Final Project: Approximate PPR Algorithm
Group 33: 106061218 李丞恩

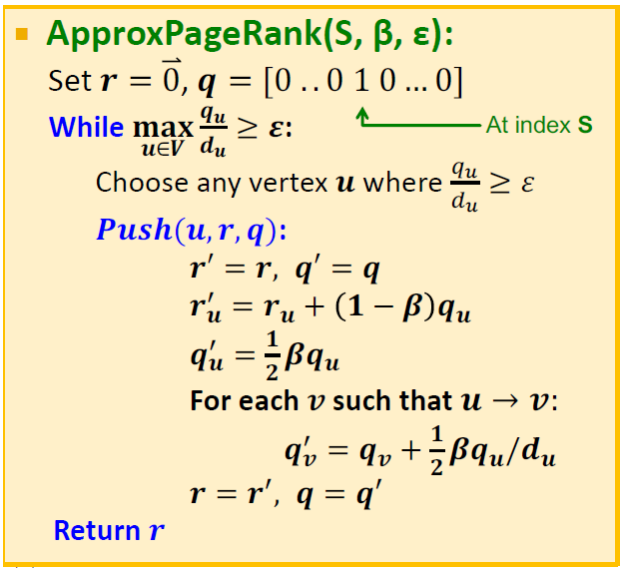
1. Specification

本專題要實作 Approximate PPR Algorithm，出自講義第 10 章的第 45 頁。在將資料讀入後，會存成 RDD 資料結構，並選取一個頂點作為 seed node。經由一系列的 map-reduce 運算後，得出資料集中每一點的 PPR。最後，利用 matplotlib 繪圖，並進行資料分析與解釋。

我所使用的 dataset 是 General Relativity and Quantum Cosmology collaboration network。[1]它是由 Leskovc 等人蒐集在 arXiv 網站上 1993 年 1 月到 2004 年 1 月的論文中，[2]學者間合作撰寫論文的關係而成的資料集。其中每個頂點代表一位作者，若兩名學者之間有一條邊，就代表兩人曾經合寫過論文。這個資料集包含 5242 個頂點，14496 個邊，兩點之間沒有多重邊（multiple edges）。但某些頂點具有 self edge。為了實作 PPR algorithm，在讀取檔案時我排除掉了所有 self edge，但由於有一個 vertex 所有的邊就是一條 self edge，因此最後實際上只有讀入 5241 個頂點。

2. Implementation

我大致上依照以下的虛擬碼實作 Approximate PPR 演算法：



▲圖一、Approximate 演算法的虛擬碼

其中d_u代表點u的 degree，r_u為點u的 PPR 值，q_u為點 u 的 residue 值，r與q

分別為所有頂點的 PPR 值或 residue 值所形成的向量。而β值設為 0.8，S 為 seed node，ε則為給定的誤差值。

我使用 Python 與 pyspark 模組，並配合 map reduce 的方法進行實作。關於各個 mapper 與 reducer 的詳細設計細節，以及程式碼的解說，我都寫在 ipynb 檔案的 markdown 裡面。首先我將 dataset 讀入一個 RDD 中，接著經由一系列 map-reduce，將各點初始d_u、r_u、q_u以及其所連到的各個 vertex 存入該 RDD 內。接著挑選 1 個 seed node 後實作 PPR 演算法。

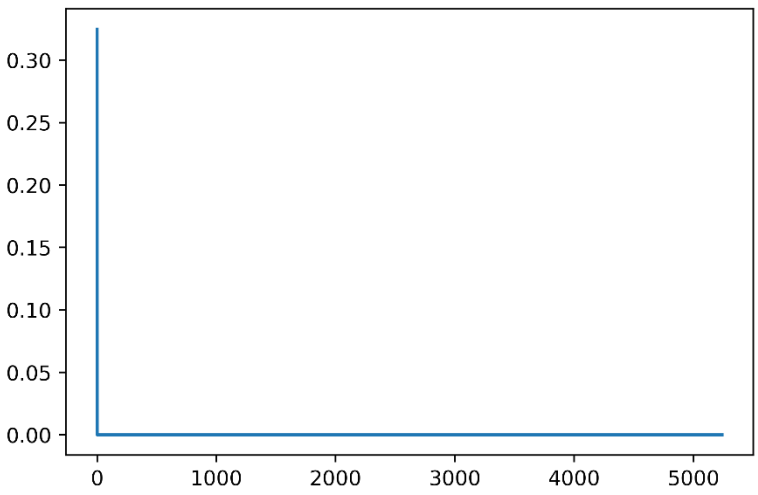
一開始，我的確是完全按照虛擬碼執行，不過可能出於我電腦效能實在太差的緣故，跑了數次迴圈便會出現 timeout 的錯誤，無法達成讓所有q_u/d_u都變成小於ε的值。經過反覆驗證後發現，問題出在我每次尋找滿足

