Università degli Studi di Salerno

Dipartimento di Informatica

Laurea Magistrale

Majority Dynamics in Networks

Progetto corso Reti Sociali

Autore:

Piero Agosto (0522501628)

Anno Accademico 2024–2025

Indice

Capitolo 1

Introduzione

La *majority domination* è una variante del problema del *dominating set* in teoria dei grafi. Dato un grafo $G = (V, E)$, dove V è l'insieme dei vertici ed E l'insieme degli archi, un sottoinsieme $D \subseteq V$ è definito **dominating set** se ogni vertice non appartenente a D ha almeno un adiacente in D.
Indicando con $N(v)$ l'insieme dei vicini di un vertice v, la condizione del dominating set è:

$$\forall v \in V \setminus D,\ |N(v) \cap D| \geq 1$$

Un insieme $D \subseteq V$ è un **majority dominating set** se ogni vertice $v \in V \setminus D$ ha almeno la metà dei suoi vicini appartenenti a D. Denotando con $d(v) = |N(v)|$ il grado del vertice v, la condizione matematica è:

$$\forall v \in V \setminus D,\ |N(v) \cap D| \geq \left\lceil \frac{d(v)}{2} \right\rceil$$

L'obiettivo della dominazione è determinare un dominating set (o un majority dominating set) D di taglia minima che domina tutto il grafo G. Questo è un problema noto per essere **NP-hard**, dunque difficile da risolvere in modo esatto su larga scala.
A partire da ciò si definisce il *Majority Cascade*, un processo dinamico di diffusione dell'influenza: a partire da un insieme iniziale di nodi, chiamato *seed set*, i nodi si attivano quando la maggioranza dei loro vicini è già attiva, rimanendo tali per sempre. Quando non ci sono altri nodi da poter rendere attivi, il processo di diffusione termina.
Se a ogni nodo si assegna un costo (cost majority cascade), il compito è scegliere un seed set che, rispettando il budget k, massimizzi le attivazioni. Essendo il problema NP-Hard, servono delle euristiche.
In questa relazione analizziamo tre approcci per la selezione del seed set in reti con majority cascade e vincoli di costo: due algoritmi noti in letteratura e un terzo algoritmo che ho proposto utilizzando come base un concetto noto. Le loro prestazioni vengono confrontate al variare delle funzioni di costo e del budget k disponibile, valutando la capacità di ciascun metodo di massimizzare la diffusione.

Capitolo 2

Metodologia

2.1 La rete utilizzata

Per la valutazione degli algoritmi ho scelto la rete *ca-GrQc*, disponibile in Snap[1], un repository dell'Università di Stanford. Si tratta di un grafo non orientato che rappresenta le collaborazioni scientifiche nella comunità di Relatività Generale e Cosmologia Quantistica. I dati provengono dalla sezione *General Relativity and Quantum Cosmology* dell'archivio elettronico arXiv e coprono il periodo che va da gennaio 1993 ad aprile 2003. I nodi del grafo corrispondono agli autori che hanno sottomesso articoli nella categoria GR-QC, mentre gli archi non orientati rappresentano una relazione di co-autoria: se un autore i ha scritto un articolo con un autore j, tra i due viene inserito un arco. Nel caso di un articolo con k autori, si genera un sottografo completamente connesso sui k nodi corrispondenti. La rete ha le seguenti proprietà strutturali:

- Numero di nodi: 5242
- Numero di archi: 14496
- Nodi nella componente connessa più grande: 4158 (79.3%)
- Numero di triangoli: 48260
- Diametro del grafo: 17
- Coefficiente medio di clustering: 0.5296

Questa rete è stata scelta perché rappresenta un contesto reale e significativo per lo studio dei processi di diffusione: le collaborazioni scientifiche possono infatti essere interpretate come un modello di propagazione di idee o comportamenti all'interno di una comunità.
Dal punto di vista computazionale, la dimensione del grafo lo rende abbastanza grande da presentare dinamiche complesse, ma comunque gestibile per la simulazione degli algoritmi proposti senza costi computazionali eccessivi.

[1] `https://snap.stanford.edu/data/ca-GrQc.html`

Inoltre, le proprietà strutturali rendono il dataset particolarmente interessante: il diametro ridotto e l'elevato coefficiente di clustering sono tipici delle reti *small-world*, dove la propagazione può diffondersi rapidamente attraverso la rete ma anche restare confinata in comunità locali molto dense. La presenza di un numero elevato di triangoli e di una componente connessa principale molto estesa permette di osservare in modo realistico l'efficacia dei diversi algoritmi di selezione del seed set.

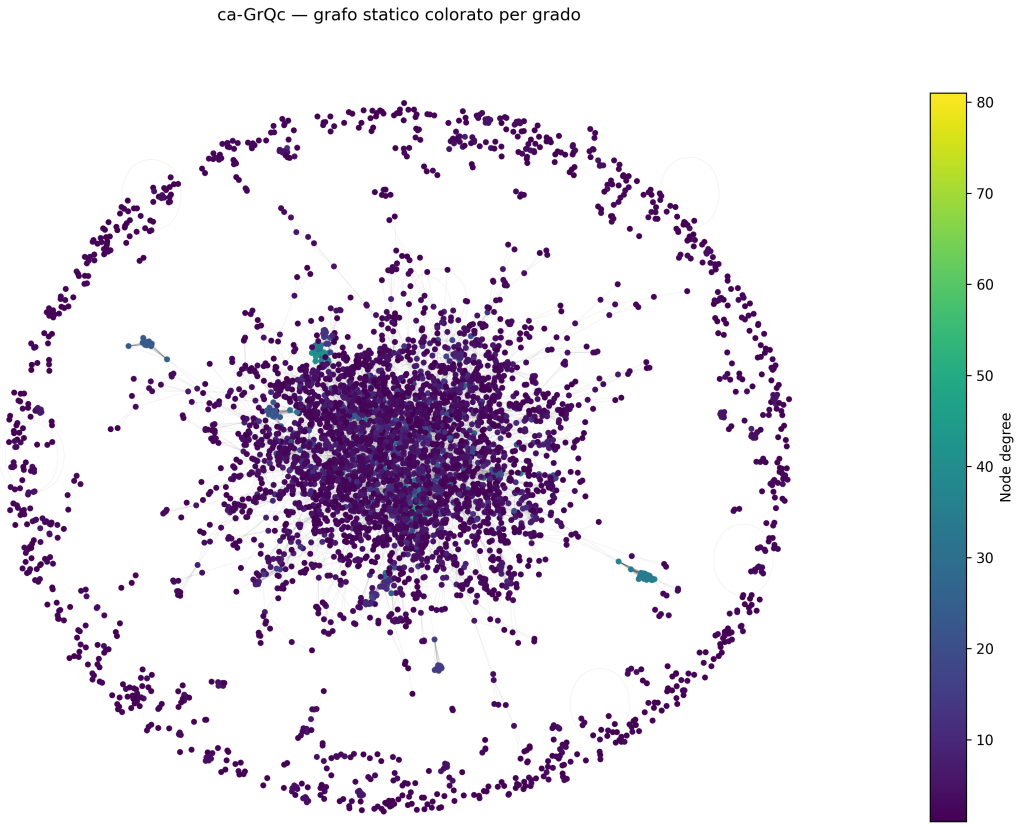

Figura 2.1: Plot della rete scelta

2.2 Funzioni di costo

Per valutare le prestazioni degli algoritmi in scenari con differenti vincoli economici, sono state considerate tre funzioni di costo.

1. Uniform Cost

$$c_{\text{uni}}(v) = 1$$

In questa configurazione ogni nodo ha lo stesso costo unitario, indipendentemente dalle sue caratteristiche strutturali (grado, posizione, ecc.). Questa scelta rappresenta il caso più semplice

e bilanciato, in cui il budget disponibile corrisponde esattamente al numero massimo di nodi selezionabili come seed.

