Mínim conjunt d'influència a una xarxa social

Àlex Domínguez, Guillem Gaya, Fernando Guirao, Haroon Rehman

1 de maig de 2023

Algorísmia 2022-2023 Q2

Resum

El Target Set Selection [1] és un problema d'optimització combinatòria proposat per Kempe, Kleinberg i Tardos. L'objectiu es trobar un conjunt d'influència màxima (IM) a una xarxa social. Sigui $G = (V, E)$ el graf que representa una xarxa social, es vol trobar un conjunt S capaç de propagar la influència al màxim nombre de nodes de G. La propagació de la influència s'estudia sota el model de cascada independent (IC) i el model de llindar lineal (LT).

Aquest problema té dues variants: la variant de maximització (coneguda com a k-MAX-influence), on donat un nombre limitat de recursos (nodes) k es pretén maximitzar la propagació de l'influència. En canvi, a la variant de maximització (coneguda com a J-MIN-Seed) es pretén trobar el mínim subconjunt capaç de propagar l'influència a J nodes de la xarxa.

En aquest document s'estudia la segona variant del problema i es pretén proporcionar una solució mitjançant un algorisme greedy, una cerca local, i una metaheurística.

Índex

1 Introducció

El màrqueting viral estudia la propagació d'idees o productes. En aquest aspecte, l'objecte principal de recerca són les xarxes socials, que es poden modelar com a grafs on els individus o les organitzacions són els nodes, i les relacions o interaccions són els vèrtexs. Les xarxes socials tenen un paper destacat en molts camps científics com les ciències socials o la medicina.

Mitjançant el màrqueting viral s'intenta treure profit de les propietats de les xarxes socials per tal d'augmentar els ingressos a certes aplicacions comercials. Això es basa en la premissa de seleccionar uns pocs individus clau que puguin provocar un fort efecte "boca a boca"que impulsi una propagació de la influència a tota la xarxa.

Els primers en estudiar aquesta qüestió des d'un punt de vista algorísmic van ser Kempe et al. al [1], que van proposar el problema del TARGET SET SELECTION primerament en la variant de maximització (coneguda com a k-MAX-Influence), on donat un graf $G = (V, E)$ que representa una xarxa social, es vol trobar el subconjunt de k nodes que maximitzi la propagació de la influència a G.

Posteriorment Long et al. al [3] van estudiar la variant anàloga de minimització (coneguda com a J-MIN-Seed), on donat un graf $G = (V, E)$ que representa una xarxa social, es vol trobar el mínim subconjunt de nodes capaç de propagar la influència a J nodes de G.

La propagació de la influència s'estudia sota els models de cascada independent (IC) i llindar lineal (LT). Sota tots dos models i tant a la variació de maximització com a la variant de minimització s'ha provat que el TARGET SET SELECTION és un problema NP-Hard.

En aquest document s'estudia la variant de minimització coneguda com **J-MIN-Seed**.

2 Models de difusió

En aquesta secció s'expliquen els models de difusió sota els que es treballa, que són el model de cascada independent (IC) i el model de llindar lineal (LT). En aquest document s'apliquen algunes simplificacions als models de manera que direm que els grafs sota el model LT són deterministes, ja que la difusió sempre dona el mateix resultat, mentre que els grafs sota el model IC són probabilístics, ja que una mateixa difusió pot donar resultats diferents. Aquesta qüestió s'afronta a la secció

3 consideracioIC

.

Donada una xarxa social representada per un graf connex i no dirigit G, denotem que V és el conjunt que conté tots els nodes de G cadascun dels quals correspon a un individu i E és el conjunt que conté totes les arestes en G. Cada aresta $e \in E$ s'associa amb un pes w_{ij}. Els diferents models de difusió tenen diferents significats en els pesos. Al model IC el pes w_{ij} indica la probabilitat que té el node i d'influir el node j i viceversa (ja que G és no dirigit), mentre que al model LT indica la capacitat o força d'influència que té i sobre j i viceversa.

