Esercizio 1

Sono stati implementati le seguenti misure di centralità:

- Shapley_degree()
- Shapley_threshold(k)
- Shapley_closeness()

Le implementazioni sono state realizzate con il supporto del paper *Shapley Centrality* che fa parte del materiale didattico.

L'implementazione originale della formula dello Shapley Value è proibitiva per reti di grandi dimensioni in quanto la sua complessità computazionale è O(2^|V|) dove V è la cardinalità dei nodi del grafo.

Lo scopo era realizzare un algoritmo che richiedesse complessità computazionale polinomiale.

L'idea è: dato un gruppo di nodi C, la funzione che definisce il valore di C nel gioco deve in qualche modo quantificare la sfera di influenza di C su tutti gli altri nodi della rete.

Game	Graph	v(C)	Complexity	Accuracy
g1	UW	V(C) è il numero di nodi in C e quelli immediatamente raggiungibili da C	O(|V|+|E|)	exact
g2	UW	V(C) è il numero di nodi in C e quelli immediatamente raggiungibili da C, ma attraverso almeno k differenti archi	O(|V|+|E|)	exact
g4	W	V(C) è la somma delle funzioni non crescenti delle distanze tra C e gli altri nodi.	O(|V||E| + |V|^2log|V|)	exact

Shapley_degree

Algorithm 1: Computing the Shapley value for Game 1

Input: Unweighted graph $G(V, E)$
Output: Shapley values of all nodes in $V(G)$ for game g_1

foreach $v \in V(G)$ **do**
  $\mathrm{SV}[v] = \frac{1}{1+deg_G(v)}$;
  foreach $u \in N_G(v)$ **do**
    $\mathrm{SV}[v]$ += $\frac{1}{1+deg_G(u)}$;
  end
end
return SV;

Ciò deriva dall'intuizione che un nodo può avere alta centralità non solo se il suo grado è alto, ma anche quando il suo grado tende ad essere più alto rispetto al grado dei nodi vicini.

Shapley_threshold(k)

Algorithm 2: Computing the Shapley value for Game 2

Input: Unweighted graph $G(V, E)$, positive integer k
Output: Shapley value of all nodes in $V(G)$ for game g_2

foreach $v \in V(G)$ **do**
  $\mathrm{SV}[v] = \min(1, \frac{k}{1+deg_G(v)})$;
  foreach $u \in N_G(v)$ **do**
    $\mathrm{SV}[v]$ += $\max(0, \frac{deg_G(u)-k+1}{deg_G(u)(1+deg_G(u))})$;
  end
end
return SV;

In questo modello ogni nodo diventa attivo se una funzione d'attivazione monotona raggiunge la threshold.

Quando k = 1 abbiamo la shapley_degree.

Shapley_closeness()

Algorithm 4: Computing the Shapley value for Game 4

Input: Weighted graph $G(V, E, W)$, function $f : \mathbb{R}^+ \to \mathbb{R}^+$
Output: Shapley value of all nodes in G for game g_4

Initialise: $\forall v \in V(G)$ **set** SV[v] = 0;
foreach $v \in V(G)$ **do**
    [Distances D, Nodes w] = Dijkstra(v,G);
    sum = 0; index = |V|-1; prevDistance = -1, prevSV = -1;
    while *index* > *0* **do**
        if *D(index)* == *prevDistance* **then**
            currSV = prevSV;
        else
            currSV = $\frac{f(D(index))}{1+index} - sum$;
        end
        SV[w(index)] += currSV;
        sum += $\frac{f(D(index))}{index(1+index)}$;
        prevDistance = D(index), prevSV = currSV;
        index--;
    end
    SV[v] += f(0) − sum;
end
return SV;

Ci si aspetta che un agente ad una certa distanza d da una coalizione contribuisca secondo un valore f(d) dipendente dalla distanza, dove f(.) è una funzione decrescente positiva.

