Report Group 1 SNA HW1

r03922096洪立遠,

**前言**

我們作此作業，可以10/21為分界，分為前期，與後期。而進入的後期的關鍵，在於一個我們解決了graph reset的bug。因為跑greedy需要作模擬的propagate，在模擬的propagate之後又要reset回原本的state。這裡有一些複雜性，所以造成greedy的錯誤。

進入後期後，配合正確的greedy已可打贏MaxWeight Player 1，很多的努力放在了greedy的加速，與找出時限內的最佳策略配對。

另外，我們基於DiffusionModel的code，自己加了很多的功能如export整張圖、simulate propagate等，變成MyDiffusionModel，這樣我們才可以做到跟上整個遊戲的進程與跑自己的演算法。

**前期**

以下是我們所實作的strategy，我們作為Player 2對決助教Player 1 MaxWeight Strategy。統計數據是在egofb.txt 這張圖上跑出來的結果。

1. DegreediscountIC:

參考[KDD 2009的論文](http://research.microsoft.com/en-us/people/weic/kdd09_influence.pdf)所得到的方法單純用他跑的話對上 S1，會輸 974 點。

1. MaxWeight:

利用助教的code 直接互相對打輸900多點。

1. MaxWeight version 2:

考慮到要阻擋對手的Activation，我們會先把敵人在沒有我們干擾的情況下，一回合能夠activate的所有的點，變成一個 set，讓Maxweight Strategy每次都只在那個set中選擇點。輸 578 點。

1. DegreediscountIC Version 2:

利用同樣的阻擋對手Activation的概念，讓DegreeDiscountIC在一回合能夠activate的所有的點，去做篩選，輸了 600 多點。

**小結論**

在我們一開始輸的情況下，限制選點在一個比較小的，對方會activate的點的集合，效果會比較好。

**後期**

Greedy:

在largest connected component的點集合中，一次選擇一個點，在每一次選擇點的考慮都是，選擇那個點後，選擇的點(包含已確定選擇的點)的集合，能造成最多的activate的點。

Greedy + MaxWeight Version 2：

利用 Greedy 跟 MaxWeight Version 2 做混和，第一回用 Greedy 後面用 MaxWeight ver 2 跑會輸 478點。

Mix\_Heuristic:

一半的點用 MaxWeight去選 一半的點用 DegreeDiscount 去選，這樣的方法會輸 800 多點。

Greedy + Mix\_Heuristic:

第一回合用 Greedy，後面用 Mix Herustic，30%的點用 DegreeDiscount 70%的點利用這個方法會輸 318 點調整兩種策略的比例可以把差距拉小到250 點左右

10/21 前基本上我們做了很多的嘗試不過後來經過一番修改後又發展出了許多新的strategy 這些 strategy 較為成功

我們的 Greedy 帶入了一個新概念，包含Giant component以及 peek Window

其中我們捨棄前期 All layer 的概念，因為用Lazy Greedy跑起來的結果發現效果較差，所以決定每次都把 graph 被影響的點刪除，

Lazy Greedy

把每個點在沒有對手的情況下做模擬，把其中最好的點抓出來再重新跑一次，做的優化是結構上的優化，能夠贏到 200 多點，但是時間很久，可能在大圖中會超時

Lazy Greedy with peek window

這個版本的 Greedy 較上一版比較起來使用的方式很像，但是第一次跑完以後會 sort 並且只對第一個點後面 n 個點做模擬找出裡面最好的點。

效果不差甚至有時候會比原本的 lazy Greedy 還好

目前所使用的 Strategy

第一回合使用 Greedy去跑，後面會將一半的點給 Greedy 一半的點給Max Weight 去跑。效果沒有全 Greedy 好，例如：在hepth.txt中，每一回都用lazy Greedy 會贏2200 多個點，但是使用一半一半會在30 秒內選擇完畢並贏1500多個點。