3.1 Cascada independet (IC)

El model de cascada independent o *independent cascade* (IC) és un model estocàstic on cada node actiu té una probabilitat p d'influir a cadascun dels seus veïns. Cada node fa un, i només un intent d'influir als seus veïns quan està actiu. Distingim llavors entre nodes actius i nodes influïts. Els nodes del subconjunt inicial S de difusió es troben actius inicialment i tracten d'influir els seus veïns, després deixen d'estar actius però romanen influïts durant la resta de la difusió. A continuació es defineix formalment el model:

Definició. Donat un graf $G = (V, E)$ que representa una xarxa, una probabilitat $p \in [0, 1]$, i un subconjunt inicial $S \subseteq V$, es denota el nombre total esperat de nodes influïts després de t passos sota el model IC com a $\sigma_t^p(S)$.

3.2 Llindar lineal (LT)

En el model de llindar lineal o *Linear Threshold* (LT), cada node té un llindar d'influència $\theta(i) = r \cdot deg(i)$ on $r \in [0, 1]$. Cada aresta té un pes w_{ij} que representa la influència del node i (si està influït) sobre el node j. Per simplicitat, el pes de totes les arestes és 1, llavors $\forall e \in E : w_{ij} = 1$. En aquest cas no distingim entre nodes i nodes actius. Tots els nodes influïts fan arribar una influència de 1 als seus veïns. Per tant, per exemple, si $r = 0.5$, un node és influït sii la meitat o més dels seus veïns es troben influïts, és a dir, si es compleix que:

$$\sum_{j \in S \wedge j \in N(i)} w_{ji} \geq \theta(i)$$

A continuació es defineix formalment el model:

Definició. Donat un graf $G = (V, E)$ amb pesos a les arestes tal que $\forall e \in E : w_{ij} = 1$ que representa una xarxa, un factor d'influència $r \in [0, 1]$ que defineix el llindar d'influència per a cada node i donat per l'expressió $\theta(i) = r \cdot deg(i)$ i un subconjunt inicial $S \subseteq V$, es denota el nombre total esperat de nodes influïts després de t passos sota el model IC com a $\sigma_t^r(S)$.

4 Definició del problema

Donat un conjunt S de nodes, definim la influència del conjunt S, denotada per $\sigma(S)$, com el nombre esperat de nodes influïts durant el procés de difusió iniciat per S.

Problema (J-MIN-Seed): donada una xarxa social $G = (V, E)$ i un enter J, trobeu un conjunt de S nodes tal que la mida de S es minimitzi i $\sigma(S) \geq J$. Diem que el node u està cobert pel conjunt S si u és influït durant el procés de difusió de la influència iniciat per S. És fàcil veure que el J-MIN-Seed té com a objectiu minimitzar el nombre de nodes de S i satisfer el requisit de cobrir almenys J nodes. Donat un node $x \in V$ i un subconjunt $S \subseteq V$, el guany marginal d'inserir x a S, denotat per $G_x(S)$, es defineix com $\sigma(S \cup x) - \sigma(S)$.

Noteu que per aquest problema, la variant de recobriment total és aquella on $J = |V|$.

5 Aproximació greedy basada en guany marginal

Kempe et al. presenten un algorisme d'aproximació greedy per a la variant de maximització del TSS (li direm KKT) que garanteix una propagació d'influència (1 -1/e) òptima. També mostren

que aquest algorisme millora significativament les heurístiques basades en grau i centralitat en la propagació d'influència.

No obstant l'algorisme té un gran inconvenient que és l'eficiència. A cada iteració, aquest algorisme greedy avalua afegir tots els nodes que no són a S. Cada cop que s'avalua afegir un node al subconjunt S l'algorisme ha de córrer la difusió sota la que s'està treballant, i a més en el cas de la difusió en cascada cal fer simulacions de Monte Carlo (més endavant es tracta aquesta qüestió) tants cops com sigui necessari per tal d'obtenir una estimació precisa de la propagació de l'influencia de cada subconjunt. Com a resultat d'això, trobar un subconjunt o *seed* petit a una xarxa gran (15000 nodes) podria portar dies.

