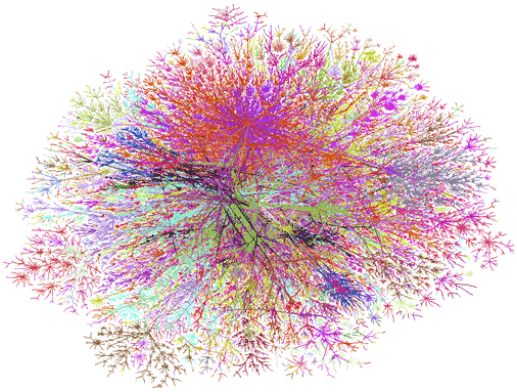
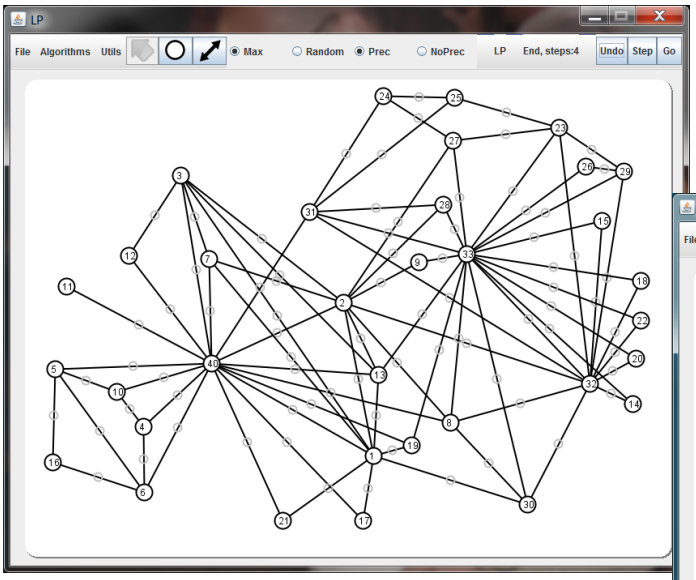
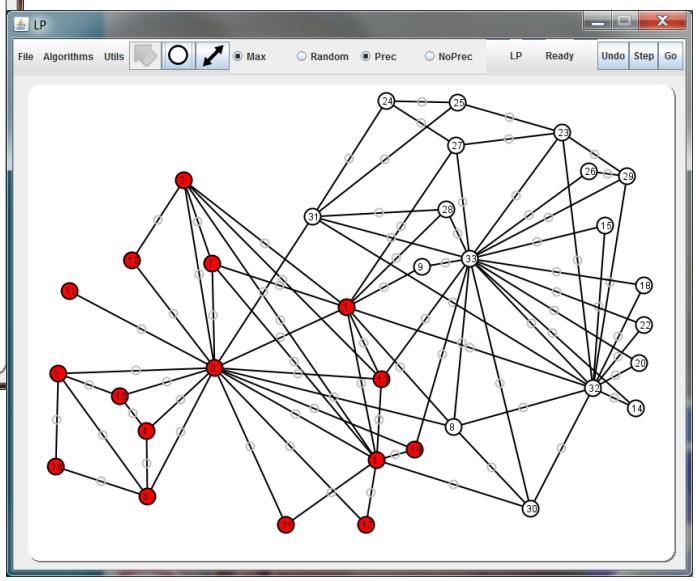


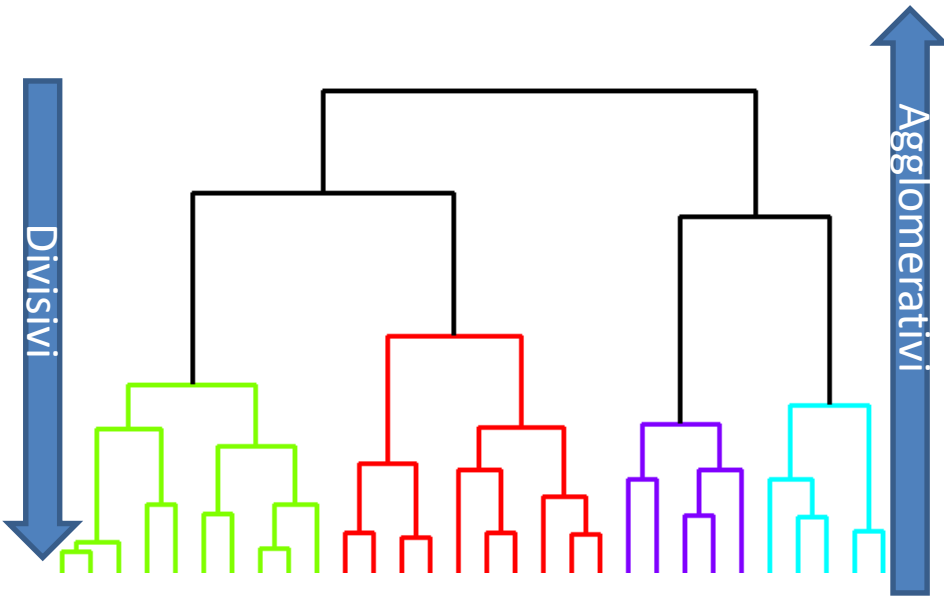
Individuazione di comunità: Label Propagation Algorithm





