Analisi dell'algoritmo di Karger per il calcolo di MINCUT

di Costantino Marco

1 Introduzione

Per l'implementazione dell'algoritmo di 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.

Per permettere la raccolta di informazioni sulle performance dell'implementazione, ho deciso di effettuare i necessari timestamp nella procedura `karger` che necessita quindi di parametri aggiuntivi rispetto ad una semplice implementazione dell'algoritmo. In particolare, necessita del risultato atteso (per calcolare l'errore relativo) e di un valore per il timeout (interpretato come numero di secondi). Il timeout utilizzato per raccogliere i dati è stato di 5 minuti. Come ho spiegato più in dettaglio nell'apposita sezione per le esecuzioni che hanno raggiunto il timeout, ho cercato di stimare il tempo di esecuzione totale.

L'esecuzione del codice produce un file .csv contenente le informazioni necessarie alla scrittura della relazione.

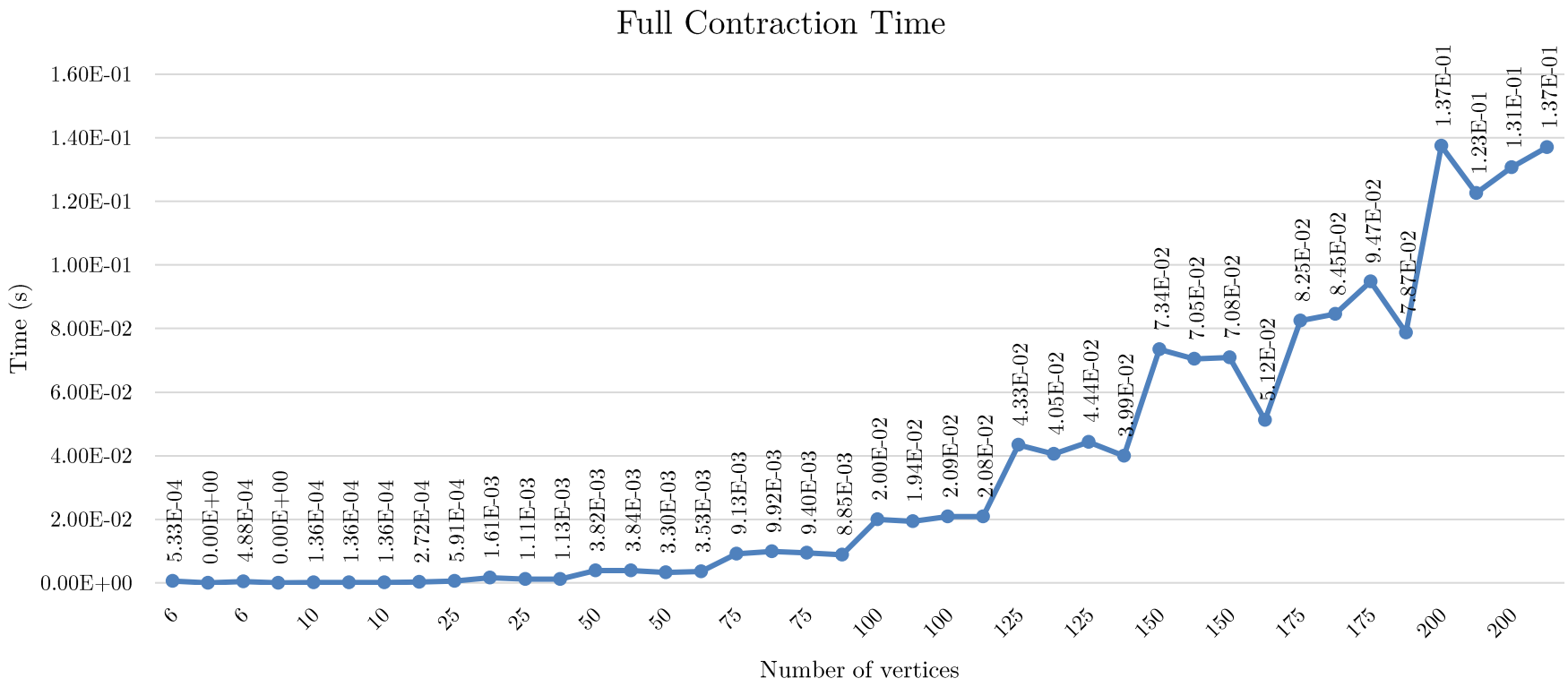
1.1 L'implementazione

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.

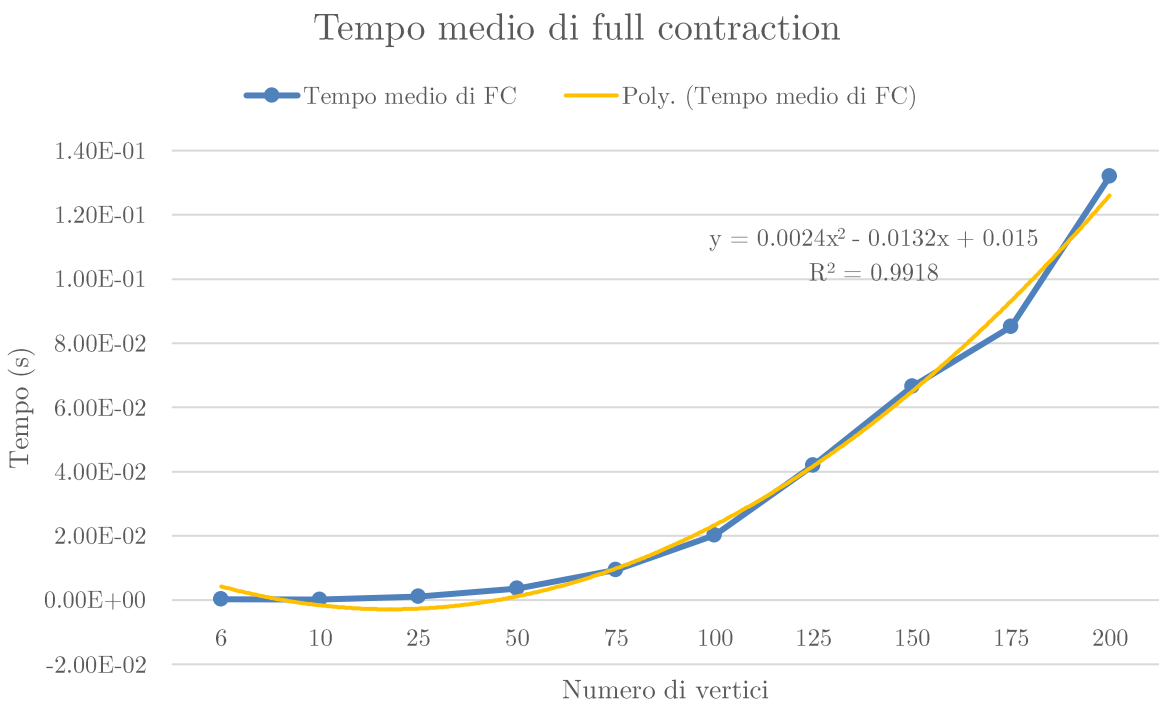
L'algoritmo è una semplice "traduzione" dallo pseudocodice visto a lezione. Unica particolarità è l'uso di una list comprehension per la cancellazione di lati dalla lista dei lati del grafo. A ragione dell'uso di una list comprehension al posto di un ciclo for, c'è l'aumento notevole delle prestazioni che ha apportato.

2 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.