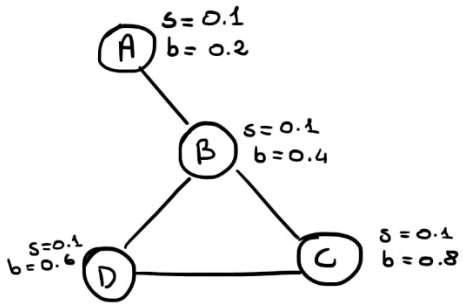
Gli algoritmi sono stati testati sulla rete Musae_Facebook_edges (22470 nodi e 341825 archi) e l'esecuzione ha richiesto pochi secondi per la shapley_degree e shapley_threshold, mentre pochi minuti per la shapley_closeness.

Friedkin-Johnsen (FJ) dynamics

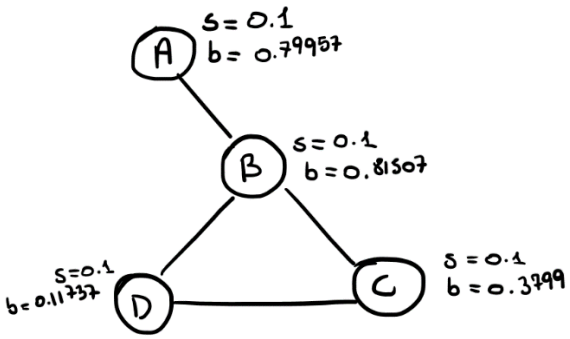
L'algoritmo è stato implementato secondo le specifiche riportate nella traccia.

Si è sperimentato che, a seconda del tipo di grafo, la dinamica non sempre converge. In particolare, si sono analizzate queste due specifiche situazioni:

G1



G2



Il grafo G1 converge ad uno stato stabile dopo circa venti iterazioni.

Il grafo G2 non converge, in quanto i nodi si influenzano all'infinito a vicenda.

Si è osservato che la dinamica FJ non converge quando nel grafo vi è una forte presenza di triangoli e/o la *stubbornness* è molto bassa, il che comporta una forte oscillazione sulla *belief* dei nodi.

Esercizio 2

Scopo di questo esercizio è di capire con che tipo di modello, la *net_7*, fosse stata generata.

Analisi net_7

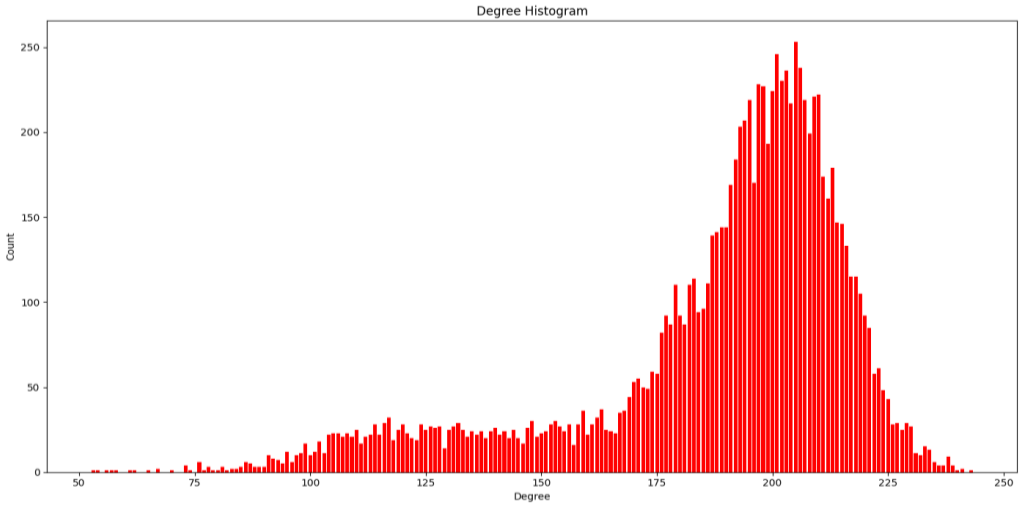


Figura 1

Type	Graph
Number of nodes	10.000
Number of edges	939129
Average degree	187.8258
Average clustering	0.6126402299894962
Number of bridges	0
Number of found component	1
Average shortest path length	6.069738073807381
Diameter	14

Random model

Una prima analisi è stata fatta generando dei grafi utilizzando il modello random. I motivi che ci hanno portato ad escludere questo modello sono:

- Il diametro ottenuto con il modello random dovrebbe essere nell'ordine di *log(n)* che, nel nostro caso per n = 10000 corrisponde a d = 9.21.
- La distribuzione dei gradi dei nodi, con questo modello, dovrebbe essere uniforme.
- I coefficienti di clustering calcolati sul grafo prodotto con questo modello sono considerevolmente minori rispetto al grafo da analizzare.