2. Random Cost

$$c_{\text{rand}}(v) \sim \mathcal{U}(1, 10)$$

dove $\mathcal{U}(1, 10)$ indica la distribuzione uniforme discreta sull'intervallo $[1, 10]$. In questo scenario, ogni nodo riceve un costo intero casuale compreso tra 1 e 10. L'obiettivo è simulare situazioni in cui i nodi presentano un peso eterogeneo, riflettendo una complessità maggiore nel processo di selezione dei seed, in quanto nodi potenzialmente equivalenti in termini di struttura possono richiedere costi differenti.

3. Threshold Cost

$$c_{\text{thr}}(v) = \left\lceil \frac{d(v)}{2} \right\rceil$$

dove $d(v)$ rappresenta il grado del nodo v. Questa funzione lega direttamente il costo al livello di connettività del nodo: i nodi con grado più elevato risultano più onerosi da selezionare come seed. In tal modo si penalizza la scelta di nodi altamente connessi, introducendo un compromesso tra potenziale diffusivo e costo associato.

2.3 Algoritmi utilizzati

La selezione del majority dominating set è stata affrontata con tre approcci principali: **Greedy**, **WTSS** e un algoritmo proposto basato su **centralità**.

2.3.1 Greedy Seed Set Selection

Il metodo *Greedy* costruisce il seed set S in maniera incrementale: ad ogni passo viene selezionato il nodo u che massimizza il *guadagno marginale per unità di costo*, ovvero:

$$u = \arg\max_{v \in V \setminus S} \frac{\Delta_v f(S)}{c(v)}$$

dove $\Delta_v f(S) = f(S \cup \{v\}) - f(S)$ rappresenta il beneficio marginale apportato dal nodo v e $c(v)$ è il costo associato a v. L'algoritmo procede iterativamente finché il budget disponibile lo consente. La qualità della scelta è guidata dall'euristica f, che valuta l'influenza del nodo rispetto ai vicini.
Sono state sperimentate tre euristiche distinte:

1. **Euristica** f_1: conta per ogni nodo v quanti dei suoi vicini sono già inclusi nel set S, senza superare la soglia necessaria per raggiungere la maggioranza tra i vicini. Sommandoli su

tutti i nodi, si misura quanto il seed set attuale influenza la rete. Formalmente:

$$f_1(S) = \sum_{v \in V} \min \left\{ |N(v) \cap S|, \left\lceil \frac{d(v)}{2} \right\rceil \right\}$$

2. **Euristica** f_2: per ogni nodo v, considera i vicini già nel seed set in ordine e attribuisce un peso decrescente. I primi vicini hanno un contributo maggiore, mentre quelli successivi sempre minore fino al raggiungimento della maggioranza. In questo modo si tiene conto del dettaglio con cui i nodi vicini contribuiscono alla diffusione.

$$f_2(S) = \sum_{v \in V} \sum_{i=1}^{|N(v) \cap S|} \max \left\{ \left\lceil \frac{d(v)}{2} \right\rceil - i + 1, 0 \right\}$$

3. **Euristica** f_3: è simile a f_2, ma introduce un fattore di normalizzazione che dipende dal numero totale di vicini rimanenti. Questo rende la misura più proporzionata e bilanciata, poiché considera non solo l'ordine di attivazione ma anche la struttura locale del grafo.

$$f_3(S) = \sum_{v \in V} \sum_{i=1}^{|N(v) \cap S|} \max \left\{ \frac{\left\lceil \frac{d(v)}{2} \right\rceil - i + 1}{d(v) - i + 1}, 0 \right\}$$

Il processo termina restituendo il seed set S che massimizza la diffusione rispettando il vincolo di budget. Lo pseudocodice dell'algoritmo è riportato di seguito:

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

$S_p = S_d = \emptyset$
repeat
    select $u = \arg\max_{v \in V - S_d} \frac{\Delta_v f_i(S_d)}{c(u)}$
    $S_p = S_d$
    $S_d = S_p \cup \{u\}$
until $c(S_d) > k$
return S_p

2.3.2 Weighted Target Set Selection (WTSS)

Il secondo algoritmo considerato è il *Weighted Target Set Selection* (WTSS), il cui obiettivo è individuare un seed set S tale che $c(S) \leq k$, massimizzando la diffusione con il modello majority cascade.
L'algoritmo si basa su un approccio iterativo che considera per ogni nodo v:

- il suo **grado residuo** $\delta(v)$, inizialmente pari al grado $d(v)$;
- la sua **soglia** $k(v)$, fissata come $k(v) = \lceil \frac{d(v)}{2} \rceil$;
- l'insieme dei suoi vicini $N(v)$.

Ad ogni passo viene selezionato un nodo secondo tre possibili casi:

1. **Caso 1**: se esiste un nodo v con $k(v) = 0$, significa che può essere attivato dall'influenza dei suoi vicini già attivi. In questo caso, v non viene incluso nel seed set ma viene attivato e contribuisce a ridurre la soglia dei suoi vicini.
2. **Caso 2**: se esiste un nodo v con $\delta(v) < k(v)$, significa che non ha abbastanza vicini per poter essere attivato. In questo caso, v viene aggiunto al seed set S, e i vicini aggiornano la propria soglia.
3. **Caso 3**: se nessuno dei due casi precedenti si verifica, si seleziona il nodo v con valore massimo della seguente espressione:

$$\frac{c(v) \cdot k(v)}{\delta(v) \cdot (\delta(v) + 1)}$$

   che bilancia il costo e la capacità del nodo di contribuire alla diffusione. Questo nodo viene poi rimosso dal grafo, aggiornando le informazioni sui vicini.

Il processo termina quando tutti i nodi sono stati eliminati o attivati, restituendo il seed set S.

Algorithm 2 WTSS(G)

$S = \emptyset$; $U = V$
for each $v \in V$ **do**
  $\delta(v) = d_G(v)$; $k(v) = t(v)$; $N(v) = \Gamma_G(v)$
end for
while $U \neq \emptyset$ **do**
  if there exists $v \in U$ such that $k(v) = 0$ **then** ▷ Case 1: v is activated by its neighbors in $V - U$
    for each $u \in N(v)$ **do**
      $k(u) = \max\{0, k(u) - 1\}$
    end for
  else
    if there exists $v \in U$ such that $\delta(v) < k(v)$ **then** ▷ Case 2: v is added to S
      $S = S \cup \{v\}$
      for each $u \in N(v)$ **do**
        $k(u) = k(u) - 1$
      end for
    else ▷ Case 3: v activated by neighbors in U
      $v = \arg\max_{u \in U} \frac{c(u)k(u)}{\delta(u)(\delta(u)+1)}$
    end if
  end if
  for each $u \in N(v)$ **do**
    $\delta(u) = \delta(u) - 1$; $N(u) = N(u) - \{v\}$
  end for
  $U = U - \{v\}$
end while

2.3.3 Centrality Seed Set Selection

L'algoritmo *Centrality Seed Set Selection* è una procedura basata sulla **betweenness centrality** dei nodi, vincolata al budget disponibile. La betweenness centrality misura quanto un nodo agisca da "ponte" tra comunità o sottogruppi della rete, calcolando il numero di cammini minimi che attraversano quel nodo. L'idea alla base è che attivare nodi con elevata centralità consenta di diffondere l'attivazione oltre i confini locali di ciascuna comunità, raggiungendo così una porzione più ampia del grafo.
Il procedimento può essere descritto nei seguenti passi:

1. Si calcola la betweenness centrality $b(v)$ per ogni nodo $v \in V$.
2. Si ordina l'insieme dei nodi secondo il rapporto $b(v)/c(v)$, in ordine decrescente. Questo garantisce che vengano privilegiati i nodi con maggiore centralità rispetto al costo.
3. Partendo dal nodo più alto in graduatoria, si aggiungono nodi al seed set S fintanto che il budget k lo consente.
4. Ad ogni inserimento, il budget residuo viene aggiornato sommando il costo $c(v)$ del nodo selezionato.