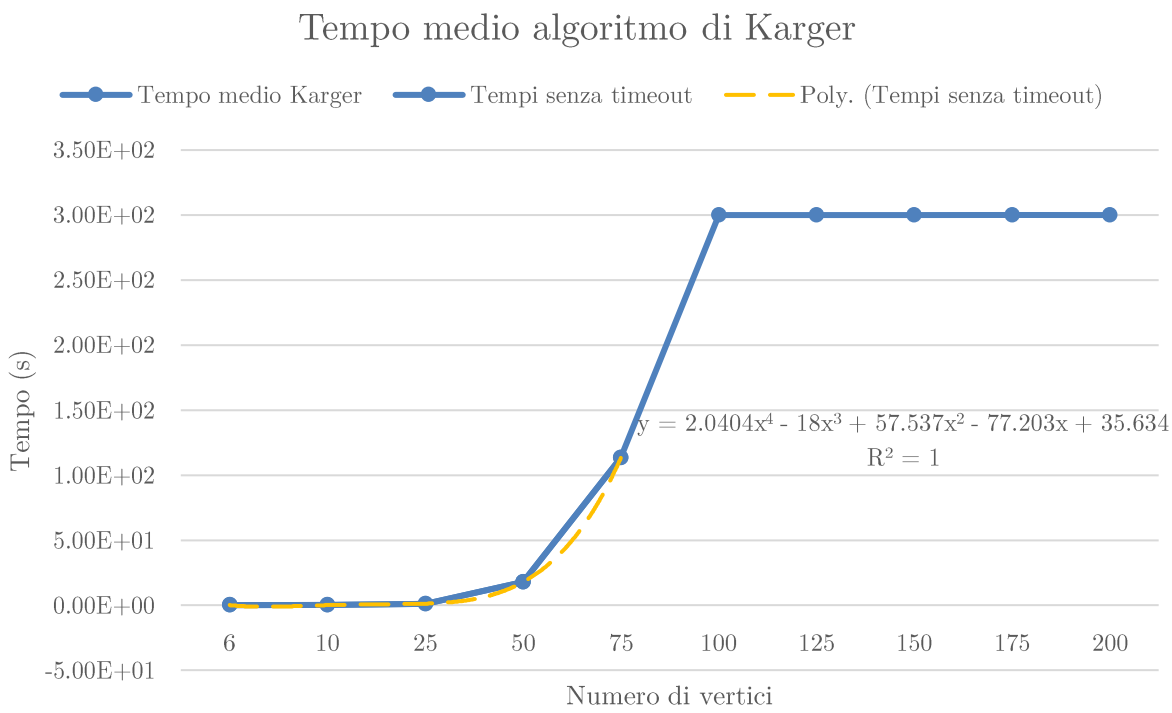
La complessità temporale dell'algoritmo non emerge bene dai dati rappresentati così, tuttavia è meglio visibile prendendo la media dei tempi raggruppati per numero di vertici del grafo:



In arancione un polinomio di grado 2 estratto dai dati. R^2 molto vicino ad 1 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 $O(n^2)$.

3 Domanda 2 – complessità temporale dell'algoritmo di Karger

Per garantire una probabilità minore o uguale ad $\frac{1}{n}$ di sbagliare, l'algoritmo di full contraction dev'essere eseguito $k = n^2/2 \ln n$ volte. Moltiplicando la complessità temporale di full contraction per il numero k di volte che dev'essere eseguito otteniamo la complessità temporale dell'algoritmo di Karger: $O(n^4 \ln n)$. Di seguito riporto i dati direttamente come tempi medi dei raggruppamenti per numero di vertici del grafo:



In questo caso ha senso estrarre un polinomio solo dai dati raccolti prima del timeout. Il timeout è stato settato a 5 minuti, per garantire il raccoglimento dei dati in un massimo (se accetto la premessa che ogni esecuzione va in timeout) di 3.3 ore. È difficile stabilire da così pochi dati se l'andamento atteso è rispettato. $R^2 = 1$ è dovuto al fatto che il polinomio incrocia tutti i punti dati, ma non è un buon indicatore con

così poche misurazioni. 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 a sinistra riporta il numero di contrazioni effettuate per ogni grafo che manda in timeout l'esecuzione.

