SDRC 2013/2014 — Sistemi Distribuiti e Reti di Calcolatori — P. Crescenzi — Antonio Passaro

1 Clustering e minimo insieme dominante

Definizione 1.1 (Clustering):

Il clustering é un processo consistente nel raggruppamento di nodi in sottoinsiemi. Tale operazione é tipica negli algoritmi che vengono utilizzati nei sistemi distribuiti in quanto consente di rappresentare un grafo in modo piú sintetico senza perdere proprietá generali e utili. Tale tecnica consente inoltre di progettare algoritmi che, demandando le loro funzioni ai vari gruppi di nodi, hanno buone prestazioni in termini di complessitá e di efficienza. L'operazione di clustering viene per esempio applicata comunemente alle reti dei sensori; infatti all'interno di questa rete i nodi(sensori) hanno il compito di raccogliere i dati proveniente dall'ambiente e di trasmetterli a una stazione che poi successivamente li elabora. Piuttosto che costringere tutti i nodi della rete a trasmettere i loro dati alla stazione che potrebbe essere molto lontana(e questo richiederebbe un consumo non indifferente per tutti i sensori), é possibile raggruppare i sensori in cluster scegliendo un sottoinsieme di nodi in modo tale che , ogni altro nodo della rete, é stato scelto per fare parte di tale sottoinsieme oppure è vicino di un nodo che pié stato scelto. A questo punto un sensore può inviare i dati a se stesso(se é stato scelto) o al nodo che fa parte del sottoinsieme di cui lui piú vicino.

