Report Group 1 SNA HW1

r03922096洪立遠, b00902104 楊勖, b00902057陳煥元

**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。

解決bug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。

1. 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。

**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。

使用了正確的greedy之後，只要第一回合有用greedy，之後配合其他的heuristic，就會獲勝。只是greedy太過耗時，所以接下來花了時間在greedy加速上面。

另外，我們捨棄前期「利用阻擋對手Activation的概念」，因為Greedy在對方會activate的點的集合跑起來，發現效果較差。

**Greedy加速**

1. 刪掉圖中的selected與activated node

因為每一回合，會有點變成selected或是activated，而這些點對整個遊戲已經沒有影響了，繼續在graph中記錄他們及其edge，會造成overhead。所以我們會刪掉這些點及其edge，然後才export成檔案，下次restore建graph時，就會比較快。

1. Giant component

因為我們有刪掉nodes，所以會造成graph變成許多weakly connected components，而我們認為主戰場就是giant connected component，不需要在那些其他的小小的weakly connected component上選擇，所以我們會限定greedy選擇的點的set在giant connected component上。

1. peek Window

greedy選擇完第一個點後，會得到一個activate node數由大排到小node list，這時選定這個list的前n個，也就是在peeking window內的這n個，做為之後選擇2~10點的set。

也就是說，之後的2~9點，只會從這個peeking window中的點中產生，而依舊是以greedy的方式來選擇，這樣的好處就是可以控制n為一定數，目前設為600，跑得node數量少，速度就快了。

這個效果甚至有時候會比原本的全部點的Greedy 還好。

1. data structure優化

把許多list, dict, reset時重複建立的data structure都去除掉。

**最終的Strategy**

第一回合10個點使用加速的Greedy去跑，後面2-10回合，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。

這樣的考量還是因為時間限制，怕全部都用greedy超時。

**總結**

Greedy真的很強，能夠讓後手得到許多優勢，滿意外的而且也不知道為什麼，像是如果兩個Player都全部使用greedy，也是後手會贏。在Future work上面，我們希望再把greedy加速(雖然感覺已經加速到不能再加速了，而且又不能用multi thread)，然後再去實作更多paper中的演算法。