Il processo termina quando l'aggiunta di ulteriori nodi eccederebbe il budget. Il seed set finale S contiene dunque nodi che, a parità di costo, massimizzano la capacità di collegare comunità diverse e diffondere l'attivazione in maniera efficiente.

Vantaggio dell'algoritmo. L'istogramma della distribuzione della betweenness centrality per la rete *ca-GrQc* (Figura 2.2) mostra come la maggior parte dei nodi presenti valori prossimi a zero, mentre solo pochi nodi hanno valori significativamente alti. Questi nodi, pur essendo rari, rappresentano veri e propri hub di collegamento tra comunità: la loro selezione nel seed set permette di superare la propagazione confinata nei cluster locali, amplificando la diffusione su scala globale.

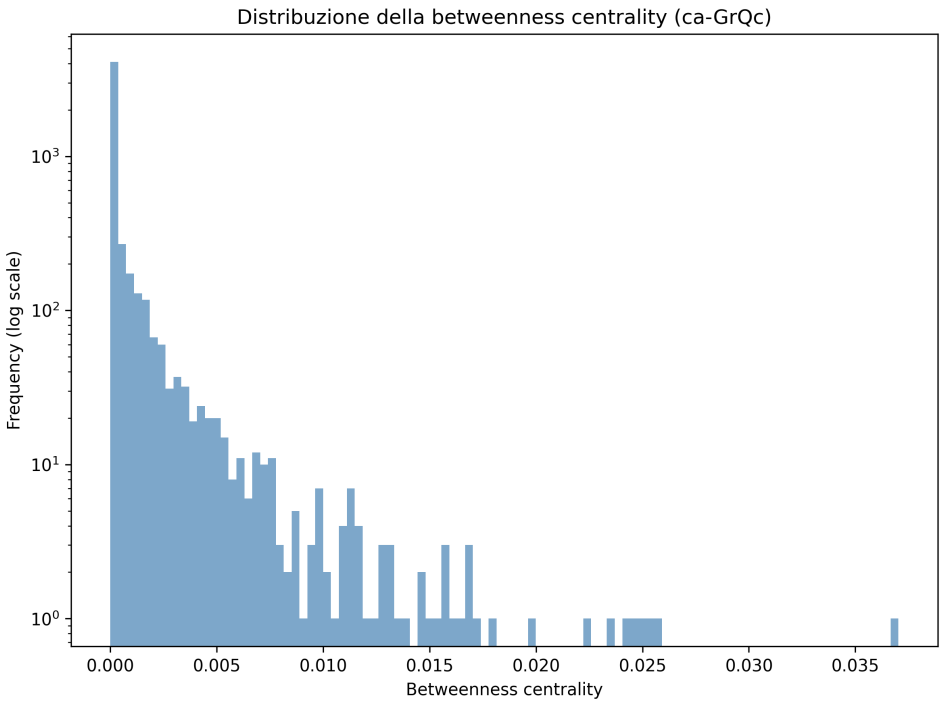

Figura 2.2: Distribuzione della betweenness centrality nella rete *ca-GrQc*.

Algorithm 3 Centrality-Seed-Set(G, k, c)

Compute betweenness centrality $b(v)$ for all $v \in V$
Sort nodes by $b(v)/c(v)$ in decreasing order
$S = \emptyset$, total_cost = 0
for each v in ranking **do**
    $c = c(v)$
    if total_cost + $c \leq k$ **then**
        $S = S \cup \{v\}$
        total_cost = total_cost + c
    end if
end for
return S

2.4 Majority Cascade in Networks

Sia dato un grafo $G = (V, E)$. Indichiamo con $N(v)$ l'insieme dei vicini di un nodo v e con $d(v) = |N(v)|$ il suo grado. Fissato un insieme di nodi iniziali $S \subseteq V$, il processo di diffusione *Majority dynamical process of influence diffusion* su G è descritto come una sequenza di insiemi:

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

dove lo stato iniziale è definito come:

$$Inf[S, 0] = S$$

e per ogni passo vale la seguente regola di aggiornamento:

$$Inf[S, r] = Inf[S, r-1] \cup \{v \in V \setminus Inf[S, r-1] : |N(v) \cap Inf[S, r-1]| \geq \lceil d(v)/2 \rceil\}.$$

Il più piccolo valore t tale che $Inf[S, t] = Inf[S, t+1]$ rappresenta l'istante di arresto del processo; in questo momento l'insieme dei nodi attivati coincide con $Inf[S] = Inf[S, t]$.
Questo modello corrisponde a una variante deterministica del *Linear Threshold Model*, dove un nodo viene attivato non appena almeno la metà dei suoi vicini è già attiva. L'attivazione è irreversibile, cioè un nodo che entra nello stato attivo vi rimane fino al termine della cascata.

2.5 Valutazione degli algoritmi

Per valutare l'efficacia delle strategie di selezione del seed set, per ciascuna funzione di costo considerata (*uniform*, *random(1,10)*, *threshold*) sono stati eseguiti i tre algoritmi:

- **Greedy**, nelle tre varianti basate sulle euristiche f_1, f_2, f_3;
- **WTSS** (Weighted Target Set Selection);
- **Centrality Seed Set Selection**, algoritmo proposto basato su betweenness centrality normalizzata rispetto al costo.

Per ogni combinazione (*funzione di costo - algoritmo*) il budget k è stato fatto variare su percentuali fissate:

$$\{0.5, 1, 2, 5, 10\}\%$$

Al fine di garantire confronti coerenti tra modelli di costo differenti, la quantità di riferimento su cui applicare la percentuale varia in base alla funzione di costo adottata:

1. **Uniform cost**: ogni nodo ha costo unitario. In questo caso k è espresso come percentuale del numero totale di nodi n.

2. **Random cost** $U(1, 10)$: i nodi hanno costi casuali interi tra 1 e 10. Anche in questo caso k è rapportato a n, in modo che il budget rimanga comparabile indipendentemente dalle estrazioni casuali dei costi.

3. **Threshold cost** $\lceil d(v)/2 \rceil$: il budget k è calcolato come percentuale del numero di archi m, poiché $\sum_v \lceil d(v)/2 \rceil \approx \frac{1}{2} \sum_v d(v) = m$. Questo assicura che il budget rifletta l'ammontare complessivo di costo indotto dai gradi dei nodi.

Pertanto, per ogni esperimento si è seguito il seguente procedimento:

1. selezionato il seed set S con l'algoritmo scelto sotto vincolo di budget k;

2. eseguita la *majority cascade* (aggiornamento sincrono) fino allo stato stazionario;

3. calcolato e registrato su file il valore $|Inf[S]|$ insieme ai metadati (funzione di costo, algoritmo, budget k);

4. calcolata la **copertura della rete** come $|Inf[S]|/|V|$ (frazione di nodi attivati), usata come metrica di confronto e come ordinata nei grafici.

