SDRC 2013/2014 — Sistemi Distribuiti e Reti di Calcolatori — P. Crescenzi — Alessandro Venturi

1 Algoritmo veloce per la determinazione dell'insieme dominante.

Nella lezione precedente è stato visto un algoritmo sequenziale per il calcolo di un Insieme Dominante (ID) approssimato con implementazione distribuita. Come si è visto questo algoritmo può risultare lento con alcune configurazioni, come nel caso di cammini di clique di grado decrescente, per cui viene impiegato un tempo lineare rispetto al numero di nodi. In realtà l'ID in questa clique può essere calcolato in un tempo minore.
In questa lezione è presentato un algoritmo con prestazioni migliori rispetto a quelli precedentemente visti. Il punto chiave è fare in modo tale che più nodi, contemporaneamente, diventino dominatori senza interferire tra di loro ed aumentare le dimensioni dell'ID.

Prima di iniziare, per il nostro grafo di grado Δ e con n nodi, definiamo:

Definizione 1.1 ($N(v)$ **):** Si definisce $N(v)$ come l'insieme dei nodi a distanza 1 da v, compreso v.

Definizione 1.2 ($N_2(v)$ **):** Si definisce $N_2(v) = \cup_{u \in N(v)} N(u)$ che rappresenta il numero di vicini a distanza 2 di v.

Definizione 1.3 ($W(v)$ **):** Si definisce $W(v) = \{u \in N(v) : u \text{ bianco}\}$. $W(v)$ ovvero è l'insieme dei nodi bianchi vicini di v.

Definizione 1.4 ($w(v)$ **):** Identifichiamo come $w(v) = |W(v)|$ la cardinalità di $W(v)$.

Per permettere ai nodi di essere selezionati in parallelo verrà fatto in modo che questi venghino selezionati in base ad un processo stocastico. Quello che vogliamo far vedere è che in questo modo non si peggiora l'insieme dominante e si riesce a diminuire il tempo di esecuzione.
In particolare, ipotizzando che ogni nodo possa calcolare $w(v)$ con un messaggio ai vicini che inizialmente sono bianchi, l'algoritmo è il seguente:

Algorithm 1: Fast Distributed Dominating Set Algorithm (per il nodo v):

```
1  begin
2      W(v) = N(v); w(v) = |W(v)|                          //Inizializzazione
3      while W(v) ≠ ∅ do
4          w̃(v) = 2^⌊log2 w(v)⌋
5          ŵ(v) = max_{u∈N2(v)} w̃(u)
6          if w(v) = w̃(v) then
7              v.attivo=TRUE
8          s(v) = |{u ∈ N(v) : u.attivo = TRUE }|           //Supporto di v. Numero dei vicini di v che sono attivi.
9          ŝ(v) = max_{u∈N(v)} s(u)
10         if v.attivo then
11             v.candidato=TRUE con probabilità 1/ŝ(v)
12         c(v) = |{u ∈ W(v) : u.candidato = TRUE}|        //Numero di nodi bianchi candidati visti da v.
13         if v.candidato=TRUE e C(v)=Σ_{u∈W(v)} c(u) ≤ 3w(v) then
14             v entra nell'ID.
15         W(v) = {u ∈ N(v) : u è bianco }; w(v)=|W(v)|     //Aggiorna W(v) e w(v)
```

Dell'algoritmo si può notare che il modo in cui viene preso $\widetilde{w}(v)$ risolve il problema dei cammini con copertura

discendente, perchè tutti quelli che sono nello stesso intervallo li identifichiamo con un unico valore. Inoltre più c'è interferenza tra i nodi, cioè più è alto $\hat{s}(v)$, meno probabilità ha v di essere candidato. Con questo algoritmo più nodi possono entrare contemporaneamente nell'insieme dominante, anche se vicini.
Per il precedente algoritmo si può dimostrare che:

Teorema 1.1: *L'algoritmo Fast Distributed Set Algorithm costruisce un ID di dimensione al più* $6\log(\Delta+1)$

Dimostrazione: L'idea della dimostrazione è quella dell'algoritmo goloso della precedente lezione, cioè ogni volta che si aggiunge un nodo v si distribuisce il costo tra v (se è bianco) ed i suoi vicini bianchi. Consideriamo un ID ottimo che possiamo vedere come un insieme di stelle. Ogni nodo appartiene alla stella che lo domina e nel caso in cui un nodo appartenga a più stelle, ne viene scelta una.

