### 課程名稱：最佳化演算法

### 題目：CDO演算法專題實現報告

### 指導老師：蕭瑛東　教授

### 姓名：林翡

### 學號：110916015

### 班級：資四甲

### 日期：2024/06/20

1. 前言：

* **背景知識**：Chernobyl Disaster Optimizer（CDO）是基於物理現象的元啟發式優化方法，靈感來自於1986年車諾比核反應堆的災難。此方法模仿由核不穩定引起的放射性衰變，其中包括三種主要輻射：γ輻射、β輻射和α輻射。
* **啟發來源：**核災害引起的大量能量釋放及其對周圍環境的長遠影響提供了對問題空間的全新理解，從而啟發了這一優化技術。
* **核心概念：**以車諾比核爆炸事故中的放射性擴散為例，描述了一種模擬輻射顆粒如何從高壓點（爆炸點）向低壓點（安全點，即人類所在地）擴散的過程。

1. CDO演算法：

* 步驟：

**Step 1. 初始化：**演算法初始化一組粒子（通常稱為“粒子群”），這些粒子將在搜索空間中移動以尋找最佳解。

**Step 2. 迭代：**使用迭代的方式進行搜索。在每一個迭代中，根據粒子的位置計算其對應的適應度（fitness），適應度通常是根據待優化問題的目標函數值來計算的。

**Step 3. 更新最佳解：**對於每個粒子，檢查其適應度是否優於三個最佳解，即γ（gamma）、β（beta）、α（alpha）粒子的適應度。如果該粒子的適應度優於γ粒子的適應度，則將γ粒子的位置更新為該粒子的位置；如果該粒子的適應度優於β粒子的適應度，則將β粒子的位置更新為該粒子的位置；如果該粒子的適應度優於α粒子的適應度，則將α粒子的位置更新為該粒子的位置。

**Step 4. 更新粒子位置：**根據每個粒子的當前位置以及一些計算得到的梯度下降因子，更新每個粒子的位置。這一步的目的是讓粒子朝著更好的方向移動，以更好地搜索解空間。

**Step 5. 更新平均位置：**在迭代的過程中，演算法會記錄每個粒子的位置，然後計算這些位置的平均值。這個平均值可以作為搜索的參考，用於指導粒子的移動方向。

* 流程：CDO演算法的程式我沒有特別改變，還是跟上一份作業一樣。

開始

輸入搜索代理數量、最大迭代次數、搜索空間下界、搜索空間上界、解的維度、目標函數

連接

Alpha\_pos=zeros(1,dim); Alpha\_score=inf;

Beta\_pos=zeros(1,dim); Beta\_score=inf; Gamma\_pos=zeros(1,dim); Gamma\_score=inf;

Positions=initialization(SearchAgents\_no,dim,ub,lb)

初始化Alpha、Beta和Gamma的位置和分數、搜索代理的位置

------

更新α、β、γ的score、位置

計算每個搜索代理的目標函數值

fitness=fobj(Positions(i,:));

fitness<Alpha\_score、Beta\_score、Gamma\_score

for i=1:size(Positions,1)

while l<Max\_iter

yes

i=1:size(Positions,1)

for j=1:size(Positions,2)

計算α、β、γ粒子的梯度下降因子，並更新位置

更新收斂曲線

Convergence\_curve(l)=Alpha\_score;

WSh=3-l\*((3)/Max\_iter);

Sa = ((log10((160001)\*rand(1,1)+16000)));