Capitolo 3

Risultati

In questo capitolo presento i risultati ottenuti dagli esperimenti di diffusione del modello majority cascade al variare dei vincoli di budget k e delle funzioni di costo considerate. Per ciascuna funzione di costo creo una sottosezione specifica, all'interno della quale vengono riportati:

- una breve descrizione dei risultati, confrontandoo i diversi algoritmi e l'andamento al crescere del budget;
- una tabella riassuntiva dei valori di **diffusion ratio**, che sarebbe la frazione dei nodi attivati per i diversi valori di budget k;
- grafici sperimentali sotto forma di curve

3.1 Uniform Cost Function

Dalla Tabella 3.1 si osserva come, per la *uniform cost function*, le strategie **Greedy** tendano a fornire valori di diffusione sensibilmente più elevati rispetto agli algoritmi **WTSS** e **Centrality**, specialmente al crescere del budget k.
Per valori piccoli di budget ($k = 26, 52$), le differenze tra gli approcci risultano contenute: tutte le strategie ottengono diffusion ratio relativamente bassi, con un leggero vantaggio per **Centrality**, che riesce a sfruttare i nodi ponte per attivare più rapidamente porzioni di rete. In questo intervallo, le funzioni **Greedy f1** e **Greedy f3** risultano già più efficaci rispetto a **Greedy f2**, mentre **WTSS** rimane il meno performante.
Con l'aumento del budget ($k = 104$), le strategie **Greedy** migliorano significativamente le proprie prestazioni, in particolare **f1** e **f3**, che iniziano a distaccarsi nettamente da **Centrality** e **WTSS**. Per valori intermedi e grandi di budget ($k = 262, 524$), **Greedy f3** diventa la funzione più efficace, raggiungendo i migliori valori di diffusione (fino a 0.603), seguita da vicino da **Greedy f1**, mentre **Greedy f2** resta meno competitiva. **Centrality** mantiene performance discrete, ma inferiori alle strategie greedy, e **WTSS** rimane l'approccio meno efficiente in tutti gli scenari.

In sintesi, per la *uniform cost function*, le strategie **Greedy f1** e soprattutto **Greedy f3** si confermano le più efficaci nella massimizzazione della diffusione, mentre gli algoritmi **WTSS** e **Centrality** mostrano un ruolo secondario e meno competitivo.

Budget	Greedy f1	Greedy f2	Greedy f3	WTSS	Centrality
26	0.021	0.012	0.022	0.0099	0.0265
52	0.039	0.026	0.038	0.0202	0.0500
104	0.086	0.069	0.079	0.0427	0.1080
262	0.261	0.242	0.344	0.1274	0.3375
524	0.579	0.553	0.603	0.2965	0.5446

Tabella 3.1: Confronto del diffusion ratio per la **uniform cost function**.

