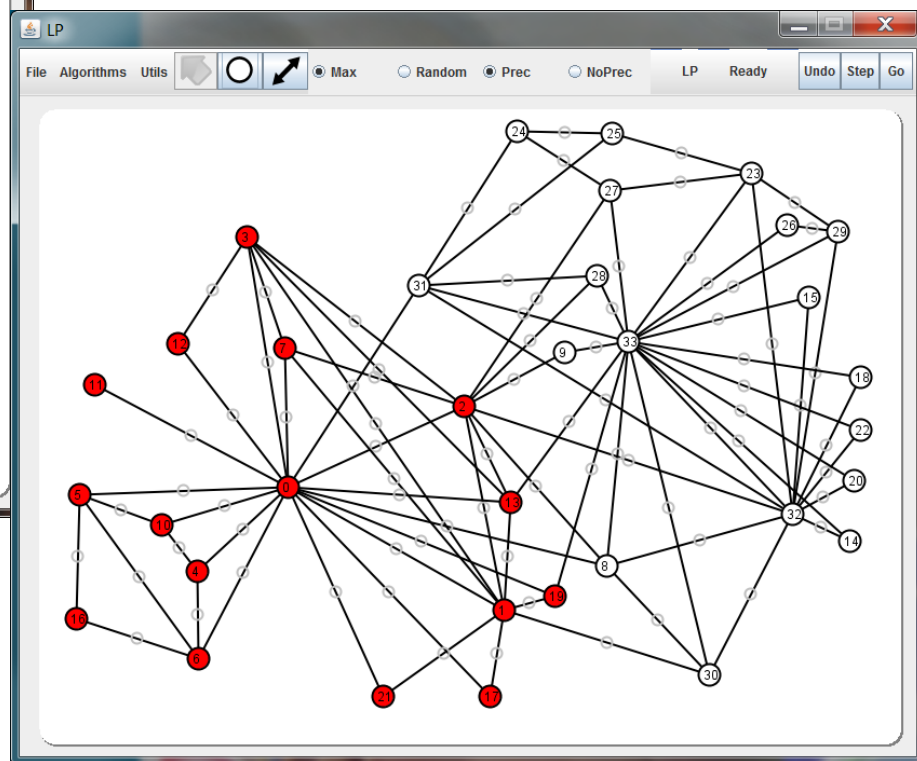
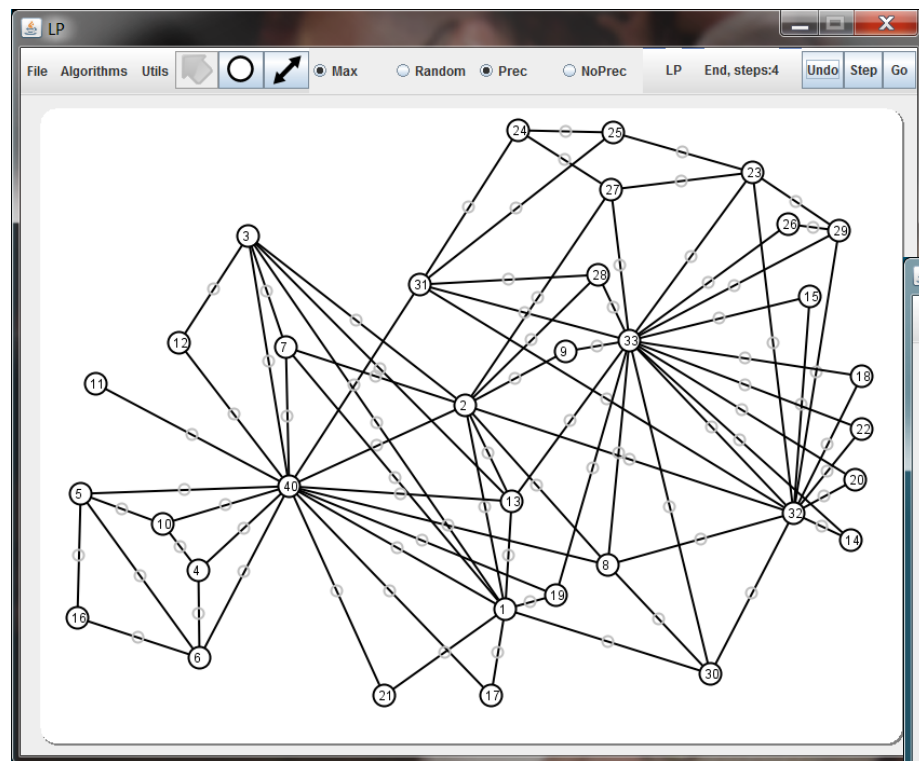


