

UNIVERSITÀ DEGLI STUDI DI MILANO  
Facoltà di Scienze e Tecnologie  
*Corso di Laurea in Informatica*

UN'APPLICAZIONE WEB PER  
DIMOSTRARE LE PROPRIETÀ DEGLI  
ALGORITMI DI LABEL  
PROPAGATION

**Relatore:** Prof. Paolo CERAVOLO

**Correlatore:** Dr. Samira MAGHOOL

Tesi di:  
Andrei Georgiani TALPALARU  
Matricola: 942824

Anno Accademico 2021-2022

# Ringraziamenti

Ringrazio molto il Professore Paolo Ceravolo per avermi permesso di lavorare su questo progetto, da cui ho imparato tante nuove tecnologie che non ho avuto modo di utilizzare fino ad ora.

Ringrazio la mia famiglia, i miei amici e colleghi, che mi hanno sostenuto e aiutato per tutto questo tempo.

# Indice

<b>Ringraziamenti</b>	<b>ii</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Obiettivo . . . . .	1
1.2 Motivazioni . . . . .	1
1.3 Divisione Documento . . . . .	2
<b>2 Social Network Analysis</b>	<b>3</b>
2.1 Reti . . . . .	3
2.2 Comunità . . . . .	3
2.3 Label Propagation . . . . .	4
2.4 Algoritmi di Label Propagation . . . . .	4
2.5 Belonging Coefficient / Coefficiente di appartenenza . . . . .	4
2.6 AVPRA . . . . .	5
2.6.1 Propagazione basata su agenti . . . . .	5
2.6.2 Aggiornamento sincrono . . . . .	5
2.6.3 Inizializzazione . . . . .	5
2.6.4 Funzione di aggiornamento . . . . .	6
2.6.5 Algoritmo . . . . .	6
2.6.6 Iterazione Finale . . . . .	6
<b>3 Lo sviluppo dell'applicazione</b>	<b>10</b>
3.1 Applicazioni di visualizzazione delle reti sociali . . . . .	10
3.1.1 Visualizzatore Reti Sociali con AVPRA . . . . .	10
3.1.2 Infomap Network Navigator . . . . .	11
3.2 Sviluppo . . . . .	12
3.2.1 Funzionalità previste . . . . .	13
3.2.1.1 Frontend . . . . .	13
3.2.1.2 Backend . . . . .	13
3.2.2 Sviluppo dell'applicazione web . . . . .	14
3.2.2.1 Frontend . . . . .	14

3.2.2.2	Backend . . . . .	16
3.2.3	Presentazione applicazione . . . . .	17
3.2.3.1	Pagina Principale . . . . .	17
3.2.3.2	Esempio Colori . . . . .	18
3.2.3.3	Pagina Caricamento Reti Sociali Personalizzate . . .	28
3.2.3.4	Esempio Personalizzato . . . . .	30
3.3	POPULITE . . . . .	32
3.3.1	Risultati . . . . .	33
<b>4</b>	<b>Sviluppi Futuri</b>	<b>35</b>
4.1	Comunità . . . . .	35
4.2	Deployment . . . . .	35
<b>5</b>	<b>Conclusioni</b>	<b>36</b>

# Capitolo 1

## Introduzione

### 1.1 Obiettivo

L'obiettivo di questo documento è quello di descrivere il contesto e lo sviluppo dell'applicazione web che è stata costruita per mostrare il funzionamento dell'algoritmo di label propagation AVPRA.

### 1.2 Motivazioni

L'applicazione per la visualizzazione delle reti sociali con AVPRA è stata sviluppata per permettere agli utilizzatori di vedere il modo in cui l'algoritmo funziona in una maniera grafica e di poter eseguire l'algoritmo in un modo più facile per gli utenti non esperti.

All'interno dell'applicazione sono visibili le informazioni riguardo all'iterazione finale e al diametro della rete, siccome si voleva capire in una maniera semplice quanto sono vicini i due valori.

## 1.3 Divisione Documento

Il documento è diviso in quattro capitoli:

- il primo capitolo introduce la nozione di Social Network Analysis, definisce alcuni termini usati spesso in questo ambito, descrive il Label Propagation e dà una descrizione dell'algoritmo su cui si basa il progetto descritto di seguito nel documento.
- il secondo capitolo offre un paragone tra l'applicazione sviluppata e altre applicazioni simili per poi andare a descrivere le tecnologie con cui è stata sviluppata l'applicazione. Il capitolo finisce presentando e descrivendo il funzionamento dell'applicazione.
- il terzo capitolo parla del modo in cui l'applicazione potrebbe evolvere nel futuro e del modo in cui l'applicazione può essere resa disponibile attraverso il deployment.
- l'ultimo capitolo descrive le conclusioni a cui sono arrivato dopo lo sviluppo dell'applicazione.

# Capitolo 2

## Social Network Analysis

La Social Network Analysis è l'analisi delle strutture sociali attraverso le reti e la teoria dei grafi. [3]

Grazie ad una rappresentazione astratta, dove le connessioni tra certi elementi di un sistema, chiamati anche nodi o agenti (quando sono capaci di fare scelte o azioni attive), la SNA può essere applicata a una varietà di domini. Il comportamento del sistema, in termini di comunicazione, propagazione ed evoluzione delle connessioni è catturato da metriche di SNA. [5]

### 2.1 Reti

Una rete è composta da due componenti di base: i nodi e gli archi. Usando questi due componenti si possono descrivere una moltitudine di strutture e relazioni (per esempio connessioni sociali, connessioni in una rete di computer, reti elettriche ...).

I nodi sono le entità presenti nella rete e che possono avere proprietà interne (per esempio un valore che rappresenta il grado del nodo, o il peso o una posizione nella rete).

Gli archi sono le connessioni tra i nodi che possono avere a loro volta delle proprietà (per esempio un peso o un altro tipo di relazione tra i nodi collegati alle estremità dell'arco). [3]

### 2.2 Comunità

Una proprietà fondamentale di molte reti è la struttura a comunità, che rappresenta la tendenza dei nodi di essere aggregati in gruppi distinti (o **comunità**) in modo che gli archi tra nodi appartenenti alla stessa comunità siano densi, ma gli archi tra nodi appartenenti a comunità diverse siano sparsi. [4]

## 2.3 Label Propagation

Label Propagation rappresenta una famiglia di algoritmi usati per predire le etichette dei nodi di una rete. Differisce da altri algoritmi di Social Network Analysis siccome, invece di adottare una visione globale della rete, le decisioni prese dall'algoritmo sono fatte rispetto a proprietà locali, quindi gli algoritmi di Label Propagation adottano una visione locale della rete. [6]

L'idea generale è che le etichette assegnate inizialmente ai nodi sono propagate attraverso la rete seguendo gli archi. La propagazione di un'etichetta a un nodo si basa sulle etichette dei suoi vicini, tipicamente ad un salto di distanza. Molteplici iterazioni permettono il raggiungimento di un'approssimazione del ottimo globale partendo da un ottimo locale. La complessità temporale varia tra tempo lineare ed esponenziale, dipendendo dalla densità della rete. L'approccio è naturalmente decentralizzato, con passaggi singoli su ogni nodo ad ogni iterazione. [5]

## 2.4 Algoritmi di Label Propagation

La scelta principale che differenzia i diversi algoritmi di Label Propagation è legata alla funzione di aggiornamento. Alcune varianti della funzione di aggiornamento che sono state usate nel tempo sono: il grado dei nodi, il coefficiente di clustering e la similarità strutturale.

La limitazione principale degli algoritmi precedenti era l'impossibilità di scegliere più di un'etichetta per ogni nodo.

Gli algoritmi sono stati generalizzati per permettere l'identificazione di comunità sovrapposte. Ogni etichetta viene pesata con un belonging coefficient associato al nodo. [5]

## 2.5 Belonging Coefficient / Coefficiente di appartenenza

Il belonging coefficient (o coefficiente di appartenenza) indica quanto forte è l'appartenenza di un nodo a una comunità definita inizialmente da un'etichetta. Tutti i coefficienti di un nodo sommano a 1. [4]



## 2.6 AVPRA

AVPRA (o Agent-based Vector-label PPropagation Algorithm) è un'algoritmo di Label Propagation basato su un'organizzazione ad agenti che permette di implementare la regola di aggiornamento rispetto a più fattori, come per esempio le proprietà di dominio.

AVPRA differisce dagli altri algoritmi tramite il fatto che l'output dell'algoritmo è un vettore compatibile col formato di input usato in maggior parte degli algoritmi di Machine Learning.

Il vettore ha una lunghezza fissata per tutti i nodi e include tutte le etichette della rete assieme al loro coefficiente di appartenenza (basato sulle proprietà strutturali della rete).