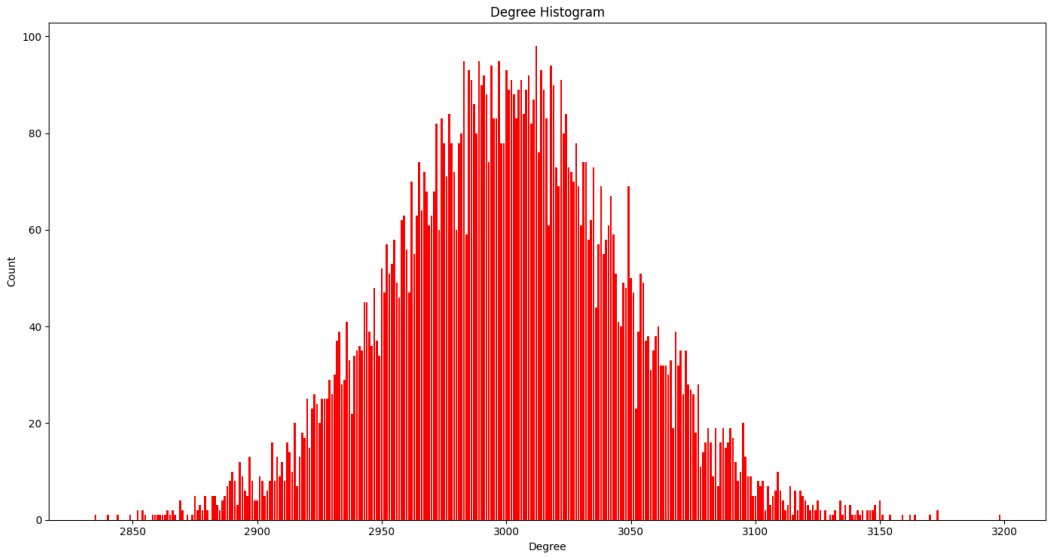
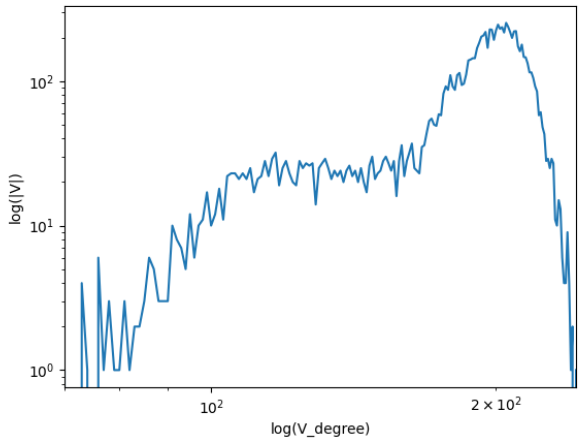


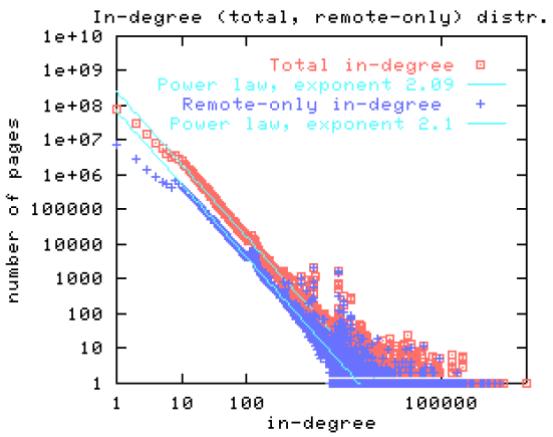
Figura 2

Configurational model

Anche questo modello, come il precedente, è stato escluso per il seguente motivo:

- Tale modello, per avere un coefficiente di clustering simile al grafo oggetto di analisi prevede che i gradi dei nodi siano distribuiti secondo una *Power Law,* quindi, in contrasto con la distribuzione *normale* dei gradi dei nodi della *net_7*.
- Se la rete fosse stata prodotta secondo questo modello e seguendo una legge di distribuzione dei gradi dei nodi Power Law, nel plot Log Log avremmo osservato una retta con pendenza negativa.