Al [2] Leskovec et al. presenten una optimització en la selecció de nous nodes, que es coneix com l'esquema "Cost-Effective Lazy Forward"(CELF). L'optimització CELF utilitza la propietat de submodularitat a la funció de propagació de la influència per reduir el nombre d'avaluacions sobre la propagació de la influència dels vèrtexs.

La submodularitat de la funció $\sigma(\cdot)$ es demostra al [3] i diu que el guany marginal d'un nou node es redueix a mesura que el conjunt creix. La funció $\sigma(\cdot)$ és submodular sii $\sigma(S \cup v) - \sigma(S) \geq \sigma(S \cup v) - \sigma(S)$ sempre que $S \subseteq T$. Això implica que quan s'afegeix un vèrtex v a un conjunt S, el guany marginal $G_v(S)$ com a resultat d'afegir v es fa més gran quan S es fa més petit. L'optimització CELF utilitza la submodularitat de manera que en cada iteració el guany marginal dels nodes no cal reavaluar-se perquè el guany marginal d'un node en la iteració actual no pot ser millor que el seu guany marginal en les iteracions anteriors.

Els resultats experimentals mostrats per Leskovec et al. demostren que l'optimització CELF podria aconseguir una velocitat de fins a 700 vegades en la selecció de vèrtexs de llavors, que és un resultat molt significatiu.

Basant-nos en l'algorisme CELF, proposem un algorisme greedy que troba un subconjunt S afegint nodes de forma iterativa.

Algorithm 1 Algorisme greedy guany marginal

Entrada: $G = (V, E)$: la xarxa social, J: el nombre mínim de nodes a influir
Sortida: S: conjunt *seed*

1: $S \leftarrow \emptyset$
2: $Q \leftarrow \emptyset$
3: **for** $v \in V$ **do**
4: $\quad Q.insert(v, G_v(S))$
5: **end for**
6: $S \leftarrow S \cup \{Q.top()\}$
7: $Q.pop()$
8: **while** $\sigma(S) < J$ **do**
9: $\quad currentNode \leftarrow Q.top()$
10: $\quad$ **if** $currentNode \notin S$ **then**
11: $\quad\quad S \leftarrow S \cup \{currentNode\}$
12: $\quad$ **end if**
13: **end while**
14: **return** S

Aquest algorisme inicialitza el conjunt S buit, i calcula el guany marginal inicial de cada node v, és a dir, el nombre de nodes que són capaços d'influir per si sols. Els nodes són emmagatzemats a una cua de prioritat Q que manté l'ordre segons el seu guany marginal de forma decreixent, de manera que al top de la cua sempre queda el node més "prometedor". Després s'agafen de manera iterativa els nodes de la cua de prioritat i es van afegint al conjunt S, alhora que es comprova si la propagació de S arriba a J o més nodes. Noteu que la condició de finalització del bucle només canvia respecte a la variant de maximització en que és $\sigma(S) \geq J$ enlloc de $\sigma(S) \geq k$. En analitzar la complexitat de l'algorisme, per una banda, veiem que el for de la línia 3 realitza $|V|$ iteracions, d'una funció que en cas pitjor és $O(n^3)$. Per tant, el cost global en cas pitjor és $O(|V|^4)$. Per altra banda, el cost del while de la línia 8, només depèn del nombre d'iteracions que pot arribar a fer, que seria si J = $|V|$. Com a resultat, el còmput total seria $O(|V|^4)$.