Il coefficiente di appartenenza nel caso dell'algoritmo AVPRA è una misura di diffusione/accumulo delle etichette dei nodi, dipendente dalla distanza rispetto ad altri nodi e dalla loro frequenza. [5]

### 2.6.1 Propagazione basata su agenti

Consideriamo che le etichette possono scorrere nella rete e che gli agenti possono tenere conto della frequenza di queste. Gli agenti sono semi-intelligenti e agiscono rispetto alla situazione riscontrata. Possono adottare diverse strategie nella ricezione o rifiuto di un'etichetta che scorre da un nodo vicino, in concordanza con certe condizioni. In questo modo, l'algoritmo può controllare gli effetti globali, come per esempio la condizione di terminazione. [5]

### 2.6.2 Aggiornamento sincrono

Per evitare i conflitti nella propagazione degli aggiornamenti, in AVPRA gli agenti sono aggiornati in modo sincrono. Il vettore delle etichette di un certo nodo a tempo  $t$  sarà ottenuto dai vettori delle etichette dei vicini a tempo  $t - 1$ . In questo modo tutti gli agenti possono aggiornarsi simultaneamente. [5]

### 2.6.3 Inizializzazione

Ogni agente viene etichettato con un vettore di  $d$  dimensioni dove  $d$  è la cardinalità dell'insieme delle etichette uniche nella rete notato  $L$ . Ogni posizione di un vettore di etichette di un nodo  $x$  viene assegnata a un'etichetta dell'insieme  $L$  e nel vettore di etichette vengono memorizzati i coefficienti d'appartenenza del nodo  $x$  rispetto a tutte le etichette.

L'unico requisito dell'inizializzazione è che i vettori di etichette assegnati ai nodi della rete riflettano proprietà reali dei nodi che agiscono come agenti. [5]

### 2.6.4 Funzione di aggiornamento

La regola di aggiornamento mostra il modo in cui i vettori di etichette sono aggiornati ad ogni iterazione. In AVPRA, la funzione di aggiornamento è la seguente:

$$VL_i[l](t) = w_1 VL_i[l](t-1) + w_2 \sum_{j \in \Gamma(i)} VL_j[l](t-1) \quad (1)$$

Dove  $VL_i[l](t)$  rappresenta il valore del coefficiente di appartenenza dell'etichetta  $l$  nel nodo  $n_i$  al tempo  $t$ ,  $w_1$  rappresenta il peso delle etichette assegnate al nodo  $n_i$  a tempo  $t-1$ ,  $w_2$  rappresenta il peso delle etichette assegnate ai nodi  $\Gamma(i)$  a tempo  $t-1$  e  $\Gamma(i)$  è l'insieme dei vicini del nodo  $n_i$ . [5]

### 2.6.5 Algoritmo

Di seguito viene presentato il pseudo-codice che rappresenta il funzionamento dell'algoritmo. [5]

**Require:** Matrice di adiacenza  $A_{i,j}$ , attributi iniziali in vettori di etichette di  $d$  dimensioni per ogni agente

**for**  $t$  in Tempi **do**

    {Iterazione su tutti gli agenti  $i$  della rete}

**for**  $i$  in  $A_{i,j}$  **do**

        {Iterazione su ogni etichetta del agente  $i$ }

**for**  $k$  in  $VL_i$  **do**

            {Se  $j$  è un vicino di  $i$ , si esegue la funzione di aggiornamento}

**if**  $A_{i,j} == 1$  **then**

$VL_i[k] = FunzioneAggiornamento(VL_i[k], VL_j[k])$

**end if**

**end for**

**end for**

**end for**

### 2.6.6 Iterazione Finale

L'algoritmo AVPRA finisce quando il sistema raggiunge un'iterazione  $s$  nella quale tutti i vettori di etichette sono rimasti stazionari per almeno un numero fissato di iterazioni prima. Il stazionamento implica che le variazioni dei coefficienti di appartenenza devono essere minori di un parametro  $p$  che viene chiamato soglia di trascuramento. [5]

La varianza ad un tempo  $t$  (dove  $t > 0$ ) per un nodo  $i$  si calcola usando la formula seguente:

$$Var_i(t) = \frac{\sum_{l \in Etti chete} (VL_i[l](t) - VL_i[l](t-1))}{k} \quad (2)$$

Dove  $k$  rappresenta il numero di etichette uniche della rete, *Etti chete* è l'insieme di etichette della rete e  $VL_i[l](t)$  è il valore del coefficiente di appartenenza dell'etichetta  $l$  al nodo  $i$  a tempo  $t$ .

Nell'esempio sottostante si può osservare come l'iterazione finale è la 4 siccome la variazione tra i vettori di etichette dell'iterazione 4 e quelli dell'iterazione 3 è minore della soglia di trascuramento (di 0.1 in questo caso), e siccome lo stazionamento è avvenuto anche nelle 2 iterazioni precedenti.



(a) Vettore etichette del nodo 1 a tempo 0



(b) Vettore etichette del nodo 2 a tempo 0



(c) Vettore etichette del nodo 3 a tempo 0



(d) Vettore etichette del nodo 4 a tempo 0

Figura 1: Vettori di etichette a tempo 0

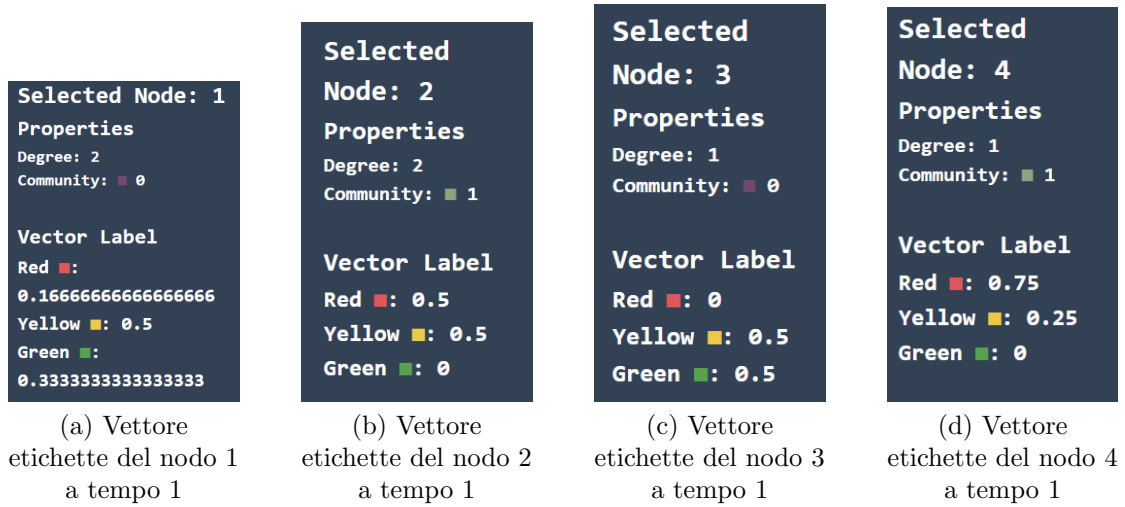


Figura 2: Vettori di etichette a tempo 1



Figura 3: Vettori di etichette a tempo 2

**Selected Node: 1**  
**Properties**  
Degree: 2  
Community: ■ 0  
  
**Vector Label**  
Red ■:  
0.25925925925925924  
Yellow ■:  
0.4722222222222222  
Green ■:  
0.2685185185185185

(a) Vettore  
etichette del nodo 1  
a tempo 3

**Selected Node: 2**  
**Properties**  
Degree: 2  
Community: ■ 1  
  
**Vector Label**  
Red ■:  
0.43981481481481477  
Yellow ■:  
0.43055555555555547  
Green ■:  
0.12962962962962962

(b) Vettore  
etichette del nodo 2  
a tempo 3

**Selected Node: 3**  
**Properties**  
Degree: 1  
Community: ■ 0  
  
**Vector Label**  
Red ■:  
0.15277777777777776  
Yellow ■: 0.5  
Green ■:  
0.3472222222222222

(c) Vettore  
etichette del nodo 3  
a tempo 3

**Selected Node: 4**  
**Properties**  
Degree: 1  
Community: ■ 1  
  
**Vector Label**  
Red ■:  
0.5486111111111112  
Yellow ■:  
0.3958333333333333  
Green ■:  
0.0555555555555555

(d) Vettore  
etichette del nodo 4  
a tempo 3

Figura 4: Vettori di etichette a tempo 3

**Selected Node: 1**  
**Properties**  
Degree: 2  
Community: ■ 0  
  
**Vector Label**  
Red ■:  
0.2839506172839506  
Yellow ■:  
0.4675925925925925  
Green ■:  
0.24845679012345676

(a) Vettore  
etichette del nodo 1  
a tempo 4

**Selected Node: 2**  
**Properties**  
Degree: 2  
Community: ■ 1  
  
**Vector Label**  
Red ■:  
0.4158950617283951  
Yellow ■:  
0.4328703703703703  
Green ■:  
0.15123456790123455

(b) Vettore  
etichette del nodo 2  
a tempo 4

**Selected Node: 3**  
**Properties**  
Degree: 1  
Community: ■ 0  
  
**Vector Label**  
Red ■:  
0.2060185185185185  
Yellow ■:  
0.4861111111111111  
Green ■:  
0.30787037037037035

(c) Vettore  
etichette del nodo 3  
a tempo 4

**Selected Node: 4**  
**Properties**  
Degree: 1  
Community: ■ 1  
  
**Vector Label**  
Red ■:  
0.49421296296296297  
Yellow ■:  
0.4131944444444444  
Green ■:  
0.09259259259259259

(d) Vettore  
etichette del nodo 4  
a tempo 4

Figura 5: Vettori di etichette a tempo 4

Il numero ottimale di iterazioni per la terminazione dell'algoritmo è molto vicino al diametro della rete. Nel esempio presentato, il diametro della rete è 3 e l'iterazione finale è 4, quindi il numero di iterazioni necessarie per la terminazione dell'algoritmo è 4, molto vicino al diametro 3.