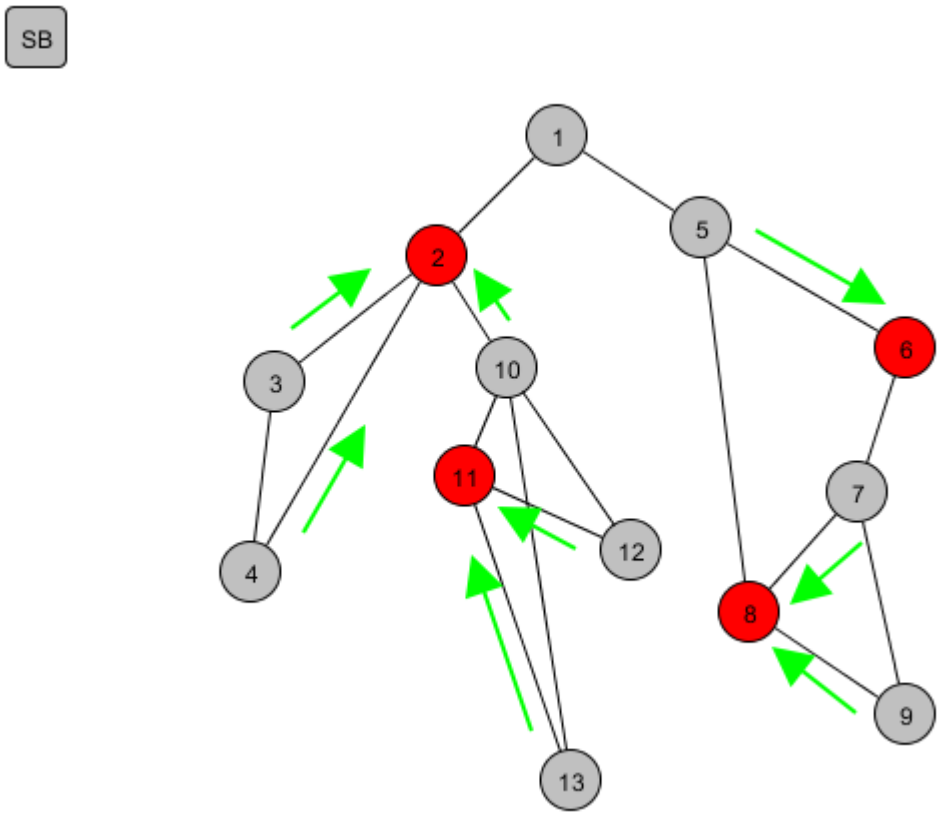

Figura 1: Rete di sensori

Definizione 1.2 (Problema del minimo insieme indipendente(MID):
Il problema appena descritto nella esempio della rete dei sensore consiste nel minimizzare il numero di nodi dei rappresentanti. Tali insieme che deve essere minimizzato si chiama minimo insieme dominante(MID). Diamo adesso una formulazione matematica di questo problema. Dato un grafo $G =< V,E >$ il problema del MID consiste nel trovare un sottoinsieme di vertici $D \subseteq V$ t.c $\forall v \in V$ v $\in D \wedge \exists\, w \in D$ t.c $(v,w) \in V$.
Il problema del MID é NP-completo, non esiste alcun algoritmo , nemmeno centralizzato capace di trovare una soluzione in un tempo polinomiale. Tuttavia, poiché non possiamo trovare il minimo del problema, possiamo cercare di trovare una soluzione che non sia troppo lontana dal minimo.
MID è NP-completo $\rightarrow$ troviamo un algoritmo che ci impiega un tempo pari a $\log\Delta \cdot$ APPROSSIMANTE.
L'algoritmo di approssimazione é basato sul concetto di rapporto di approssimazione , ossia sul rapporto tra la soluzione ottima e quella calcolata dall'algoritmo o viceversa a seconda del tipo di problema(che puó essere di minimizzazione o di massimizzazione). Sia OPT il valore della soluzione ottima e A la soluzione trovata dall'algoritmo. Poiché il MID è un problema di minimizzazione vogliamo calcolare quanto é piú grande A rispetto a OPT. Pertanto calcolo r=$\frac{A}{OPT}$

2 Algoritmi di risoluzione del MID

In questo capitolo analizzeremo in dettaglio i due algoritmi(centralizzato e distribuito) che risolvono il problema del MID trovando una soluzione approssimata di questo problema.

2.1 Algoritmo centralizzato

Sia w(v) la copertura di v, ossia il numero di vicini bianchi di v incluso v. Associamo vari colori ai nodi v, in particolare si ha che: Nodi= $\left\{ \begin{array}{c} \text{Bianchi} \\ \text{Grigi} \\ \text{Neri} \end{array} \right\}$.
Diamo un significato a ciascun colore che il nodo puó assumere.I nodi bianchi sono i nodi che non stati ancora coperti; i nodi grigi sono stati coperti e dunque possono essere ancora selezionati; i nodi neri sono invece quelli selezionati. L'immagine seguente riporta lo pseudo-codice dell'algoritmo appena descritto.

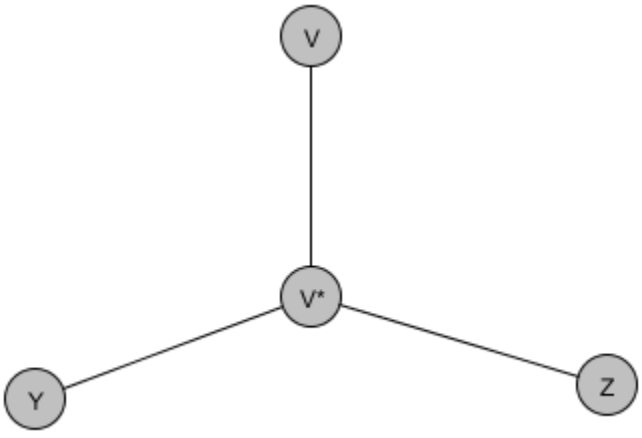

Figura 2: Stella di centro v^*

Teorema 1.1:
Il numero di nodi selezionati n$\leq log(A)\cdot$*OTTIMO*

Algorithm 1 Calcola il MID

while ci sono nodi biachi **do**
  seleziona v non nero con w(v) massimo
  $v \rightarrow nero$
  vicini di v non neri$\rightarrow$ *grigi*
end while

Dimostrazione: Sia v il nodo che viene selezionato a una data iterazione dall'algoritmo 2.1 Quando il nodo v viene selezionato esso diventa nero e, pertanto , la complessitá della soluzione aumenta di 1 in quanto sto aggiungendo un altro nodo all'insieme dominante. Piú formalmente si ha che $A \rightarrow A+1$.
Tuttavia, anziché caricare questo costo aggiuntivo solo sul nodo v che viene selezionato, distribuiamo tale costo su tutti i nodi che vengono dominati da v. In particolare si possono verificare due casi:

$$\begin{cases} \text{associo ad ogni vicino bianco un costo pari a } \frac{1}{w(v)+1} & \text{v non è bianco} \\ \text{associo anche a v un costo pari a } \frac{1}{w(v)}, & \text{v è bianco} \end{cases}$$

