Reti Sociali
Majority cascade in cost networks

Autore Gagliarde Nicolapio
Anno Accademico 2023/2024
Corso di Reti Sociali

1 Introduzione

1.1 Descrizione del problema

Nell'ambito delle reti sociali, il concetto di **cascade** permette lo studio di numerosi fenomeni sociali tra cui la **Influence Diffusion**. Nel campo della Influence Diffusion troviamo il problema del **Majority dynamical process**. Ricordando che con $N(v)$ si indica l'insieme degli adiacenti di v in G (dove G è un grafo) e con $d(v)$ il grado di v, cioè $d(v) = |N(v)|$, il Majority dynamical process di Influence Diffusion su un grafo $G = (V, E)$ e $S \subseteq V$ (sottoinsieme di nodi chiamato "seed set") è definito come una sequenza di sottinsiemi:

$$Inf[S, 0], Inf[S, 1], ..., Inf[S, r], ... \subseteq V$$

dove $Inf[S, 0] = S$ e $Inf[S, r] = Inf[S, r-1] \cup \{v \in V - Inf[S, r-1] : |N(v) \cap Inf[S, r-1]| \geq \lceil d(v)/2 \rceil\}$.
Il fenomeno cascade di arresta quando $Inf[S, t] = Inf[S, t+1]$, dove t è un istante di tempo.

Inoltre, nelle reti con costo, viene definita una funzione $c : V \to N$ che per ogni nodo definisce il costo di attivazione di tale nodo. Quindi il costo del seed set S è definito con $c(S) = \sum_{u \in S} c(u)$.

Infine viene stabilito un budget k che limita il costo di S, cioè $c(S) \leq k$.
L'obiettivo di tale problema è determinare il seed set S con $c(S) \leq k$ che massimizzi $|Inf[S]|$, tuttavia il problema risulta essere NP-hard. Di conseguenza in questo documento vengono analizzati algoritmi in grado di determinare un seed set S massimale tale che $c(S) \leq k$ e verrà valutato il numero di vertici influenzati a partire da tale seed set, cioè $|Inf[S]|$.

1.2 Struttura del documento

Nel capitolo 2 è presente la descrizione della rete sociale su cui sono stati svolti gli esperimenti.
Nel capitolo 3 vengono riportate le funzioni di costo utilizzate, gli algoritmi per l'individuazione del seed set e il processo per il calcolo dell'Influence Diffusion.
Nel capitolo 4 e 5 vengono descritti i risultati ottenuti e le conclusioni.

2 Descrizione della Rete Sociale

Per gli esperimenti condotti in questo progetto si è scelta una rete sociale che descrive un particolare comportamento osservabile in una colonia di formiche. Tale rete è stata realizzata osservando gli scambi di cibo che avvengono tra un esemplare e un altro all'interno della stessa colonia, tale fenomeno prende il nome di "trofallassi". Quindi i nodi della rete rappresentano le formiche, mentre gli archi rappresentano gli scambi di cibo.
La rete è composta da 39 nodi e 330 archi, come osservabile dalla Figura 1 e dalla Tabella 1 ottenute dalla documentazione del dataset, all'interno della rete è presente un elevato numero di triangoli da cui si ricava un coefficiente di clustering pari a 0.45 e un numero medio di triangoli pari a 74. Infatti anche il grado medio di ogni nodo è elevato essendo pari a 16, contro i 35 possibili vicini.

I dati numerici si traducono graficamente in un grafo composto da un unico cluster composto da 35 nodi altamente collegati tra di loro e 4 nodi più esterni con grado 1 o 2.