5.1 Consideració pel model IC

Degut a la naturalesa probabilística del model de cascada independent, a l'implementació de l'algorisme greedy sota aquest model es decideixen definir dos paràmetres:

`optimality`: és un real entre 0 i 1, idefineix la proporció de vèrtexs de G que volem arribar a influir amb S. L'algorisme per defecte defineix aquest paràmetre en 1, per tant en aquest cas estem definint el problema J-MIN-Seed on $J = |V|$. Llavors, per exemple, si definim el paràmetre en 0.5, estem definint el problema J-MIN-Seed on $J = |V|/2$.

`nMonteCarlo`: defineix el nombre d'iteracions que s'executa la difusió en cascada cada vegada que es fa a l'algorisme. Utilitzar el mètode Monte Carlo ens permet estimar una propagació $\sigma(S)$ més precisa. És evident que quantes més iteracions es facin, augmentem el cost temporal de l'algorisme. Long et al. defineixen al [3] un nombre d'iteracions a partir de la desigualtat de Hoeffding amb el que s'aconsegueix una $(1 \pm e)$ aproximació de $\sigma(S)$. Aquest nombre d'iteracions ve donat per la següent expressió, on c és un nombre real entre 0 i 1:

$$\frac{(|V| - 1)^2 \ln \frac{2}{1-c}}{2e^2|S|^2}$$

6 Aproximació greedy basada en grau (model LT)

Durant la implementació de l'algorisme greedy per aquest projecte, abans de proposar-se una versió basada en el guany marginal es va provar una implementació basada en el grau dels nodes. Aquest criteri local tria com a candidats als nodes amb grau més elevat, ja que són els més difícils d'influenciar. Si després de triar un node no s'ha pogut influenciar tot el graf, es tria el següent node amb grau més elevat que no s'hagi influenciat en la simulació de difusió anterior. Al ser un criteri tan bàsic, dubtàvem de que trobés un conjunt proper al mínim, no obstant, sabíem que el seu cost no seria molt alt, ja que només s'aplicaria la simulació de difusió una vegada per cada node del conjunt S, per tant el cost en cas pitjor és: $O(|S|(V+E))$.

Algorithm 2 Algorisme greedy grau més gran

Entrada: $G = (V, E)$: la xarxa social, J: el nombre mínim de nodes a influir
Sortida: S: conjunt *seed*

```
1: S ← ∅
2: Q ← ∅
3: for v ∈ V do
4:    Q.insert(v, deg(v))
5: end for
6: S ← S ∪ {Q.top()}
7: Q.pop()
8: while σ(S) < J do
9:    currentNode ← Q.top()
10:   if currentNode ∉ S ∧ currentNode ∉ Bool(S) then
11:      S ← S ∪ {currentNode}
12:   end if
13: end while
14: return S
```

Com podem observar en el pseudocodi, la única diferència amb el *Algorithm*1 és que la selecció de nodes es basa amb el seu grau en comptes del guany marginal, i al ser una cua de prioritat sempre rebrem primer aquells nodes amb grau major.

7 Cerca local

Per millorar la solució inicialment generada per l'algorisme greedy, proposem un algorisme que implementa una estratègia *Hill Climbing* simple, basada en *first improvement*. L'elecció d'implementar *Hill Climbing* en la variant de *first improvement* enlloc de *best improvement* és degut a la manera en que realitzem la cerca local en aquest cas.

Per descriure l'exploració que fa l'algorisme, primer definim que cada estat (solució) es composa simplement del conjunt de nodes escollits per al subconjunt S de G. L'exploració a través de l'espai de solucions es fa mitjançant l'operador de treure node i una heurística que simplement mesura la mida del subconjunt S a cada estat (en aquest cas l'heurística coincideix amb la funció objectiu a minimitzar). D'aquesta manera, tots els estats successors que siguin solució (és a dir, cobreixin els nodes demanats) tenen la mateixa heurística i per tant només cal seleccionar el primer successor que sigui solució, i continuar la cerca local a partir d'aquest.

