Massive Graph Management and Analytics (MGMA)
Delivery 4

Alaa Almutawa and Rishika Gupta
4th March 2024

Github repo: link

Community detection and Influence Maximization

Datasets: We have selected two datasets from SNAP.

- /data/email-Eu-core.txt (source)
- /data/wiki-Vote.txt (source)
- /data/web-Google.txt* (source) (We have looked into this but unfortunately, we were not able to run it fully)

Challenges:

1. The size of the network and limited computational resources.
   ⇒ Solution was to pick graphs that are less than 10k nodes. To enable us to run experiments with the number of seeds k > 2 and to also run greedy approach.
2. Using web-Google.txt (source) from SNAP was our original plan, however, due to the size of the network and our desire to interpret the results we could not use it
   ⇒ Initially we wanted to reduce the graph size, however, we have observed that once the graph is reduced in size, the results and their interpretation is hard to formulate and understand, especially without any domain experts.

Community detection

Heuristics:

- Louvian
- Girvan* (We have looked into this but unfortunately, we were not able to run it fully)

Machine Learning based:

- Spectral Clustering(with number of communities k=10)

Metrics:

- Modularity

Script:

- community_detection.py

	Louvain	Spectral Clustering
email-Eu-core.txt	0.430968	1.0
wiki-Vote	0.4299802861418711	

When comparing modularity results, Louvain approach scores lower than spectral clustering method. Indicating that machine learning based method

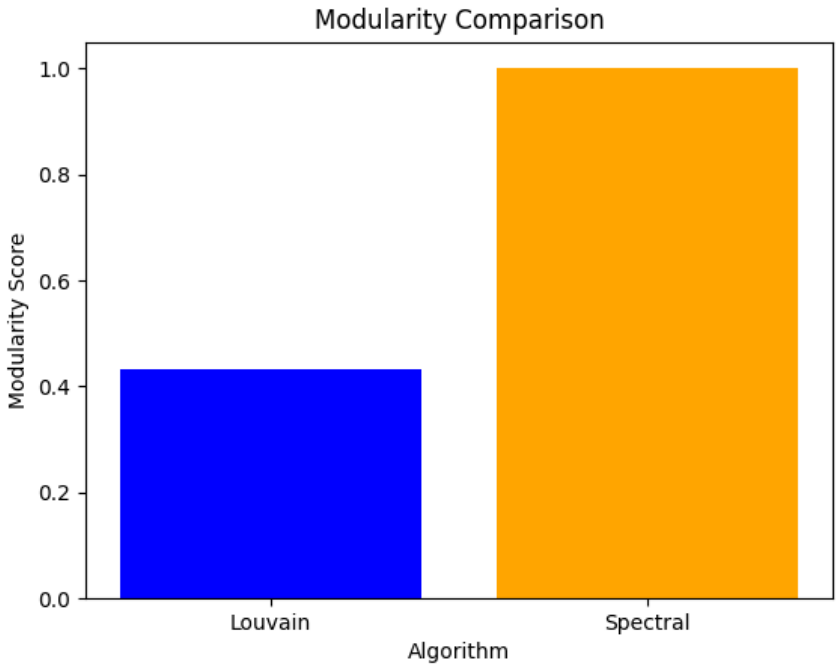
Interpretation:

email-Eu-core:

Louvain communities: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27}

Spectral communities: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

This graph represents email communication between departments in an EU institution. Thus, the communities can detected through community detection algorithm can be interpreted as employees belonging to the same departments or employees that communicate regularly together (as they might have interdependent tasks)