□ La seguente immagine mostra il grafo al quale viene applicato l'algoritmo

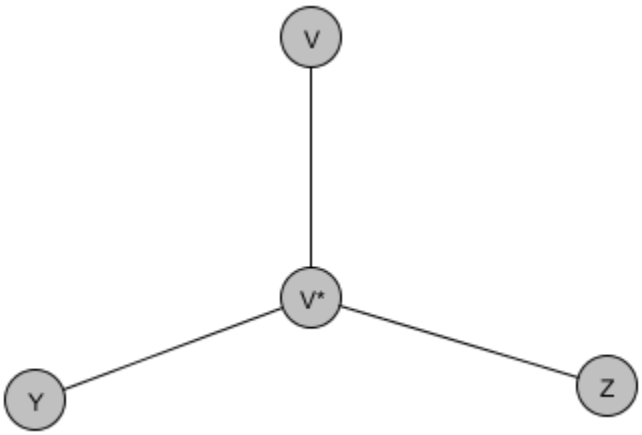

Figura 3: Stella di centro v^*

Definizione 1.3 (Definizione D^*): Sia D^* l'insieme dominante ottimo(MID). D^* è una collezione di stelle tante quanti sono i nodi presenti nell'insieme dominante. Assumiamo inoltre che ogni nodo dell'insieme dominante sia il centro della stella.

Teorema 1.2:
L'algoritmo A domina la stella con al piú $\leq log(A+1)$ *nodi.*

Teorema 1.3: *L'insieme dominante ha cardinalitá* $\mid D^* \mid$. *In base a 2.1* $A \leq log(\Delta) \cdot \mid D^* \mid$

Dimostrazione: Sia v^* un nodo dominante centro di una stella, e sia v un vicino di v^*. Supponiamo che, a una data iterazione, l'algoritmo scelga il nodo v con w(v) massimo. Poiché w(v) è la copertura massima, vale la seguente relazione: w(v)$\geq$ w(v^*) $\rightarrow \frac{1}{w(v)} \leq \frac{1}{w(v^*)}$. Ma, dal momento che w($v^*$)=d($v^*$)+1 essendo d($v^*$) il grado di v^*(il fattore 1 viene utilizzato per includere v^* stesso), si ha che vale la seguente relazione $\frac{1}{w(v)} \leq \frac{1}{w(v^*)} = \frac{1}{d(v^*)+1}$. A questo punto si ripete questo ragionamento per tutti i restanti nodi ; per esempio supponiamo di selezionare, all'iterazione successiva , il nodo z il cui costo associato diviene pertanto pari a $\frac{1}{d(v^*)}$.Supponiamo di selezionare, all'iterazione successiva il nodo y il cui costo associato diviene pari ad a $\frac{1}{d(v^*)-1}$. In generale, man mano che vengono selezionati dei nodi, il costo associato a quel rispettivo nodo diminuisce di una unitá. Pertanto all'i-esimo nodo posso associare un costo pari ad $\frac{1}{d(v^*)-i+2}$. Pertanto il costo distribuito sui vari nodi vale: costo $\leq \frac{1}{d(v^*)+1}+\frac{1}{d(v^*)+}+\frac{1}{d(v^*)-1}$ +.....$\frac{1}{2}$+1 Riscriviamo tale costo utilizzando la seguente sommatoria:
Costo $\leq \sum_{i=1}^{d(v^*)+1} \frac{1}{i}$=$H_{\Delta+1} \leq log(\Delta+1)$.

Dal momento che la soluzione ottima é costituita da D^* stelle si ha che A$\leq log(\Delta+1)\cdot \mid D^* \mid$.

□

2.2 Algoritmo distribuito

L'algoritmo distribuito, a differenza di quella centralizzato, esamina una zona locale dei nodi; in particolare, dato un nodo, considera i nodi a distanza 2. Se v ha il massimo grado di copertura tra quelli a distanza 2, l'algoritmo seleziona v prima di tutti gli altri nodi che si trovano nel vicinato a distanza 1 o 2. L'idea dell'algoritmo é la seguente: i nodi a distanza 1 possono essere dominati dai nodi a distanza 2 o dal nodo v se v é il noto che ha la copertura massima. Pertanto se é è il nodo avente la copertura massima, v viene scelto dall'algoritmo. Quindi l'algoritmo risulta essere corretto perché simula l'algoritmo goloso ottenendo lo stesso rapporto di approssimazione. Tuttavia il seguente algoritmo rischia di essere lento. Nel caso peggiore il tempo puó essere lineare nel numero di nodi(pertanto la sua complessitá é uguale a quella dell'algoritmo centralizzato). Riportiamo lo pseudo-codice dell'algoritmo nel paragrafo sottostante:

Algorithm 2 Calcola il MID

```
while ci sono nodi biachi do
  raccoglie w(u) per ogni u a distanza ≤ 2
  if w(v) é il massimo then
    v diventa nero
  end if
end while
```