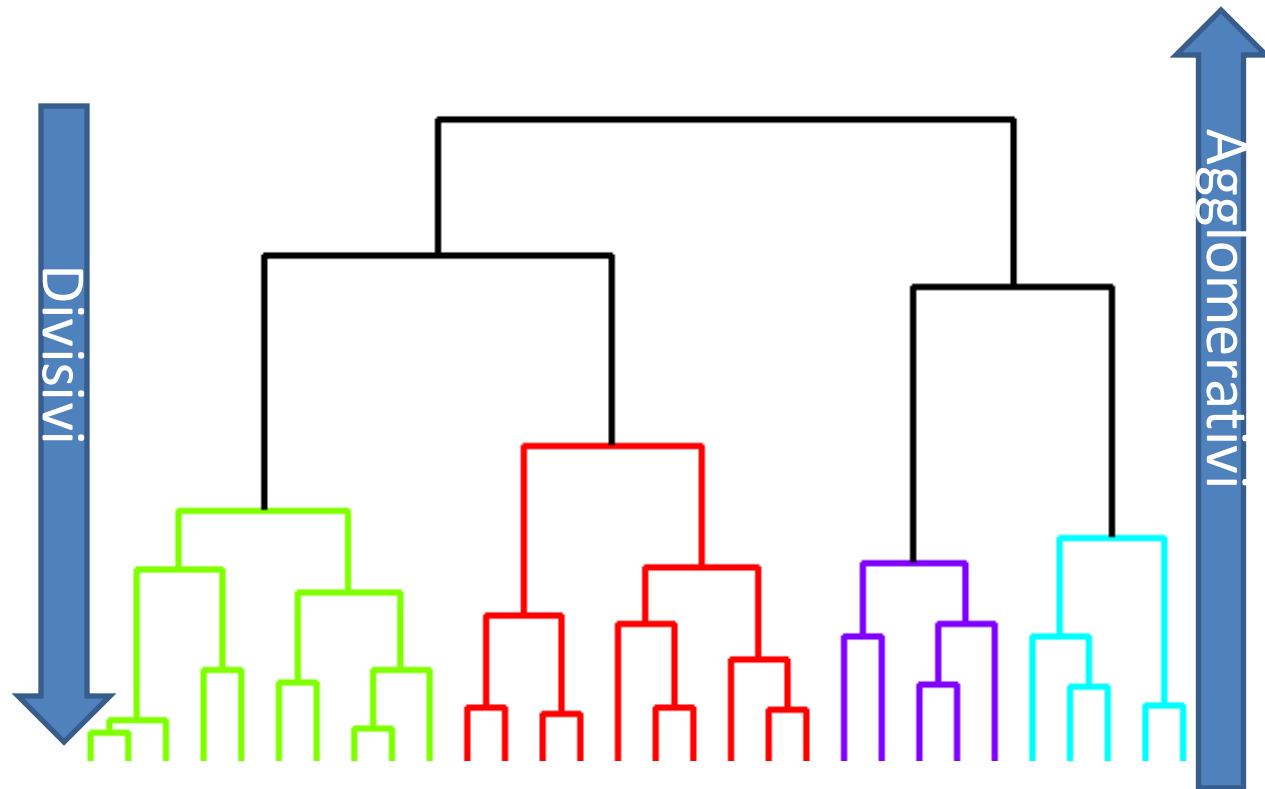
Individuazione di comunità: Label Propagation Algorithm

Comunità



Algoritmi per il rilevamento delle comunità

- Input: una rete $G = (V, E)$
- Output: una partizione di V in comunità
- Due famiglie di metodi:
 - divisivi
 - **agglomerativi**



Label Propagation Algorithm (LPA)

Proposto nel 2007

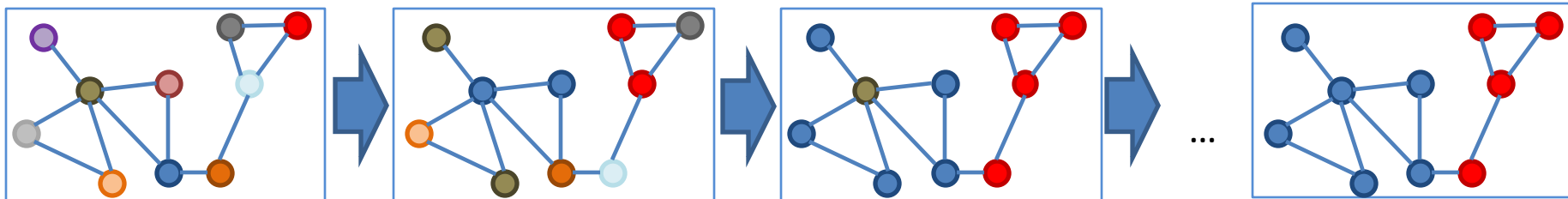
- È un algoritmo **veloce** per la ricerca di comunità in un grafo.
 - Rileva le comunità utilizzando solo la struttura della rete come guida e
 - non richiede una funzione obiettivo predefinita o
 - informazioni preliminari sulla struttura della rete o sulle comunità

Label Propagation Algorithm (LPA)

Come funziona?