Preferential Attachment model e Affiliation model

Entrambi i modelli sono stati scartati per il seguente motivo:

- Questa tipologia di algoritmo genera dei grafi con distribuzione dei gradi dei nodi che segue una Power Law.

Watts-Strogatz

La nostra scelta è ricaduta su questa implementazione per i seguenti motivi:

- Una delle proprietà di WS è che dati N nodi e un grado medio K, il numero di archi è pari a NK/2:
  10000*187.8258/2 = 939129
- La degree distribution segue una Dirac Delta Function centrata in K e com'è osservabile dalla figura 2 la proprietà è rispettata.

Analisi Parametri

I parametri da ricercare erano i seguenti:

- R: il raggio di ogni nodo (un nodo u è connesso con ogni altro nodo a distanza al più r) – strong ties
- K: è il numero di random edges per ogni nodo u – weak ties
- Q: è un termine per valutare l'importanza della distanza, in particolare la probabilità di un edge tra u e v è proporzionale a 1/dist(u,v)**q

Basandoci sulla definizione stessa di raggio, abbiamo effettuato i primi esperimenti con valori interi che fossero attorno alla metà del diametro della rete in analisi.

1. **R = 7; K = 0; Q = 0**
   Number of nodes: 10000

Number of edges	726350
Average degree	145.27

   Come possiamo notare il numero di edges è molto minore del numero di edges della rete in analisi (circa 210.000 edges)

2. **R = 7; K = 21; Q = 1**
   Number of nodes: 10000

Number of edges	904944
Average degree	180.9888
Average clustering	0.4015283887126754
Number of bridges	0
Number of found component	1
Average shortest path length	2.33
Diameter	3

   Come possiamo notare dai dati di Average Shortest Path Length e Diameter, nonostante avessimo un numero di edges simile a quello della rete originale, un K molto elevato diminuisce parecchio il diametro e l'average shortest path

length. Per cui si è aumentato il raggio a 8.

3. **R = 8; K = 0; Q = 0**

   Number of nodes: 10000

Number of edges	935621
Average degree	187.1242
Average clustering	0.6127206837949042
Number of bridges	0
Number of found component	1
Diameter	19

   Con raggio pari a 8 si è ottenuto un numero di edges paragonabile a quello della rete in esame, così come per gli altri parametri. L'unica differenza si è riscontrata nel diametro.
   Questa grossa disparità tra i diametri ci ha portato ad utilizzare anche i parametri K e Q, in modo da ottenere dei weak ties che possano permettere ad un dato nodo di poter fare un salto considerevole invece di considerare solo i propri vicini, questa operazione dovrebbe contribuire ad abbassare il diametro, poiché in media i nodi target saranno raggiungibili in un numero minore di hop.
   Questa grossa disparità tra i diametri ci ha portato ad utilizzare anche i parametri K e Q, in modo da ottenere dei weak ties che possano permettere ad un dato nodo di poter fare un salto considerevole invece di considerare solo i propri vicini. Questa operazione dovrebbe contribuire ad abbassare il diametro, poiché in media i nodi target saranno raggiungibili in un numero minore di hop.

4. **R = 8; K = 1; Q = 3.3**

   Number of nodes: 10000

Number of edges	938708
Average degree	187.7416
Average clustering	0.6142751116260949
Number of bridges	0
Number of found component	1
Average shortest path length	5.969431623162317
Diameter	14

   All'aumentare di K, si è provato che il diametro diminuisse molto velocemente, (es. K = 2 à diametro = 5) per cui la scelta è ricaduta su 1.
   Dopo vari esperimenti con K fissato a 1 si è giunti alla scelta di Q pari a 3.3, ottenendo dei risultati molto vicini a quelli della rete originale, come riportato in figura 3.