## Capitolo 3

# Lo sviluppo di un'applicazione per dimostrare le proprietà dell'algoritmo AVPRA

### 3.1 Applicazioni di visualizzazione delle reti sociali

Tra le poche applicazioni che visualizzano reti sociali è stato deciso di confrontare l'applicazione creata con un'altra che si comporta in modo simile, ma che non ha la capacità di eseguire l'algoritmo AVPRA sulle reti visualizzate.

Per ogni applicazione presa in studio viene data una piccola descrizione ed in seguito vengono elencati **I Pro** ed **I Contro** di ogni una.

#### 3.1.1 Visualizzatore Reti Sociali con AVPRA

AVPRA è un algoritmo di Label Propagation basato su Agenti.

Il visualizzatore delle Reti Sociali per AVPRA è un'applicazione web che permette all'utente di eseguire e visualizzare i risultati dell'algoritmo in un modo interattivo nel browser web.

##### **I Pro**

- Ha tanti parametri che permettono all'utente di controllare il comportamento dell'algoritmo e dell'input.
- Permette l'esportazione della rete e delle informazioni visibili sullo schermo in formato PNG.

- Permette di esportare le informazioni riguardanti la rete e il output in formato JSON.
- Singoli nodi possono essere selezionati in modo da poter vedere più dettagli e misure riguardanti quel specifico nodo.
- Offre la possibilità di selezionare, comparare ed esportare la comparazione tra due nodi diversi della rete.
- Sopporta la visualizzazione di due viste diverse: Vector Label View (mostra in modo grafico la distribuzione delle etichette su ogni nodo) e Community View (mostra in modo grafico la divisione della rete in comunità basate sulla struttura).
- Mostra graficamente la divisione in comunità raggruppando i nodi della stessa comunità più vicini uno all'altro.
- Permette all'utente di eseguire l'algoritmo per più iterazioni e di vedere graficamente l'evoluzione dei Vector Label nel tempo (feature specifica di AVPRA).
- La paletta di colori utilizzata per le etichette nei vettori di etichette può essere modificata usando un'ulteriore file nella fase di caricamento della propria rete.
- Permette all'utente di selezionare un nodo specifico tramite una barra di ricerca dove si può inserire l'id del nodo.

## I Contro

- I dataset dell'utente e i risultati delle esecuzioni dell'algoritmo su quei dataset sono tenute sul server dell'applicazione, anche se per un tempo limitato.
- Performance più scarse nel caso venga usato con reti molto grandi.

### 3.1.2 Infomap Network Navigator

Infomap è un'algoritmo di clustering su reti basato su Map Equation.

Infomap Online è un'applicazione web che permette agli utenti di eseguire l'algoritmo Infomap nel browser web. [1]

Infomap Network Navigator è un'applicazione che permette di visualizzare i risultati dell'algoritmo in un modo interattivo. [2]

### **I Pro**

- I dataset dell'utente e i risultati dell'esecuzione dell'algoritmo su quei dataset non sono mantenute sul server di Infomap Online.
- Ha tanti parametri per controllare il comportamento dell'algoritmo, dell'input, del output e dell'accuratezza.
- Permette di esportare la visualizzazione nei formati SVG e PNG.
- Permette di esportare il file di output.
- Singoli nodi possono essere selezionati in modo da vedere più dettagli e misure riguardanti quel specifico nodo.
- Permette all'utente di selezionare un nodo specifico tramite una barra di ricerca.
- Mostra graficamente il raggruppamento dei nodi in comunità e sotto-comunità attraverso un modulo (caratteristica specifica Map Equation).

### **I Contro**

- L'esportazione della visualizzazione non include nient'altro oltre la rete stessa.
- Non offre la possibilità di vedere il modo in cui due nodi diversi della rete siano correlati.
- Esiste un limite sulla dimensione dei file di input che si possono utilizzare.
- Non esiste un modo per personalizzare la paletta di colori che viene usata nella visualizzazione.

## **3.2 Sviluppo**

Il progetto è un'applicazione web che è divisa in 2 parti:

- Frontend - la componente dell'applicazione con cui l'utente interagisce direttamente.
- Backend - la componente dell'applicazione con cui il frontend interagisce e che si occupa di esporre un'API che esegue le funzionalità richieste dal frontend.



### 3.2.1 Funzionalità previste

#### 3.2.1.1 Frontend

- Deve permettere all'utente di visualizzare in modo interattivo la situazione, alla iterazione corrente, della rete e dei suoi vettori di etichette.
- Deve implementare un modo in cui l'utente possa caricare la sua propria rete.
- Deve dare la possibilità all'utente di avere più informazioni riguardo lo stato corrente di un vettore di etichette di un nodo specifico.
- Deve permettere di avere informazioni riguardo l'iterazione corrente e l'iterazione finale dell'algoritmo.
- Deve implementare un modo in cui si possa andare avanti e indietro tra le iterazioni in modo da vedere l'evoluzione dei vettori di etichette.
- Deve fare possibile la visualizzazione delle comunità associate alla rete.
- Deve permettere di selezionare 2 nodi e di vedere la situazione corrente di entrambi.
- Deve implementare un modo di esportare i file che si possono esportare attraverso l'API esposta dal backend.

#### 3.2.1.2 Backend

- Dev'essere capace di eseguire l'algoritmo AVPRA dato attraverso un script Python.
- Deve esporre un'API che:
  - permette all'utilizzatore di eseguire l'algoritmo.
  - permette all'utilizzatore di controllare i parametri dell'algoritmo come: numero di iterazioni, soglia di trascuramento, interpretazione della rete (diretto, non diretto), modo di propagazione delle label (da predecessori, da successori) e formula dei pesi della regola di aggiornamento.
  - permette di ottenere i dettagli riguardanti un solo nodo ad una certa iterazione.
  - permette di ottenere i dettagli riguardanti la relazione tra due nodi ad una certa iterazione.

- permette di esportare i risultati e i dettagli dell'esecuzione in un formato definito.
- permette all'utente di caricare ed eseguire l'algoritmo sulla propria rete usando un formato definito.

## 3.2.2 Sviluppo dell'applicazione web

### 3.2.2.1 Frontend

Per lo sviluppo della parte Frontend dell'applicazione, contenente l'interfaccia utente e la comunicazione col backend, sono state utilizzate diverse tecnologie e librerie. che sono presentate in seguito.

**React.js** è una libreria di JavaScript che viene usata per creare in modo facile delle interfacce utente interattive. I componenti più importanti di questa libreria, che sono utilizzati all'interno dell'applicazione, sono presentati in seguito:

- **Componenti**

I componenti sono le parti che permettono di organizzare l'interfaccia utente in React. Questi possono essere definiti in due modi: Classi Componente e Funzioni Componente. [7]

Le **Funzioni Componente** sono funzioni JavaScript che prendono in input dei dati (chiamati props) e ritornano degli elementi React che descrivono, tramite JSX, il modo in cui dovrebbe essere visualizzato sullo schermo. [8]

Le **Classi Componente** sono classi che estendono **React.Component** e che implementano un metodo **render** che restituisce un elemento React. [8]

- **Stato**

Gli aggiornamenti della UI in JavaScript base dovevano accadere trovando l'elemento nel DOM e aggiornandolo manualmente. React, invece, aggiorna automaticamente l'interfaccia utente in base allo stato interno del componente.

Prima dell'aggiunta dei Hooks, per modificare il stato interno di un componente era necessario che esso venga scritto come Classe Componente. Dopo l'aggiunta del hook **useState**, il stato poteva essere utilizzato anche all'interno di una Funzione Componente.

È necessario usare **useState** (Funzione Componente) o **setState** (Classe Componente) in modo che React capisca che l'interfaccia utente che usa il stato interno debba essere renderizzato nuovamente per mostrare il valore aggiornato dello stato. [9]

- **Hooks**

Gli Hooks sono stati aggiunti per permettere di utilizzare stato e altre funzioni che esistevano nelle Classi Componente anche nelle Funzioni Componente [11].