$$\max_{u \in V} \frac{q_u}{d_u}$$

的頂點u時耗費太多運算資源，我進一步發現前三次迴圈都能執行的相當快速，但第 4 次迴圈就會跑很久，而第 5 次迴圈就會 timeout。因此只好改採折衷方式，只跑 4 次迴圈。程式的運行結果會一個資料夾，裡面的 part-00000、part-00001、part-00002、part-00003 由手動改為.txt 檔後就是每個點與所對應的 PPR 值。

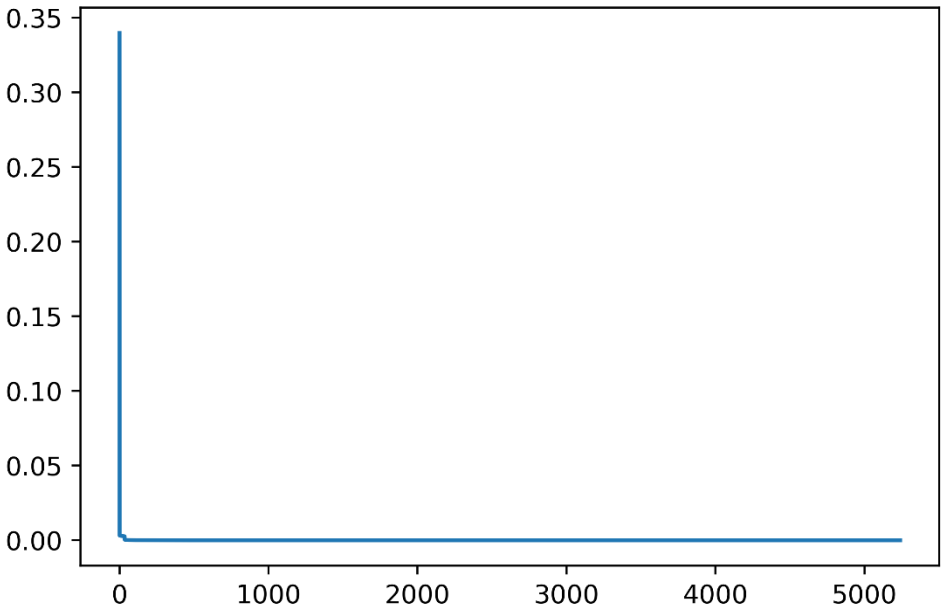
3. Discussion

只跑 4 次迴圈明顯是不夠的，將排序後的 PPR 值對排序名次畫出來的圖形會長成這樣：