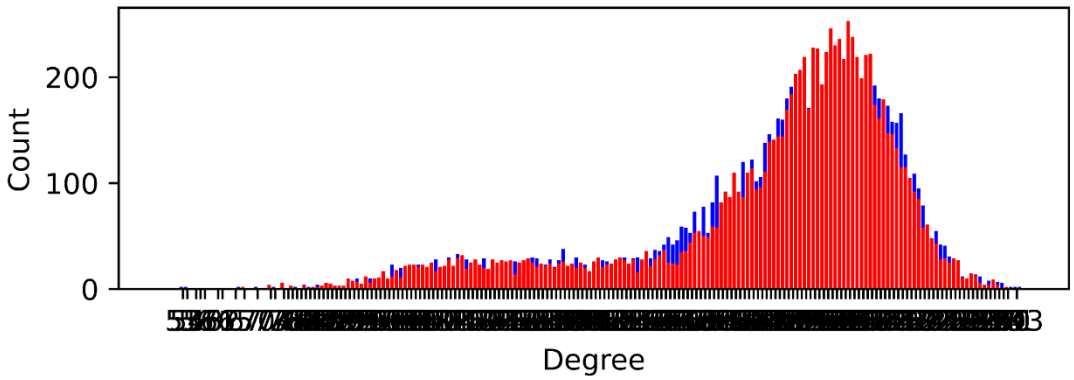
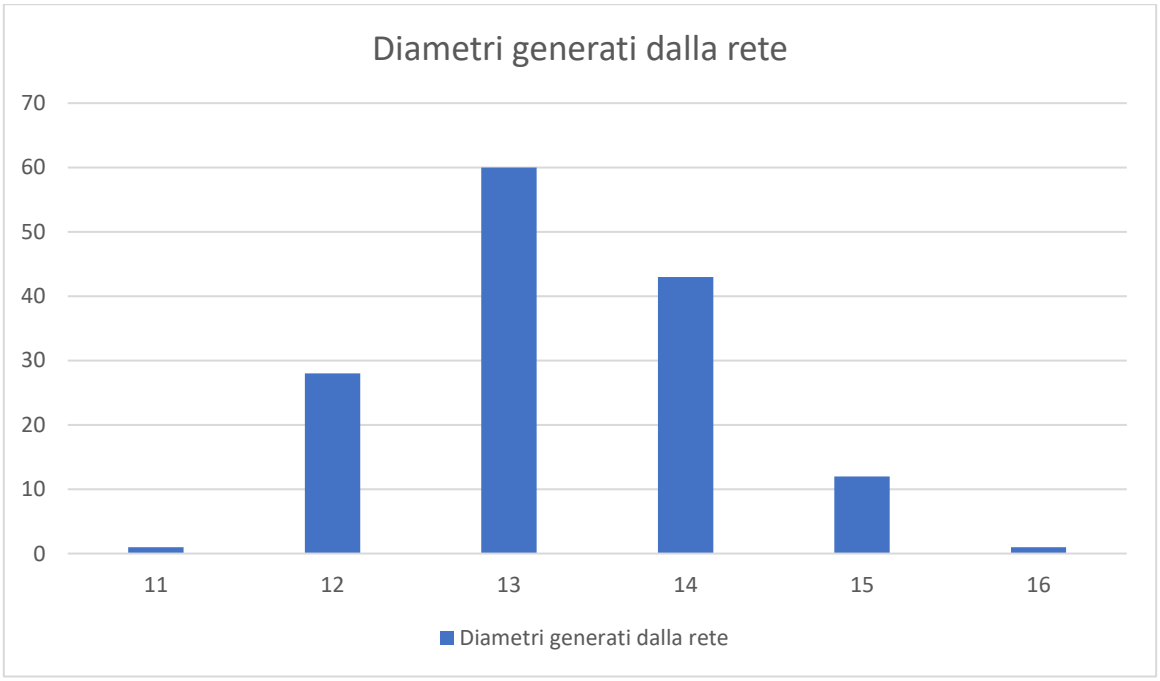


Figura 3

Abbiamo sperimentato che il diametro delle reti generate con questo set di parametri, data la componente randomica dell'algoritmo, si concentrano in un intorno [12; 15] con alcuni casi isolati che hanno visto un diametro di 11 o 16. Come evidenziato dal grafico sottostante sono presenti molte occorrenze anche di 12,13,15. Ciò è facilmente spiegabile col fatto che una sola coppia di nodi può cambiare il diametro.
L'Average Shortest Path Length, invece, si concentra in un intorno [5.55; 6.1] quando il numero di edges (differenza attorno a 500) e l'average degree sono molto simili a quelli della rete originale.