L'utilització de *first improvement* també comporta beneficis en termes de memòria, ja que no es necessari emmagatzemar tots els estats successors per seleccionar-ne el millor, donat que

simplement seleccionant el primer que millora la solució de l'estat pare (és a dir, és solució amb un node menys) ja estem seleccionant el millor successor.

Si estudiem la complexitat de l'algorisme, ens adonem que el for, de la línia 4, és el que determina el nombre d'iteracions que es faran en aquest algorisme, totes les altres instruccions són de cost constant. A com el que fem es anar tractant $node \in S$, aquest bucle en cas pitjor serien $|V|$ iteracions, ja que $S \subseteq V$. Per tant, la complexitat és O($|V|$).

Algorithm 3 *First improvement hill climbing*

Entrada: $G = (V, E)$: la xarxa social
J: el nombre mínim de nodes a influir
S: subconjunt solució inicial
Sortida: S: subconjunt solució final

```
1:  Improving ← true
2:  while Improving do
3:    Improving ← false
4:    for node ∈ S do
5:      S' ← S' − {node}
6:      if σ(S') = σ(S) then
7:        S ← S'
8:        Improving ← true
9:        BREAK
10:     end if
11:   end for
12: end while
13: return S
```

8 Metaheurística

L'algorisme de la secció anterior ens proporciona una millor solució a partir de la solució generada per l'algorisme greedy, però aquesta solució millorada es correspon amb un òptim local que en la majoria dels casos no serà l'òptim global. Per poder ser capaços de cercar l'òptim global implementarem un algorisme metaheurístic. En aquest cas optem per un *simulated annealing*.

Definim l'estat novament com el subconjunt de nodes escollits per a S. El *simulated annealing* permet cercar l'òptim global ja que pot acceptar estats solució pitjors amb una certa probabilitat. Per poder generar solucions pitjors, a part de l'operador de treure node cal afegir l'operador d'afegir node. En aquest cas definim l'heurística com la relació entre els nodes influïts pel subconjunt S i la mida del subconjunt S, és a dir:

$$h(S) = \frac{\sigma(S)}{|S|}$$

A diferència de l'algorisme anterior de *hill climbing*, en aquest algorisme de *simulated annealing*, a cada iteració s'afegeix o elimina un node aleatori a S i es genera un nou subconjunt S'. S'avalua $h(S)$ i si la diferència (Δ) entre $h(S')$ i $h(S)$ és positiva es continua la cerca a partir de S', sino S' pot ser acceptat amb una una probabilitat donada per una funció que definim més endavant. La probabilitat d'acceptació és més alta quan la temperatura T és alta. La temperatura és defineix al principi de forma arbitrària i va baixant a mesura que augmenta el nombre d'iteracions de l'algorisme, també definides de forma arbitrària. El valor de la temperatura ve donat per la següent expressió, on α és un real entre 0 i 1 que també es defineix de forma arbitrària:

$$T(iter) = \alpha \cdot T(iter - 1)$$

La funció que defineix la probabilitat d'acceptació d'un estat solució pitjor a una certa iteració, ve donada per l'expressió:

$$P(S', iter) = e^{\Delta/T(iter)}$$

Per tant, en pensar en la complexitat d'aquesta solució, no podem estudiar-la com les anteriors, ja que depenem del nombre d'iteracions que fem, a més del component aleatori que té aquesta solució.