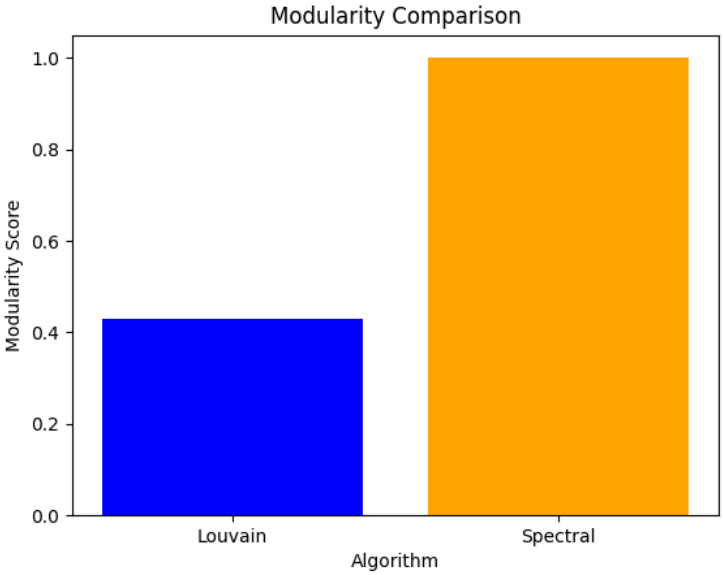


wiki-Vote.txt:

Louvain communities: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29}

Spectral communities: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

This graph represents the admin voting on other users for their adminship requests on wikipedia. Running a community detection algorithm on this graph results in identifying the admins that are closely connected to each other and have similar voting patterns (vote on similar individuals). Thus, the communities represent that voter group.



Output can be found in: /results/{dataset}/community_detection

Influence Maximization

Spread process:

- Independent Cascades

Algorithms:

- CELF (paper)
- Greedy (paper)

Experiment 1:

Goal: Study the computational time of two different implementations. (reference)