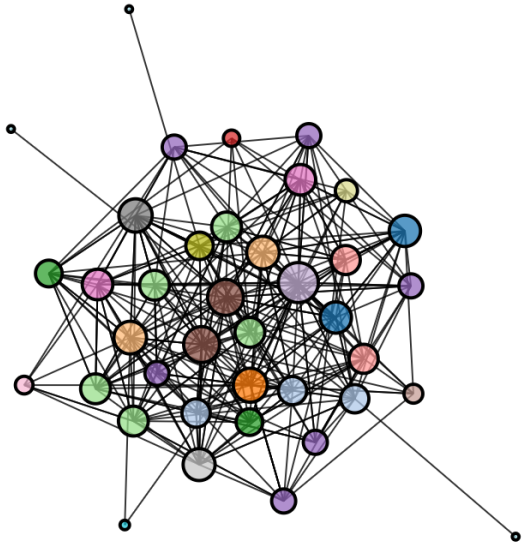

Figure 1: Rappresentazione grafica della rete sociale

Nodi	39
Archi	330
Densità	0,445344
Grado massimo	36
Grado minimo	1
Grado medio	16
Numero di triangoli	2900
Numero medio di triangoli	74
Numero massimo di triangoli	196
Coefficiente di clustering medio	0,4565
Frazione di triangoli chiusi	0,443693
k-score massimo	15

Table 1: Caratteristiche della rete sociale

Il dataset è presente al seguente link: https:networkrepository.cominsecta-ant-trophallaxis-colony2.php

3 Esperimenti e Metodologie

Gli esperimenti condotti prevedono l'applicazione di tre funzioni di costo sulla rete e l'uso di tre diversi algoritmi per l'individuazione dei seed sets. Infine è stata calcolata l'influenza ottenuta partendo dai suddetti seed sets.

3.1 Funzioni di costo

Per il calcolo del costo dei singoli nodi sono state usate tre differenti funzioni:

- La prima associa un valore casuale in un determinato range ad ogni nodo. Negli esperimenti effettuati il range va da 1 a 10;
- La seconda associa ad ogni nodo un costo pari a $\lceil \frac{d(u)}{2} \rceil$, dove $d(u)$ è il grado del nodo u;
- La terza associa ad ogni nodo un costo che è inversamente proporzionale al suo grado. In particolare il costo di un nodo v è pari a $\lceil \frac{100}{d(v)} \rceil$. La costante 100 è stata selezionata in maniera sperimentale basandosi sul grado medio, massimo, minimo e sui parametri delle precedenti funzioni di costo.

3.2 Individuazione del seed set

Per l'individuazione del seed set sono stati utilizzati tre algoritmi: Cost-Seeds-Greedy, Weighted Target Set Selection e un terzo algoritmo progettato appositamente per tali esperimenti, di seguito chiamato "CostMajorityCascadeProblem".

3.2.1 Cost-seeds-greedy

Questo algoritmo punta a massimizzare la funzione $f(S) = \sum_{v \in V} min\{|N(v) \cap S|, \lceil \frac{d(v)}{2} \rceil\}$, tenendo in considerazione il budget k. Di seguito lo pseudocodice:

Algorithm 1: Cost-Seeds-Greedy (G, k, c, f_i)

$S_p = S_d = \emptyset$
repeat
  $select\ u = argmax_{v \in V - S_d} \frac{\delta_v f(S_d)}{c(u)}$
  $S_p = S_d$
  $S_d = S_p \cup \{u\}$
until $c(S_d) > k$
  return S_p

L'algoritmo prevede l'uso dei due insiemi vuoti S_p e S_d, utilizza un ciclo che si interrompe se $c(S_d)$ supera il budget, ad ogni iterazione seleziona il nodo che massimizza la funzione $f(S)$ e lo aggiunge al seed set. Il valore $\delta_v f(S_d)$ viene calcolato come $f(S_d \cup \{v\}) - f(S_d)$.

3.2.2 Weighted Target Set Selection

Dato il grafo G, la soglia di attivazione t e il costo c di ogni nodo, l'algoritmo elimina dal grafo i nodi che hanno una soglia pari a zero (cioè vengono attivati dai vicini), aggiunge al seed set i nodi che hanno un grado minore della soglia (non ci sono abbastanza vicini per attivare quel nodo, quindi bisogna aggiungerlo al seed set) oppure rimuove il nodo con il massimo rapporto costo-efficienza aggiornando le soglie dei vicini.