L'algoritmo procede in round


- Inizialmente, a ciascun vertice della rete viene assegnata un'etichetta univoca (è la propria comunità iniziale)
- Le etichette si propagano attraverso la rete.
 - Ad ogni round della propagazione, ciascun nodo aggiorna la sua etichetta a quella a cui appartiene il numero massimo di vicini.
 - I pareggi vengono risolti in modo casuale uniforme.
 - LPA raggiunge la convergenza quando ogni nodo ha l'etichetta maggioritaria tra suoi vicini.
 - LPA **termina** se viene raggiunta la **convergenza** o *il numero massimo di iterazioni* definito dall'utente.
 - Comunità \equiv nodi aventi la stessa etichetta finale



Label Propagation Algorithm (LPA)

Algorithm 1: Label Propagation (Synchronous)

```
1 Initialize labels: for each  $v \in V$ ,  $l_v(0) = v$ ;  
2  $i=0$ ;  
3 while the stop criterion is not met do  
4    $i++$ ;  
5   Propagation:  
6   foreach  $v \in V$  do  
7      $l_v(i) = \operatorname{argmax}_l \sum_{u \in N(v)} [l_u(i-1) == l],$   
8   end  
9 end  
10 return Final labeling:  $l_v(t)$  for each  $v \in V$ , where  $t$  is  
    the last executed step.
```



Vale 1 se la
label di u al
round $i-1$ è l

Label Propagation Algorithm (LPA)

IDEA:

Etichetta cambia in base alle etichette possedute dai vicini al round precedente



i gruppi densamente connessi raggiungono rapidamente un'etichetta comune.

Quando molti di questi gruppi densi (di consenso) vengono creati in tutta la rete, continuano ad espandersi verso l'esterno fino a quando possibile farlo.

Label Propagation Algorithm (LPA)

Gli algoritmi LPA hanno dimostrato di essere rapidi (sperimentalmente) ed efficaci

Es. LPA usato per

- **assegnare la polarità** (sentimento positivo o negativo) **ai tweet**
In uno studio sulla classificazione della polarità di Twitter collegamenti lessicali e grafico del follower
- **stimare combinazioni potenzialmente pericolose di farmaci da co-prescrivere** a un paziente, in base alla somiglianza chimica e ai profili degli effetti collaterali.
In uno studio delle interazioni farmaco-farmaco basate sugli effetti collaterali clinici
- **inferire le caratteristiche delle espressioni in un dialogo**, per un modello di apprendimento automatico per monitorare le intenzioni dell'utente con l'aiuto del grafo di conoscenza di Wikidata dei concetti e delle loro relazioni.
In uno studio su "Inferenza delle caratteristiche sul grafo Wikidata per l'ora legale"

LPA Sincrono

PRO

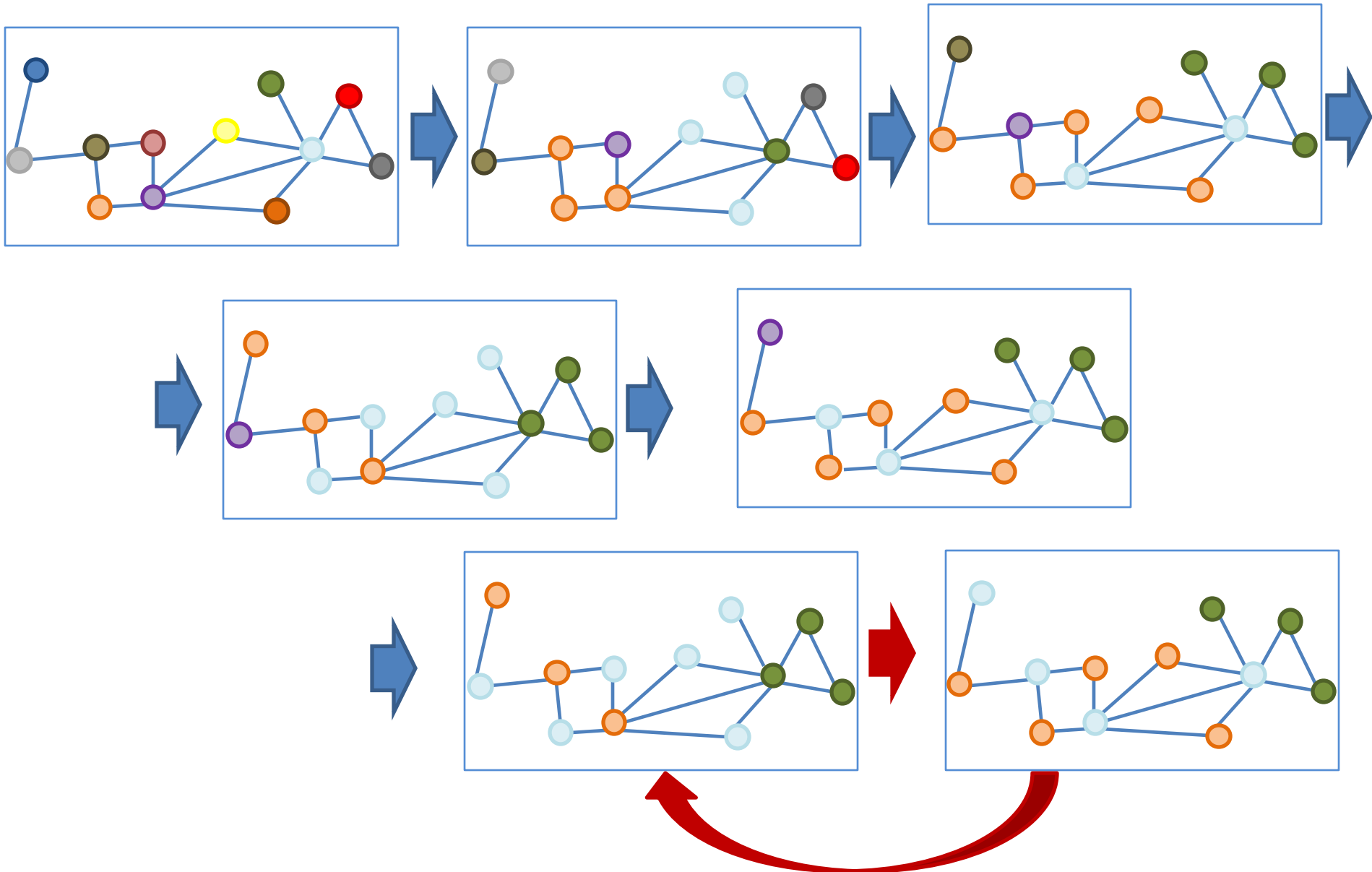
- Semplice
- Veloce
 - In pratica bastano pochi round
 - nodi eseguono round in parallelo
- Stabile (sempre lo stesso risultato)