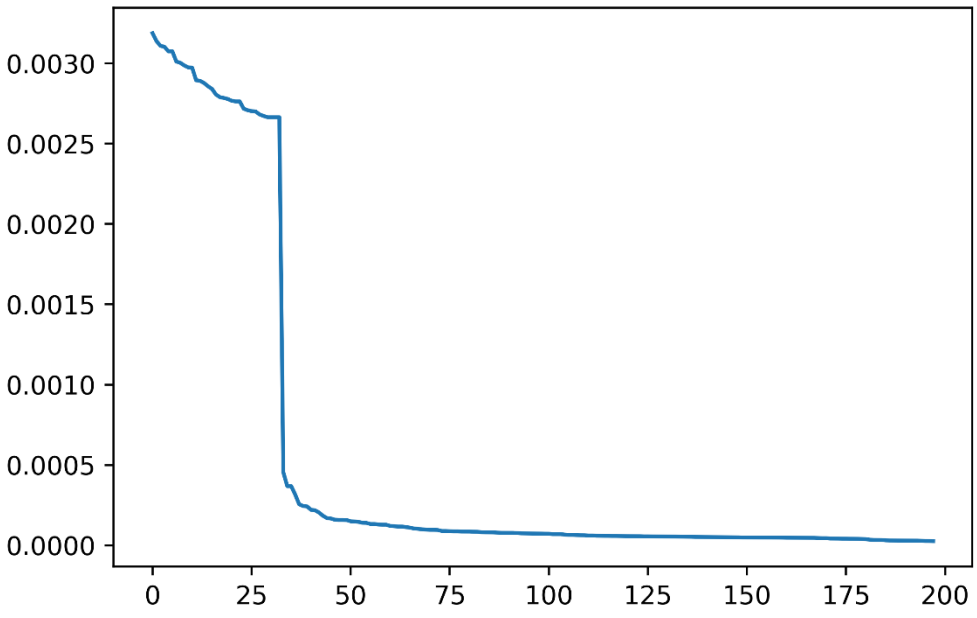


▲圖二、PPR 值仍集中在 seed node 上

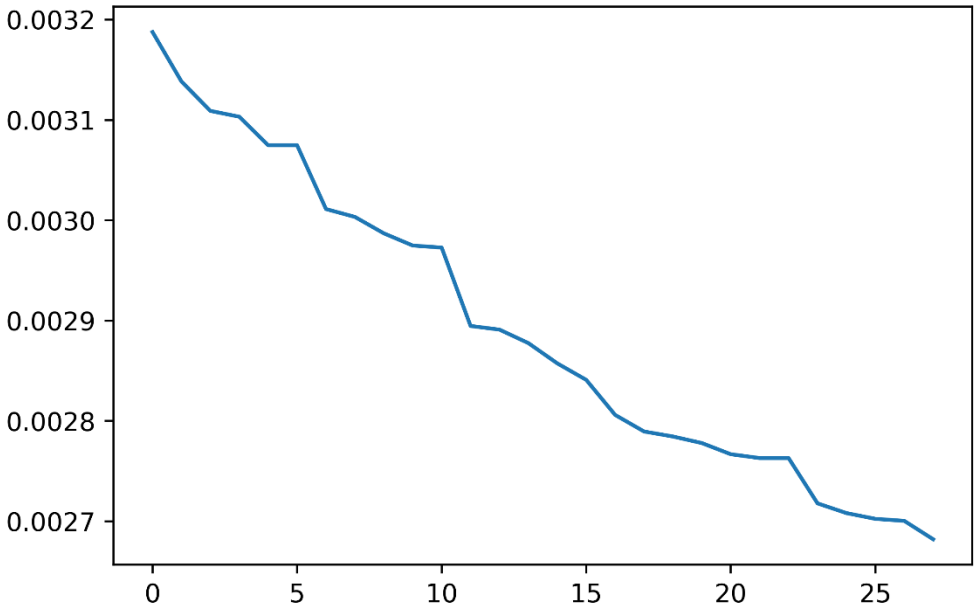
也就是 PPR score 仍集中在 seed node 上面，沒有擴散出去。因此我用另一個 package networkX 重新實作了 PPR，輸出後在 google colab 上做圖。設定 seed node 為 9572，β值為 0.8，ε為10^{-7}，得出各點 PPR 值後將其排序，以橫軸為名次，縱軸為 PPR 值做圖，得出以下結果：



