Algoritmi Avansați
Seminar 2

Gabriel Majeri

1 Introducere

În seminarul trecut am discutat principalele tipuri de probleme computaționale și am văzut că, pentru unele probleme, putem calcula în mod eficient soluția optimă. În acest seminar vom investiga câteva probleme dificile din punct de vedere computațional, pe care nu le putem rezolva în mod eficient și pentru care putem cel mult să obținem rapid o soluție **aproximativă** [1].

2 Exerciții

1. (Vertex cover [2]) Problema acoperirii unui graf ne cere să găsim o **submulțime de noduri din graf** astfel încât orice muchie din graf să fie învecinată cu (cel puțin) un nod din această mulțime. Problema de optimizare asociată implică găsimea mulțimii de cardinal **minim**.

În cele ce urmează, vom presupune că avem un graf cu n și m muchii, reprezentat prin liste de adiacență.

1. Fiind dată o mulțime de noduri dintr-un graf, cum ați **verifica** că mulțimea dată este o acoperire? Ce complexitate de timp are soluția propusă?

2. Propuneți o soluție de tip brute-force care să rezolve problema (i.e. să verifice toate submulțimile posibile de noduri). Ce complexitate de timp are soluția?

3. Să presupunem că alegeți o muchie aleatoare din graf, cu capetele u și v. Având în vedere că într-o soluție (optimă) a problemei *vertex cover* trebuie „atinse” toate muchiile, ce puteți spune despre nodurile u și v în raport cu mulțimea de noduri care constituie soluția (optimă)?

4. Ne gândim la următorul algoritm pentru a construi o soluție la problema acoperirii:

   (a) Inițial, acoperirea noastră este mulțimea vidă $\varnothing$.

   (b) Cât timp mai sunt muchii neacoperite în graf:

      i. Luăm (aleator) orice muchie din graf.

      ii. Includem capetele muchiei în mulțimea care va reprezenta acoperirea noastră.

      iii. Ștergem toate muchiile incidente la cele două noduri menționate.

   (c) Mulțimea de noduri construită este soluția la problema noastră.

   Acest algoritm construiește o soluție *aproximativă* la problema de optim pe care vrem să o rezolvăm; în unele cazuri, s-ar putea să fie posibil să construim o acoperire folosind mai puține noduri.

   Determinați factorul de aproximare al acestui algoritm, dați un exemplu când algoritmul obține soluția optimă și unul în care obține soluția cea mai proastă în raport cu soluția optimă.

2. (Maximum cut [3]) Fiind dat un graf neorientat (V, E), problema tăieturii maxime ne cere să găsim o **submulțime de noduri din graf**, astfel încât numărul de muchii care o leagă pe aceasta de complementul ei să fie maxim. Cu alte cuvinte, vrem două mulțimi de noduri A și B, cu $A \cup B = V$ și $A = V \setminus B$.

Această probleme are aplicații în fizica statistică și în proiectarea de circuite electronice [4].

Un algoritm aproximativ care rezolvă problema, descris în [5], este următorul:

1. Alegem aleator două noduri diferite din graf, v_1 și v_2.
2. Inițializăm $A \coloneqq \{ v_1 \}$, $B \coloneqq \{ v_2 \}$.
3. Cât timp mai sunt noduri care nu se află nici în A, nici în B:

   (a) Luăm aleator un astfel de nod.

   (b) Dacă are mai multe muchii care duc către A decât muchii care duc către B, îl punem în B; altfel, în A.

4. Soluția finală sunt mulțimiile A și B.

Determinați ce **complexitate de timp** și **de spațiu** are algoritmul propus, ce **factor de aproximare** oferă și dați un **exemplu** în care soluția aproximativă găsită de algoritm este cea mai îndepărtată de soluția optimă.

Referințe

[1] Wikipedia contributors, *Approximation algorithm*, URL: https://en.wikipedia.org/wiki/Approximation_algorithm.

[2] Wikipedia contributors, *Vertex cover*, URL: https://en.wikipedia.org/wiki/Vertex_cover.

[3] Wikipedia contributors, *Maximum cut*, URL: https://en.wikipedia.org/wiki/Maximum_cut.

[4] Francisco Barahona et al., „An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design", în *Operations Research* 36.3 (1988), pp. 493–513, URL: http://www.jstor.org/stable/170992.

[5] Vijay V. Vazirani, *Approximation algorithms*, Springer, 2001, ISBN: 978-3642084690.