Algorithm 4 *Simulated annealing*

Entrada: $G = (V, E)$: la xarxa social
J: el nombre mínim de nodes a influir
S: subconjunt solució inicial
Sortida: S: subconjunt solució final

1: $bestSolution \leftarrow S$
2: **while** $iter > 0$ **do**
3: $S' \leftarrow S$
4: $randomNode = randomNode(G)$
5: **if** $randomNode \in S'$ **then**
6: $S' \leftarrow S' - \{randomNode\}$
7: **else**
$S' \leftarrow S' \cup \{randomNode\}$
8: **end if**
9: **if** $\Delta > 0$ or $P(S', iter)$ **then**
10: $S \leftarrow S'$
11: **if** $h(S) > h(bestSolution)$ **then**
12: $bestSolution \leftarrow S$
13: **end if**
14: **end if**
15: $-iter$
16: **end while**
17: **return** $bestSolution$

9 Consideració pel model LT

La implementació de la generació de l'estat successor està modificada per a aquest model. La modificació consisteix en que el 75% de les vegades que es genera l'estat successor es fa esborrant un node aleatori de S, i el 25% restant, s'afegeix un node aleatori de G que no pertanyi a S. A aquesta proporció s'ha arribat de manera experimental. Aquesta decisió es pren donat que amb el comportament original de l'algorisme, com l'algorisme greedy original selecciona ja un subconjunt S prou reduït, és poc probable reduir més el subconjunt.

10 Experimentació

Tot el procés d'experimentació s'ha dut a terme en un mateix ordinador, tot i que el procés ha estat diferent en base al model.

L'elecció dels grafs per a l'experimentació s'ha fet en base al seu nombre d'arestes, on hem buscat experimentar amb grafs petits, mitjans i grans. Si l'algorisme tardava massa amb un graf a causa del seu nombre d'arestes, descartàvem el graf pels algorismes més lents ja que sinó l'experimentació hagués estat massa costosa.

Nom grafs	nodes	arestes
Karate	62	159
Jazz	198	2742
socfb-nips-ego	2888	2981
CA-GrQc	5242	14484
CA-HepTh	9877	25973
soc-gplus	23628	39194
socfb-Mich67	3748	81903
musae-git	37700	289003
deezer-HR	54573	498202
gemsec_facebook_artist	50515	819090

10.1 Experimentant sota el model IC

L'experimentació amb els algorismes implementats sota el model IC és una tasca més complexa. Principalment cal ajustar bé els paràmetres d'optimalitat i nombre d'iteracions de Monte Carlo, comentats a la secció 5.1.

Durant l'experimentació vam treure diverses conclusions sobre aquests dos paràmetres. Es va observar que donar valor 1 al paràmetre d'optimalitat (és a dir, trobar un subconjunt capaç d'influir en tot el graf durant totes les iteracions de Monte Carlo) comportava l'elecció de pràcticament tos els nodes per part dels algorismes per tal d'assegurar la propagació a tot el graf. Davant aquesta situació, es decideix fixar l'optimalitat en 0.99. Per tant les solucions obtingudes asseguren cobrir el **99% dels nodes del graf**.

Pel que fa al nombre d'iteracions de Monte Carlo, aquest és un paràmetre que augmenta molt significativament el cost temporal. Hem tractat de fixar un nombre alt d'iteracions per assegurar qualitat a les solucions però no sempre ha estat possible, per això a partir de grafs grans no hem pogut posar un gran nombre d'iteracions.

Els primers 3 grafs s'han fet amb nMontecarlo = 100, ja que són grafs petits, els altres estan fets amb 10.

	Greedy IC			Local Search IC			Metaheuristica IC		
Nom grafs	Best	%	Time (s)	Best	%	Time (s)	Best	%	Time (s)
Karate	56	90%	0,003	45	73%	0,004	30	48%	2,203
Jazz	196	99%	3,937	133	67%	13,363	96	48%	20,161
socfb-nips-ego	2834	98%	87,438	2829	98%	91,828	2110	73%	60,442
soc-gplus	23052	98%	116,227	23015	97%	343,725	22251	94%	71, 172
socfb-Mich67	3244	87%	20,851	2904	77%	225,236	N/A	N/A	N/A
musae-git	36571	97%	1.136,661	N/A	N/A	N/A	N/A	N/A	N/A
deezer-HR	52798	97%	3.076,234	N/A	N/A	N/A	N/A	N/A	N/A
	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

10.2 Experimentant sota el model LT

Com que en el model LT no hi ha probabilitat, d'una execució a un altre el conjunt S sempre és el mateix, la única diferencia que es troba és en el temps d'execució. Així que per a calcular el temps s'ha decidit fer la mitjana ponderada de 10 execucions, ja que la diferència de temps entre execucions era tan petita que fer-ne moltes no ens hagués aportat gaire informació.

