Report sull'Algoritmo di Clustering Bilanciato con Nodi Condivisi

1. Introduzione

Questo algoritmo suddivide n nodi in k cluster rispettando i seguenti vincoli:

- Ogni cluster deve contenere almeno `min_shared_nodes` nodi condivisi con altri cluster.
- Ogni cluster deve contenere almeno `min_exclusive_nodes` nodi esclusivi.
- Tutti i nodi devono essere assegnati a uno o più cluster.

Una caratteristica fondamentale dell'algoritmo è la capacità di gestire situazioni in cui un cluster non soddisfa il numero minimo di nodi condivisi (`min_shared_nodes`). In tali casi, l'algoritmo forza l'aggiunta di nodi condivisi al cluster interessato considerando i nodi non condivisi degli altri cluster, minimizzando la distanza dal centroide del cluster corrente. Inoltre, durante questo processo, viene garantito che ogni cluster mantenga almeno `min_exclusive_nodes` nodi esclusivi.

2. Fasi dell'Algoritmo

2.1 Selezione dei centroidi iniziali

Scopo: Selezionare k centroidi iniziali che siano il più lontani possibile tra loro.
Algoritmo:

1. Il primo centroide è scelto arbitrariamente.
2. Iterativamente, si seleziona il nodo più distante dai centroidi già scelti:

$$\texttt{next\_centroid} = \arg\max_i(\texttt{cached\_dists}[i]),$$

   dove `cached_dists` mantiene le distanze minime dai centroidi già scelti.

Complessità: $O(k \cdot n)$.

2.2 Popolamento bilanciato dei cluster

Scopo: Assegnare i nodi ai cluster in modo bilanciato, mantenendo una dimensione uniforme per ciascun cluster.
Metodo: L'algoritmo utilizza una strategia basata su heap per popolare i cluster in modo bilanciato. Ecco i passi principali:

1. **Inizializzazione degli heap:**
   - Viene costruito un `heap` (una coda con priorità) per ogni cluster.
   - Ogni heap contiene i nodi non assegnati, ordinati in base alla distanza dal centroide del cluster corrispondente.
   - La priorità nell'heap è determinata dalla distanza: il nodo con la distanza minima dal centroide è il primo ad essere estratto.
2. **Assegnazione dei nodi ai cluster:**
   - Per ogni cluster, viene estratto il nodo con la distanza minima dall'heap del cluster.
   - Il nodo viene aggiunto al cluster se non è già stato assegnato ad altri cluster.
   - Una volta assegnato un nodo a un cluster, questo nodo viene rimosso dagli heap degli altri cluster per garantire che non venga assegnato più volte.
3. **Controllo delle dimensioni del cluster:**
   - Il processo di assegnazione continua iterativamente finché ogni cluster raggiunge la dimensione desiderata (`cluster_size`).
   - Se un heap non ha più nodi da estrarre, il processo passa al cluster successivo.

Complessità: $O(k \cdot n \log n)$.

2.3 Ricerca nodi condivisi

Scopo: Garantire che ogni cluster abbia almeno `min_shared_nodes` nodi condivisi mantenendo il vincolo sui nodi esclusivi.
Metodo:

1. Si costruisce un `heap` per ogni cluster, contenente i nodi non condivisi di altri cluster, ordinati in base alla distanza dal centroide del cluster corrente.
2. Per ogni nodo candidato, si verifica che il cluster di origine abbia un numero sufficiente di nodi esclusivi ($>$ `min_exclusive_nodes`).
3. Si seleziona iterativamente il nodo più vicino dal `heap` e lo si aggiunge come nodo condiviso tra il cluster corrente e il cluster di origine del nodo.
4. Questo processo continua finché ogni cluster soddisfa il vincolo di `min_shared_nodes`.

Modifica introdotta: Durante l'assegnazione dei nodi condivisi, il controllo aggiuntivo garantisce che i cluster di origine mantengano sempre almeno `min_exclusive_nodes` nodi esclusivi. Se un cluster ha un numero di nodi esclusivi $\leq$ `min_exclusive_nodes`, i suoi nodi non possono essere selezionati come condivisi.
Complessità: $O(k \cdot n \log n)$.

2.4 Calcolo dei nodi esclusivi

Scopo: Identificare i nodi esclusivi di ciascun cluster.
Metodo: Per ogni nodo in un cluster, si verifica se non è presente nell'elenco dei nodi condivisi.
Complessità: $O(n)$.

2.5 Visualizzazione avanzata dei cluster

Scopo: Visualizzare i cluster con un'ulteriore evidenza grafica dei nodi condivisi.
Metodo:

- Ogni nodo condiviso è rappresentato con due colori, ognuno corrispondente a uno dei due cluster più vicini.
- Viene utilizzata la libreria `matplotlib` e la funzione `Wedge` per creare un effetto visivo a doppio colore.

—

3. Complessità Complessiva

La complessità totale dell'algoritmo aggiornato (senza considerare la generazione della matrice delle distanze) è:

$$O(k \cdot n \log n) \text{ (popolamento dei cluster)} + O(k \cdot n \log n) \text{ (forzatura dei nodi condivisi)}.$$

Pertanto, la complessità complessiva è:

$$O(k \cdot n \log n).$$

—

4. Conclusioni

L'algoritmo di clustering bilanciato aggiornato:

- Suddivide i nodi in cluster bilanciati rispettando i vincoli di nodi condivisi ed esclusivi.
- Durante la ricerca dei nodi condivisi, utilizza un approccio basato sulla minimizzazione delle distanze, selezionando i nodi più vicini al centroide del cluster corrente tramite heap.
- Garantisce che ogni cluster mantenga almeno `min_exclusive_nodes` nodi esclusivi anche dopo l'assegnazione dei nodi condivisi.
- Utilizza strutture dati efficienti (heap) e un processo iterativo ottimizzato per mantenere le prestazioni scalabili.
- Fornisce un output visivo avanzato con evidenza grafica dei nodi condivisi, permettendo una migliore interpretazione dei risultati.