Sia v^* un nodo che fa parte della soluzione ottima, il quale domina alcuni nodi in una stella. Supponiamo che il nodo v venga aggiunto all'ID ad una certa iterazione con un certo $W(v)$ sottoinsieme di $N(v)$. Per un nodo $u \in w(v)$ sia $c(u)$ il numero di candidati bianchi in $N(u)$. Definiamo inoltre:

$$C(v) = \sum_{u \in W(v)} c(u) \tag{1}$$

dove necessariamente $C(v) \leq 3w(v)$ perchè altrimenti v non sarebbe stato aggiunto all'ID (vedi Algoritmo1, linea 13). Il costo che viene attribuito a u è $\dfrac{3}{c(u)w(v)}$. Con questa assegnazione riusciamo a coprire il costo 1 per l'aggiunta di v nell'ID facendo si che il costo sia non minore rispetto a quello dell'algoritmo greedy. Dobbiamo quindi far vedere che:

$$\sum_{u \in W(v)} \frac{3}{c(u)w(v)} \geq 1 \tag{2}$$

Se vale il maggiore stretto è meglio al fine dell'analisi in quanto ci mettiamo in un caso più pessimista. Svolgendo otteniamo:

$$\sum_{u \in W(v)} \frac{3}{c(u)w(v)} = 3 \sum_{u \in W(v)} \frac{1}{c(u)w(v)} = 3\frac{1}{w(v)} \sum_{u \in W(v)} \frac{1}{c(u)} \tag{3}$$

Un risultato che ci viene in aiuto è il seguente:

$$\sum_{i=1}^{k} \frac{1}{a_i} \geq \frac{k}{\sum_{i=1}^{k} \frac{a_i}{k}} \qquad \forall i \; a_i > 0 \tag{4}$$

Riportandoci al nostro caso abbiamo che: $a_i = c(u)$ *e* $k = w(v)$. Si ha quindi:

$$3\frac{1}{w(v)} \sum_{u \in W(v)} \frac{1}{c(u)} \geq 3\frac{1}{w(v)} \frac{w(v)}{\sum_{u \in W(v)}^{k} \frac{c(u)}{w(v)}} = \frac{3w(v)}{C(v)} \geq 1 \tag{5}$$

in quanto $C(v) \leq 3w(v)$.
Con questo valore riusciamo a coprire il costo che, come abbiamo detto, doveva essere almeno 1.

Scegliamo adesso un nodo z che sta nella stella, e supponiamo che ad un certo passo questo è dominato da un nodo u, vicino bianco di z (come in Figura 1). Se z è attribuito a u, il suo costo è $\dfrac{3}{c(z)w(u)}$ che vale anche per ogni nodo u_i che cerca di entrare nell'ID nello stesso passo.
Dal momento che un nodo può solo unirsi all'ID se $w(u_i) \geq \dfrac{w(v^*)}{2}$ (dove il fattore 2 è dovuto all'arrotondamento), segue che il costo che viene attribuito ad un nodo è minore o uguale a $\dfrac{6}{c(z)w(v^*)}$. Dato che solo i candidati visti

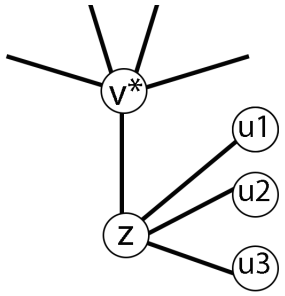

Figura 1: Rete all'istante t. Nodo z appartenente alla stella di v^* e dominato anche dai nodi $u1$, $u2$ e $u3$.

da z possono entrare a far parte dell'ID, segue che complessivamente il costo attribuito a z è minore o uguale a $C(z)\frac{6}{C(z)w(v^*)} = \frac{6}{w(v^*)}$

Adesso, ragionando come per l'algoritmo non probabilistico, si ottiene:

$$|ID| = \sum_{i=1}^{|N(v^*)|} \frac{6}{i} \leq 6H(|N(v^*)|) \leq 6H(\Delta+1) \leq 6ln(\Delta+1) \tag{6}$$

□

Per ora non abbiamo usato l'ipotesi di casualità della scelta dei candidati. La probabilità entra in gioco per il calcolo del tempo medio.

Teorema 1.2: *L'algoritmo probabilistico per l'insieme dominante termina in* $T = O(\log^2 \Delta \log n)$

Prima di proseguire si dimostra un risultato intermedio.

Lemma 1.1: *Supponiamo di essere in un'iterazione del ciclo while, ed il nodo u è bianco con:*

$$s(u) \geq max_{v \in F(u)} \frac{\hat{s}(v)}{2} \tag{7}$$

dove $F(u) = \{z \in N(u) : z.candidato = TRUE\}$*, cioè* $F(u)$ *è l'insieme dei nodi attivi candidati vicini di u.*
Segue che il nodo u diventa grigio o nero con probabilità $\geq \frac{1}{9}$.