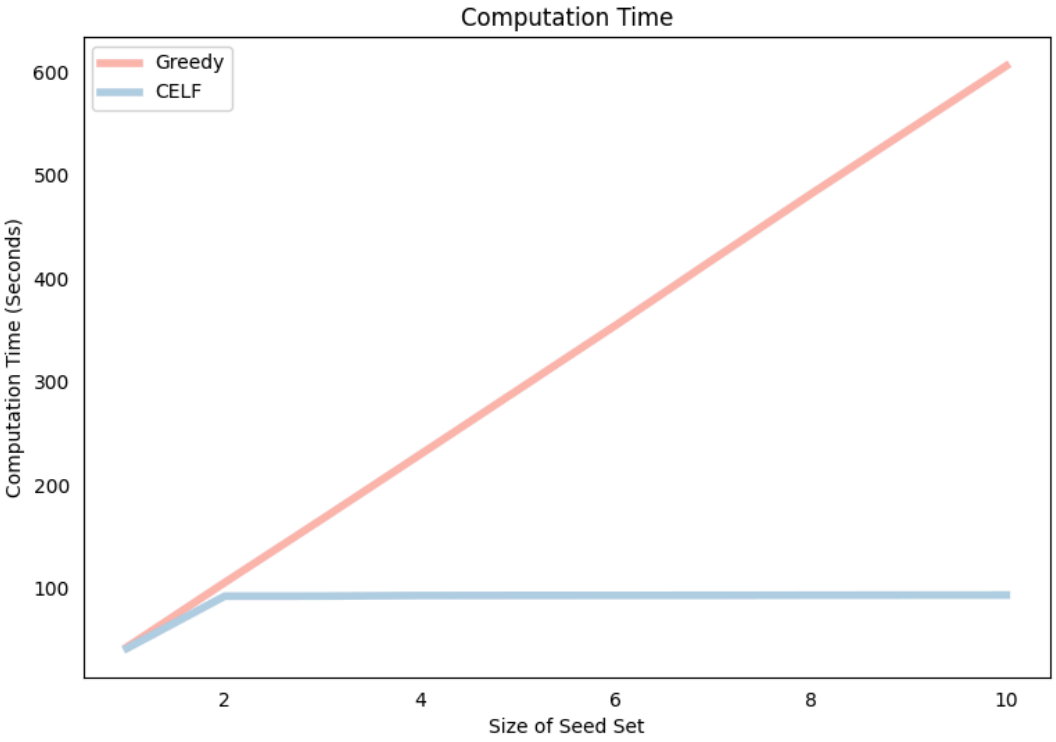
Script: greedy_vs_celf.py

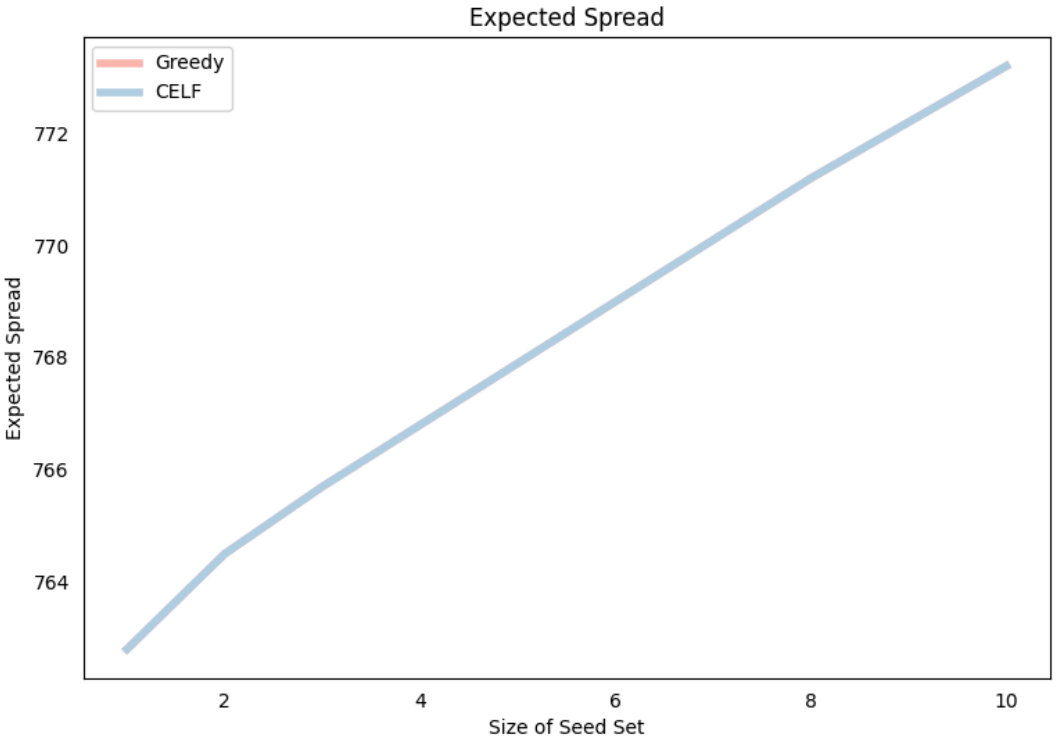
1. Email EU core:

Greedy	Seed Set [2, 5, 979, 7, 435, 524, 634, 995, 414, 475] Influences [762.8, 764.5, 765.7, 766.8, 767.9, 769.0, 770.1, 771.2, 772.2, 773.2]
CELF	Seed Set [2, 5, 979, 7, 635, 435, 524, 995, 634, 846] Influences [762.8, 764.5, 765.7, 766.8, 767.9, 769.0, 770.1, 771.2, 772.2, 773.2]

Interpretation:

The graph represents the email communication between employees in an EU institution. The selected seeds demonstrate the employees within the European institution that have the most influential spread. Thus, these employees (nodes) have the required connection to efficiently spread information across the network making them good candidates to disseminate information across the network.


Computation Time
Greedy
CELF
600
500
400
300
200
100
Computation Time (Seconds)
2
4
6
8
10
Size of Seed Set