	Greedy LT - Guany Marg.			Local Search LT - Guany Marg.		
Nom grafs	Best	%	Time (s)	Best	%	Time (s)
Karate	12	19%	0	10	16%	0
Jazz	54	27%	0,004	37	19%	0,065
socfb-nips-ego	10	0%	0,003	10	0%	0,006
CA-GrQc	1099	10%	1,012	976	19%	11,519
CA-HepTh	1628	6%	2,987	1397	14%	31,128
soc-gplus	68	0%	0,728	64	0%	1,234
socfb-Mich67	1020	27%	1,42	640	17%	40,399
musae-git	441	1%	5,725	369	1%	64,316
deezer-HR	9251	17%	70,977	5952	11%	2720,29
gemsec_facebook_artist	4067	8%	75,963	N/A	N/A	N/A

	Greedy LT - Grau max			Local Search LT - Grau Max		
Nom grafs	Best	%	Time (s)	Best	%	Time (s)
Karate	8	13%	0	6	10%	0
Jazz	30	15%	0,003	26	13%	0,222
socfb-nips-ego	10	0%	0,003	10	0%	0,006
CA-GrQc	1031	28%	1,347	922	18%	11,162
CA-HepTh	1388	4%	3,617	1236	13%	28,251
soc-gplus	69	0%	0,263	62	0%	1,389
socfb-Mich67	203	5%	0,249	184	5%	6,931
musae-git	196	1%	2,028	188	0%	22,824
deezer-HR	2373	4%	12,597	2000	4%	651,545
gemsec_facebook_artist	788	2%	12,586	704	1%	332,242

	Metaheuristica LT - Grau max		
Nom grafs	Best	%	Time (s)
Karate	7	11%	0,211
Jazz	22	11%	12,923
socfb-nips-ego	10	0%	0,4
CA-GrQc	1029	20%	2,837
CA-HepTh	1384	14%	5,428
soc-gplus	61	0%	59,135
socfb-Mich67	200	5%	75,817
musae-git	191	1%	310,778
deezer-HR	2311	4%	690,235
gemsec_facebook_artist	777	2%	1006,64

11 Glossari

A continuació es mostra un glossari amb les notacions més utilitzades en aquest document:

Notació	Descripció
$G = (V, E)$	Xarxa d'influència amb un conjunt de nodes V i un conjunt d'arestes E.
p	Probabilitat de propagació sota el model IC
r	Factor d'influència sota el model LT
S	Subconjunt inicial de difusió (*seed*)
$\sigma^p(S)$	Nodes influïts per un subconjunt S sota el model IC
$\sigma^t(S)$	Nodes influïts per un subconjunt S sota el model LT
$Bool(S)$	vector de booleans amb els nodes que s'activen en la difusio LT
$\sigma(S)$	Nodes influïts per un subconjunt S sota qualsevol model
$G_x(S)$	Guany marginal d'inserir x a S, el cost en cas pitjor és n^2.
$h(S)$	Heurística del estat definit per un subconjunt S
$T(iter)$	Temperatura del *simulated annealing* a l'iteració *iter*
α	Factor de descens de la temperatura al *simulated annealing*

Referències

[1] Kleinberg, J., & Tardos, E. (2003). Maximizing the Spread of Influence through a Social Network.

[2] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 420–429, 2007.

[3] C. Long and R.C.-W. Wong. Minimizing seed set for viral marketing. In 2011 IEEE 11th International Conference on Data Mining, pages 427–436. IEEE press, 2011.