Dimostrazione: Consideriamo $D(u)$ come l'evento che u diventi grigio o nero. La probabilità di $D(u)$ si può scomporre in due parti, considerando l'evento $D(u)$ condizionato da $c(u)$ e dal suo complementare.

$$Pr[D(u)] = Pr[D(u)|c(u) > 0]\ Pr[c(u) > 0] + Pr[D(u)|c(u) = 0]\ Pr[c(u) = 0] \tag{8}$$

Dato che $Pr[D(u)|c(u) = 0]\ Pr[c(u) = 0] = 0$ perchè per ipotesi si ha almeno un candidato, risulta:

$$Pr[D(u)] = Pr[D(u)|c(u) > 0]\ Pr[c(u) > 0] \tag{9}$$

La probabilità che un vicino si candidi è Pr[c(u) > 0] = $\frac{1}{\hat{s}(z)}$. Per le assunzioni fatte si ha:
$\frac{1}{\hat{s}(z)} \geq \frac{1}{2s(u)}$. Quindi, la probabilità che nessuno dei vicini di u si candidi è: $Pr[c(u) = 0] \leq (1 - \frac{1}{2s(u)})^{s(u)} < \frac{1}{\sqrt{e}}$
da cui segue:

$$Pr[c(u) > 0] > 1 - \frac{1}{\sqrt{e}} > \frac{1}{3} \tag{10}$$

Adesso facciamo vedere che $Pr[D(u)|c(u) > 0] > \frac{1}{3}$ in modo che $Pr[D(u)] > \frac{1}{9}$. Abbiamo:
$Pr[D(u)|c(u) > 0] \geq min_{v \in N(u)} Pr[v$ entri in ID$|$ v.candidato = TRUE]
Vediamo se riusciamo a dare un bound. Chiamiamo:
$C(u) = \sum_{v' \in W(v)} c(v')$
Se v è un candidato allora entra in ID se $C(v) \leq 3w(v)$. Ci riconduciamo a stimare: $Pr[C(v) \leq 3$w(v)]
Il valore medio:
$E[c(v')|v$ candidato = TRUE $] = 1 + E[c(v') - 1] \leq 1 + \frac{s(v') - 1}{s(v')} < 2$
Nel penultimo passaggio sostanzialmente stiamo considerando un processo stocastico dove: $s(v') - 1$ è il numero di tentativi che si possono fare nella ricerca dei candidati tra i vicini di v', e $\frac{1}{s(v')}$ è la probabilità di successo di ogni tentativo.
Consideriamo adesso $E[C(v)|v.candidato = TRUE]$. Dato che ciascun nodo nei vicini bianchi ha valore atteso ≤ 2 e dato che questi in numero sono pari a $w(v)$, per la linearità del valore atteso, otteniamo:
$E[C(v)|v.candidato = TRUE] < 2w(v)$
A questo punto limitiamo, mediante Markov, la probabilità di andare oltre $2w(v)$:
$\Pr[C(v) > 3w(v)] < \frac{2}{3}$. Da cui segue: $Pr[C(v) \leq 3w(v)] \geq \frac{1}{3}$.
Si ha quindi: $Pr[D(u)|c(u) > 0] > \frac{1}{3}$ ed infine: $Pr[D(u)] > \frac{1}{9}$
□
A questo punto, sfruttando questa proprietà, possiamo valutare il numero di passi dell'algoritmo.

Dimostrazione: Supponiamo di essere al tempo t e di selezionare $s_{max}(t)$, il massimo s al tempo t e $\widetilde{w}_{max}(t)$, il massimo $\widetilde{w}$ al tempo t. Si può notare che, $\widetilde{w}_{max}$ può essere uguale in istanti di tempo differenti e in questi istanti s_{max} non può aumentare. Questo succede perchè al passare del tempo alcuni nodi possono non essere più attivi.
Consideriamo le fasi in cui $\widetilde{w}_{max}$ rimane uguale per un certo lasso di tempo. Per un $\widetilde{w}_{max}$ avrò quindi più s_{max} che al passare del tempo vanno a diminuire. Prima o poi avremo un $\widetilde{w}'_{max} < \widetilde{w}_{max}$ perchè stiamo comunque continuando ad inserire nodi nell'ID. Anche per $\widetilde{w}'_{max}$, per un certo lasso di tempo, avremo s'_{max} che decresce al passare del tempo.
Osserviamo che il numero di passi per arrivare a $\widetilde{w}_{max} = 0$ è $\log\Delta$ perchè si diminuisce di potenze di 2 ogni volta. In ogni fase in cui $\widetilde{w}_{max}$ è costante si può dimezzare s_{max} al più $\log\Delta$ volte. Se adesso facciamo vedere che dimezzare s_{max} si può fare in tempo $\log n$ abbiamo dimostrato il teorema.