Illustriamo due esempi all'interno dei quali si verifica questa evenienza: nel primo di questi due esempi consideriamo un insieme di cammini composti da nodi il cui grado decresce linearmente da sinistra verso destra. Analizzando il seguente cammino possiamo notare che la copertura dell'ultimo nodo della lista é minore dei vicini a distanza 1 e 2 che lo precedono. Pertanto l'ultimo nodo della lista non verrá mai selezionato per primo. Il primo nodo che l'algoritmo seleziona é il primo nodo della lista che diventa nero in quanto non ha vicini alla sinistra di grado maggiore di lui. Successivamente l'algoritmo selezionerá sequenzialmente tutti gli altri nodi della lista. Calcoliamo formalmente la complessitá di questa soluzione: Sia n=numero totale di nodi lungo il cammino di lunghezza k. Supponiamo che ogni nodo al centro della stella ha k vicini. Pertanto si ha che n=$k^2 \rightarrow$ O(k)=($\sqrt{n}$).
Una soluzione possibile di questa particolare configurazione consiste nell'arrotondare le coperture approssimandole con le potenze di due successive; pertanto, dividendo, ad ogni iterazione il numero di nodi per due, diminuisco la complessitá dell'algoritmo che, da lineare, diventa , logaritmica.

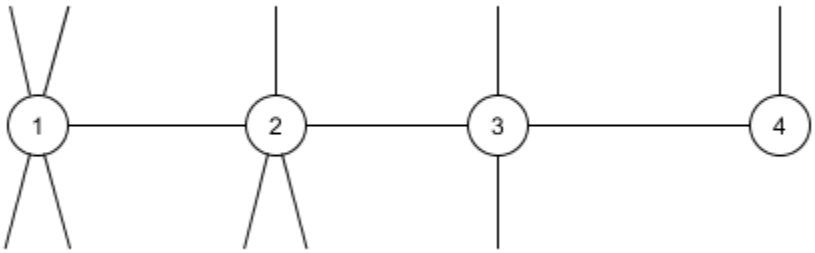

Figura 4: Cammini decrescenti non colorati

Tuttavia, anche utilizzando questa soluzione, non sempre si riesce a garantire una soluzione logaritmica; infatti consideriamo una clicke all'interno della quale ciascun nodo ha due foglie bianche.
L'algoritmo , scegliendo un nodo qualsiasi della clicke, copri tutti i nodi della clicke in quanto, in questa particolare configurazione , tutti i nodi sono vicini tra di loro. Tuttavia ogni nodo, che é stato dominato dal nodo scelto alla prima iterazione, ha ancora una copertura massima pari a due(le due foglie bianche non sono state ancora coperte). Pertanto

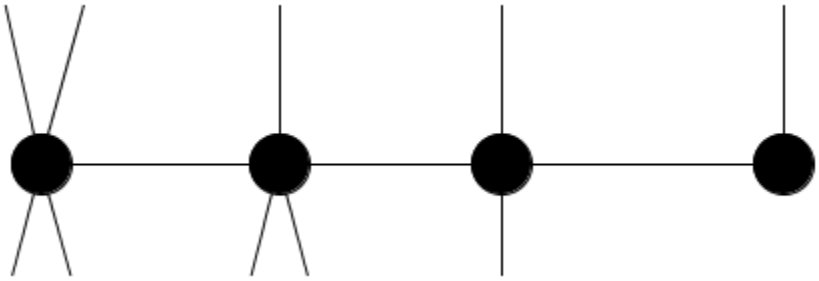

Figura 5: Cammini decrescenti colorati

l'algoritmo é costretto a selezionare tutti i nodi della clicke per coprire tutte le rimanenti foglie bianche associate ad ogni nodo. Tale operazione richiede un costo pari a O(n), dove n é il numero di nodi della clicke.

La seguente figura mostra la clicke di nodi: a ciascun nodo sono associate due foglie che vengono via via coperte durante l'esecuzione dell'algoritmo .

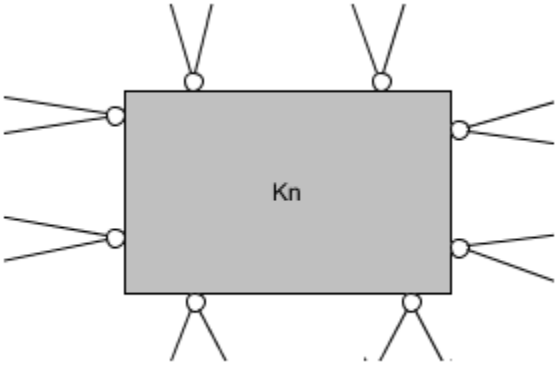

Figura 6: Clicke