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**



Un algoritmo agglomerativo: Label Propagation Algorithm (LPA)

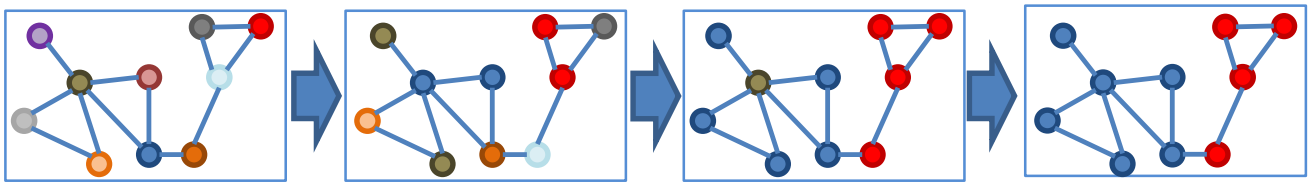
Proposto nel 2007

- 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

Descrizione dell'algoritmo.

L'algoritmo, descritto formalmente in seguito come Algorithm 1, 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 spezzati in modo uniforme e casuale.
  - 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.



Algorithm 1: Label Propagation (Synchronous)

1 **Inizialize 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 $l_v(i) = \underset{l}{\operatorname{argmax}} \sum_{u \in N(v)} [l_u(i-1) == l]$,
8 **end**
9 **end**
10 **return** Final labeling: $l_v(t)$ for each $v \in V$, where t is the last executed step.

Note (valide anche per il seguito).

Nell'esempio l'etichetta di un nodo è rappresentata dal colore del nodo stesso

Nell'algoritmo [x==y] assume valore 1 se x=y, valore 0 se gli argomenti sono diversi

Idea alla base del funzionamento.

L'etichetta cambia in base alle etichette possedute dai vicini al round precedente. Questo implica che i gruppi densamente connessi raggiungono rapidamente un'etichetta comune.

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

Al termine del processo i gruppi di nodi aventi un'etichetta comune formano le comunità cercate

LPA in pratica

Nonostante la mancanza di validazioni teoriche, gli algoritmi LPA hanno dimostrato sperimentalmente di essere rapidi 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 grafico di conoscenza di Wikidata dei concetti e delle loro relazioni.

  In uno studio su "Inferenza delle caratteristiche sul grafo Wikidata per l'ora legale«.

Per l'algoritmo LPA non esistono valutazioni teoriche, ma solo sperimentali.