Algorithm 2: Weighted Target Set Selection

Input
$G = (V, E)$, $t(v)$, $c(v)$ $\forall v \in V$
Output
S Target set per G
$S = \emptyset$
$U = V$
for $v \in V$ **do** $\delta(v) = d_G(v)$; $k(v) = t(v)$; $N(v) = \Gamma_G(v)$
while $U \neq \emptyset$ **do**
  if $\exists v \in U$ s.t. $k(v) = 0$ **then**
    for $u \in N(v)$ **do** $k(u) = max\{0, k(u) - 1\}$
  else
    if $\exists v \in U$ s.t. $\delta(v) < k(v)$ **then**
      $S = S \cup \{v\}$
      for $u \in N(v)$ **do** $k(u) = k(u) - 1$
    else
      $v = argmax_{u \in U} \frac{c(u)k(u)}{\delta(u)(\delta(u)+1)}$
  for $u \in N(v)$ **do** $\delta(u) = \delta(u) - 1$; $N(u) = N(u) - \{v\}$
  $U = U - \{v\}$

3.3 CostMajorityCascadeProblem

Il terzo algoritmo si basa sulla logica del problema dello zaino, dove lo zaino è paragonabile al seedset, il budget k è paragonabile alla grandezza dello zaino e il valore di ogni nodo (oggetto nel problema dello zaino) è pari al numero di nodi vicini. Nel seed set si inseriscono i nodi con valore massimo il cui costo è minore del budget rimanente. Inoltre durante la scelta del nodo da inserire nel seed set non vengono considerati i vicini del nodo inserito nello step precedente.

Algorithm 3: CostMajorityCascadeProblem

Input
$G = (V, E)$, k, $c(v)\ \forall v \in V$
Output
S Target set per G
//costo del seed set, seed set e nodo inserito nello step precedente
$total\_cost = 0$
$S = \emptyset$
$prev\_node = null$
//calcolo del valore di ogni nodo
for $v \in V$ **do**
$val[v] = |N(v)|$
//finché c'è qualche nodo il cui costo rientra nel budget restante
while $\exists v \in V$ s.t. $c(v) < k - total\_cost$ **do**
//rimuove il nodo inserito nello step precedente nel seed set da V, ricalcola i valori di ogni nodo, non considera i vicini del nodo precedente, seleziona il nodo con valore massimo il cui costo è minore del budget rimanente
if $prev\_node \neq null$ **then**
$neigh\_prev\_node = N(prev\_node)$
$V = V - \{prev\_node\}$
for $v \in V$ **do**
$val[v] = |N(v)|$
$val\_no\_neigh = \{\forall node \in val$ s.t. $node \notin neigh\_prev\_node\}$
$node\_val\_max = argmax_{v \in val\_no\_neigh} val[v]$ & $c(v) < k - total\_cost$
else
//Else presente per il primo nodo inserito nel seed set
$node\_val\_max = argmax_{v \in val} val[v]$ & $c(v) < k - total\_cost$
//non c'è nessun altro nodo attivabile
if $node\_val\_max = null$ **then** $break$;
//salvataggio del nodo inserito per la prossima iterazione
$prev\_node = node\_val\_max$
//inserimento in S del nodo
if $c(node\_val\_max) + total\_cost \leq k$ **then**
$S = S \cup \{node\_val\_max\}$
$total\_cost = total\_cost + c(node\_val\_max)$
else
$break$;
return S

La logica di tale algoritmo punta a massimizzare l'influenza inserendo nel seed set prima i nodi con più vicini, quindi con valore massimo e tenendo in considerazione il budget. Di conseguenza anche questo algoritmo rientra tra gli algoritmi greedy. Inoltre aggiornando i valori dei nodi ad ogni iterazione ed escludendo i vicini del nodo precedente, l'algoritmo cerca di ottenere un seed set i cui nodi si trovano distanti tra di loro e non raggruppati in una singola area del grafo, con lo scopo di non far arrestare l'influenza in un eventuale cluster che potrebbe contenere i nodi del seed set.