CONTRO

- Le etichette possono oscillare

Ricorda criterio di stop: convergenza o raggiunto max numero di round

Oscillazioni



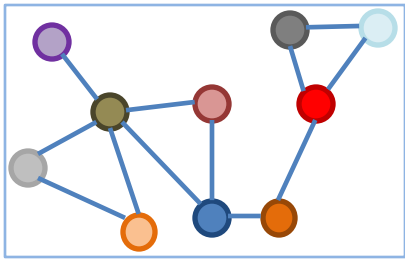
LPA Asincrono

Nell'algoritmo LP asincrono, per ogni passaggio, le etichette dei vertici vengono aggiornate in sequenza in base all'etichettatura corrente..

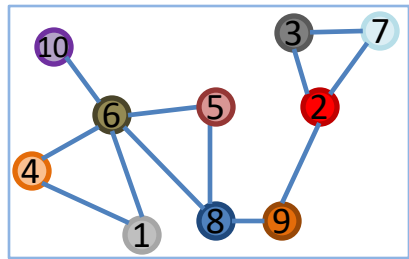
Algorithm 2: Label Propagation (Asynchronous)

```
1 Initialize labels: for each  $v \in V$ ,  $l_v(0) = v$ ;  
2  $i=0$ ;  
3 while the stop criterion is not met do  
4    $i++$ ;  
5   let  $\pi = (v_{\pi(1)}, v_{\pi(2)}, \dots, v_{\pi(n)})$  be a random  
   permutation of the vertices.  
6   Propagation:  
7   for  $j = 1$  to  $n$  do  
8      $l_{v_{\pi(j)}}(i) = \operatorname{argmax}_l \sum_{u \in N(v_{\pi(j)})} [l_u == l],$   
9     where  $l_u = \begin{cases} l_u(i) & \text{if } u = v_{\pi(k)}, k < j \\ l_u(i - i) & \text{if } u = v_{\pi(k)}, k > j \end{cases}$   
10  end  
11 end  
12 return Final labeling:  $l_v(t)$  for each  $v \in V$ , where  $t$  is  
    the last executed step.
```