La tabella in centro riporta invece il numero medio di contrazioni eseguite per numero di vertici del grafo.

L'ultima tabella riporta le stime di tempo necessario per l'esecuzione completa.

Vertices	Tempo Medio stimato
100	426.108
125	1624.97
150	4172.65
175	9202.17
200	18860.8

Per la stima dei tempi ho usato la seguente equazione:

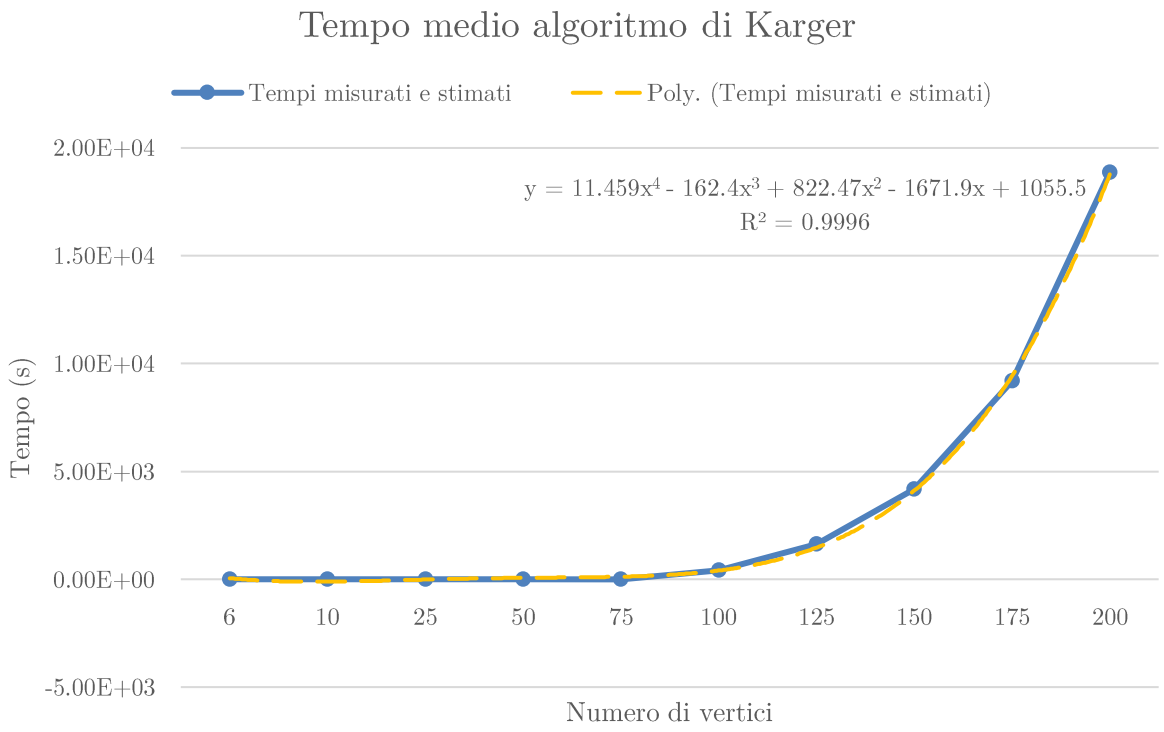
$$tempo\ stimato = \frac{300\ k}{media\ FC\ al\ timeout}$$

Che risulta dalla seguente proporzione:

$$300 : media\ FC\ al\ timeout = tempo\ stimato : k$$

Il risultato è espresso in secondi.