In conclusione, si è osservato, che in media i valori dei parametri ottenuti sono molto più simili alla rete originale quando il diametro ottenuto è pari a 14, mentre, per gli altri valori di diametro ottenuti, più frequentemente, si è notata una certa disparità rispetto ai valori dei parametri del grafo originale.

Esercizio 3

In questo esercizio è stato richiesto di simulare un'elezione i cui votanti sono connessi attraverso un grafo.

Si suppone avere:

- n votanti, rappresentati dai nodi del grafo
- m candidati
- p_i *in* $[0, 1]$, rappresenta la tendenza politica dell'i-esimo candidato
- b_u, rappresenta il picco di preferenza del votante u

L'obiettivo è di simulare l'elezione in accordo alla regola di *plurality voting* e di manipolare ogni votante u affinché i voti per un certo candidato c siano massimizzati rispetto alle preferenze ottenute dalla b_u iniziale.

La opinioni dei votanti sono diffuse attraverso la rete secondo la dinamica *FJ* indicata in traccia.

Si è implementato un algoritmo che prende in input i seguenti parametri:

- grafo G
- un set di candidati con la loro posizione $(p1, ..., p_m)$
- uno speciale candidato c
- un budget B
- un set iniziale di beliefs di tutti i votanti $(b1, ..., b_n)$

Suddetta implementazione restituisce come output:

- un set S di al più B seed
- un set di $\mathbf{b'_u}$ per ogni seed u in S, tale che la differenza tra il numero di voti ottenuti per il candidato c nell'elezione manipolata e l'elezione truthful è massimizzata.

Implementazioni progettate

Greedy con marginal influence
Considerando un grafo con un numero di nodi di circa 20k, il tempo computazionale richiesto da questo tipo di implementazione è oneroso. Inoltre, si è provveduto a realizzare una implementazione *custom* per questo tipo di approccio che comunque non ha risolto lo svantaggio legato ai tempi di computazione.
La maggiore differenza tra le due versioni è che la prima doveva computare 20k*budget operazioni, in quanto, ad ogni iterazione era calcolata nuovamente l'influenza del nodo, a seconda del set di seed utilizzato. La seconda, invece, computava

solo 20k operazioni e considerare solo i budget nodi che avessero influenza maggiore sull'elezione del candidato.
L'implementazione *custom* prevedeva l'utilizzo di una *priority queue*, la cui priorità era data dal numero di voti ricevuti dal candidato, utilizzando il nodo come manipolatore.

