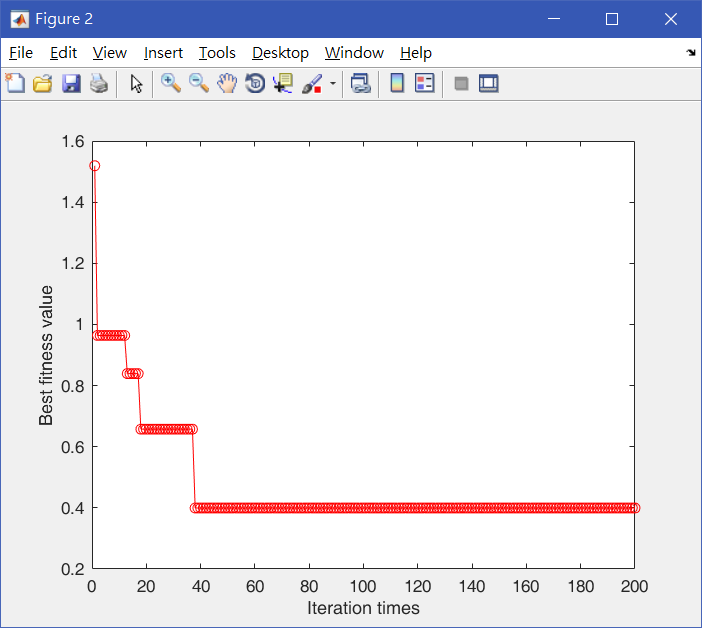
相鄰的48bits取轉換後fitness第二高的值。

1. 實驗結果

實驗結果將進行200次的迭代呈現每一代的最佳解，並畫出其中幾代爬山成功的結果圖。



收斂狀況



以下為執行程式5次，並且以表格來呈現最佳的結果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 次數 | 第一次 | 第二次 | 第三次 | **第四次** | 第五次 |
| (x1, x2) | **(3.159180, 2.255859)** | (9.395924, 2.285155) | (9.399414, 2.453689) | (-3.125000, 12.421876) | (3.232422, 2.343750) |
| 最佳解 | **0.399402** | 0.429312 | 0.400976 | 0.434072 | 0.456656 |
| CPU time | **23.589** | 24.030 | 24.381 | 24.271 | 24.603 |

1. 結論

從實驗結果圖可以看出其實爬山演算法收斂速度很快，在約四十代以內就已經收斂到最佳值。而因為有隨機產生值的關係可以讓最佳解跳脫出區域最小值，但如果一開始隨機點取的不夠好或者鄰近點太接近，就可能得到比較差的值因而比較慢得到收斂效果。

模擬退火法

先選定隨機的位置，反轉該位置附近的位元值得到不同組的解答，隨機找出一組解，如果比原先解更好或者在某些機率下更好就更新原先解，接著判斷是否降溫，最後一直重複這樣的步驟直到收斂。

1. 初始條件設定

Generation = 200

Genetic representation: Binary string

bit

Decoding:

Temperature T = 1

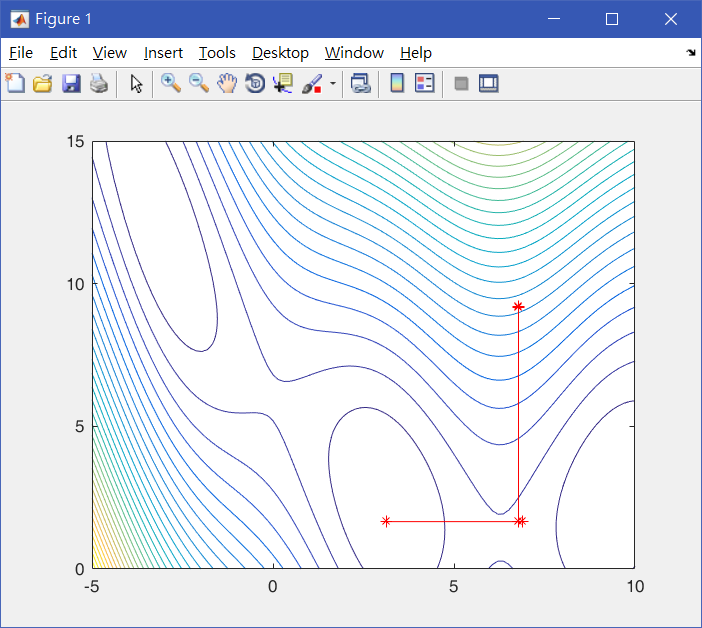
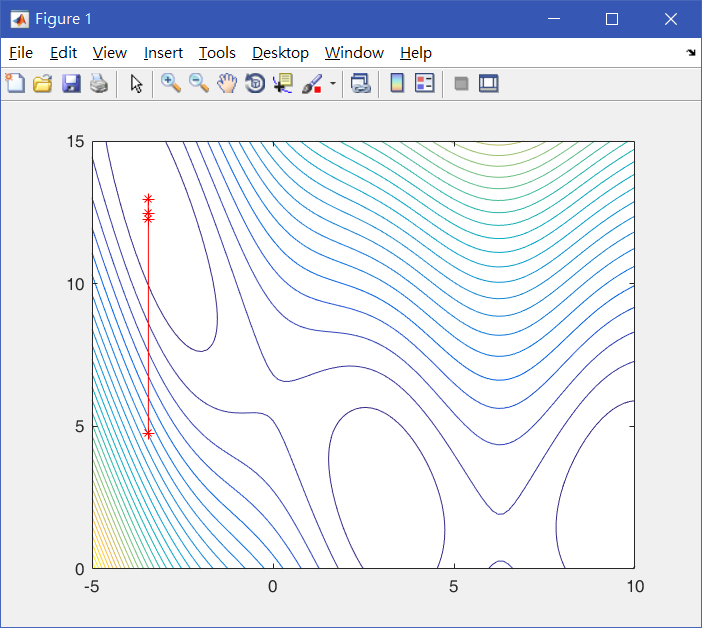
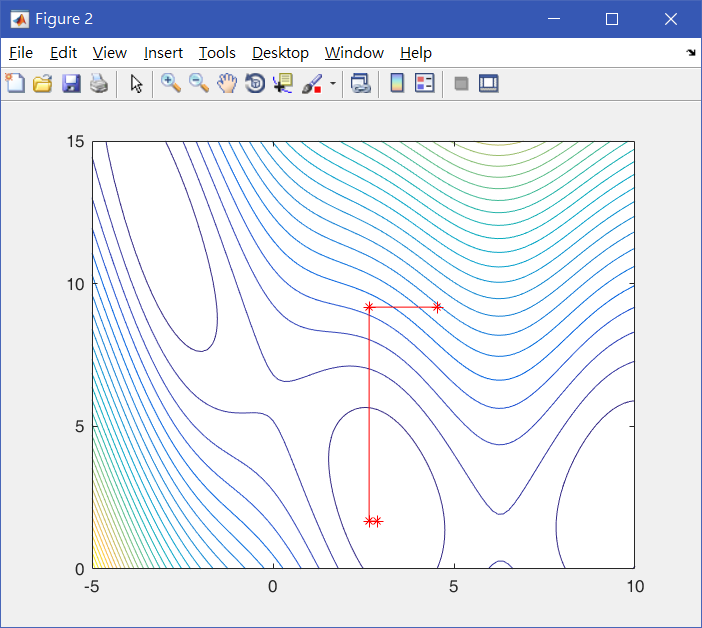
1. 終止條件設定