3.4 Calcolo dei seed set

I seed set totali ottenuti dai tre algoritmi precedentemente presentati ammontano a 72. In particolare per ognuna delle tre funzioni di costo e per ognuno dei tre algoritmi sono stati calcolati 8 seed set, con budget pari a 20, 30, 40, 50, 60, 70, 80 e 90. La figura di seguito mostra le dimensioni dei seed sets.

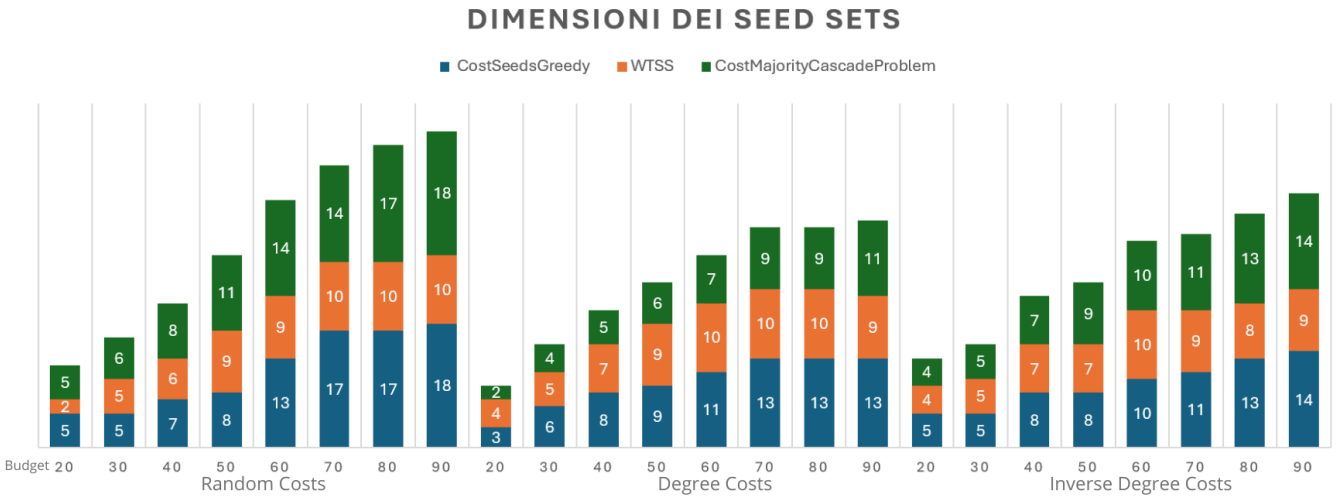

Figure 2: Numero di nodi nei seed sets

3.5 Calcolo dell'Influence Diffusion

Per il calcolo dell'Influence Diffusion si è utilizzato un modello in cui un determinato nodo viene influenzato se almeno la metà dei suoi vicini sono stati influenzati. Si riporta di seguito lo pseudocodice.

Algorithm 4: Calcolo dell'Influence Diffusion

Input
    $G = (V, E)$, *Seed set* S
Output
    $stato\_nodi$, $count$
$count = 0$
$stato\_dei\_nodi = \emptyset$
for $v \in V$ **do**
    $stato\_dei\_nodi[v] = 0$
for $v \in S$ **do**
    $stato\_dei\_nodi[v] = 1$
while $True$ **do**
    $flag = False$
    for $v \in V$ **do**
        if $|\{x \in N(v) : stato\_dei\_nodi[x] = 1\}| \geq |N(v)|/2$ && $stato\_dei\_nodi[v] = 0$ **then**
            stato_dei_nodi[v]=1
            flag=True
            count++
    if !flag **then**
        break
return stato_dei_nodi, count