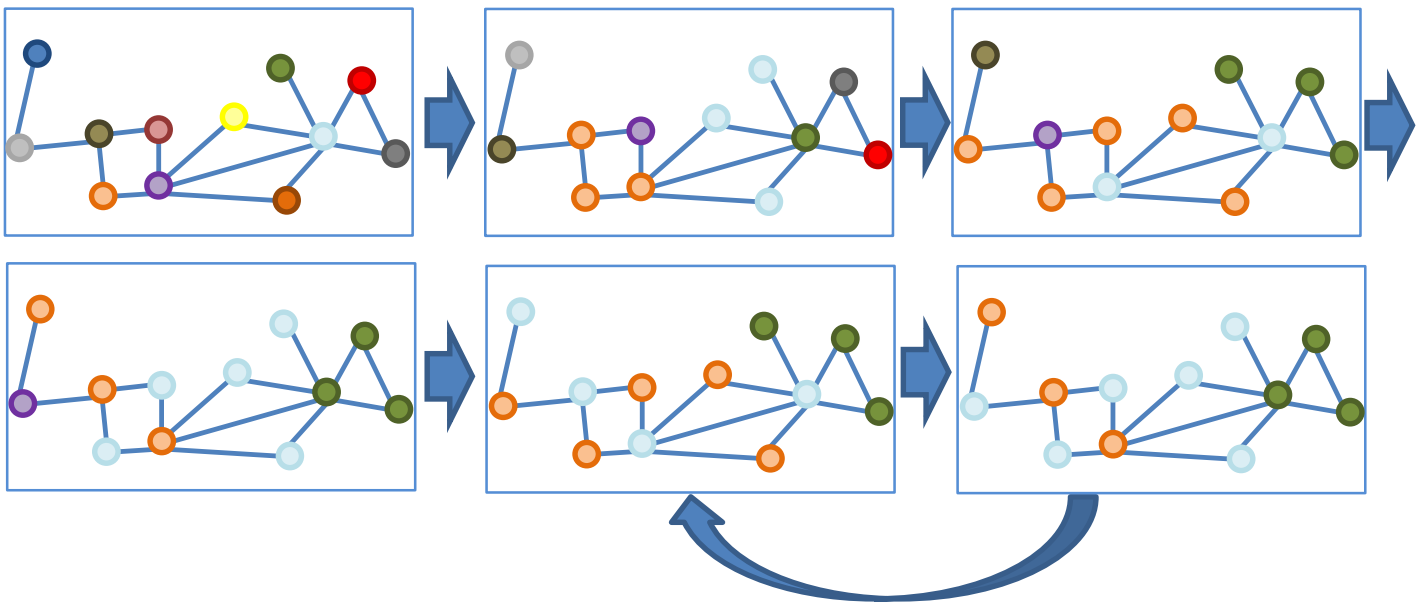
Dalle valutazioni sperimentali risulta che l'algoritmo ha i seguenti punti a favore:

- È semplice,
- È veloce (in pratica bastano pochi round ed i nodi eseguono ogni round in parallelo;
- È stabile (sempre lo stesso risultato).

Presenta però un problema: *le etichette possono oscillare*

(*Ricordare il criterio di stop: convergenza o raggiunto max numero di round – cosa che non garantisce che si siano già formate le comunità)*

Il seguente è un esempio in cui il processo non converge



LPA Asincrono

Per ovviare al problema delle oscillazioni è stata quindi introdotta una versione dell'algoritmo, in cui i nodi non eseguono i round in parallelo. Ad ogni round, le etichette dei vertici vengono aggiornate in maniera sequenziale (in accordo ad un ordinamento casuale dei vertici) in base all'etichettatura corrente. L'algoritmo è formalmente descritto come Algorithm 2.

Algorithm 2: Label Propagation (Asynchronous)