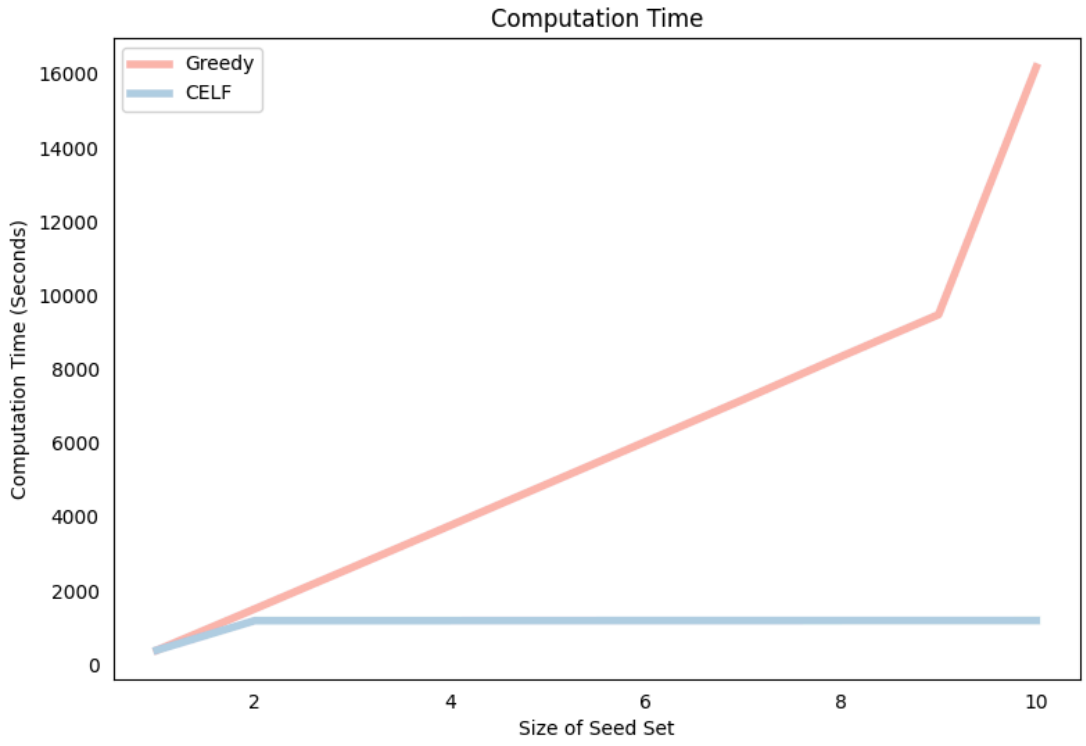

Expected Spread
Greedy
CELF
772
770
768
766
764
Expected Spread
2
4
6
8
10
Size of Seed Set