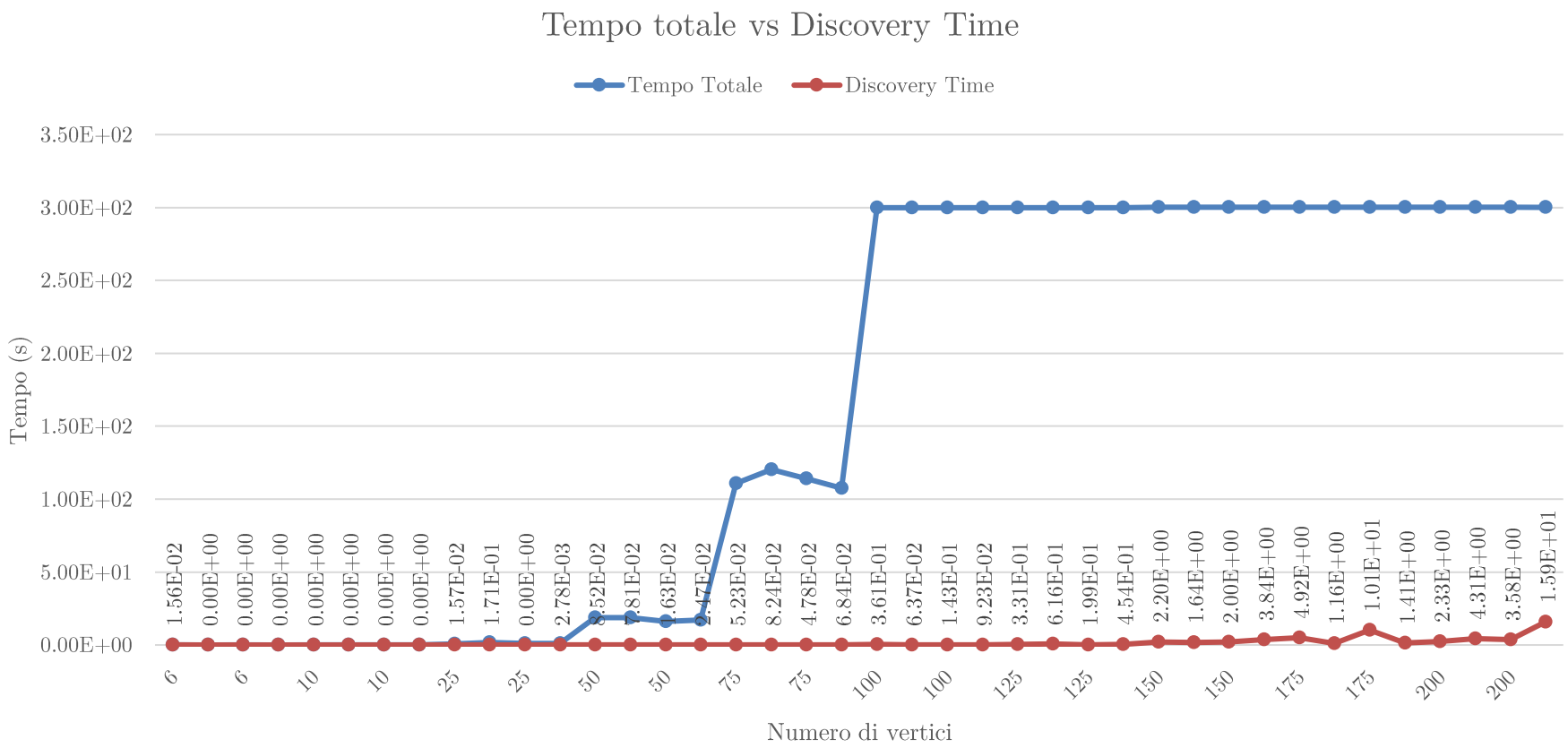
Utilizzando i tempi misurati (fin dove ottenuti) e quelli stimati, il grafico ottenuto è il seguente:



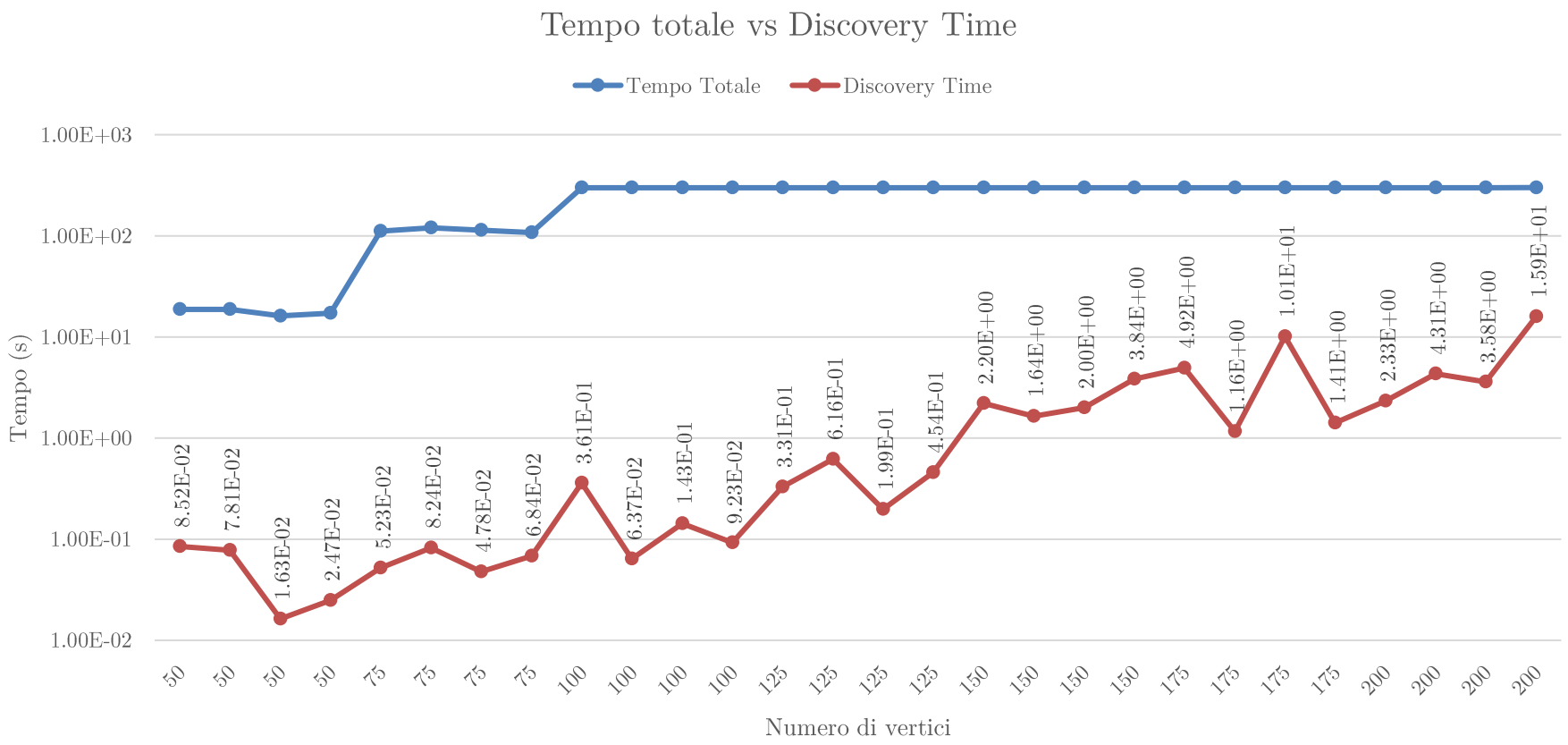
Il polinomio di quarto grado cattura bene i dati, tuttavia questo è da aspettarsi visto la flessibilità del polinomio di quarto grado. L'andamento dei tempi evidenzia comunque la complessità temporale dell'algoritmo che è piuttosto oneroso.

