Analisi dell’algoritmo di Karger per il calcolo di MINCUT

di Costantino Marco

# Introduzione

Per l’implementazione dell’algoritmo id Karger, il linguaggio scelto è *python*.

Per migliorare le prestazioni ho usato, al posto dell’interprete standard di python, *pypy* un compilatore just-in-time per codice python.

Non vi sono particolari originalità nell’implementazione, che è “traduzione” diretta dello pseudocodice visto a lezione.

Unica particolarità è l’uso di list comprehension per rimuovere lati dal grafo nella procedura contraction(G, v1, v2, next\_node)

La ragione per l’uso di list comprehension al posto di un ciclo for è il notevole miglioramento delle prestazioni temporali apportato.

## La struttura dati Graph

La struttura dati che ho usato per rappresentare i grafi è molto semplice e consiste in una lista dei vertici ed una lista dei lati del grafo.

# Domanda 1 – complessità temporale di Full Contraction

I dati raccolti rappresentano il tempo medio di esecuzione della procedura di full contraction per ogni grafo. I valori effettivi sono indicati sopra ogni punto, per riportare completamente i dati.

La complessità temporale dell’algoritmo non emerge bene dai dati rappresentati così, tuttavia è meglio visibile prendendo la media dei tempi raggruppati per numero di vertici del grafo:

In arancione un polinomio di grado 2 che interpola i dati. molto vicino ad ci indica che il modello spiega bene i dati. Il polinomio ci permette di dire oggettivamente che l’andamento delle misurazioni è consistente con l’analisi di complessità della procedura che individua come complessità temporale .

# Domanda 2 – complessità temporale dell’algoritmo di Karger

Per garantire una probabilità minore o uguale ad di sbagliare l’algoritmo di full contraction dev’essere eseguito volte. Moltiplicando la complessità temporale di full contraction per il numero di volte che dev’essere eseguito otteniamo la complessità temporale dell’algoritmo di Karger: . Di seguito riporto i dati direttamente come tempi medi dei raggruppamenti per numero di vertici del grafo:

In questo caso l’interpolazione ha senso solo sui dati raccolti prima del timeout. Il timeout è stato settato a 3 minuti, per garantire il raccoglimento dei dati in un massimo di ore. È difficile stabilire da così pochi dati se l’andamento atteso è rispettato. è dovuto al fatto che l’interpolazione incrocia tutti i punti dati, ma non è un buon indicatore con così pochi dati.

Per stimare i tempi di esecuzione completa, ho salvato per ogni grafo il numero di esecuzioni di full contraction eseguite al momento del timeout:

|  |  |  |
| --- | --- | --- |
| Vertici | k | FC al timeout |
| 100 | 23025 | 14514 |
| 100 | 23025 | 14981 |
| 100 | 23025 | 13866 |
| 100 | 23025 | 13878 |
| 125 | 37721 | 6700 |
| 125 | 37721 | 7184 |
| 125 | 37721 | 6659 |
| 125 | 37721 | 7313 |
| 150 | 56369 | 3958 |
| 150 | 56369 | 4110 |
| 150 | 56369 | 4116 |
| 150 | 56369 | 4027 |
| 175 | 79085 | 2575 |
| 175 | 79085 | 2601 |
| 175 | 79085 | 2338 |
| 175 | 79085 | 2799 |
| 200 | 105966 | 1614 |
| 200 | 105966 | 1795 |
| 200 | 105966 | 1709 |
| 200 | 105966 | 1624 |
| Vertices | k | Media FC al timeout |
| 100 | 20325 | 14309.75 |
| 125 | 37721 | 6964 |
| 150 | 56369 | 4052.75 |
| 175 | 79085 | 2578.25 |
| 200 | 105966 | 1685.5 |

La tabella riporta il numero di full contractions eseguite in media raggruppando per numero di vertici del grafo.

|  |  |
| --- | --- |
| Vertices | Tempo  Medio stimato |
| 100 | 4.261081 |
| 125 | 16.24971 |
| 150 | 41.72648 |
| 175 | 92.02172 |
| 200 | 188.6075 |

Per la stima dei tempi ho usato la seguente equazione:

Il risultato è espresso in secondi.

Utilizzando i tempi misurati (fin dove ottenuti) e quelli stimati, il grafico ottenuto è il seguente:

Il polinomio di quarto grado interpola bene i dati, tuttavia questo è da aspettarsi visto la flessibilità del polinomio di quarto grado. L’andamento dei tempi evidenzia comunque la complessità temporale dell’algoritmo che è piuttosto oneroso.

# Domanda 3 – discovery time

Il discovery time cresce molto più lentamente del tempo di esecuzione dell’algoritmo:

Per osservare meglio l’andamento del discovery time, escludo i primi grafi, che spesso hanno e riporto il resto dei dati in scala logaritmica:

Adesso è ben visibile la variabilità del discovery time, dovuta alla natura casuale dell’algoritmo. È anche visibile l’andamento deterministico di crescita, sebbene molto piccola, del discovery time al crescere della dimensione del grafo.

# Domanda 4 – l’errore