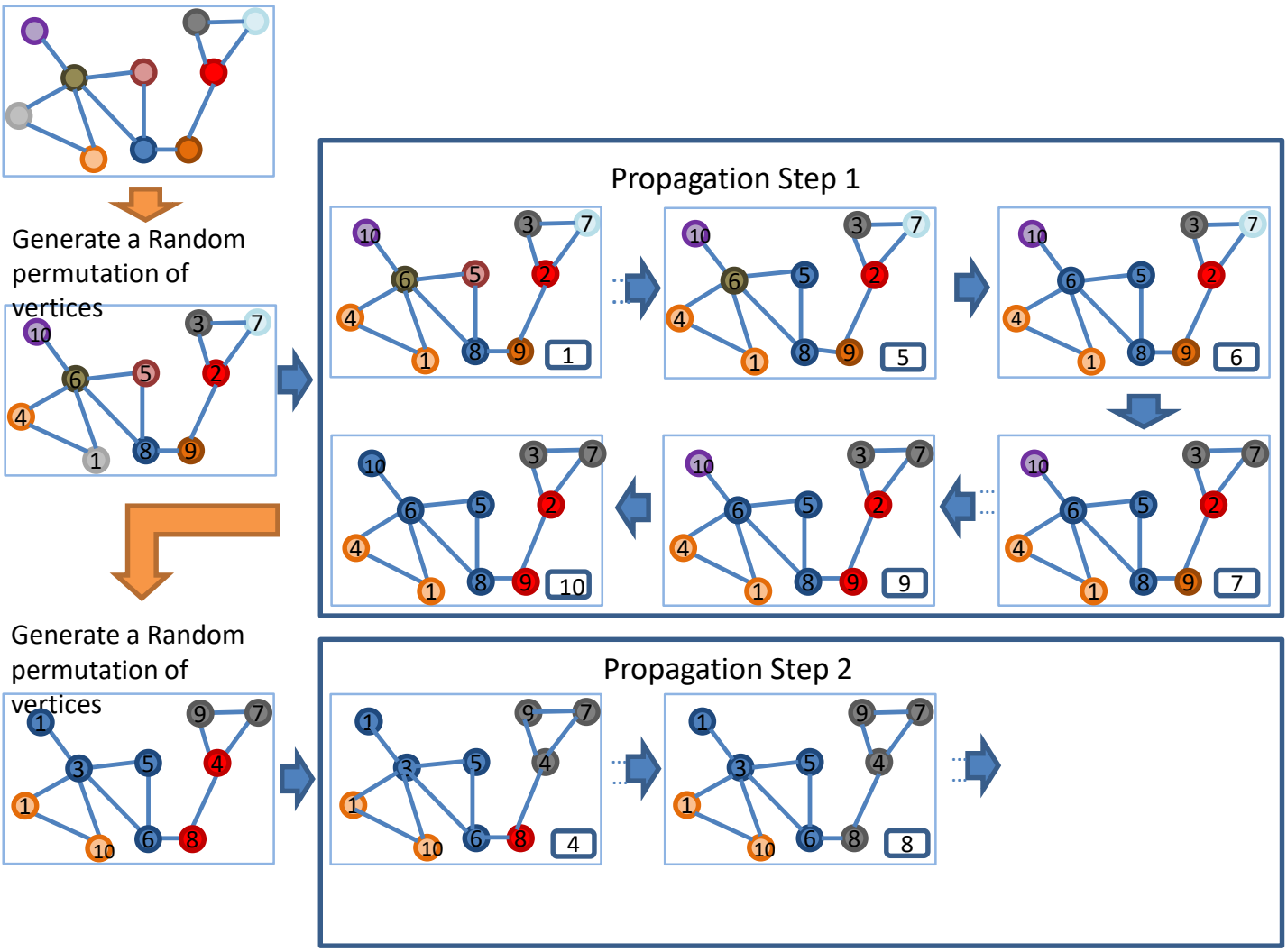
1 **Inizialize labels:** for each $v \in V$, $l_v(0) = v$;
2 i=0;
3 **while** *the stop criterion is not met* **do**
4 i++;
5 let $\pi = (v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)})$ be a random permutation of the vertices.
6 **Propagation:**
7 **for** $j = 1$ **to** n **do**
8 $l_{v_{\pi(j)}}(i) = \underset{l}{\operatorname{argmax}} \sum_{u \in N(v_{\pi(j)})} [l_u == l]$,
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.

Ne risulta un algoritmo che non presenta più il problema delle oscillazioni. Tuttavia esso presenta alcune altre problematiche rispetto alla versione sincrona:

- E' difficile da parallelizzare: infatti per aggiornare l'etichetta del nodo j-mo devono essere considerate le dipendenze dagli aggiornamenti dei nodi precedenti, cioè con indice <j nell'ordinamento scelto per eseguire il round corrente.

- E' instabile: a causa dell'uso massivo della randomizzazione, diverse esecuzioni possono fornire risultati diversi, cioè partizioni in comunità molto differenti; inoltre alcune esecuzioni potrebbero fornire una comunità ''enorme«, cioè con moltissimi nodi, priva di significato reale.

La figura mostra un esempio di esecuzione. La permutazione casuale generata per eseguire ciascun round è indicata dai numeri all'interno dei nodi.



LPA per grafi bipartiti

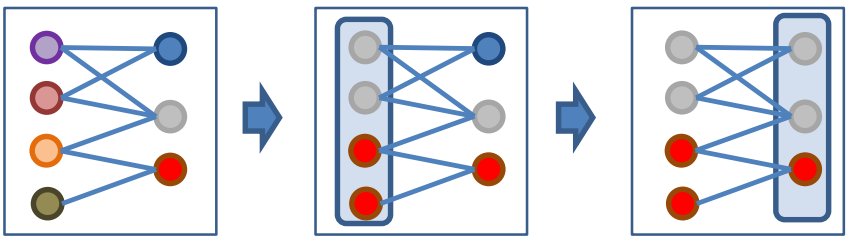
E’ possibile un adattamento particolare dell’algoritmo nel caso di grafi bipartiti. L'algoritmo divide ogni round di propagazione in due fasi sincronizzate che corrispondono alla partizione dei nodi del rafo bipartito.