Alcuni esempi di hooks maggiormente utilizzati sono:

- **useState** - hook che permette di aggiungere stato a funzioni componente.
- **useEffect** - hook che permette di aggiungere side effects in funzioni componente, il che significa poter far succedere qualcosa in base al cambiamento del stato delle dipendenze dichiarate . Può essere usato anche per simulare i comportamenti **componentDidMount** e **componentDidUpdate** delle Classi Componenti.

All'interno dell'applicazione sono stati creati alcuni hook che hanno permesso di facilitare il processo di sviluppo.

I hook che sono stati implementati sono:

- **useGUIControls** che mette a disposizione le funzioni e i dati da utilizzare per i controlli delle pagine riguardanti la visualizzazione dei grafi.
- **usePolling** che permette di chiamare un endpoint dell'API in modo ripetuto e a intervalli regolari nel caso esso risponde con un codice che indica che i dati non sono pronti per il recupero.
- **useSS** che permette di fotografare il schermo e viene utilizzato nel caso in cui si preme il tasto di Screenshot sull'applicazione.

- **JSX**

JSX rappresenta un'estensione di JavaScript che permette di scrivere HTML direttamente all'interno di JavaScript. Usando Babel, JSX viene poi compilato in chiamate **React.createElement()**. Ciò significa che JSX può essere usato anche in strutture condizionali, per poter produrre risultati diversi rispetto a una decisione. [10]

**Typescript** è un'estensione di JavaScript che aggiunge una sintassi per i tipi e una migliore integrazione coi editor, aiutando in questo modo il sviluppatore a trovare velocemente gli errori dovuti ai tipi. [12]

**Next.js** è un framework di React che risolve maggior parte dei problemi che si incontrano quando si cerca di lavorare solo con la libreria React. Per framework si intende che Next.js gestisce in modo autonomo gli strumenti e le configurazioni che React necessita, offrendo però anche funzionalità aggiuntive. [13]

**D3.js** è una libreria per manipolazione di documenti in base a dati. D3.js permette di associare dati al DOM e costruire visualizzazioni per quei dati sull'applicazione che si sta creando. Nel caso presente D3.js viene utilizzato per costruire la visualizzazione del grafo e per aggiungere interattività ad esso. [14]

**Katex** è una libreria JavaScript usata per visualizzare formule matematiche sulla pagina. Nel caso presente, questa libreria è stata usata per poter mostrare all'utente la formula coi pesi che si stanno inserendo quando si usa la visualizzazione delle reti personalizzate. [15]

**Tailwind CSS** è un framework di CSS che offre classi di utilità che aiutano il sviluppatore a evitare di scrivere CSS direttamente. Usando solo Tailwind CSS si può evitare di usare valori arbitrari per maggior parte delle misure, e quindi si arriva ad essere più consistenti col design dell'applicazione. Nell'applicazione presentata, questo framework è stato usato per fare tutto il layout del sito. [16]

### 3.2.2.2 Backend

Per lo sviluppo della parte backend dell'applicazione, contenente l'API che ha la capacità di rispondere alle richieste del frontend, sono state utilizzate diverse tecnologie e librerie che sono presentate di seguito.

**Flask** è un microframework web basato su Python che è stato usato, nel caso presente, per lo sviluppo del backend come un REST API. [18]

Flask permette di creare un'API nel modo più semplice possibile, ed è stato scelto come tecnologia di backend poiché l'algoritmo che viene usato per la elaborazione delle reti è scritto in Python, e quindi la scelta più facile per un linguaggio di backend era Python.

Nel caso dell'esempio dei colori, il server di backend risponde puramente da file statici con la rete già processata. Nel caso degli esempi personalizzati, i file necessari vengono caricati sul server, vengono processati, e fino alla scadenza della sessione, si può visualizzare la rete caricata e processata.

**Sympy** è una libreria di Python per l'interpretazione di matematica simbolica. È stato usato per poter interpretare le formule per i pesi che vengono mandate dal Frontend nel caso di esempi personalizzati. [19]

**CDlib** è una libreria che aggrega tanti algoritmi di community detection. È stato utilizzato solamente per l'implementazione dell'algoritmo di community detection di Leiden che era necessario nel caso delle reti dirette. [20]

### 3.2.3 Presentazione applicazione

#### 3.2.3.1 Pagina Principale

La pagina principale serve per permettere all'utente di scegliere tra due modalità diverse di provare e visualizzare i risultati dell'algoritmo. Ogni modo ha una piccola descrizione del come funziona e un tasto per andare alla pagina specifica per quella modalità. Ogni modalità sarà descritta in dettaglio nelle sezioni successive.

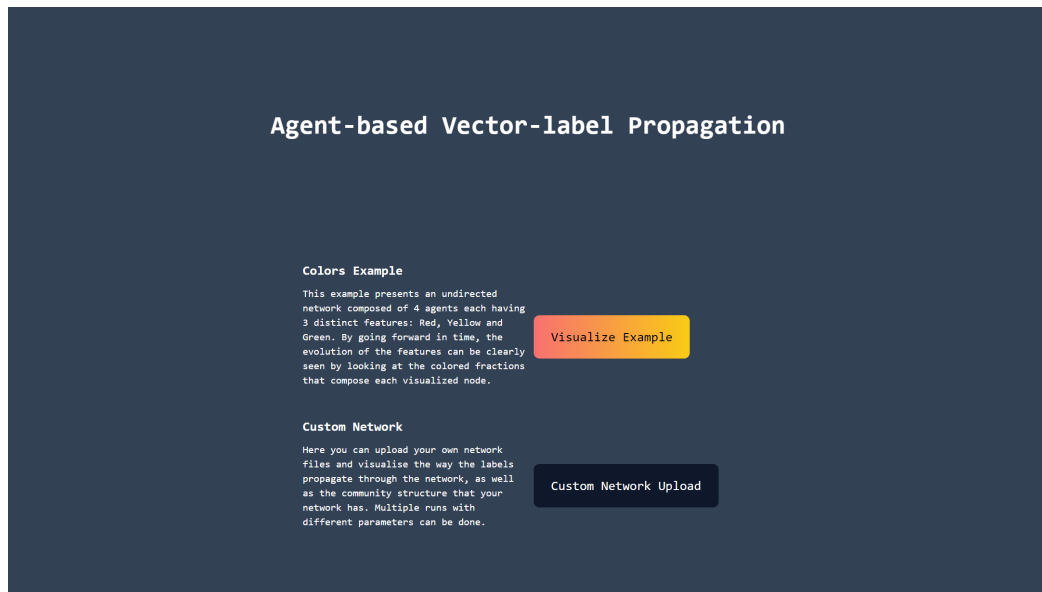


Figura 6: Pagina Principale

### 3.2.3.2 Esempio Colori

La pagina dell'esempio colori, che si può raggiungere premendo il bottone "Visualize Example" vicino a "Colors Example" sulla pagina principale, ha tutte le funzionalità dell'applicazione, dimostrate su un semplice esempio di grafo non diretto con 4 nodi e 3 archi. In questo esempio si può vedere il modo in cui le etichette si propagano ad ogni iterazione attraverso i colori.



Figura 7: Esempio Colori

I componenti che compongono l'interfaccia di visualizzazione della rete sono presentati di seguito.

**La barra di controllo** contiene tutte le modalità che si possono usare per controllare la visualizzazione della rete.

Iniziando da sinistra e andando verso destra, la barra di controllo contiene per prima cosa un rettangolo con dei tasti per controllare l'iterazione che viene visualizzata correntemente, assieme alla iterazione che viene considerata finale dall'algoritmo (nel caso dell'esempio colori, la soglia di trascuramento viene fissata a 0.1).

Una delle proprietà dell'algoritmo che viene presentata in questo componente è l'iterazione finale.

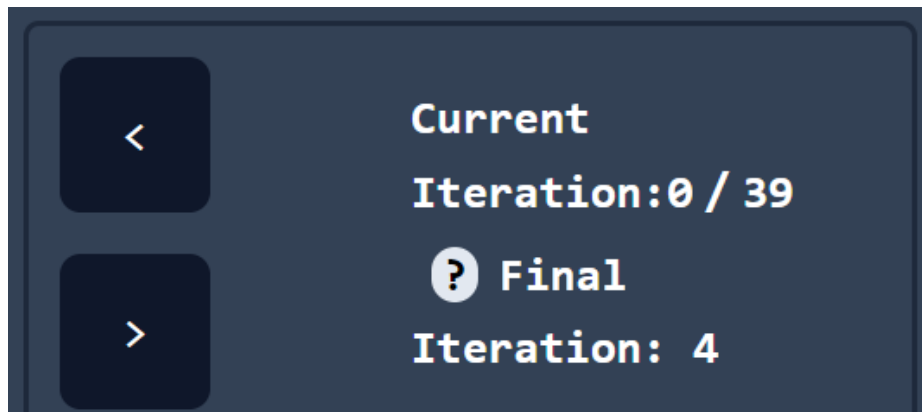


Figura 8: Controlli Iterazione

La seconda parte della barra di controllo viene utilizzata per cambiare il modo di selezione nodo e per cercare un nodo specifico nella rete. Se si preme il tasto per cambiare il modo di selezione nodo, si può selezionare un secondo nodo, in modo che, ulteriormente, si possono comparare i vettori di etichette dei due nodi selezionati.