▲圖三、所有 vertex 的 PPR 排序

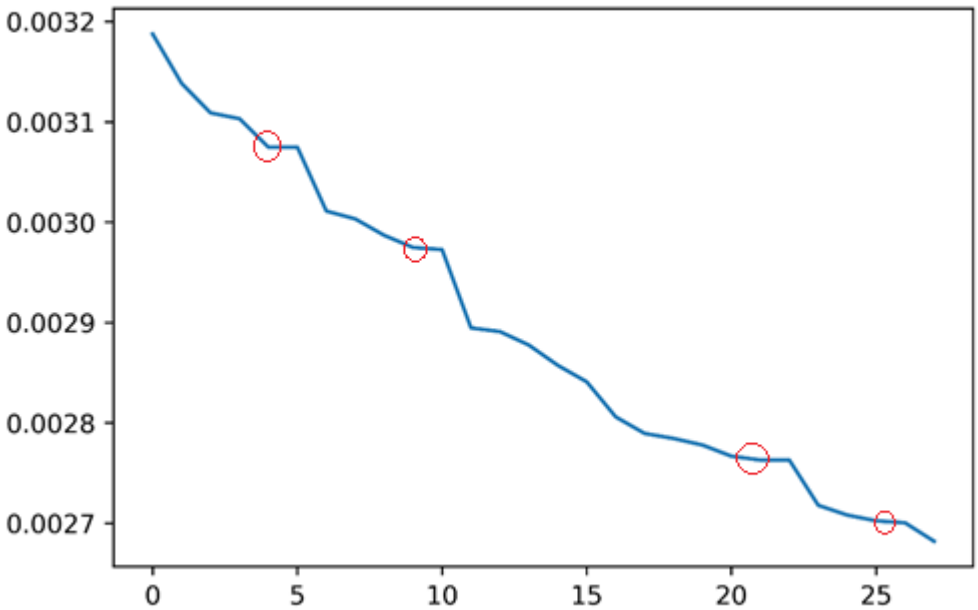


▲圖四、第 2 到第 200 名的 vertex，其 PPR 排序



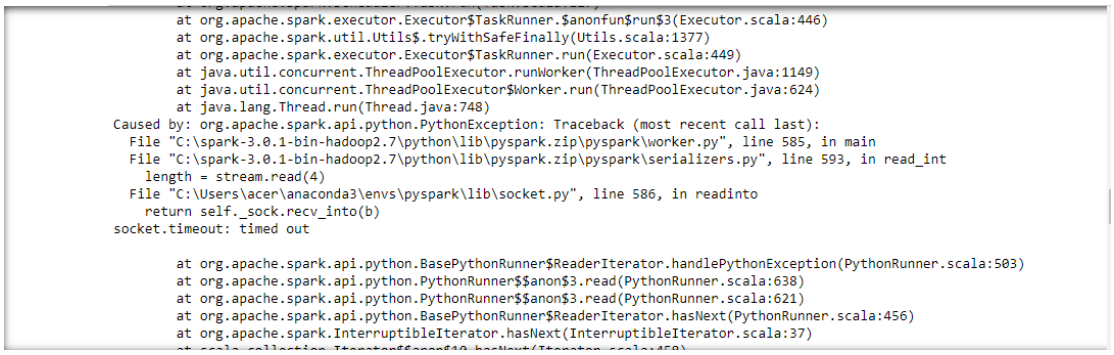
▲圖五、第 2 到第 30 名的 vertex，其 PPR 排序

由上圖可知，雖然大部分的 PPR 值仍集中在 seed node 上，但是藉由 lazy random walk，已經將部分的 PPR 分數給傳遞出去。接著，藉由觀察圖五中的 local minima 數量，我們可以發現這 30 個點大致可分為 4 群，如下圖所示：



▲圖六、local minima 數決定分群之數量

而關於為何 pyspark 無法正常運行的問題，我曾經想過是否可能是 dataset 太大的所致，因此我下載了另一個非常小的 dataset，例如 dolphins，[3]總共只有 62 個頂點與 159 個邊，但是也發現同樣的問題，即迴圈最多只能執行 4 次後就會出現 timeout 的錯誤。



▲圖七、timeout 錯誤

我另外嘗試了不同的解決方法：換到 google colab 上運行，也會出現同樣的問題，另外也在網路上搜尋相關解決方案，嘗試了增加運行的 cpu 數量以及 sparkconf 的 heartbeat 時間，但都無法解決我遇到的問題。我最後嘗試把迴圈內的 code 提出來一行一行運行，仍會發生一樣的問題。不過，若從逐行運行的結果顯示出來，仍可以發現我還是有正確地做出了 Approximate PPR 演算法，不過就是沒有辦法跑出一個可接受的結果。

4. Conclusion

簡單講解一下各個檔案是什麼：

檔案	說明
CA-GrQc.txt	dataset
Term_Project_Group33.ipynb	利用 pyspark 實作 PPR 演算法的程式
PPR_use_networkx.ipynb	利用 networkX 實作 PPR 演算法
plot.ipynb	繪製圖三～圖五的程式，於 google colab 運行（因為我的電腦跑不動）
part-00000.txt	Term_Project_Group33.ipynb 的輸出
part-00001.txt	
part-00002.txt	
part-00003.txt	

output.txt	PPR_use_networkx.ipynb 的輸出
Term_Project_Group33.pdf	本報告

▲表一、各個檔案的內容

5. References

[1] General Relativity and Quantum Cosmology collaboration network https://snap.stanford.edu/data/ca-GrQc.html

[2] J. Leskovec, J. Kleinberg and C. Faloutsos. Graph Evolution: Densification and Shrinking Diameters. *ACM Transactions on Knowledge Discovery from Data (ACM TKDD)*, 1(1), 2007.

[3] Dolphins, http://networkrepository.com/soc-dolphins.php