Siano X e Y le due parti (formate da nodi a due a due indipendenti) che compongono l’insieme V=X U Y dei nodi .

Poiché l’aggiornamento della label di un nodo x in X dipende solo dai vicini di , che sono nodi in Y, segue che tutti i nodi in X possono aggiornare contemporaneamente le proprie label; analogamente per i nodi in Y.

Possiamo quindi considerare ogni round dell’algoritmo LPA come composto da due fasi successive:

- una prima fase in cui tutti i nodi di X aggiornano le loro label in parallelo ed
- una seconda fase in cui tutti i nodi in Y aggiornano le loro label in parallelo.



Questo algoritmo ha dimostrato sperimentalmente di essere

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

LPA Semi–Syncrono

L'idea alla base dell'algoritmo per grafi bipartiti può essere generalizzata a grafi generici. In un grafo G=(V,E) possiamo colorare i nodi in modo che due nodi adiacenti non abbiano lo stesso colore assegnato.

Nel caso di un grafo bipartito con V=X U Y, possiamo usare 2 colori, A per colorare i nodi in X e il colore B per colorare i nodi in Y. Quindi l'algoritmo precedente può essere descritto come segue: ad ogni round dispari i nodi di colore A aggiornano le loro label; ad ogni round pari i nodi di colore B aggiornano le loro label.

L'algoritmo, descritto formalmente come Algorithm 3, è quindi composto 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 delle etichette è divisa in round (tanti quanti sono i colori utilizzati per la colorazione): Al round c, le etichette tutti i vertici a cui è stato assegnato il colore c durante la fase di colorazione aggiornano le proprie etichette in contemporanea.

Algorithm 3: LPA (Semi–synchronous)

Initialize labels: for each $v \in V$, $l_v(0) = v$;
Network coloring: assign a color to the vertices of the network such that no two adjacent vertices share the same color. Let $\mathcal{D} = \{D_1, D_2, \ldots, D_\ell\}$ be the color partitioning obtained.
i=0;
while *the stop criterion is not met* **do**
    i++;
    Propagation:
    for $j = 1$ **to** ℓ **do**
        foreach $v \in D_j$ **do**

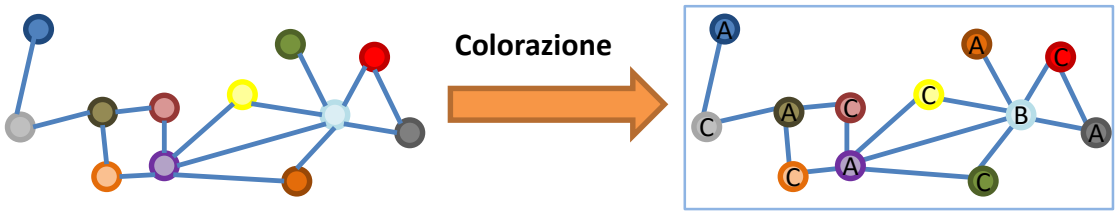
$$l_v(i) = \underset{l}{\operatorname{argmax}} \sum_{u \in N(v)} [l_u == l],$$