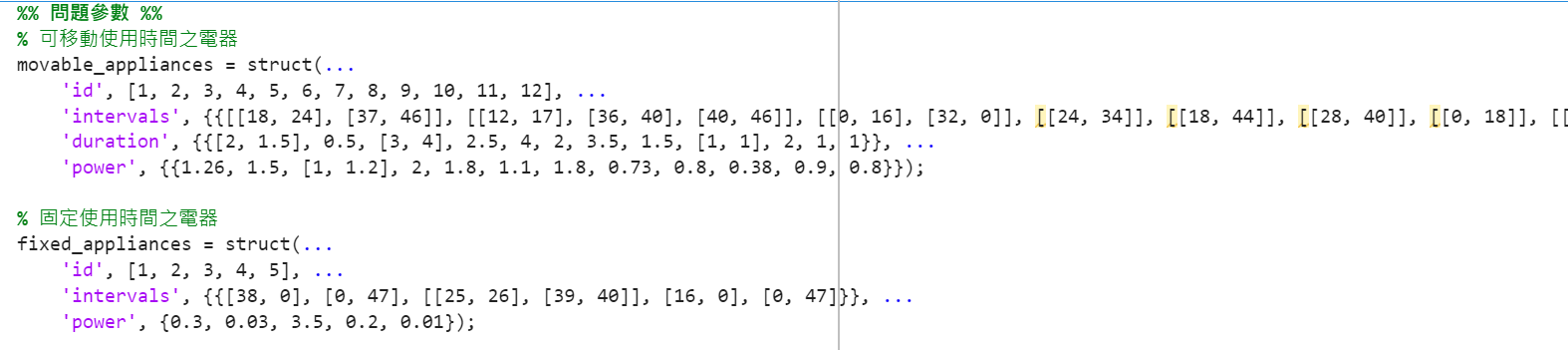
Sb = ((log10((270000-1)\*rand(1,1)+270000)));

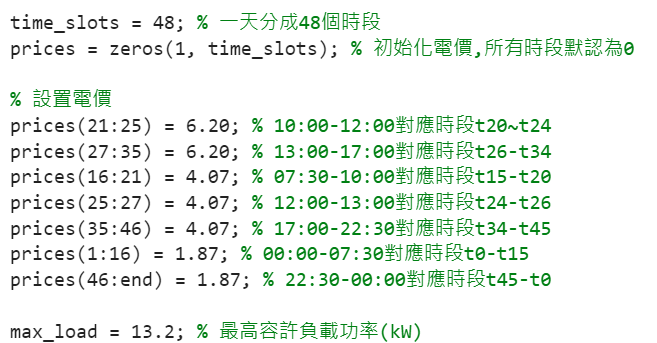
Sr = ((log10((300000-1)\*rand(1,1)+300000)));

連接

1. 應用例子：

* 定義問題參數:例如電器的相關資訊、時段數、時間電價、最高容許負載功率等等。



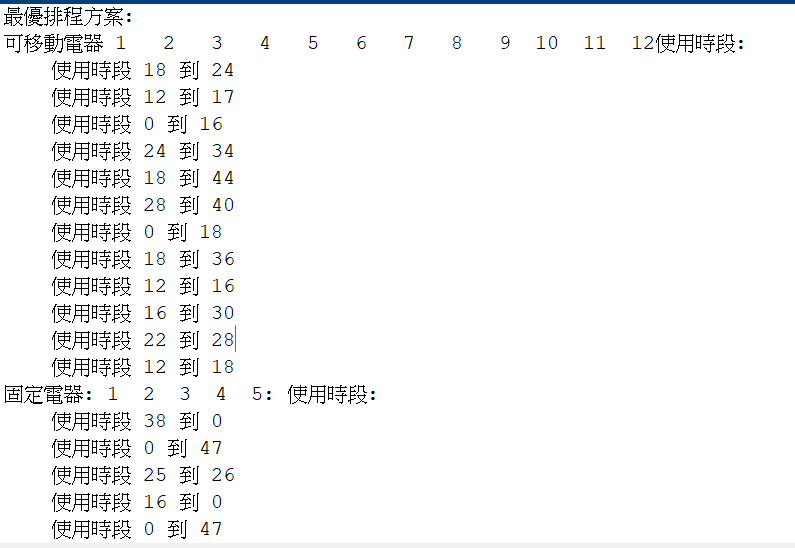
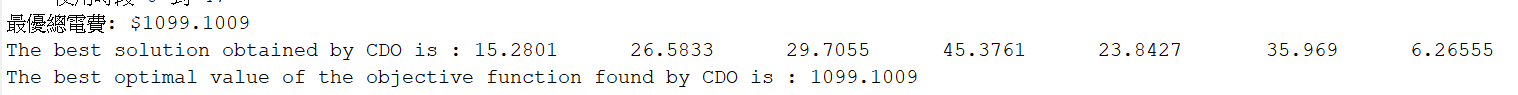
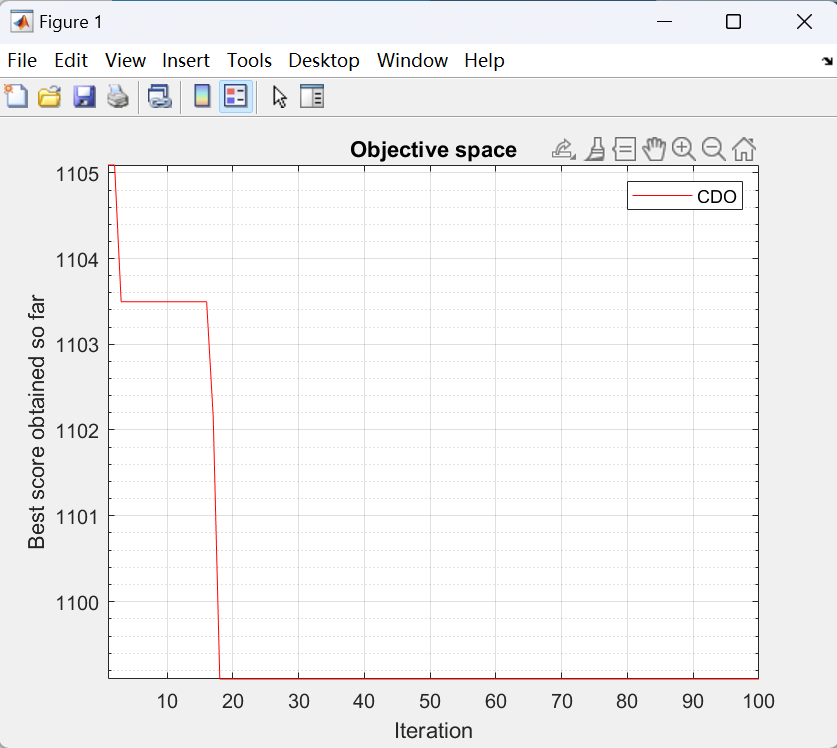