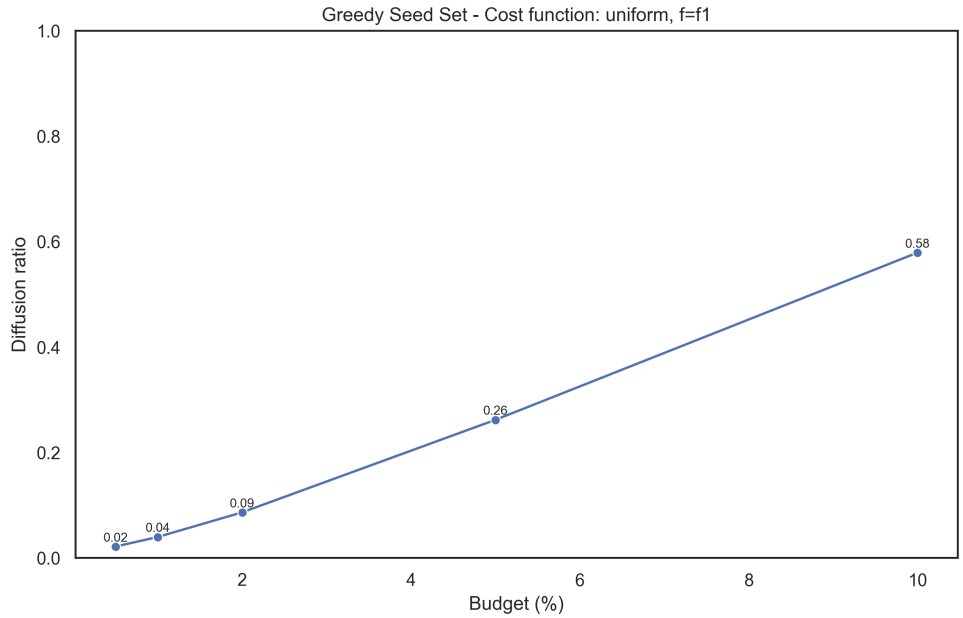

Figura 3.1: Greedy f1

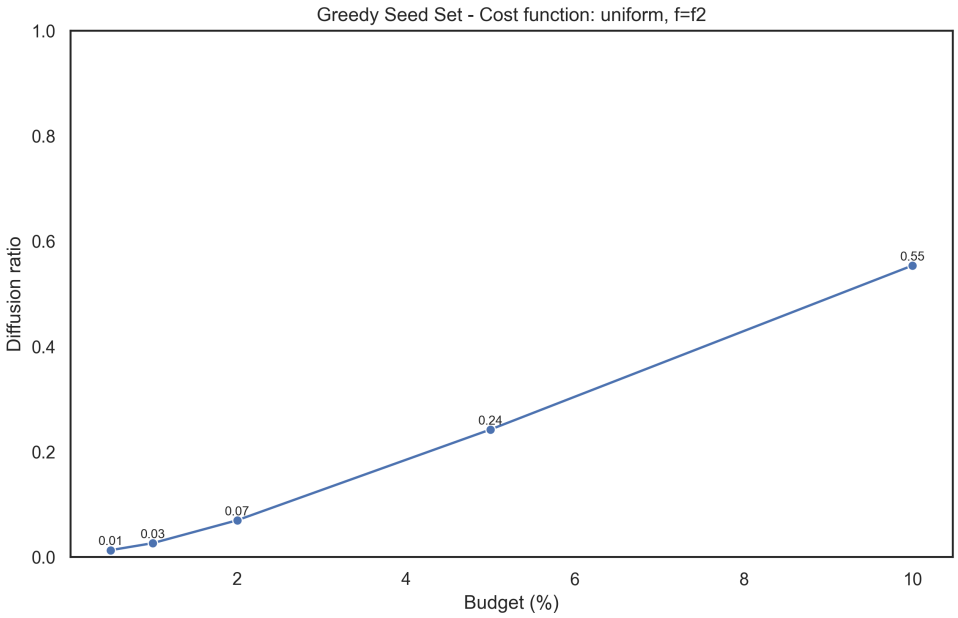

Figura 3.2: Greedy f2

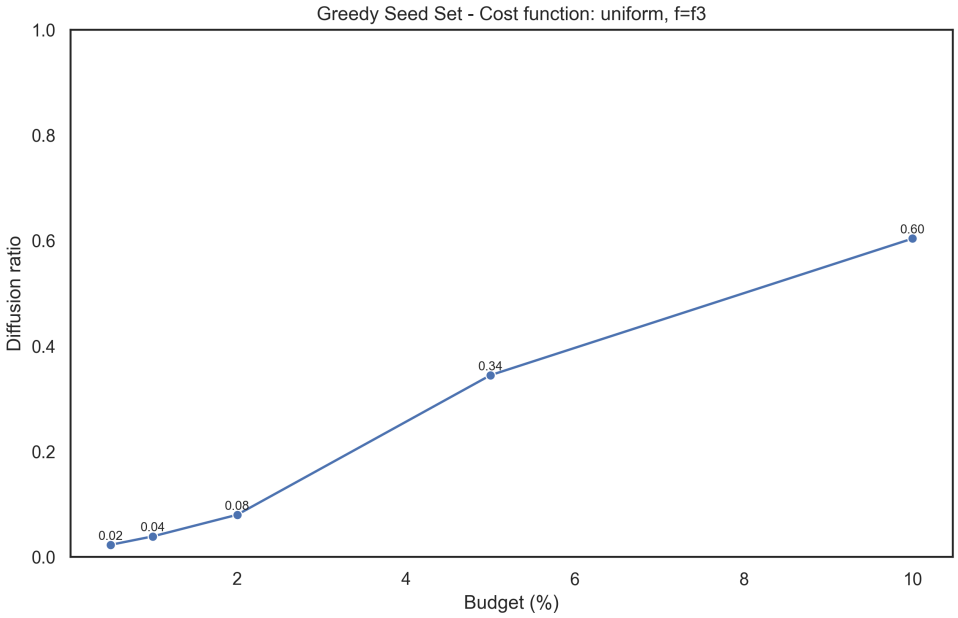

Figura 3.3: Greedy f3

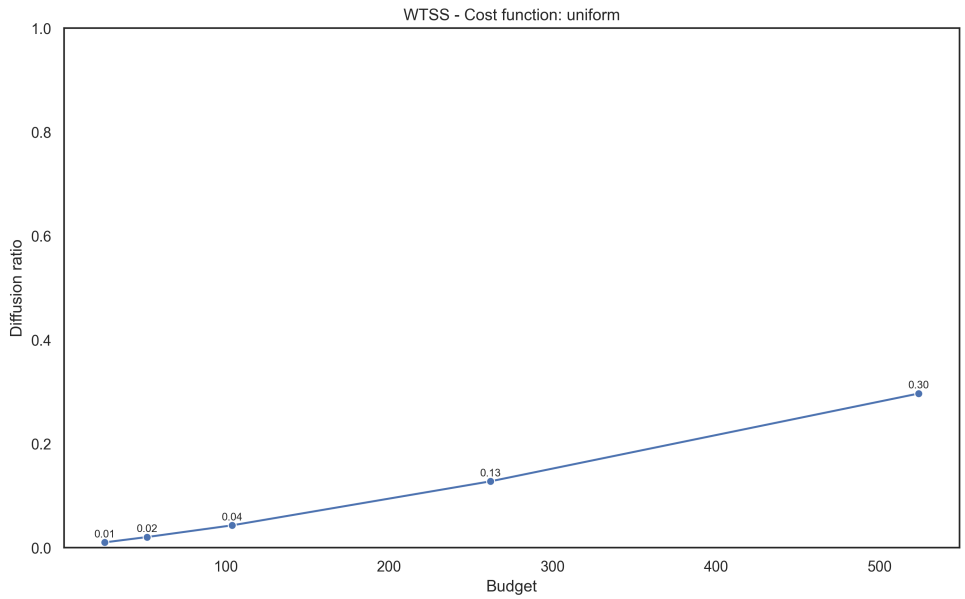

Figura 3.4: WTSS

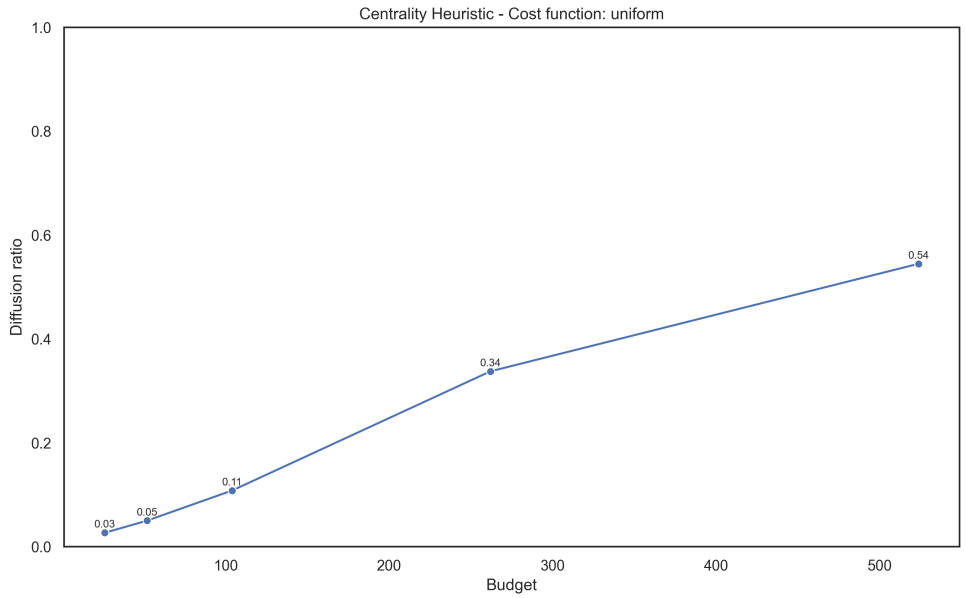

Figura 3.5: Centrality

3.2 Random Cost Function

Dalla Tabella 3.2 si osserva come le diverse strategie *Greedy*, insieme agli algoritmi *WTSS* e *Centrality*, si comportino al variare del budget k. Per budget più piccoli ($k = 144, 288$), la strategia *Greedy f1* raggiunge i valori di diffusione più elevati, superando significativamente le altre funzioni greedy e gli altri algoritmi, che mostrano performance quasi nulle. Con l'aumento del budget ($k = 576$), *Greedy f1* mantiene ancora un vantaggio, ma *Greedy f2* e *Greedy f3* riducono il gap, mentre *WTSS* e *Centrality* iniziano a migliorare leggermente.

Per budget maggiori ($k = 1441, 2883$), si nota un cambiamento nella performance relativa: *Greedy f3* diventa la strategia più efficace a $k = 2883$, mentre *Greedy f1* resta competitiva a $k = 1441$. Gli algoritmi *WTSS* e *Centrality*, pur aumentando i valori di diffusione con il budget, rimangono sempre significativamente inferiori rispetto alle strategie greedy. Complessivamente, le strategie *Greedy*, in particolare *f1* e *f3*, risultano più efficaci nella massimizzazione della diffusione rispetto agli altri approcci.

Budget	**Greedy (f1,f2,f3)**	**WTSS**	**Centrality**
144	0.021 / 0.010 / 0.019	0.009	0.013
288	0.039 / 0.025 / 0.038	0.022	0.043
576	0.082 / 0.069 / 0.081	0.043	0.084
1441	0.270 / 0.222 / 0.350	0.124	0.315
2883	0.580 / 0.551 / 0.599	0.291	0.532

Tabella 3.2: Diffusion ratio per i tre algoritmi sulla funzione di costo **random**.