Consideriamo il nodo u tale che:
(1) u è bianco
(2) $\exists v \in N(u) : w(v) \geq \widetilde{w}_{max}$
(3) $s(u) \geq \frac{s_{max}}{2}$
Se u soddisfa la (3), dal lemma precedente, potrebbe essere selezionato con probabilità almeno $\frac{1}{9}$. E quindi al passo successivo potrebbe non essere più bianco, cioè:
$Pr[u$ non sia bianco al passo successivo$] \leq \frac{8}{9}$
e generalizzando a τ passi: $Pr[u$ non sia bianco ai τ passi successivi$] \leq \left(\frac{8}{9}\right)^{\tau}$.
Se ora si sceglie $\tau = \log_{\frac{9}{8}} 2n$ abbiamo che la $Pr[u$ non sia bianco ai τ passi successivi$] \leq \frac{1}{2n}$.
Per ogni u, facendo l'Union Bound, la probabilità che sia ancora soddisfatta la (3) è $\leq \frac{1}{2}$.
Se nessun nodo la soddisfa vuol dire che s_{max} si è dimezzata. Se adesso prendiamo $\tau = k\log_{\frac{9}{8}} 2n$ otteniamo:
$Pr[u$ non sia bianco ai τ passi successivi$] \leq \frac{1}{2^k}$ che possiamo rendere arbitrariamente piccola aumentando k.
Quindi con probabilità arbitrariamente piccola, aumentando il numero di passi, avremo $s(u) < \frac{s_{max}}{2}$ per tutti i nodi, e quindi s_{max} si dimezza. Dalle considerazioni fatte sui tempi, dato che di passi per dimezzare s_{max} ne abbiamo $O(\log\Delta)$ totali per s_{max} e $O(\log\Delta)$ totali per w_{max} si ottiene infine: $T \cong O(\log^2\Delta\log n)$.

□

A seguire la dimostrazione di (4)

Dimostrazione: Dimostriamo che: $\sum_{i=1}^{k} \frac{1}{a_i} \geq \frac{k}{\sum_{i=1}^{k} \frac{a_i}{k}} \quad \forall i \; a_i > 0$

La disequazione può essere riscritta come $\sum_{i=1}^{k} \frac{1}{a_i} \geq \frac{k^2}{\sum_{i=1}^{k} a_i}$

Moltiplichiamo e dividiamo per il reciproco del secondo membro, ottenendo $\frac{\sum_{i=1}^{k} \frac{1}{a_i} \sum_{i=1}^{k} a_i}{k^2} \geq 1$.

Sviluppando le sommatorie al primo membro si ha $\frac{(a_1 + a_2 + ... + a_k) \cdot (a_1^{-1} + a_2^{-1} + ... + a_k^{-1})}{k^2}$

Possiamo notare che al numeratore ogni a_i viene moltiplicato per il suo reciproco e dato che $a_i \cdot a_i^{-1}$=1 $\forall i$, il loro contributo è pari a k. Gli altri termini del prodotto si possono sommare a due a due $\frac{a_i}{a_j} + \frac{a_j}{a_i}$. Essendo $\frac{a_i}{a_j}$ e $\frac{a_j}{a_i}$ due numeri reali e positivi l'uno reciproco dell'altro, si può dimostrare che

$$\frac{a_i}{a_j} + \frac{a_j}{a_i} \geq 2 \tag{11}$$

Le possibili combinazioni che possiamo avere sono $\binom{k}{2}$. Utilizzando l'uguaglianza del precedente risultato come bound, otteniamo:

$$\frac{k \; + \; 2 \cdot \binom{k}{2}}{k^2} = \frac{k \; + \; 2 \cdot \frac{k!}{2 \cdot (k-2)!}}{k^2} = \frac{k \; + \; k \cdot (k-1)}{k^2} = \frac{1 \; + (k-1)}{k} = 1 \tag{12}$$

□

Dimostrazione: Dimostriamo la (11) che possiamo anche scrivere come:

$$n + \frac{1}{n} \geq 2 \qquad n \in \mathbb{R}_+ - \{0\} \tag{13}$$

Moltiplichiamo per n il primo ed il secondo membro: $n^2 + 1 \geq 2n$ che possiamo riscrivere come $n^2 + 1 - 2n \geq 0$ cioè $(n-1)^2 \geq 0$ che è sempre vera.
□