A MapReduce-Based Parallel Clustering Algorithm for Large Protein-Protein Interaction Networks

Gruppo 5: Marco Cavallo, Romeo Silvestri, Vittoria Zagarese

Introduzione

L'obbiettivo del progetto è quello di fare un clustering su un network di molecole di tipo PPI utilizzando l'algoritmo di Girvan and Newmans basato sul calcolo dell'Edge-Betweenness utilizzando un approccio Map-Reduce. La PPI network è una rete che rappresenta l'interazione tra proteine, essa contiene archi e nodi per descrivere il rapporto di relazione tra le proteine stesse; in questo specifico caso i nodi rappresentano le proteine e gli archi le relazioni tra esse.
L'utilizzo di un clustering è utile per ricercare i gruppi di proteine: proteine in uno stesso gruppo avranno delle relazioni più forti tra loro e proteine in gruppi differenti saranno invece collegate con una relazione più debole.

L'Algoritmo

L'approccio di clustering è basato sul concetto di edge-betweenness, ovvero una misura di centralità di un arco all'interno di un grafo.
Essa è calcolata come:

$$BC(e) = \sum_{s \neq t \in V} \delta_{s,t}(e)$$

dove $\delta_{s,t}(e)$ rappresenta il rapporto di due quantità, in cui il denominatore è il numero totale di percorsi più brevi che partono dal nodo s e arrivano al nodo t, indicato con $\sigma_{s,t}$, e il numeratore è il numero di tali percorsi che comprendono l'arco e, indicato come $\sigma_{s,t}(e)$.
Un arco con un'alta edge-betweenness indica che un elevato numero di percorsi più brevi tra due nodi passano attraverso di esso. Rimuovendo gli edge con la più alta betweenness in ordine decrescente, possiamo separare il grafico PPI in diversi sottografi. Tale sottografo conterrà proteine interconnesse che hanno forti relazioni funzionali.
Essendo il numero di proteine molto elevato l'algoritmo viene implementato in parallelo con l'utilizzo del calcolo distribuito. Il modello MapReduce contiene principalmente due fasi di elaborazione dei dati: Map e Reduce.
Durante la fase di Map, i dati di input vengono suddivisi in segmenti di dati più piccoli e distribuiti a più unità di elaborazione utilizzando una funzione Map. L'output intermedio prodotto dalla funzione Map è una collezione di tuple di coppie chiave-valore. Durante la fase di Reduce, gli output intermedi di ogni funzione Map sono trasferiti gli elaboratori che eseguono una funzione Reduce.
I dati formati del tipo chiave-valore sono ordinati usando le chiavi e aggregati usando la funzione Reduce. Il framework di supporto è Apache Spark che si basa sulla realizzazione di RDD (Resilient Distributed Dataset) e viene applicato attraverso il linguaggio di programmazione Java.

Struttura del Dataset

Per implementare l'algoritmo viene realizzata una struttura di input, nella quale ogni osservazione contiene sei informazioni di interesse:

- *NodeId*: ID del nodo;
- *Neighbors*: lista dei nodi vicini collegati direttamente al NodeId;

- *Root*:ID del nodo radice;
- *Path*:lista dei nodi che formano il percorso più breve dal NodeID al nodo Root;
- *Distance*:distanza del Path;
- *Color*: status del NodeID
  - WHITE: il nodo non è stato raggiunto;
  - GRAY: il nodo è stato raggiunto e sta per essere elaborato;
  - BLACK quando il nodo è stato raggiunto ed elaborato;

In particolare, il Root è stato scelto come nodo più centrale tra quelli a disposizione. Il Path è stato inizializzato a NULL, poiché non sono stati ancora definiti i percorsi più brevi tra i nodi e il Root, di conseguenza la distanza è inizializzata a MAX e il colore viene impostato a WHITE per tutti i nodi, in quanto nessuno di essi è stato inizialmente raggiunto.
Quanto descritto è valido per tutti i nodi, ad esclusione del nodo radice Root che si presenta con distanza 0 e colore GRAY.

Fasi dell'Algoritmo

Fase 1: Forward MR
Questa fase lavora esclusivamente sui nodi e ha come obbiettivo l'aggiornamento di Path, Distance e Color di ogni singolo nodo.

Map:
La fase di Map individua i nodi GRAY in quanto i nodi di interesse risulteranno i vicini dei nodi stessi. Innanzitutto, si individuano i NodeID dei vicini utilizzando un'apposita classe in Java denominata 'EstrapolaVicini' e, con una serie di operazioni intermedie, sono state recuperate le informazioni di interesse.
La funzione di map genera come output una serie di coppie chiave-valore. Per il nodo di lettura la chiave è NodeID e Root che rimangono invariati; il valore è composto invece da Path e Distance che rimangono anch'essi invariati e da Color che da GRAY passa a BLACK.
Per i nodi vicini che sono stati precedentemente estrapolati, la chiave sarà formata dal proprio NodeID e dal Root, mentre il valore sarà formato dal Path che contiene il NodeID del nodo di lettura, Distance+1 e Color che da WHITE passa temporaneamente a GRAY.
Tali cambiamenti vengono effettuati solo se il colore dei nodi non è di tipo BLACK, se Color è BLACK, infatti, non viene apportato alcun cambiamento.

Reduce:
La fase di Reduce prende in input l'output della fase di Map e procede con l'aggiornamento delle informazioni. In particolare, produce un output in cui ogni riga è formata da NodeId, Root, Neighbors, Distance, Color e Path, dove Distance è la distanza minima trovata con un'apposita classe 'DistanzaMinima', Color è l'ultimo status del nodo e Path è il percorso più breve.
Se il nodo non viene raggiunto, allora la distanza rimane MAX e il colore rimane WHITE.