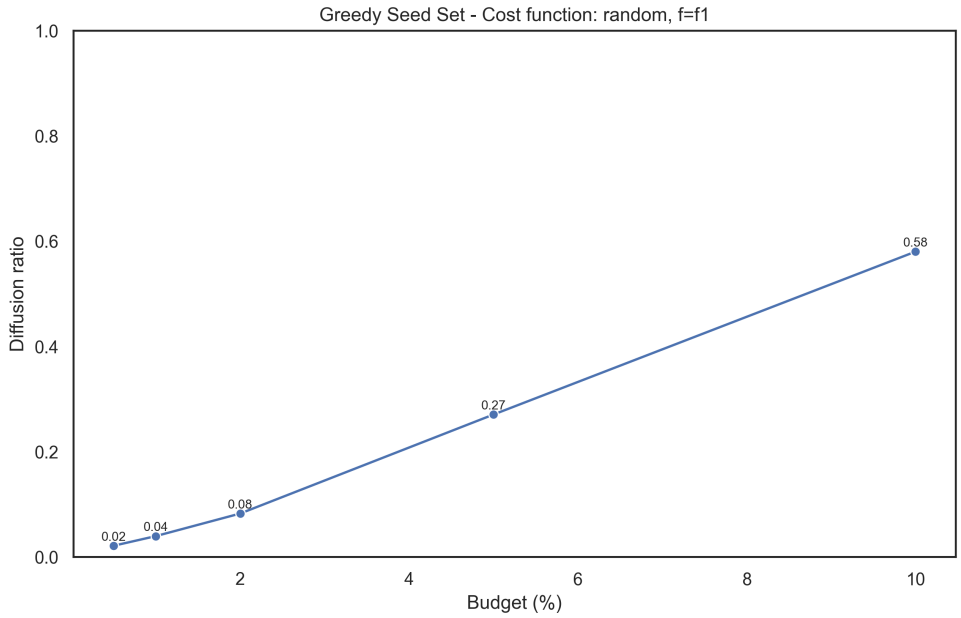

Figura 3.6: Greedy f1

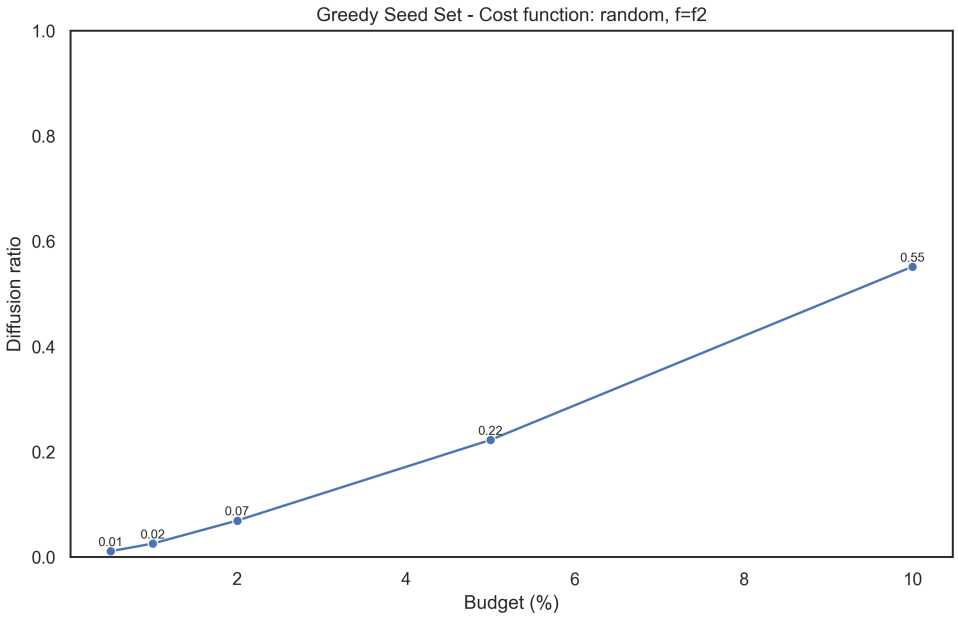

Figura 3.7: Greedy f2

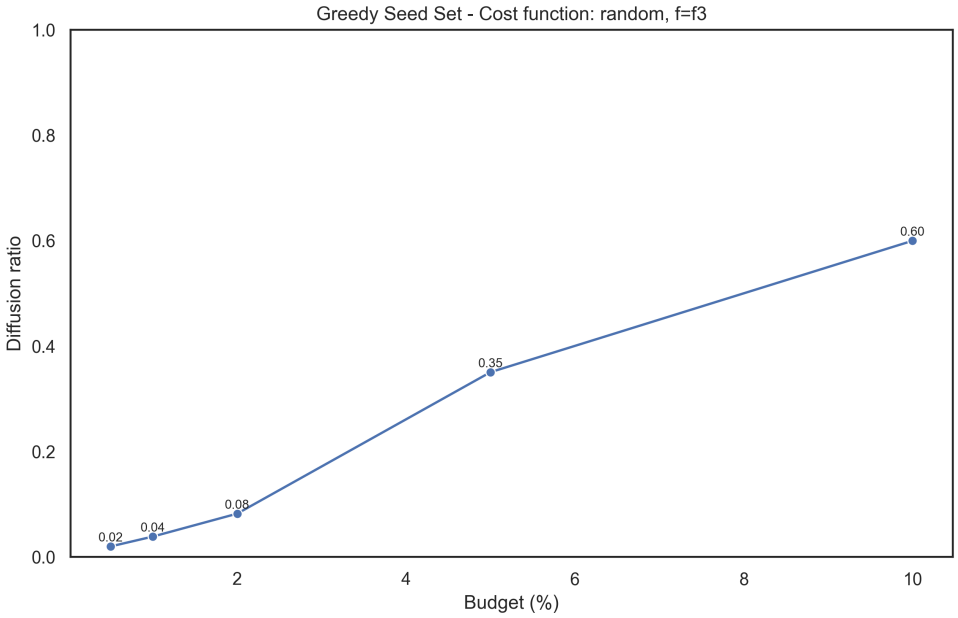

Figura 3.8: Greedy f3

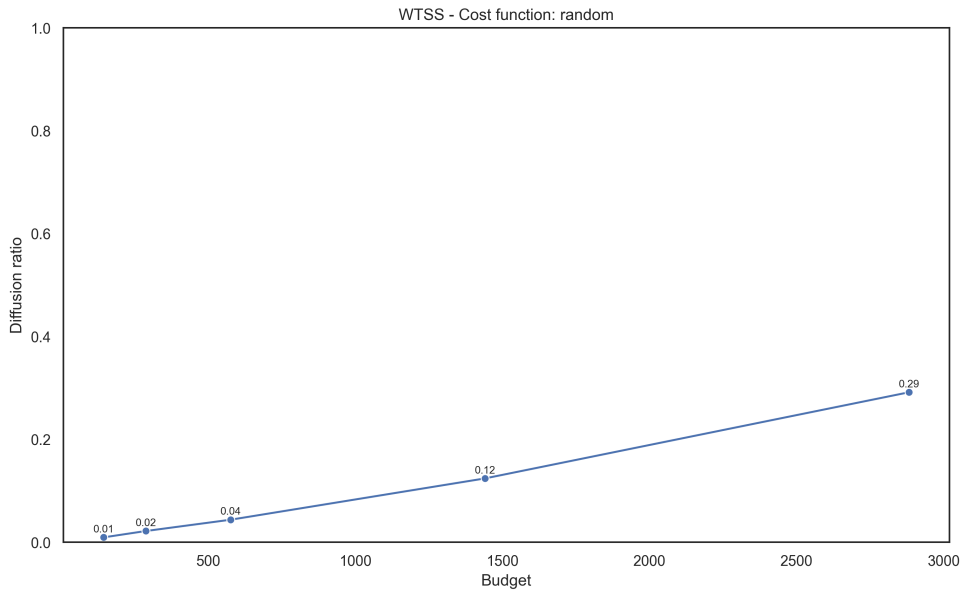

Figura 3.9: WTSS

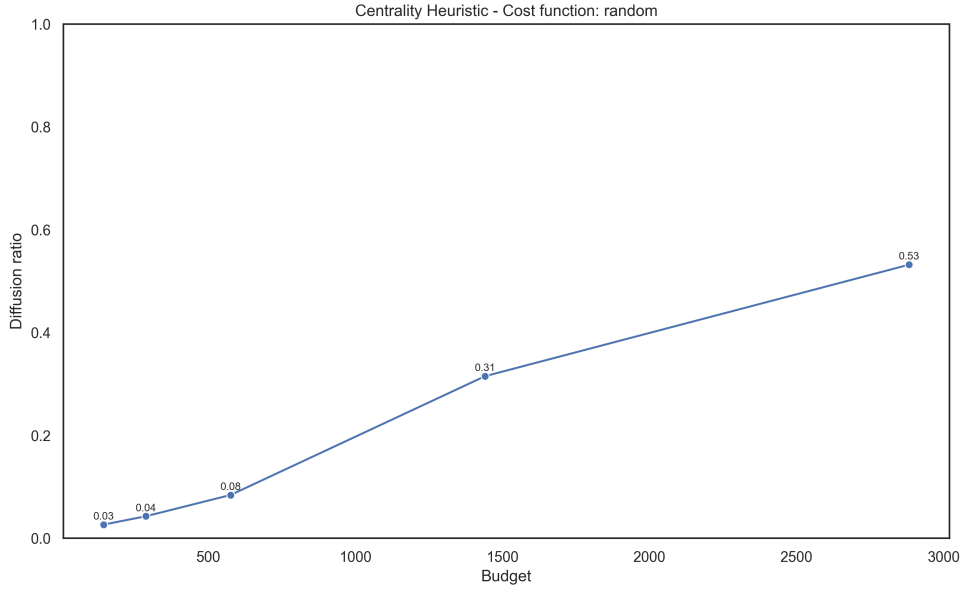

Figura 3.10: Centrality

3.3 Threshold Cost Function

Dalla Tabella 3.3 si osserva che, per la funzione di costo *threshold*, le strategie *Greedy* mostrano prestazioni inizialmente molto basse. Per valori ridotti di budget ($k = 79, 159, 318$), i valori di diffusione ottenuti da *Greedy f1* e *Greedy f2* sono pressoché nulli (rispettivamente 0.0099 e 0.0019 per $k = 79$), mentre *Greedy f3* evidenzia una crescita più rapida (0.0355 per $k = 79$ e 0.1023 per $k = 318$), pur restando distante dalle altre strategie. Con l'aumentare del budget, in particolare a $k = 797$ e $k = 1594$, *Greedy f3* raggiunge valori più competitivi (0.1927 e 0.3129), superando nettamente le altre due varianti greedy.
Gli algoritmi *WTSS* e *Centrality*, al contrario, partono da valori leggermente più elevati fin dai budget più bassi. In particolare, *WTSS* si mantiene costantemente il migliore, passando da 0.0298 per $k = 79$ a 0.3512 per $k = 1594$. *Centrality*, pur mostrando una crescita regolare (da 0.0065 a 0.1997), rimane sempre su valori più bassi rispetto a *WTSS* e non raggiunge mai la competitività di quest'ultimo.
Nel complesso, la funzione di costo *threshold* riduce notevolmente l'efficacia di *Greedy f1* e *Greedy f2*, mentre valorizza *Greedy f3* soltanto a budget più alti. In questo scenario, *WTSS* emerge come la strategia dominante, mentre *Centrality*, pur migliorando progressivamente, rimane complessivamente meno efficace.

Budget	**Greedy (f1,f2,f3)**	**WTSS**	**Centrality**
79	0.0099 / 0.0019 / 0.0355	0.0298	0.0065
159	0.0170 / 0.0034 / 0.0652	0.0626	0.0126
318	0.0233 / 0.0053 / 0.1023	0.1255	0.0259
797	0.0572 / 0.0166 / 0.1927	0.2083	0.0845
1594	0.1524 / 0.0528 / 0.3129	0.3512	0.1997

Tabella 3.3: Diffusion ratio per i tre algoritmi sulla funzione di costo **threshold**.