La barra di ricerca di un nodo specifico diventa molto utile nel caso in cui la rete utilizzata ha molti nodi e quindi dove la selezione del nodo dalla VL View diventa difficile.

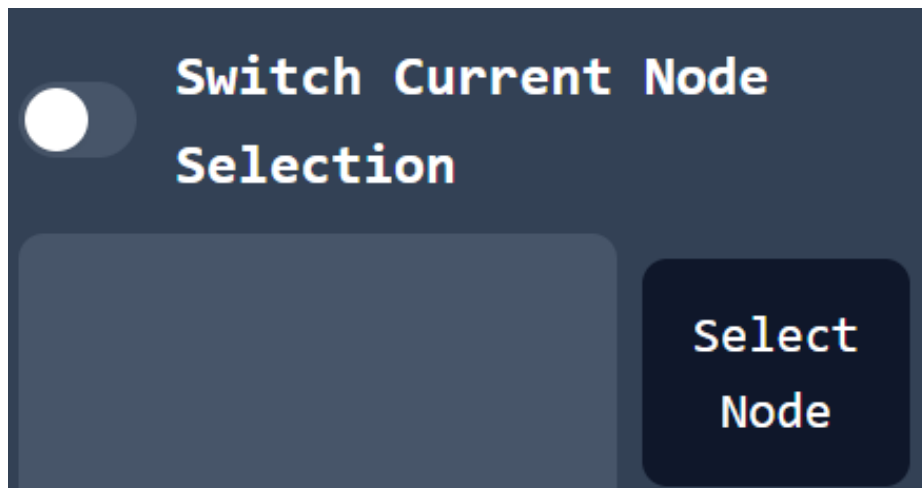


Figura 9: Controlli Selezione Nodo

L'ultima parte della barra di controllo contiene alcuni controlli per cambiare il layout di visualizzazione e alcuni tasti per: catturare la configurazione corrente dell'interfaccia, scaricare il file dei risultati dell'algoritmo, scaricare il file contenente i dettagli del grafo e scaricare un file contenente la comparazione di due nodi selezionati (tasto attivo solo nel caso in cui sono stati già selezionati due nodi).

Le opzioni per cambiare il layout dell'applicazione sono: Show VL View (che viene utilizzato per nascondere o per visualizzare la vista che presenta graficamente la configurazione dei vettori di etichette), Show Community View (che viene utilizzato per nascondere o per visualizzare la vista che presenta graficamente la divisione in comunità della rete), Show Info Bar (che viene utilizzato per nascondere e far comparire la barra laterale che presenta le informazioni sulla rete, o sui nodi selezionati) e Show Ids (che viene utilizzato per nascondere o per visualizzare gli ID all'interno di ogni nodo in una vista).

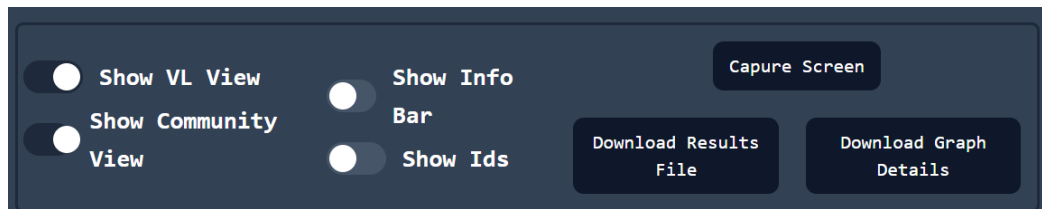


Figura 10: Controlli Vista



Il formato del file JSON in cui i dettagli della rete viene esportata e il seguente:

```
{
  "directed": ...,
  "multigraph": ...,
  "graph": {
    "labels": [...],
    "diameter": ...,
    "communities": ...,
    "nodesCount": ...,
    "edgesCount": ...,
    "centrality": {
      "1": ...,
      "2": ...,
      ...
    },
    "timestamps": ...,
    "finalTs": ...
  },
  "nodes": [
    {
      "community": ...,
      "degree": ...,
      "neigh": [...],
      "id": ...
    },
    ...
  ],
  "links": [
    {
      "source": ...,
      "target": ...
    },
    ...
  ]
}
```

Il formato del file JSON in cui la comparazione nodi viene esportata è il seguente:

```
{
  "node1":
  {
    "community": ...,
    "degree": ...,
    "neigh": [...],
    "id": ...,
    "VL": [...],
    "betweenness centrality": ...
  },
  "node2":
  {
    "community": ...,
    "degree": ...,
    "neigh": [...],
    "id": ...,
    "VL": [...],
    "betweenness centrality": ...
  },
  "cosineSimilarity": ...,
  "timestamp": ...
}
```

**La VL View** rappresenta la vista che visualizza la rete dal punto di vista dei vettori delle etichette. Tutti i belonging coefficient sommano a 1 e quindi ogni etichetta viene rappresentata da un colore che occupa la percentuale specifica sul nodo.