* 定義CDO參數：我把搜索代理數量設定為100，跟原先的作業一樣；迭代次數設為100，設太大程式會跑得有點久；lb設為0，表示每個時段的開始時間是從0開始，ub設為time\_slots-1=47，上下界設為0~47；dim是計算所有可移動電器的時段數量之和，對於每個可移動電器，它可以有多個工作時段，每個工作時段都需要一個變數來表示是否使用該時段。
* 解碼函數(DecodeChromosome)：用於從CDO算法得到的個體編碼中解碼出電器的使用時段和持續時間，主要分為兩部分：解碼可移動使用時間之電器和解碼固定使用時間之電器。
* 總電費函數(total\_cost)：用來計算給定排程方案的總電費,並考慮了最高容許負載的限制。接受五個參數：

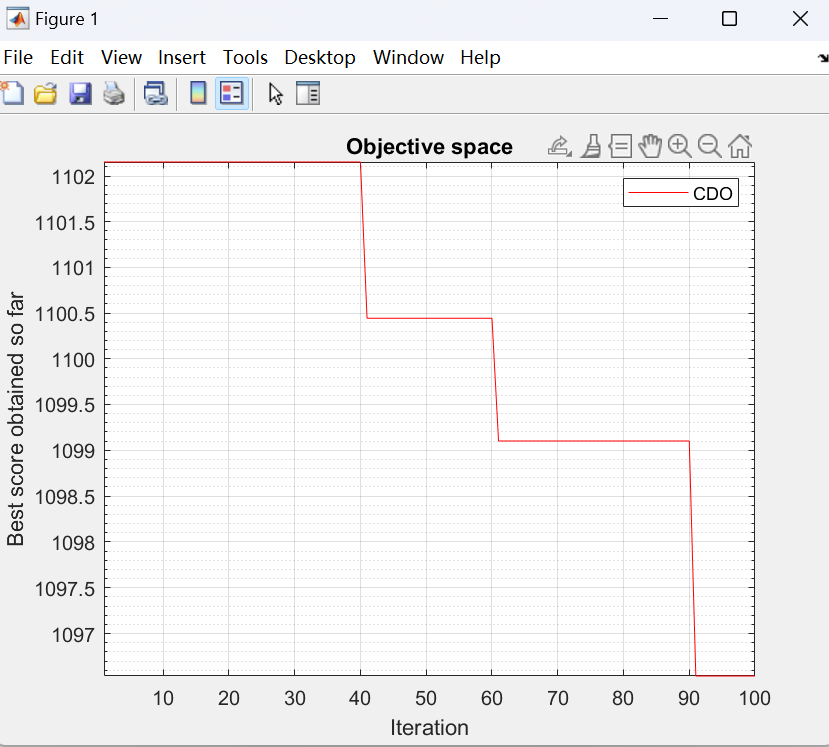
1. chrom：表示一個排程方案的個體編碼。
2. prices：一個長度為48的向量，表示一天中每個半小時的電價。
3. movable\_appliances：一個結構體，包含可移動使用時間之電器的ID、可用工作時段範圍、持續時間和功率。
4. fixed\_appliances：一個結構體，包含固定使用時間之電器的ID、工作時段範圍和功率。
5. max\_load：表示最高容許的總負載功率。