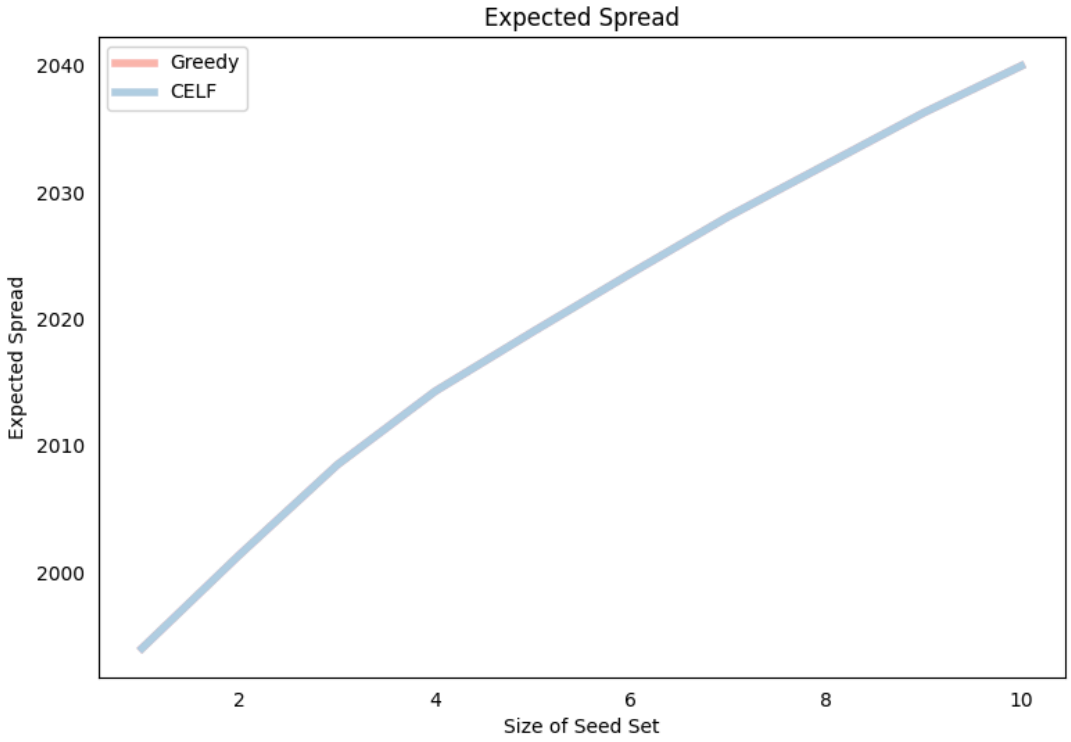
2. Wiki-vote

Greedy	Seed Set [408, 1093, 2013, 29, 1853, 5201, 892, 915, 3714, 958] Influences [1994.0, 2001.4, 2008.5, 2014.3, 2019.0, 2023.6, 2028.1, 2032.2, 2036.3, 2040.0]
CELF	Seet Set [408, 1093, 2013, 29, 1853, 5201, 892, 915, 3714, 958] Influences [1994.0, 2001.4, 2008.5, 2014.3, 2019.0, 2023.6, 2028.1, 2032.2, 2036.3, 2040.0]

Interpretation:

The graph data represents the admin votes on other users to receive adminship rights. Thus, in the context of this graph data, the influential seed represents the admin users that are participating highly in the selection of other admins. Thus, these users (nodes) can be seen as a good target to sway election results as they have influential power in the election.





Conclusions:

From these experiments, we can conclude:

1. CELF approach is more computationally friendly and achieves results that are close to greedy approach.
2. Initial selected seed have the biggest spread influence. The seeds that are selected later have marginal increase on the spread
3. The context of the data that we use to run the algorithm determines the interpretation of the output.

Experiment 2:

Goal: Compare different seed selection methods to greedy approaches.(reference)

Spread process:

- Independent Cascades

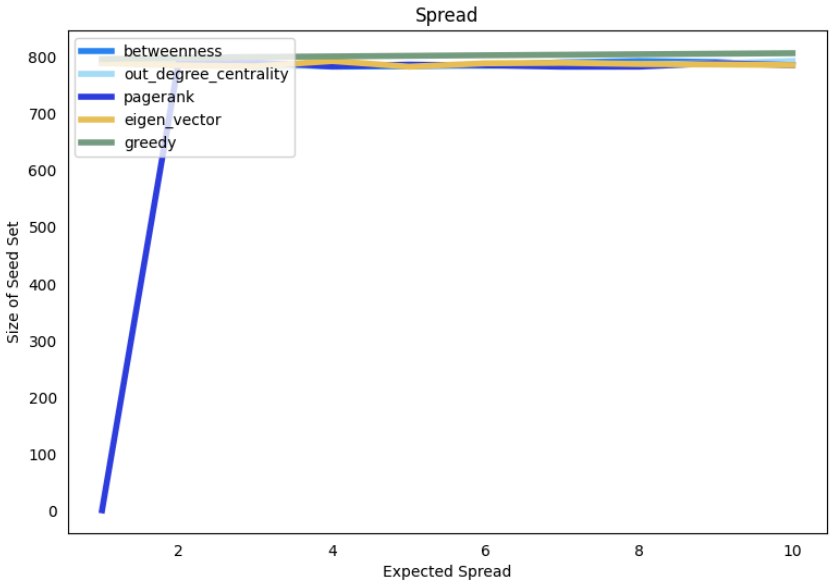
Algorithms:

- Greedy (paper)

- Other seed selection: Betweenness, Eigenvector, Pagerank, Outdegree using Networkx implementation

Script: greedy_vs_centrality.py

1. Email EU core:



Conclusion:

- Using centrality measures can help decide on the influential nodes for some network topologies. In our example, we can observe that outdegree centrality achieves almost the same spread as a greedy approach without requiring the extensive computational resources.

Output can be found in: /results/{dataset}/influence_max