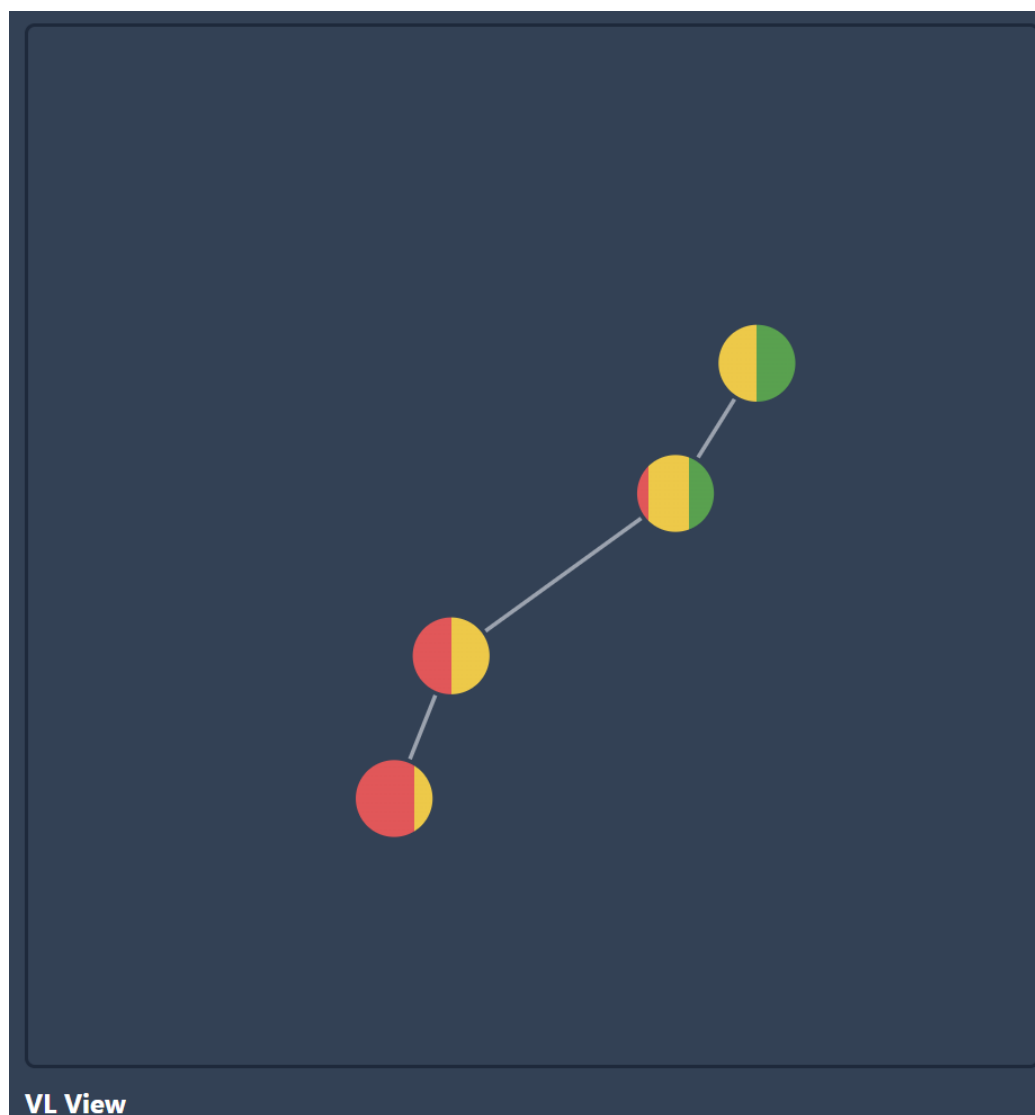


Figura 11: Vista Vettori di etichette

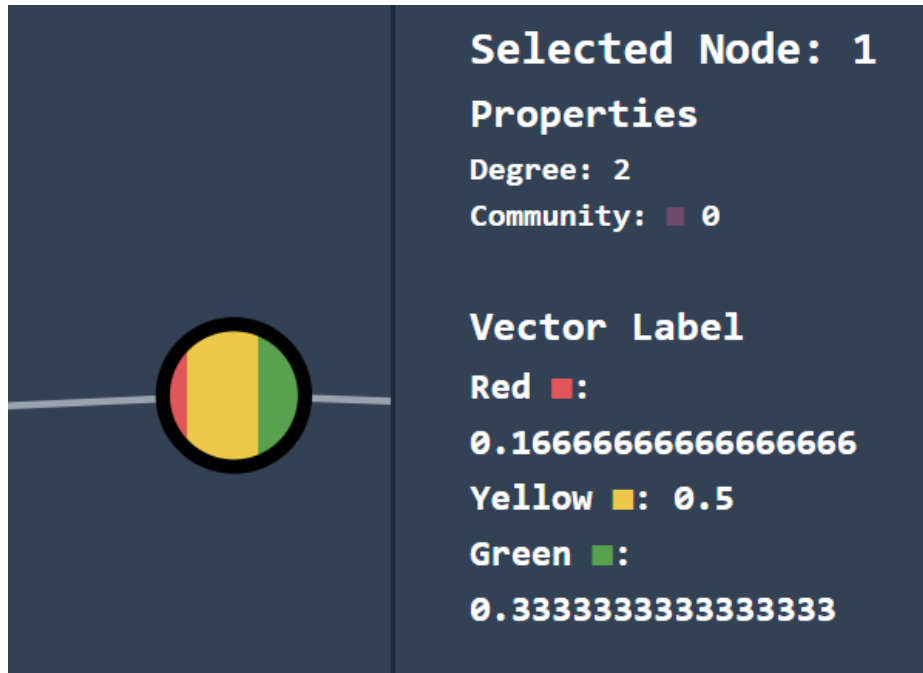


Figura 12: Un nodo nella vista Vettori di etichette

Nella figura 12 si può vedere come ad ogni etichetta viene associato un colore vicino al suo nome, e un valore che rappresenta il belonging coefficient di quell'etichetta per il nodo selezionato, all'iterazione corrente.

I colori dentro il nodo sono rappresentati in questo modo siccome l'etichetta **Red** ha un belonging coefficient di 0.166... (quindi 16.6...% del nodo viene riempito col colore rosso), l'etichetta **Yellow** ha un belonging coefficient di 0.5 (quindi 50% del nodo viene riempito col colore giallo) e l'etichetta **Green** ha un belonging coefficient di 0.333... (quindi 33.3...% del nodo viene riempito col colore verde).

**La Community View** rappresenta la vista che visualizza la rete dal punto di vista delle comunità che compongono la rete. I nodi che si trovano nella stessa comunità avranno lo stesso colore, in modo da identificare facilmente i nodi appartenenti alla stessa comunità dal punto di vista strutturale. La comunità di ogni nodo può essere conosciuta dalle informazioni specifiche di ogni nodo, quando esso viene selezionato.

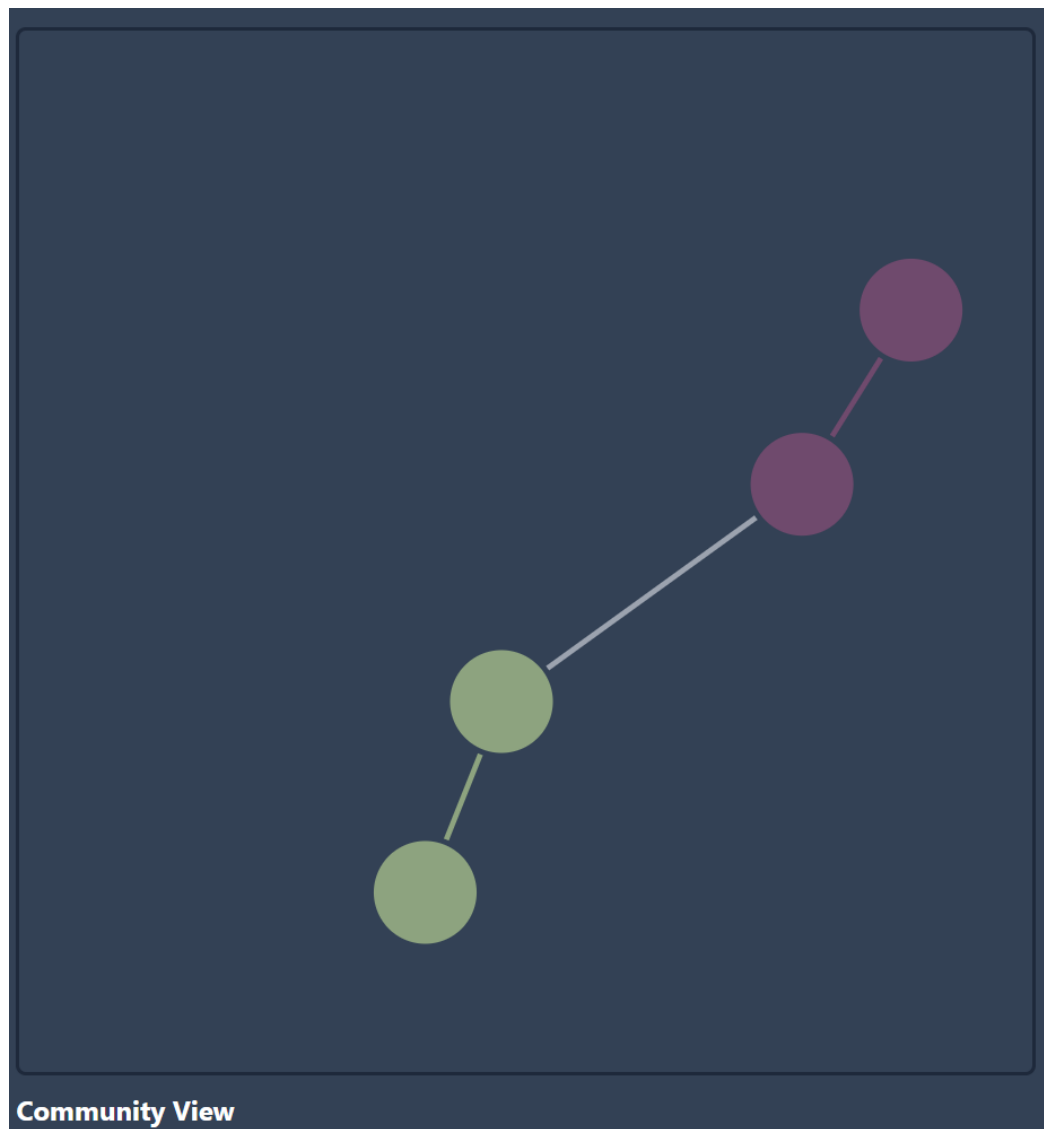


Figura 13: Vista comunità

**La Barra Informazioni** visualizza, nel caso in cui nessun nodo viene selezionato, le informazioni generali riguardanti la rete, come tipo di rete, numero di nodi, numero di archi e diametro. Nel caso in cui un nodo viene selezionato, la barra delle informazioni mostra il grado di quel nodo, la comunità appartenente, assieme al suo colore e il vettore delle etichette coi belonging coefficient di ogni etichetta, assieme al colore dell'etichetta. Nel caso in cui due nodi vengono selezionati, ci saranno due barre di informazioni che mostrano le stesse informazioni di prima per entrambi i nodi.

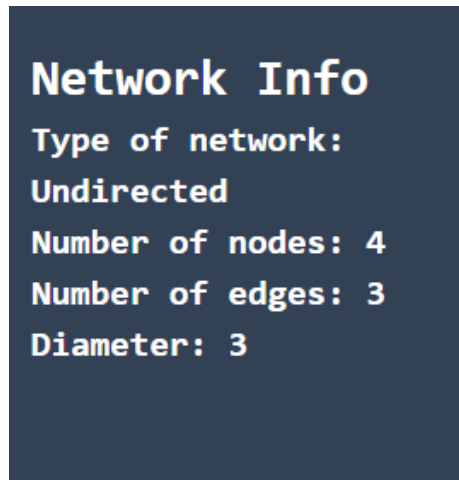


Figura 14: Barra Informazioni senza un nodo selezionato



Figura 15: Barra Informazioni con un solo nodo selezionato

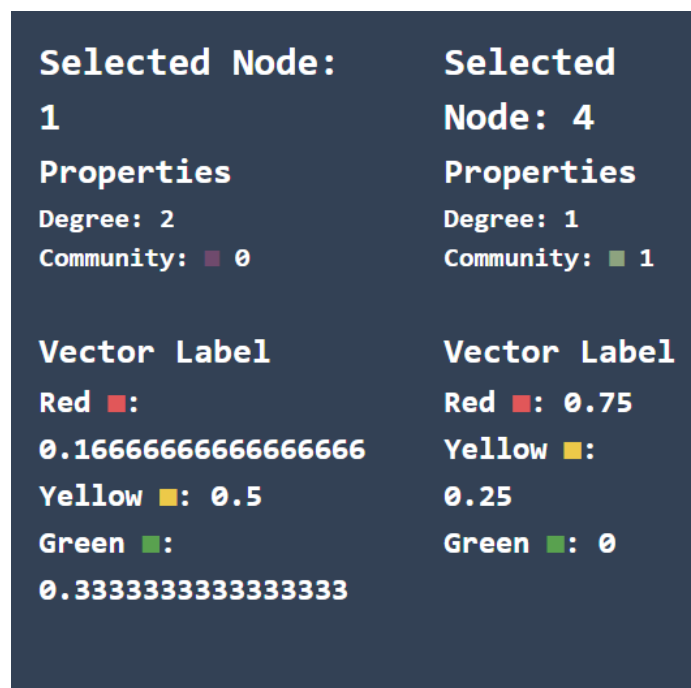


Figura 16: Barra Informazioni con due nodi selezionati

### 3.2.3.3 Pagina Caricamento Reti Sociali Personalizzate

Nel caso della pagina di caricamento delle reti sociali personalizzate, l'utente può caricare i suoi dataset personali, formattati in un formato descritto nella pagina stessa e di seguito, e visualizzare i risultati di AVPRA sulla sua rete.