Fase 2: Backward MR
Questo algoritmo prende in input l'output dell'algoritmo precedente e lavora sugli archi estratti dal Path di ciascun nodo con l'obbiettivo di calcolare l'Edge-Betweenness di ogni arco.

Map:
In questa fase, per ogni riga vengono estratti gli archi dal Path con una specifica classe 'EstraiArchi' generando una coppia chiave-valore, dove le chiavi sono gli archi all'interno del Path e ogni valore viene impostato pari a 1.

Reduce:
Nella fase di Reduce vengono inizialmente calcolati gli $\sigma_{s,t}$ e i $\sigma_{s,t}(e)$ con tre apposite classi denominate 'Estrai st', 'Sigma ste' e 'Sigma ste2'. Successivamente viene calcolato $\delta_{s,t}(e)$ sulla base della quale

viene determinata la Edge-Betweenness di ogni arco.

Fase 3:
Si individua il valore massimo dell'Edge-Betweenness e viene eliminato l'arco corrispondente ricreando il formato di input inizialmente utilizzato.
L'operazione di eliminazione degli archi con Edge-Betweenness massima presente nella fase 3 si arresta se quest'ultima risulta minore o uguale di 2 (valida per ogni possibile dataset di input) o minore o uguale del numero totale di nodi diviso per una costante pari a 20; questo avviene per garantire una dimensione minima adeguata del numero di nodi presenti in ogni gruppo tenendo in considerazione l'obbiettivo di clustering.
La prima condizione è posta indipendentemente dalla grandezza del dataset, al contrario, la seconda tiene conto del numero di nodi del dataset di input e mette, di conseguenza, in relazione l'Edge-Betweenness massima con l'entità del numero di nodi.

Fase 4: Condizione di arresto
Si implementano due livelli di arresto:

1. le operazioni previste della fase 1 si interrompono quando tutti i vicini del nodo di lettura con Color GRAY sono BLACK perché non sarà più possibile aggiornare i nodi.

2. per implementare l'algoritmo si sono utilizzati due cicli, uno annidato nell'altro: il primo ciclo annidato è stato realizzato per la prima fase, mentre il secondo più generale fa riferimento alle due fasi seguenti. Per motivi di efficienza dell'algoritmo è stato impostato un numero massimo di iterazioni per il secondo ciclo pari a 5, in modo tale da evitare che l'algoritmo cicli un numero eccessivo di volte.

Applicazione del Modello

Per costruire l'algoritmo si è utilizzato un dataset ridotto di sole 11 osservazioni creato direttamente con una struttura analoga a quanto spiegato precedentemente.
Il dataset finale su cui si svolge il clustering è preso invece dall'IMEx (International Molecular Exchange Consortium) e scaricato direttamente dal sito di repository: $http://dip.doe-mbi.ucla.edu/$.
Il dataset in analisi presenta 22600 osservazioni, ciascuna delle quali rappresenta un'interazione tra proteine; al fine dell'analisi questo è stato rielaborato per assumere la struttura in linea con l'applicazione del modello.
Per mostrare a livello grafico i risultati dell'algoritmo si è stabilita una connessione con il software Neo4j per visualizzare l'input iniziale e l'output finale del network di molecole. Questa operazione è stata riprodotta soltanto per il dataset ridotto in modo tale da avere una visione più chiara dei risultati. Si riporta il grafo di input del dataset ridotto nella Figura 1:

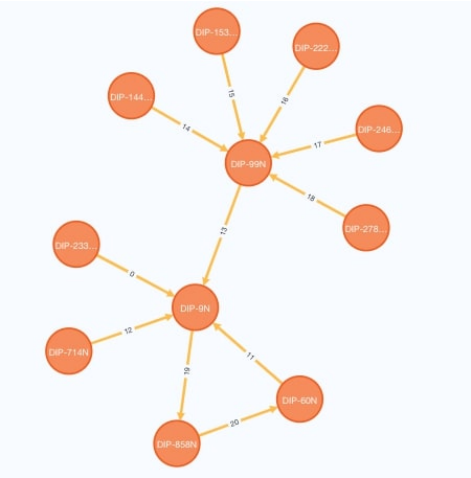

Figura 1: Grafo di input

Dopo l'applicazione dell'algoritmo, l'arco 13, avendo l'edge-betweenness più elevata, viene eliminato come mostrato nella Figura 2:

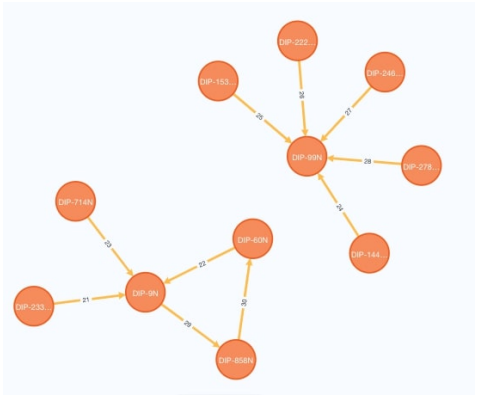

Figura 2: Grafo di output

Costo computazionele:

Costo computazionale: in termini di tempo l'algoritmo ha impiegato 30 minuti sul dataset finale con un processore di tipo Intel-Core i5 Dual-Core da 1,6 GHz con RAM da 8GB.