4 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 *discovery time* = 0 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.

5 Domanda 4 – l'errore

Per ognuno dei grafi analizzati, l'algoritmo ha trovato esattamente il risultato atteso. Per tanto l'errore è per ogni grafo 0. Per quanto detto in laboratorio, mi aspettavo di avere degli errori negativi (poichè le soluzioni attese non sono ottime) tuttavia ciò non è avvenuto. Al momento non so indicarne la ragione. Utilizzo questa sezione per riportare la totalità dei dati raccolti:

Indice	Vertici	k	Mincut	Media FC	Karger Time	Disc. time	Errore
1	6	32	2	5.33E-04	1.71E-02	1.56E-02	0
2	6	32	1	0.00E+00	0.00E+00	0.00E+00	0
3	6	32	3	4.88E-04	1.56E-02	0.00E+00	0
4	6	32	4	0.00E+00	0.00E+00	0.00E+00	0
5	10	115	4	1.36E-04	1.56E-02	0.00E+00	0
6	10	115	3	1.36E-04	1.56E-02	0.00E+00	0
7	10	115	2	1.36E-04	1.56E-02	0.00E+00	0
8	10	115	1	2.72E-04	3.12E-02	0.00E+00	0
9	25	1005	7	5.91E-04	5.94E-01	1.57E-02	0
10	25	1005	6	1.61E-03	1.61E+00	1.71E-01	0
11	25	1005	8	1.11E-03	1.13E+00	0.00E+00	0
12	25	1005	9	1.13E-03	1.14E+00	2.78E-03	0
13	50	4890	15	3.82E-03	1.87E+01	8.52E-02	0
14	50	4890	16	3.84E-03	1.88E+01	7.81E-02	0
15	50	4890	14	3.30E-03	1.62E+01	1.63E-02	0
16	50	4890	10	3.53E-03	1.73E+01	2.47E-02	0
17	75	12142	19	9.13E-03	1.11E+02	5.23E-02	0
18	75	12142	15	9.92E-03	1.20E+02	8.24E-02	0
19	75	12142	18	9.40E-03	1.14E+02	4.78E-02	0
20	75	12142	16	8.85E-03	1.08E+02	6.84E-02	0
21	100	23025	22	2.00E-02	3.00E+02	3.61E-01	0
22	100	23025	23	1.94E-02	3.00E+02	6.37E-02	0
23	100	23025	19	2.09E-02	3.00E+02	1.43E-01	0
24	100	23025	24	2.08E-02	3.00E+02	9.23E-02	0
25	125	37721	34	4.33E-02	3.00E+02	3.31E-01	0
26	125	37721	29	4.05E-02	3.00E+02	6.16E-01	0
27	125	37721	36	4.44E-02	3.00E+02	1.99E-01	0
28	125	37721	31	3.99E-02	3.00E+02	4.54E-01	0
29	150	56369	37	7.34E-02	3.00E+02	2.20E+00	0
30	150	56369	35	7.05E-02	3.00E+02	1.64E+00	0
31	150	56369	41	7.08E-02	3.00E+02	2.00E+00	0

32	150	56369	39	5.12E-02	3.00E+02	3.84E+00	0
33	175	79085	42	8.25E-02	3.00E+02	4.92E+00	0
34	175	79085	45	8.45E-02	3.00E+02	1.16E+00	0
35	175	79085	53	9.47E-02	3.00E+02	1.01E+01	0
36	175	79085	43	7.87E-02	3.00E+02	1.41E+00	0
37	200	105966	54	1.37E-01	3.00E+02	2.33E+00	0
38	200	105966	52	1.23E-01	3.00E+02	4.31E+00	0
39	200	105966	51	1.31E-01	3.00E+02	3.58E+00	0
40	200	105966	61	1.37E-01	3.00E+02	1.59E+01	0