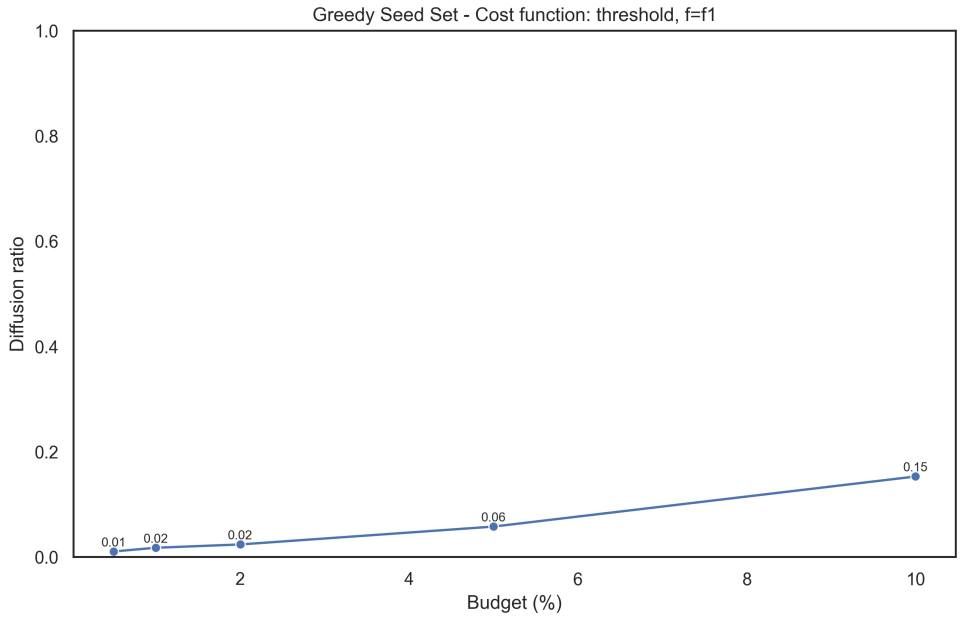

Figura 3.11: Greedy f1

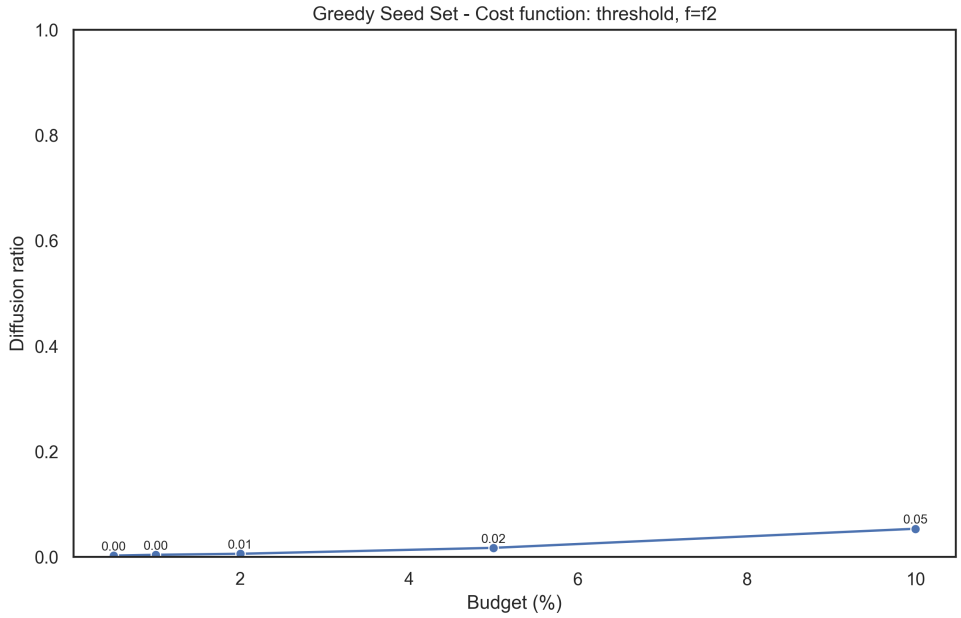

Figura 3.12: Greedy f2

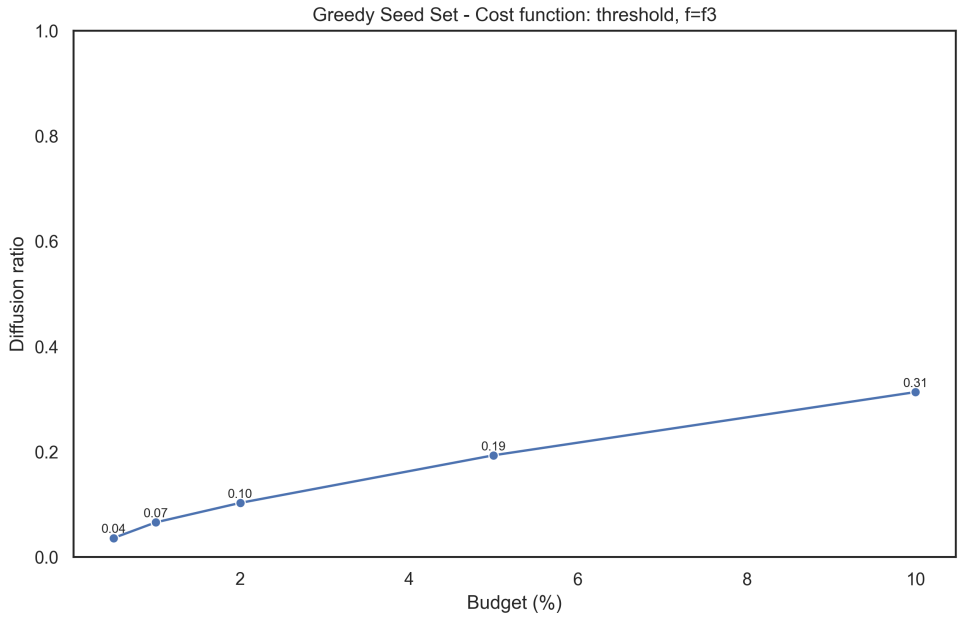

Figura 3.13: Greedy f3

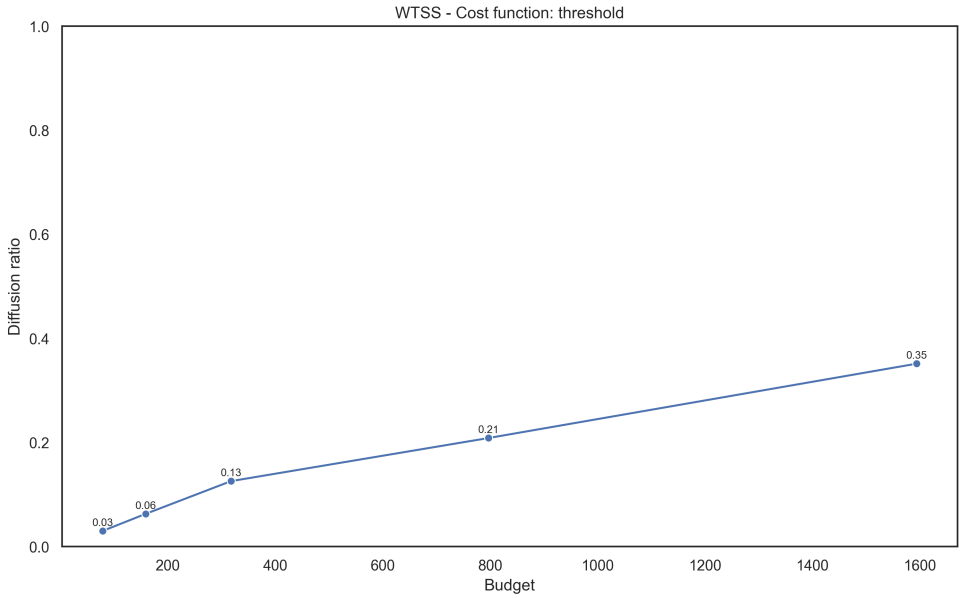

Figura 3.14: WTSS

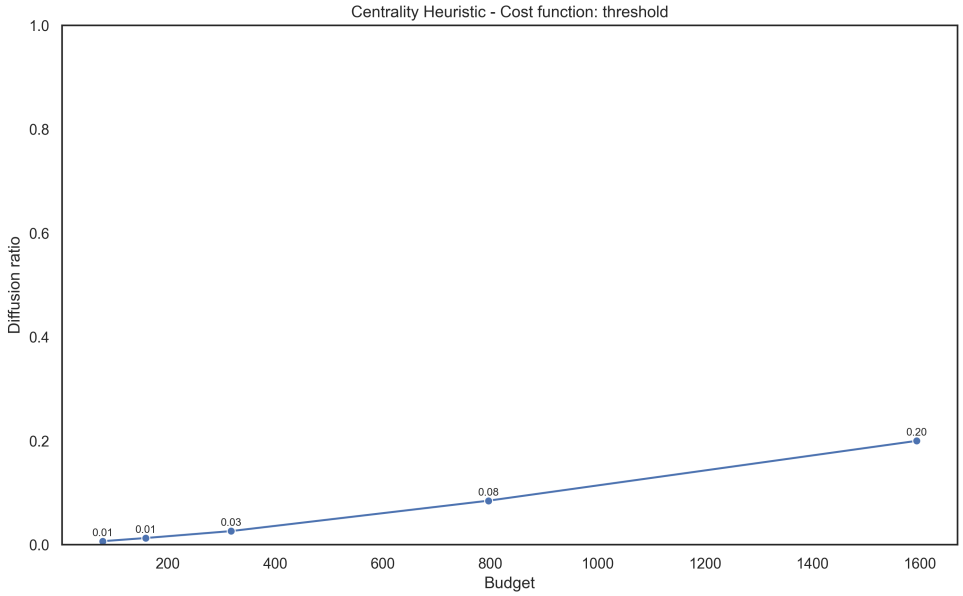

Figura 3.15: Centrality

Capitolo 4

Conclusioni

In questo lavoro ho confrontato tre approcci per la selezione del *seed set* sotto vincolo di budget: le varianti Greedy (f_1, f_2, f_3), l'algoritmo WTSS e una procedura Centrality basata su *betweenness* normalizzata rispetto al costo. Gli esperimenti sono stati eseguiti sulla rete `ca-GrQc` variando sistematicamente il budget in percentuale rispetto a n (per *uniform* e *random*) o a m (per *threshold*), in modo da mantenere confronti coerenti tra modelli di costo diversi.

Sintesi dei risultati.

- **Costo *uniform*.** Le varianti Greedy dominano per budget medio–alti: f_3 è la più efficace e raggiunge una copertura fino a 0.603 con $k = 524$, seguita da f_1 (0.579); Centrality resta leggermente dietro (0.545), mentre WTSS è complessivamente il meno performante (0.296). A budget minimi, il vantaggio di Centrality è sottile ma presente.
- **Costo *random* $U(1, 10)$.** Per budget piccoli–medi f_1 guida la diffusione; crescendo il budget f_3 passa al comando e raggiunge 0.599 con $k = 2883$. Centrality migliora (fino a 0.532) ma rimane dietro a Greedy; WTSS cresce ma senza colmare il divario (fino a 0.291).
- **Costo *threshold* $\lceil d(v)/2 \rceil$.** Le Greedy sono penalizzate ai budget bassi; f_3 è l'unica a recuperare a budget alti (fino a 0.3129 con $k = 1594$). In questo regime WTSS risulta sistematicamente migliore lungo tutto l'intervallo di budget (da 0.0298 a 0.3512), mentre Centrality cresce regolarmente ma resta meno competitiva.