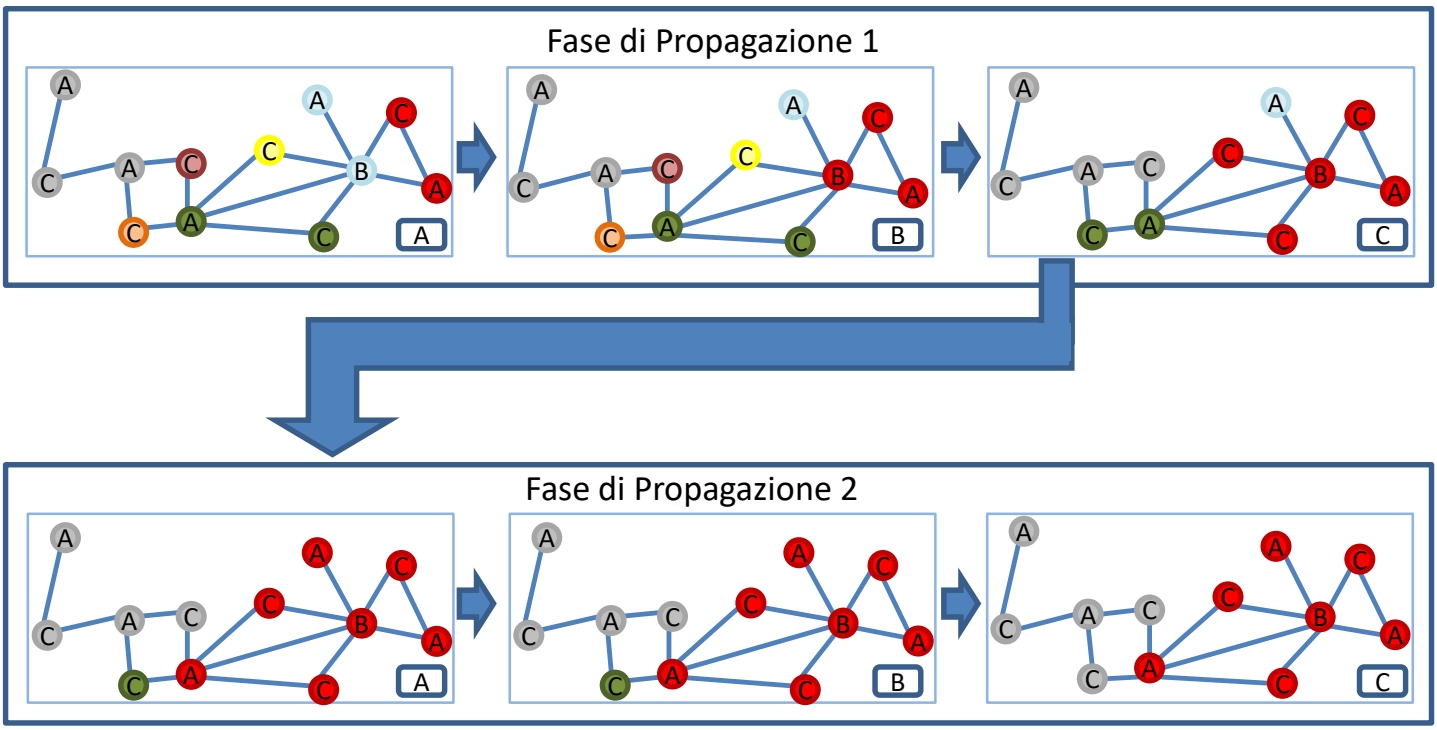
$$\text{where } l_u = \begin{cases} l_u(i) & \text{if } u \in D_k,\ k < j \\ l_u(i-i) & \text{if } u \in D_k,\ k > j \end{cases}$$

        end
    end
end
return Final labeling: $l_v(t)$ for each $v \in V$, where t is the last executed step.

Notiamo che poiché due vertici dello stesso colore sono indipendenti, durante l'esecuzione dell'algoritmo, le etichette di due nodi vicini non vengono mai aggiornate contemporaneamente

Esempio. Notare che, come negli esempi precedenti, le etichette coincidono con il colore del nodo, mentre I colori assegnati ad I nodi sono indicate dale lettere A,B,C





Convergenza

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

Supponiamo che LPA semi-sincrono termina al primo passo t in modo tale che per ogni v ∈ 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 considerate per la gestione dei pareggi

LPA-Random (LPA): una delle etichette che massimizza la somma viene scelta casualmente.
Prec. LPA: se l'etichetta corrente soddisfa l'equazione di etichettatura, 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 l'etichetta viene scelta etichetta mssima.

E' possibile dimostrare il seguente risultato

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.

L'*idea* alla base della dimostrazione del teorema è la seguente.
Notiamo come prima cosa che fintantoché il criterio di stop non è soddisfatto, almeno un vertice v aggiorna la sua etichetta senza che si verifichi un pareggio.

L'algoritmo assicura che quando un vertice v viene rietichettato, i suoi vicini non cambiano le loro etichette (avendo essi un colore differente da quello assegnato a v) . Da questo, è possibile dedurre il numero di edge uniformi (cioè i cui estremi hanno la stessa etichetta/sono nella stessa comunità) aumenta strettamente durante un qualsiasi round i. Quindi, l'algoritmo necessariamente termina. ∎

Risultati sperimentali hanno inoltre mostrato che l'algoritmo LPA-SemiSincrono Risolve il problema della convergenza dell'algoritmo LPA, mantenendone le qualità di efficienza, efficacia e stabilità