Il processo per il calcolo dell'influenza riceve in input il grafo G e il seed set S, utilizza la variabile $count$ per conteggiare il numero di nodi infettati e $stato\_dei\_nodi$ per associare ad ogni nodo il valore 1 se il nodo è infetto, 0 altrimenti. Infatti $stato\_dei\_nodi$ viene prima inizializzato con tutti 0, successivamente ad ogni nodo presente nel seed set viene associato 1. Dopodiché viene avviato un ciclo infinito in cui viene settato $flag = False$ siccome ancora nessun nodo è stato infettato in questa iterazione, per ogni nodo del grafo si controlla se il numero di vicini infetti è maggiore o uguale alla metà dei vicini, se questa condizione di verifica allora si considera il nodo in questione come infetto e si setta il suo stato a 1, si aggiorna $flag$ a $True$ e si incrementa $count$. Se dopo aver

iterato ogni nodo del grafo, $flag$ risulta essere $False$, significa che nessun altro nodo può essere infettato, quindi il processo di influenza termina.

4 Risultati

In questo capitolo vengono discussi i risultati ottenuti eseguendo il processo di Influence Diffusion usando i seed set generati da tre diversi algoritmi con tre funzioni di costo. I dati rappresentati nei grafici mostrano il numero di nuovi nodi infettati a partire da un determinato seed set.

4.1 Degree Costs

Usando la funzione che ad ogni nodo associa un costo pari al suo grado, si osserva un andamento comune, in termini di nodi infettati, dei tre algoritmi fino a un budget pari a 50. Analizzando in maniera manuale i nodi infettati con budget 20, 30, 40 e 50 non risultano pattern evidenti.

Settando dei budget superiori a 60 il numero di nodi infettati dai seed sets degli algoritmi CostSeedsGreedy e CostMajorityCascadeProblem cresce molto rapidamente. Infatti incrementando il budget da 60 a 70, l'algoritmo CostMajorityCascadeProblem infetta 27 nuovi nodi. Invece con CostSeedsGreedy, spostando il budget da 70 a 80, si ottengono 24 nuovi nodi infetti.

In tutti i casi il seed set selezionato dall'algoritmo WTSS infetta meno di cinque nodi.

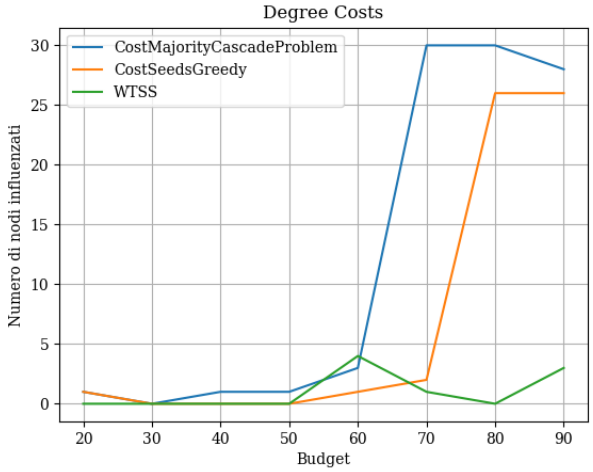

Figure 3: Numero di nodi infettati; Funzione di costo utilizzata: Degree Costs.

4.2 Inverse Degree Costs

Anche utilizzando la funzione di costo che ad ogni nodo associa un costo pari a $\lceil \frac{100}{d(v)} \rceil$, i seed set che riescono ad infettare più nodi sono quelli creati dal CostMajorityCascadeProblem. Infatti basta un budget pari a 40 per riuscire ad infettare 32 nodi. Tale risultato viene raggiunto anche dal seed set selezionato da WTSS con budget 80.

Invece il numero massimo di nodi infettati (pari a 28) grazie all'algoritmo CostSeedsGreedy si ottiene con un budget pari a 70. Inoltre risulta evidente un andamento comune, in termini di nodi infettati, tra i tre algoritmi.