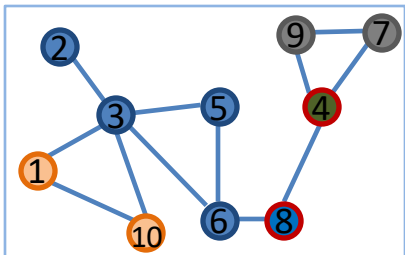
Asynchronous LPA



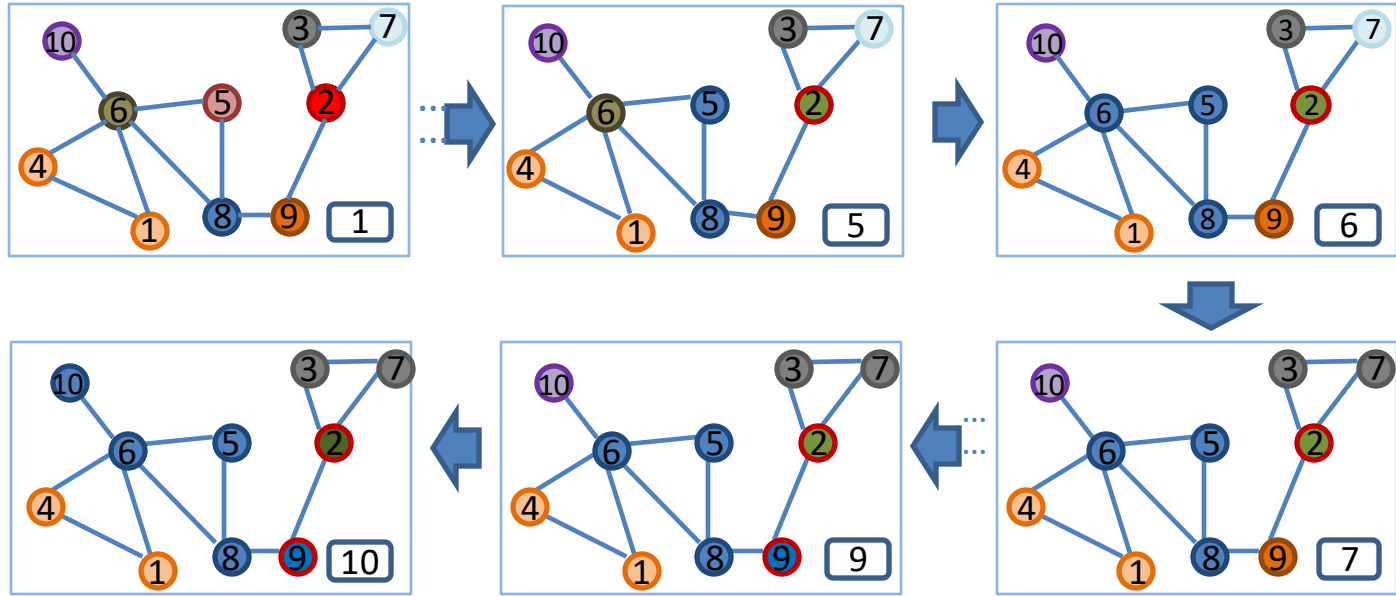
Permutazione random



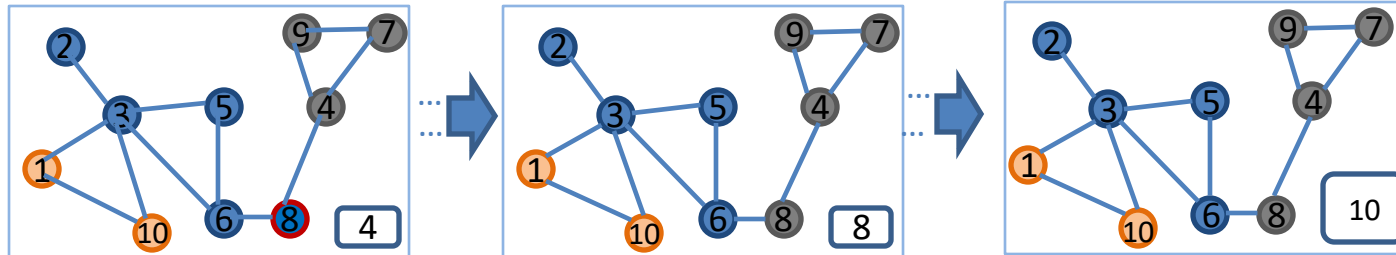
Permutazione random



Step 1



Step 2



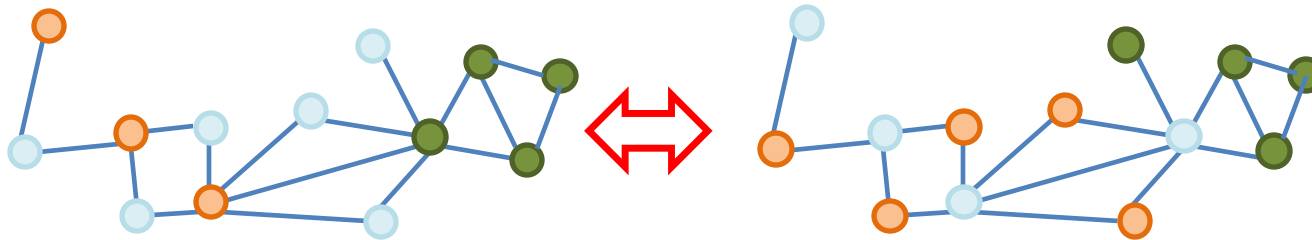
LPA Asincrono

Nell'algoritmo LP asincrono, per ogni passaggio, le etichette dei vertici vengono aggiornate in sequenza in base all'etichettatura corrente..