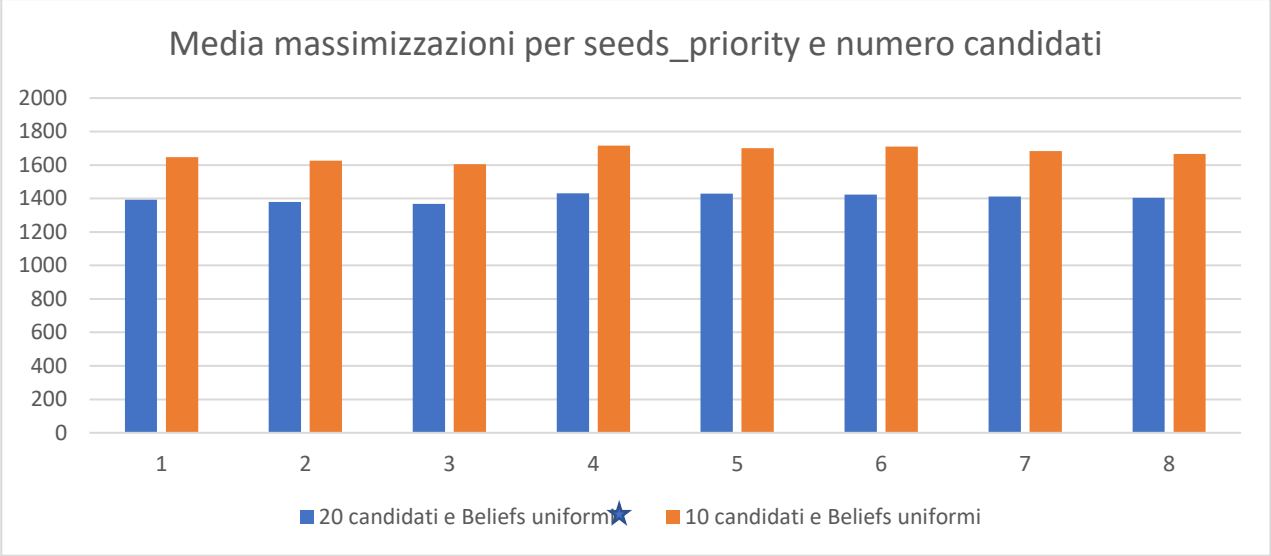
Page Rank - Shapley degree - Shapley closeness – Shapley Threshold
Ognuno di questi algoritmi è stato eseguito su un grafo di circa 20k nodi. Per ognuno di essi sono stati estratti i primi budget nodi che avessero *value* maggiore (es. nel caso di Page Rank i primi budget nodi con page rank maggiore).
Dopo vari test si è scelto di utilizzare l'algoritmo di Shapley degree, in quanto influiva maggiormente sulle votazioni ricevute dal candidato.
Questo tipo di implementazione dipende unicamente dalla velocità degli algoritmi che in questo caso sono molto performanti (qualche secondo di esecuzione).
Calcolo Influenza
Dato un set di candidati m, le prove sono state eseguite scegliendo ognuno dei candidati come candidato per cui massimizzare le preferenze e infine facendo una media della differenza dei voti ricevuti prima e dopo la manipolazione. Ogni prova è indipendente dalle altre.
Si sono effettuati test con vari set di candidati e si è sperimentato che i candidati più vicini allo 0 e all'1 fossero quelli che ottenevano minore massimizzazione, dato il funzionamento della dinamica FJ che tende a concentrare le preferenze attorno alla media aritmetica delle opinioni dei votanti. Infatti, si sono riscontrati degli scenari in cui alcuni candidati ottengono un numero di voti minore rispetto ai voti prima della manipolazione ottenendo quindi una differenza negativa. In particolare, i candidati affetti da una massimizzazione negativa sono i candidati il cui orientamento politico è molto vicino allo 0 o all'1.

Scelta dei seeds
Si è effettuato un ordinamento dei seeds secondo la misura di centralità Shapley Degree, nel codice si vedrà l'utilizzo di Shapley Threshold con $k = 1$.
Si è però scelto di utilizzare un approccio *ibrido*:

- seeds_priority: priorità del seed
- shapley_value: grado shapley del nodo
- beliefs: set di beliefs degli elettori
- v: elettore
- c: orientamento politico del candidato per cui massimizzare le preferenze

seeds_priority: shapley_value*0.2 + abs(beliefs[v]-c)

Lo scopo è considerare anche la distanza di orientamento politico dell'elettore per cui forzare opinion e stubbornness, coprendo quindi una parte di elettorato distante dalla politica del candidato che altrimenti sarebbe difficilmente influenzabile in quanto più lontano dal pensiero politico di c.
Si è scelto, poi, di considerare una certa percentuale di shapley_value (sperimentalmente del 20%) in quanto si è riscontrato che il candidato, in questo modo, ottenesse maggiori preferenze rispetto, ad esempio, alla sola somma dei due criteri.
Questo criterio non esclude però che nel set di seeds siano presenti nodi le cui preferenze siano già per il candidato da votare, per cui si è scelto di escluderli laddove non fossero necessari al raggiungimento del numero massimo di seeds.
Qui di seguito sono riportati i risultati di alcune prove in riferimento alla scelta del seeds_priority.



1	shapley_value + abs(beliefs[v]-c)
2	shapley_value + abs(beliefs[v]-c)*0.5
3	shapley_value
4	shapley_value*0.2 + abs(beliefs[v]-c)
5	shapley_value*0.15 + abs(beliefs[v]-c)
6	shapley_value*0.3 + abs(beliefs[v]-c)
7	shapley_value*0.4 + abs(beliefs[v]-c)
8	shapley_value*0.5 + abs(beliefs[v]-c)