4.3 Random Costs

Anche con una funzione di costo che associa costi random, il seed set che garantisce un numero di nodi infettati pari a 31 con budget pari a 40 è un seed set generato dal CostMajorityCascadeProblem. Aumentando il budget è evidente la decrescita dei nodi infettati, questo avviene perché aumentando il budget si incrementa anche la dimensione del seed set e quindi diminuisce il numero di nodi rimanenti. Per gli algoritmi WTSS e CostSeedsGreedy l'incremento dei nodi infettati avviene con un budget pari a 70, infatti il seed set ottenuto dal primo algoritmo infetta 23 nodi, il seed det dell'algoritmo WTSS ne infetta 29.

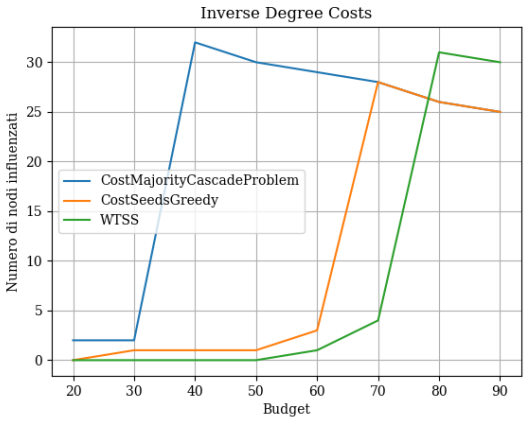

Figure 4: Numero di nodi infettati; Funzione di costo utilizzata: Inverse Degree Costs.

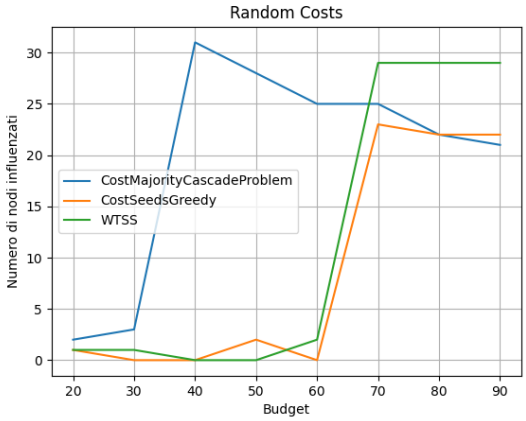

Figure 5: Numero di nodi infettati; Funzione di costo utilizzata: Random Costs.

4.4 Conclusioni

Il primo risultato più evidente è dato dai seed set selezionati utilizzando il CostMajorityCascadeProblem, infatti in tutti i casi permette di influenzare il numero maggiore di nodi con il budget minore rispetto gli altri due algoritmi. Questo risultato potrebbe essere dovuto al fatto di escludere il vicinato del nodo inserito nel seed set durante lo step precedente, infatti date le caratteristiche della rete presa in esame (in particolare coefficiente di clustering, grado medio e numero di triangoli), il comportamento dell'algoritmo implica l'esclusione di un numero (in media 16) di nodi che limita la scelta del nodo da inserire nel seed set a un sottoinsieme di nodi della rete. Nello step successivo l'algoritmo sceglierà il nodo da inserire nel seed set tenendo in considerazione solo il sottoinsieme "opposto" a quello dello step precedente. Di conseguenza il seed set conterrà nodi sparsi in tutta la rete e non raggruppati in una singola area, quindi la probabilità che l'influenza termini in un cluster della rete è ridotta.

Inoltre dai grafici si notano due fenomeni: l'andamento comune dei tre algoritmi in termini di nodi infettati, infatti i grafici dei tre algoritmi presentano una forma molto simile tra di loro con la differenza che i grafici di WTSS e CostSeedsGreedy risultano traslati verso destra siccome risultano efficaci aumentando il budget; e l'andamento decrescente dopo aver raggiunto il numero massimo di nodi infettabili, questo è dato dal crescere dei seed set e quindi dalla riduzione di nodi infettabili.

Tutto il necessario per la replicabilità degli esperimenti è presente al seguente link: https://github.com/GagliardeNicolapio/MajorityCascadeNetworks.