- Difficile parallelizzare:
devono essere considerate
le dipendenze
- Instabile:
 - diverse esecuzioni possono
fornire risultati diversi
- Meno efficace:
- alcune esecuzioni potrebbero
fornire una comunità "enorme"

Algorithm 2: Label Propagation (Asynchronous)

```
1 Initialize labels: for each  $v \in V$ ,  $l_v(0) = v$ ;  
2  $i=0$ ;  
3 while the stop criterion is not met do  
4    $i++$ ;  
5   let  $\pi = (v_{\pi(1)}, v_{\pi(2)}, \dots, v_{\pi(n)})$  be a random  
   permutation of the vertices.  
6   Propagation:  
7   for  $j = 1$  to  $n$  do  
8      $l_{v_{\pi(j)}}(i) = \operatorname{argmax}_l \sum_{u \in N(v_{\pi(j)})} [l_u == l],$   
9     where  $l_u = \begin{cases} l_u(i) & \text{if } u = v_{\pi(k)}, k < j \\ l_u(i-1) & \text{if } u = v_{\pi(k)}, k > j \end{cases}$   
10  end  
11 end  
12 return Final labeling:  $l_v(t)$  for each  $v \in V$ , where  $t$  is  
    the last executed step.
```

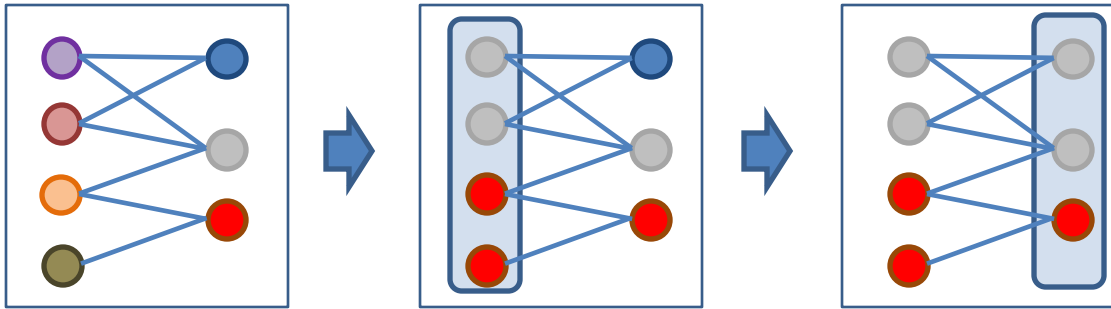


Sincrono: Su alcuni nodi le label si “scambiano”

Asincrono: nodi non cambiano label contemporaneamente ➔ no “scambi”

LPA per grafi bipartiti

L'algorithm divide ogni round di propagazione in due passi sincronizzati che corrispondono alla partizione dei nodi del grafo bipartito $G=(V_1 \cup V_2, E)$



Questo algoritmo ha dimostrato sperimentalmente di essere

- efficiente come LPA sincrono
- facilmente parallelizzabile, e
- stabile.

LPA Semi-Sincrono

Due fasi:

- **Fase di colorazione:**

Colora i vertici della rete

- in modo che non vi siano due vertici adiacenti che hanno lo stesso colore (usando un qualsiasi algoritmo distribuito di colorazione di grafi).

- **Fase di propagazione.**

La fase di propagazione dell'etichetta è divisa in round

(ognuno suddiviso in tanti passi quanti sono i colori utilizzati per la colorazione)

- Al passo c , le etichette vengono propagate simultaneamente ai vertici a cui è stato assegnato il colore c durante la fase di colorazione.

LPA Semi-Sincrono

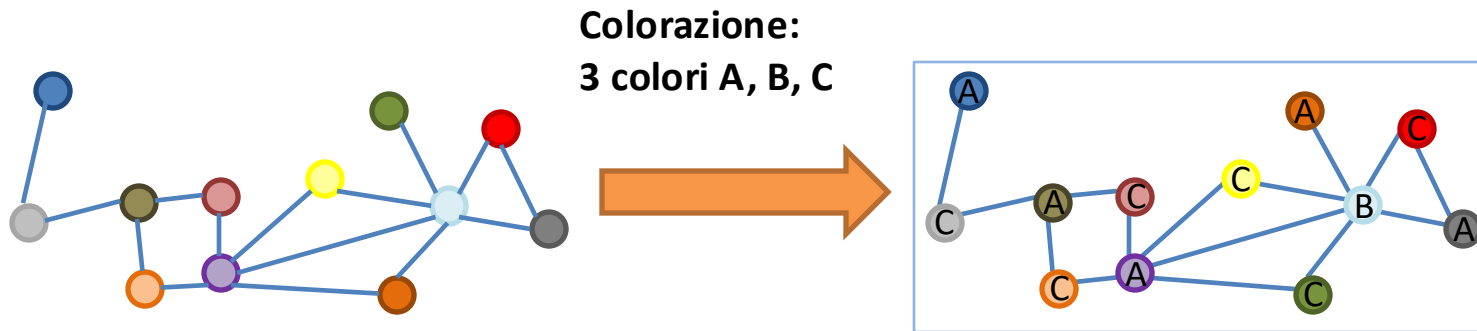
Idea:

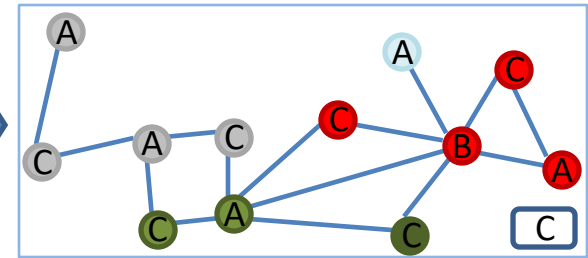
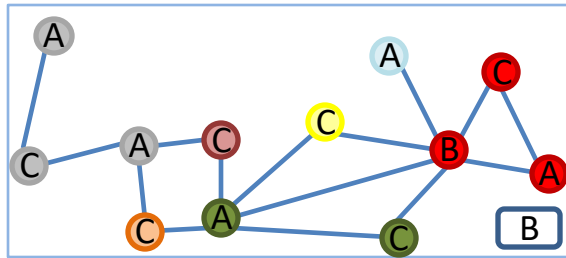
- Colora la rete
- Aggiorna vertici per classe di colore
 - due nodi vicini non vengono mai aggiornati contemporaneamente

Algorithm 3: LPA (Semi-synchronous)

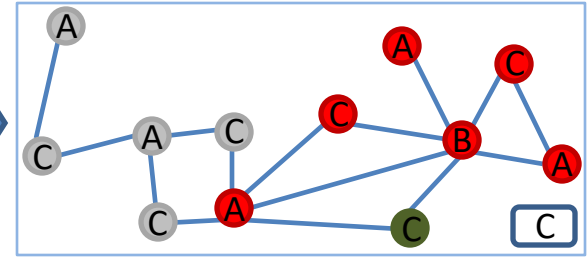
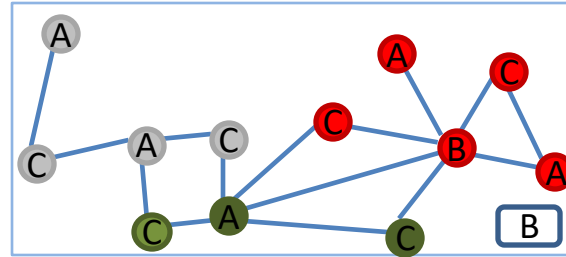
```
1 Initialize labels: for each  $v \in V$ ,  $l_v(0) = v$ ;  
2 Network coloring: assign a color to the vertices of the  
   network such that no two adjacent vertices share the  
   same color. Let  $\mathcal{D} = \{D_1, D_2, \dots, D_\ell\}$  be the color  
   partitioning obtained.  
3  $i=0$ ;  
4 while the stop criterion is not met do  
5    $i++$ ;  
6   Propagation:  
7   for  $j = 1$  to  $\ell$  do  
8     foreach  $v \in D_j$  do  
9        $l_v(i) = \operatorname{argmax}_l \sum_{u \in N(v)} [l_u == l],$   
10      where  $l_u = \begin{cases} l_u(i) & \text{if } u \in D_k, k < j \\ l_u(i-1) & \text{if } u \in D_k, k > j \end{cases}$   
11     end  
12   end  
13 end  
14 return Final labeling:  $l_v(t)$  for each  $v \in V$ , where  $t$  is  
   the last executed step.
```

LPA Semi-Sincrono

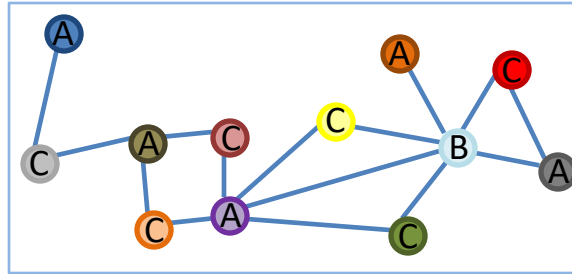




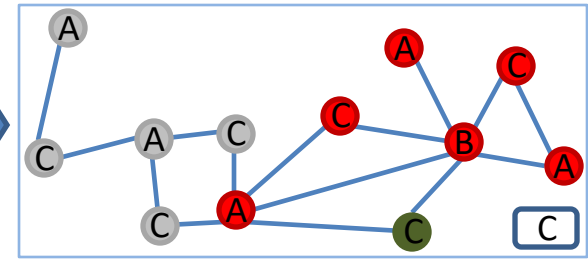
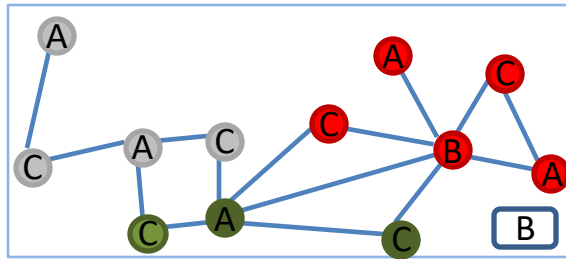
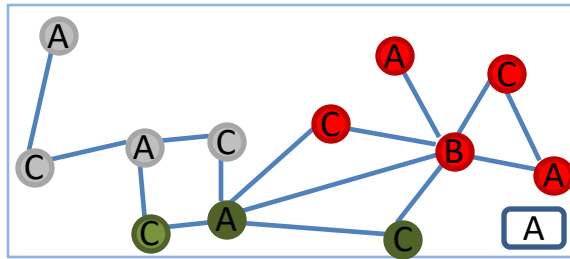
A graph with 10 nodes and 15 edges. The nodes are labeled A, B, C, and A. The nodes are colored: 3 gray, 3 red, 3 green, and 1 blue. The blue node is in a box.