* 輸出最佳解：

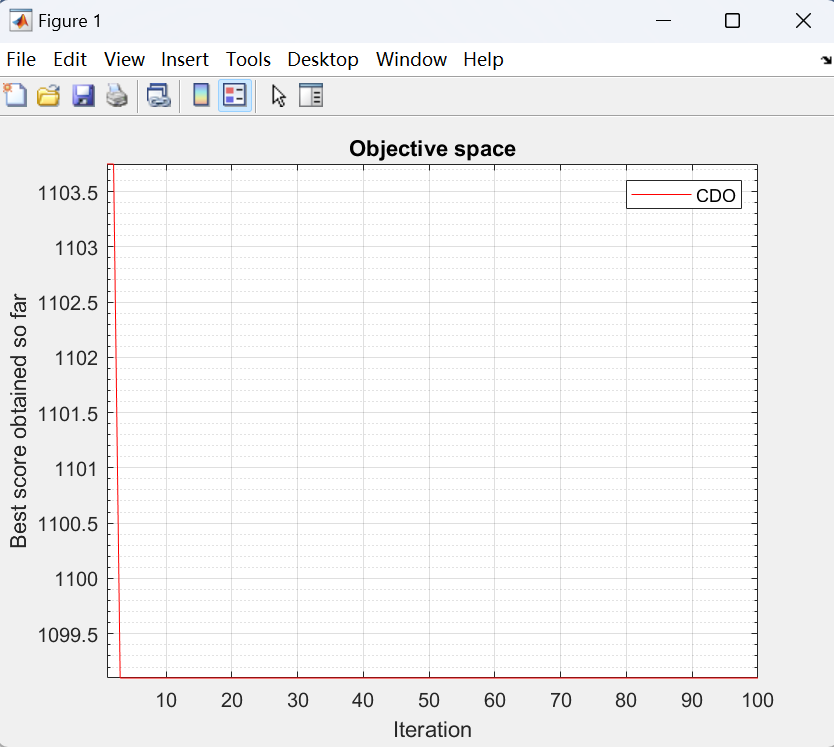
1. 將Best\_pos(CDO算法得到的最優位置)通過reshape函數轉換為適當維度的矩陣best\_schedule。
2. 遍歷所有可移動使用時間之電器與固定使用時間之電器：獲取該電器的ID、可用時段範圍、持續時間。
3. 輸出該可移動使用時間之與固定使用時間之電器的ID和使用時段。
4. 輸出最佳總電費Best\_score。
5. 繪製CDO算法的收斂曲線CDO\_cg\_curve。
6. 最後顯示CDO算法獲得的最佳解Best\_pos和最優目標函數值Best\_score。
7. 實現結果：

* 答案：
* 收斂過程：

1. 檢討與心得：

* 檢討：單看收斂曲線我覺得表現得好像還可以，但排程似乎沒有排成功，我真的debug了好久，最後只排出每個電器的使用時段而已，而且固定使用時間之電器還是原先定義的樣子，也不知道正不正確，最後得出的最佳總電費也覺得值有點大，排程排不出正確的，在debug的過程中遇到最多次狀況就是陣列索引超過的問題；然後我發現調整 chrom = randi([0, 1], 1, 40);的染色體的長度會有不同的最佳總電費(如下圖)。

染色體長度為40的最佳總電費與收斂曲線



染色體長度為50的最佳總電費與收斂曲線

* + 心得：這次作業真的是大魔王題，是個複雜問題又非常有挑戰性，花了好多時間在debug程式才可以執行，雖然結果是不盡理想的樣子，但有努力過了，上網爬了很多文也找不太到解這個問題的高手，chatGPT也被問倒了，算是提前見識研究所要解問題的威力了，因為以後只會有更複雜的問題出現。

1. 參考文獻：
   * https://blog.csdn.net/woaipythonmeme/article/details/131288863