降溫機制T = T\*0.1 每50代降溫一次

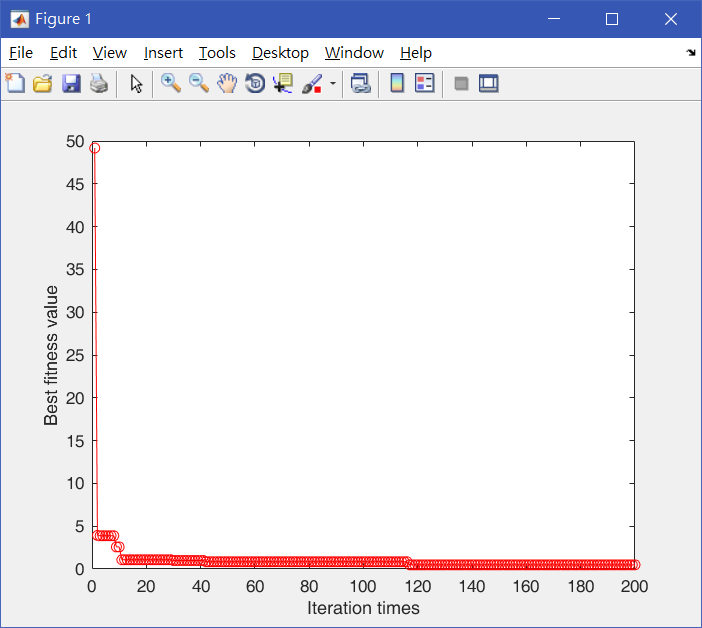
Temperature Tend =

1. 實驗結果

實驗結果將進行200次的迭代呈現每一代的最佳解，並畫出其中幾代找到最佳解的結果圖。



收斂狀況



以下為執行程式5次，並且以表格來呈現最佳的結果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 次數 | 第一次 | 第二次 | 第三次 | 第四次 | **第五次** |
| (x1, x2) | (-3.096546, 11.885665) | (-3.195999, 12.805043) | **(9.402449, 2.467975)** | (9.244634, 2.286507) | (-3.126043, 11.985221) |
| 最佳解 | 0.486780 | 0.571224 | **0.400419** | 0.554928 | 0.462775 |
| CPU time | 10.515 | 10.382 | **10.276** | 11.026 | 11.289 |

1. 結論

從實驗結果圖可以看出模擬退火法收斂速度稍微慢了一點，在約120代左右有個小振盪，因為它不像爬山演算法一樣直接選取相鄰最好的解答，取而代之的是用了機率函數和控制溫度參數來跳脫區域解。

綜合比較

以下實驗表格將基因演算法、模擬退火法初始迭代次數設為300、爬山演算法設為20、基因演算法母群體數量設為300，以方便比較

基因演算法

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 次數 | **第一次** | 第二次 | 第三次 | 第四次 | 第五次 |
| (x1, x2) | **(3.142863, 2.275443)** | (3.143457, 2.280967) | (3.146121, 2.279586) | (-3.140931, 12.265170) | (3.146080, 2.272316) |
| 最佳解 | **0.397897** | 0.397959 | 0.398052 | 0.397957 | 0.397985 |
| CPU time | **137.334** | 131.153 | 135.347 | 146.735 | 137.337 |

爬山演算法

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 次數 | 第一次 | 第二次 | 第三次 | **第四次** | 第五次 |
| (x1, x2) | **(3.159180, 2.255859)** | (9.395924, 2.285155) | (9.399414, 2.453689) | (-3.125000, 12.421876) | (3.232422, 2.343750) |
| 最佳解 | **0.399402** | 0.429312 | 0.400976 | 0.434072 | 0.456656 |
| CPU time | **23.589** | 24.030 | 24.381 | 24.271 | 24.603 |

模擬退火法

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 次數 | 第一次 | 第二次 | 第三次 | 第四次 | **第五次** |
| (x1, x2) | (-3.096546, 11.885665) | (-3.195999, 12.805043) | **(9.402449, 2.467975)** | (9.244634, 2.286507) | (-3.126043, 11.985221) |
| 最佳解 | 0.486780 | 0.571224 | **0.400419** | 0.554928 | 0.462775 |
| CPU time | 10.515 | 10.382 | **10.276** | 11.026 | 11.289 |

從實驗結果可以發現，如果要得到最佳解的話最好的方法為基因演算法，因為初始母群體數量夠大且得到的值很穩定，但缺點是需要花費比較多時間、記憶體也占用比較多空間，爬山演算法的優勢在於可以快速收斂到最佳解和模擬退火法相比收斂速度快很多，因為爬山演算法是直接找鄰近點最好的值，而模擬退火法則是隨機選一個值，且有可能因為滿足機率涵式的關係讓值變差，但因為這個設定可以比較容易跳脫出區域解。