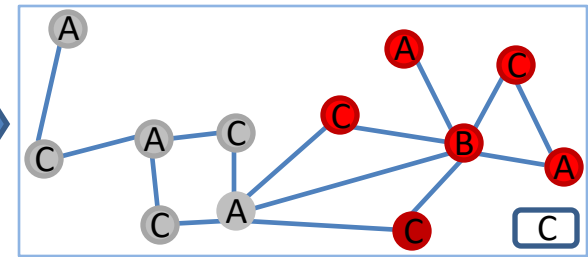
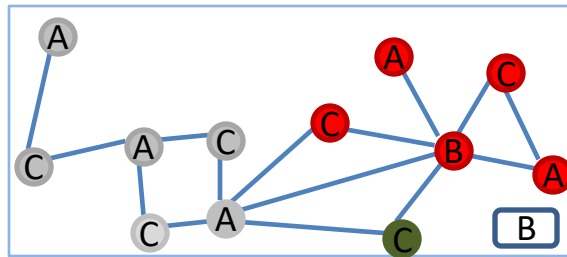
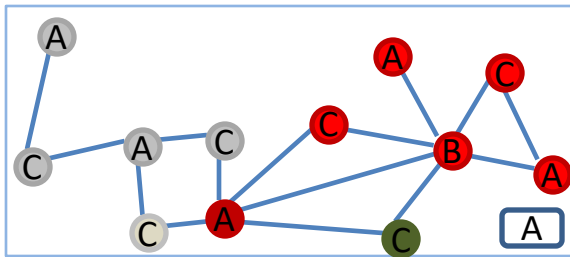
LPA Semi-Sincrono



Round di Propagazione 2



Round di Propagazione 3



Convergenza

Teorema. Consideriamo una rete $G = (V, E)$.

Supponiamo che LPA semi-sincrono termina al primo round t in modo tale che per ogni $v \in V$ vale una delle seguenti condizioni:

i) $l_v(t) = l_v(t - 1)$

ii) $l_v(t) \neq l_v(t - 1)$ ma questa modifica è dovuta a un ex equo

Allora l'algoritmo converge, indipendentemente dalla regola di gestione degli ex equo

Regole gestione pareggi

LPA-Random (LPA): una delle etichette che massimizza la somma viene scelta casualmente.

Prec. LPA: se l'etichetta corrente soddisfa è tra le maggioritarie, il vertice mantiene la sua etichetta corrente, altrimenti l'etichetta viene scelta in modo casuale.

LPA-Max: scelta l'etichetta con valore più alto.

LPA-Prec-Max: se l'etichetta corrente soddisfa l'equazione di etichettatura, il vertice mantiene la sua etichetta corrente, altrimenti viene scelta l'etichetta massima.

Convergenza: Idea

Lemma: Considera una rete $G = (V, E)$. L'algoritmo semisincrono con le strategie di risoluzione del legame LPA-Prec, LPAMax e LPA-Prec-Max, non genera alcun ciclo (ritorno ad un'etichettatura precedente).

Ricorda: fintanto che il criterio di stop non è soddisfatto, almeno un vertice v aggiorna la sua etichetta senza che si verifichi un pareggio.

Consideriamo un generico round $i > 1$. Distinguiamo due casi

Caso 1: $l_v(i-1)$ soddisfa l'equazione

$$l_v = \operatorname{argmax}_l \sum_{u \in N(v)} [l_u == l]$$

Sappiamo che $l_v(i)$ e $l_v(i-1)$ compaiono lo stesso numero di volte tra le attuali etichette dei vicini di v .

Cambio di etichetta avviene solo se si utilizza la strategia di risoluzione del legame **LPA-Max**, altrimenti v manterrebbe la sua etichetta attuale (cioè, $l_v(i) = l_v(i-1)$)

→ il numero di edge monocromatici incidenti su v non cambia
(ricordiamo che nessun vicino di v aggiorna la sua etichetta durante lo stesso passaggio)

Caso 2: $l_v(i-1)$ non soddisfa l'equazione

Ora $l_v(i)$ appare più volte di $l_v(i-1)$, tra le attuali etichette dei vicini di v .

→, il numero di edge monocromatici incidenti su v aumenta almeno di 1
(poiché nessun vicino di v aggiorna la sua etichetta in concomitanza);

Quindi
il numero di edge monocromatici (che collegano due nodi con la stessa etichetta) aumenta ad ogni round

→ Non è possibile ciclare, cioè ritornare all'etichettatura di un round precedente $j < i$