L'utente ha la possibilità di personalizzare diverse opzioni riguardanti l'interpretazione dei dati e i parametri con cui l'algoritmo sarà eseguito.

I file riguardanti la rete che l'utente può caricare sull'applicazione sono i seguenti:

- Il file degli archi

Questo file dev'essere un file .csv col seguente formato:

$$\begin{array}{ccc} & \dots & \\ \text{nodo}_1 & \text{nodo}_2 & \{p\} \\ & \dots & \end{array} \quad (3)$$

Questo formato mostra che il  $\text{nodo}_1$  è collegato al  $\text{nodo}_2$  e che l'arco ha il peso  $p$ . Il peso è un parametro opzionale e quindi il file può essere definito anche nel seguente modo:

$$\begin{array}{ccc} & \dots & \\ \text{nodo}_1 & \text{nodo}_2 & \\ \text{nodo}_2 & \text{nodo}_4 & \\ & \dots & \end{array} \quad (4)$$

- Il file dei vettori delle etichette con cui i nodi vengono inizializzati.

Questo file dev'essere un file .csv col seguente formato:

$$\begin{array}{c} b_1(l_1); b_1(l_2); \dots; b_1(l_k) \\ b_2(l_1); b_2(l_2); \dots; b_2(l_k) \\ b_3(l_1); b_3(l_2); \dots; b_3(l_k) \\ \dots \end{array} \quad (5)$$

In questo caso  $b_1, b_2, b_3, \dots$  sono le funzioni del coefficiente d'appartenenza dei nodi  $1, 2, 3, \dots$  e  $l_1, l_2, \dots, l_k$  sono le etichette della rete. I valori di ogni riga devono sommare a 1.



- Il file dei nomi delle etichette

Questo file dev'essere un file .csv col seguente formato:

$$l_1; l_2; l_3; l_4; \dots \quad (6)$$

- Il file dei colori con cui vengono rappresentate le etichette.

Questo file dev'essere un file .csv che ha il seguente formato:

$$hex_1; hex_2; \dots \quad (7)$$

In questo formato  $hex_1, hex_2, \dots$  sono i codici esadecimali dei colori con cui saranno rappresentate le etichette  $l_1, l_2, \dots$  nella visualizzazione.

Un esempio del formato di questo file è:

$$\#433A3F; \#404A56; \#3D5A6C; \#58827E; \#659687; \dots \quad (8)$$

Le opzioni che l'utente può scegliere sono le seguenti:

- Grafo diretto o non diretto

Quest'opzione permette all'utente di scegliere se il grafo costruito dal file degli archi viene interpretato come un grafo diretto o non diretto.

- Propagazione delle etichette dai predecessori o successori

Quest'opzione permette all'utente di scegliere se la propagazione delle etichette viene influenzata dagli nodi vicini entranti o dagli nodi vicini uscenti. Nel caso in cui il grafo viene interpretato come un grafo non diretto, quest'opzione non è disponibile.

- Negligibly Threshold / Soglia di trascuramento

Quest'opzione permette all'utente di scegliere il valore del parametro negligibly threshold associato all'algoritmo, da cui dipende l'iterazione finale dell'algoritmo.

- Numero di iterazioni

Quest'opzione permette all'utente di scegliere il numero di iterazioni per cui l'algoritmo sarà eseguito.

- Formula per il peso nella funzione di aggiornamento

Quest'opzione permette all'utente di scegliere una formula (dipendente dal numero di vicini del nodo) per i pesi nella funzione di aggiornamento dell'algoritmo AVPRA.

$$VL_i[l](t) = (1 - w)VL_i[l](t - 1) + w \sum_{j \in \Gamma(i)} VL_j[l](t - 1) \quad (9)$$

Nella formula 9, il peso che viene scelto dall'utente è  $w$ . Se l'utente non sceglie un peso personalizzato per  $w$ , allora  $w = \frac{1}{k+1}$ , dove  $k$  è il numero di vicini del nodo  $i$ .

### 3.2.3.4 Esempio Personalizzato

Nel caso dell'esempio personalizzato, l'interfaccia utente è praticamente uguale all'interfaccia presente nell'Esempio Colori, con qualche piccola differenza data dalle opzioni che l'utente ha scelto nella fase di caricamento della rete.

**La barra di informazioni** si comporta nello stesso modo della barra di informazioni dell'Esempio Colori, ma offre più informazioni generali riguardo la rete caricata sull'applicazione.

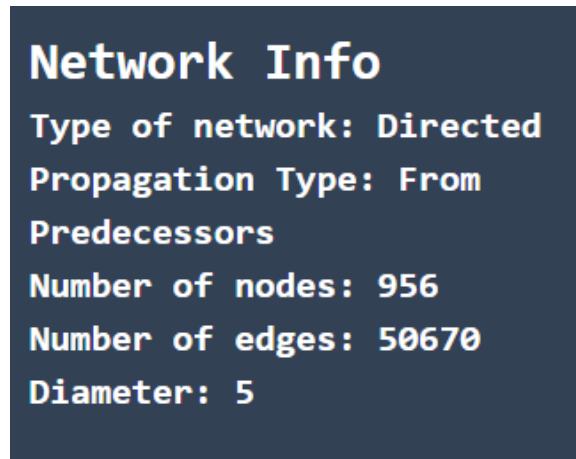
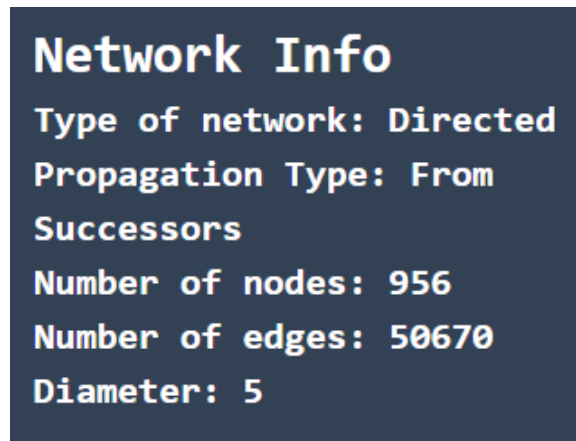
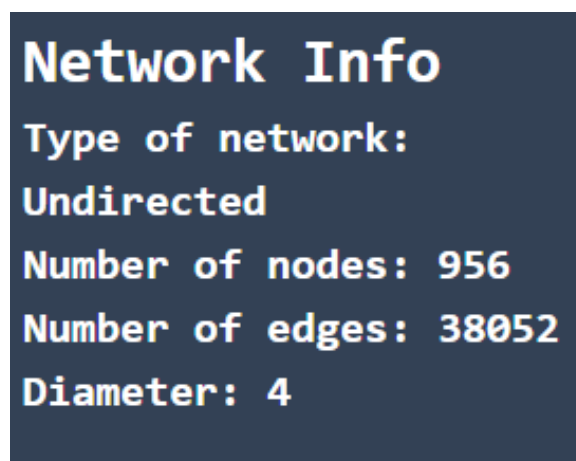


Figura 17: Barra Informazioni Generale per una rete diretta e con propagazione delle etichette da predecessori



**Network Info**  
Type of network: Directed  
Propagation Type: From  
Successors  
Number of nodes: 956  
Number of edges: 50670  
Diameter: 5

Figura 18: Barra Informazioni Generale per una rete diretta e con propagazione delle etichette da successori



**Network Info**  
Type of network:  
Undirected  
Number of nodes: 956  
Number of edges: 38052  
Diameter: 4

Figura 19: Barra Informazioni Generale per una rete non diretta

**La barra di controllo** subisce modifiche solo nella sezione di Controlli Iterazione. In questo caso, il numero massimo delle iterazioni viene sostituito col numero di iterazioni scelto dall'utente durante il caricamento della rete, e l'iterazione finale cambia in modo che il numero d'ordine mostrato sia l'iterazione dove avviene la condizione di terminazione.