Due tendenze principali. Primo: le varianti *greedy* — soprattutto f_3 — funzionano meglio quando il modello di costo non penalizza troppo i nodi ad alto grado, perché stimano con più precisione il contributo marginale dei vicini prossimi alla soglia e trasformano guadagni locali in cascate globali. Secondo: quando il costo è proporzionale al grado (*threshold*), WTSS diventa la scelta naturale; l'algoritmo sfrutta la logica delle soglie residue e dei gradi residui per "forzare" attivazioni che le greedy trascurano ai budget ridotti.

Lettura dei comportamenti. Con costo *uniform* o *random*, le Greedy (in particolare f_3) capitalizzano rapidamente sui cluster locali; CENTRALITY offre un vantaggio solo quando il budget è troppo basso per innescare comunità numerose, sfruttando pochi nodi ponte per raggiungere porzioni lontane del grafo. Con costo *threshold*, l'onerosità degli hub rende inefficiente selezionarli: WTSS risulta competitivo anche ai budget bassi e resta in testa fino ai budget più alti, mentre f_3 diventa rilevante solo oltre una certa soglia.

Conclusione operativa. Nella maggior parte degli scenari pratici con costi *uniform* o *random*, le **greedy basate su f_3** rappresentano la scelta di riferimento; f_1 resta una valida opzione ai budget ridotti. Quando il costo è legato al grado (*threshold*), **WTSS** è preferibile lungo tutto il range di budget, con f_3 che torna competitiva solo in coda. La selezione dell'algoritmo va dunque guidata *dal modello di costo e dall'entità del budget*, più che da una preferenza universale per una singola tecnica.