Relazione MonteCarlo MinCut

Pezzano Enrico

Il programma implementato per questo compito sfrutta l'algoritmo MinCut di tipo Monte Carlo (chiamato tramite codice python)

---

for $i = n$ to 2

1. campiona un arco G con probabilità uniforme e identifica i suoi vertici, u e v, in un nuovo vertice uv (vedi Figura 3 con $u = a$ e $v = b$)
2. rimuovi tutti gli archi che univano i vertici u e v, incluso quello campionato, e mantieni tutti gli archi che incidono sul nuovo vertice uv
3. $i \leftarrow i - 1$

C è costituito dagli archi che uniscono gli ultimi due vertici rimasti di G

---

su un multigrade non orientato di 9 vertici e 21 archi.
In particolare il codice ha fatto 100 mila run, andando a calcolare una buona stima della frequenza empirica attesa, che si attesta a circa 0.22478, siccome si avvicina di molto a 2/n2, dove n=9 è il numero dei vertici.

Siccome MCMinCut è un algoritmo randomizzato di tipo Monte Carlo, non è sicuro che ad ogni esecuzione ci restituisca effettivamente il taglio minimo reale, ma eseguendolo un numero molto elevato di volte (105) siamo sicuri che il valore che comparirà più spesso sarà l'effettivo taglio minimo. Come riscontrato nei test eseguiti durante l'implementazione.

Di seguito uno screenshot dei risultati ottenuti:

```
[enrico@Macbook-Air-M1 2.1 MCMinCut % python3 MCMinCut.py

7       -->   191503
8       -->   226515
9       -->   102125
10      -->   63910
12      -->   18814
5       -->   30441
4       -->   22478
11      -->   33555
13      -->   9145
14      -->   1514

Il taglio minimo è  4
La frequenza empirica è  0.22478
Il numero di run per calcolare il taglio minimo con la probabilità del 99,9% è  28
```

Il taglio minimo naturale del grafo di Fritsch è 4, di conseguenza il risultato ottenuto è conforme alla teoria; per quanto riguarda la frequenza empirica pur essendo di un ordine superiore è in linea alla stima, confermando il risultato. Infine, il numero di run minimo per calcolare il taglio minimo essendone praticamente certi varia a seconda del codice o della macchina implementata: in questo caso, con python, si aggira intorno alle 20-30 esecuzioni, mentre su c++ si attesta intorno alle 40.