Anche in questo caso, la proprietà dell'algoritmo che viene mostrata è l'iterazione finale.

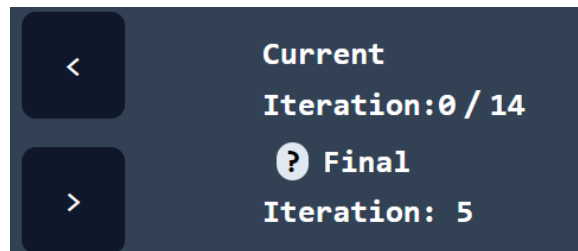


Figura 20: Controlli Iterazione per reti personalizzate

### 3.3 POPULITE

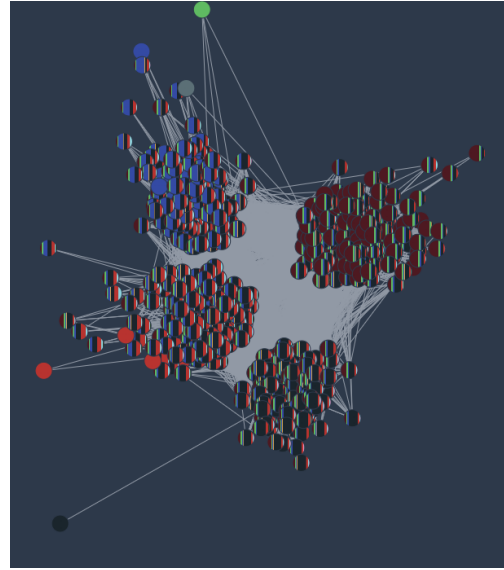
POPULITE (POPUList Language in ITalian political Elites) è un dataset in cui ogni nodo rappresenta un politico che appartiene ad un solo partito. I politici possono essere collegati con un arco ed ogni arco  $(P, Q)$  ha il significato che il politico  $P$  segue e prende influenza dal politico  $Q$ .

### 3.3.1 Risultati

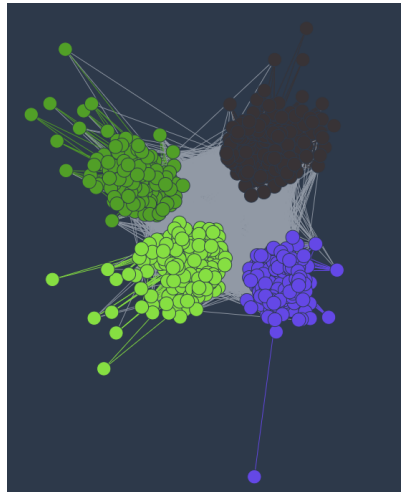
Di seguito sono presentati dei screenshot che mostrano che, allo stesso modo in cui, strutturalmente, i nodi si aggregano in diverse comunità, anche i vettori di etichette di ogni nodo della stessa comunità sono simili. Dal punto di vista semantico si può vedere come membri dello stesso partito (stesso colore all'iterazione 0) hanno più probabilità di finire nella stessa comunità grazie al fatto che si seguono.



(a) Vista Vettori di Etichette  
all'iterazione 0



(b) Vista Vettori di Etichette  
all'iterazione 5



(c) Vista Comunità

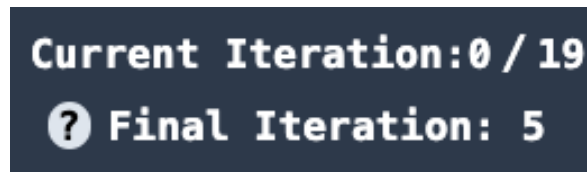


Figura 21: Controllo Iterazione

Il diametro della rete di questo dataset è 5 e come si può vedere dalla sezione di controllo iterazione, l'iterazione finale è la 5, quindi è stato necessario un numero di iterazioni vicino al diametro per arrivare al stazionamento dei vettori di etichette. Nel caso presentato è stato usata una soglia di trascuramento di 0.1.

## Capitolo 4

# Sviluppi Futuri

### 4.1 Comunità

La sezione riguardante le comunità può essere estesa in modo da permettere l'utilizzo di un modello di Machine Learning per poter riconoscere le comunità rispetto ai vettori di etichette ad ogni iterazione, invece della struttura.

### 4.2 Deployment

Una delle cose che vanno fatte nel futuro è la pubblicazione dell'applicazione web in modo che gli utenti possano utilizzare l'applicazione in modo semplice.

Il frontend, che è stato sviluppato con l'aiuto di Next.js, può essere facilmente distribuito attraverso un server Node.js basato, per esempio, su **AWS EC2** o un **DigitalOcean Droplet** [21]. Un altro modo molto semplice per la distribuzione del frontend è attraverso **Firebase Hosting** di **Google Firebase**, che dopo qualche semplice configurazione, permette la pubblicazione dell'applicazione web in modo gratuito (entro certi limiti d'utilizzo) [22].

Il backend, sviluppato usando Flask e scritto in Python, può essere distribuito usando le **Google Cloud Functions** di **Google Firebase**, in modo che ogni endpoint sia collegato a una funzione in cloud che può essere chiamata dal frontend [23].

# Capitolo 5

## Conclusioni

Lo scopo del documento presentato è quello di descrivere lo sviluppo e l'applicazione sviluppata in se.

L'applicazione è stata sviluppata per mostrare alcune proprietà dell'algoritmo AVPRA in un modo grafico e facile da capire per gli utenti di tutti i tipi.

Le proprietà che sono state mostrate nell'applicazione sono le seguenti:

- L'iterazione finale dell'algoritmo viene sempre mostrata nella barra di controllo vicino all'iterazione correntemente visualizzata.
- All'iterazione 0 si può vedere visivamente il modo in cui i vettori di etichette sono inizializzati con un solo belonging coefficient a 1 e il resto a 0. (quest'affermazione non vale nel caso in cui nell'esempio personalizzato viene caricato un file d'inizializzazione che non rispetta questa regola).
- Ad ogni iterazione si può vedere come i valori dei belonging coefficient all'interno dei vettori di etichette di ogni nodo cambiano in modo corretto rispetto alla regola di aggiornamento coi pesi selezionati.
- Ad ogni iterazione si può vedere come l'aggiornamento si fa in modo sincrono, poiché, rifacendo i calcoli, i valori risultati ad un tempo  $t$  sono calcolati rispetto ai valori del tempo  $t - 1$ .
- Dopo aver eseguito l'algoritmo su più reti è stato osservato che il numero di iterazioni necessarie per arrivare all'iterazione finale dell'algoritmo è molto vicina al diametro della rete. Questa affermazione è vera sia per l'esempio colori e sia per POPULITE.



# Bibliografia

- [1] Map Equation, <https://www.mapequation.org/infomap/>
- [2] Map Equation Navigator, <https://www.mapequation.org/navigator/>
- [3] Social Network Analysis: From Graph Theory to Applications with Python, <https://towardsdatascience.com/social-network-analysis-from-theory-to-applications-with-python-d12e9a34c2c7>
- [4] Gregory, S.: Finding overlapping communities in networks by label propagation. New journal of Physics 12(10), 103018 (2010)
- [5] Valerio Bellandi, Paolo Ceravolo, Ernesto Damiani, and Samira Maghool: Agent-based Vector-label Propagation for Explaining Social Network Structures. In: Springer Verlag (Lecture Notes in Communications in Computer and Information Science (CCIS)) (2022)
- [6] Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Physical review E 76(3), 036106 (2007)
- [7] React.Component, <https://it.reactjs.org/docs/react-component.html>
- [8] Componenti e Props, <https://it.reactjs.org/docs/components-and-props.html>
- [9] State e Lifecycle, <https://it.reactjs.org/docs/state-and-lifecycle.html>
- [10] Introduzione a JSX, <https://it.reactjs.org/docs/introducing-jsx.html>
- [11] Introduzione agli Hooks, <https://it.reactjs.org/docs/hooks-intro.html>
- [12] Typescript, <https://www.typescriptlang.org/>
- [13] What is Next.js, <https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs>

- [14] Data-Driven Documents, <https://d3js.org/>
- [15] katex, <https://www.npmjs.com/package/katex>
- [16] Tailwind CSS, <https://tailwindcss.com/>
- [17] Flask, <https://flask.palletsprojects.com/en/2.1.x/>
- [18] Flask Foreword, <https://flask.palletsprojects.com/en/2.1.x/foreword/>
- [19] SymPy, <https://www.sympy.org/en/index.html>
- [20] CDlib - Community Discovery Library, <https://cdlib.readthedocs.io/en/latest/>
- [21] Next.js Deployment, <https://nextjs.org/docs/deployment>
- [22] Firebase Hosting, <https://firebase.google.com/docs/hosting>
- [23] Google Cloud Functions, <https://cloud.google.com/functions>