Algoritmi Avansați
Seminar 3

Gabriel Majeri

1 Introducere

În acest seminar vom discuta câteva tipuri de algoritmi care nu se încadrează în tiparele obișnuite de rezolvare ale problemelor, dar care pot fi extrem de utili și versatili în unele situații.

2 Algoritmi probabiliști

O metodă de a rezolva mai eficient unele probleme dificile din punct de vedere computațional este să ne folosim de *șansă*. Astfel obținem algoritmi „probabiliști" / „randomizați" [1].

Un scurt istoric

În cadrul Proiectului Manhattan, cercetătorii Stanislaw Ulam și John von Neumann aveau nevoie să rezolve niște probleme foarte complicate legate de difuzia neutronilor. Așa că au venit cu ideea să modeleze pe calculator o simulare a reacțiilor care au loc în nucleul unei arme de fisiune nucleară, ca să înțeleagă ce se întâmplă fără să mai fie nevoie de un experiment fizic. Astfel a fost creată și folosită pentru prima dată *metoda Monte Carlo* [2, 3]: modelarea pe calculator a unei probleme teoretice pentru a o rezolva numeric.

Generarea numerelor aleatoare

Primul pas în implementarea unui algoritm probabilist este să avem acces la o sursă de numere aleatoare [4].

Calculatoarele digitale obișnuite sunt mașinării *deterministe*, care nu pot produce numere cu adevărat aleatoare. Un calculator poate folosi un **generator de numere pseudo-aleatoare** (PRNG [5]) pentru a genera un șir de

numere care aparent nu au nicio legătură unul cu celălalt, plecând de la un număr dat (denumit *seed*); dar dacă numărul inițial este dezvăluit, se pot determina toate numerele care urmează. Din acest motiv, dacă aveți nevoie să generați valori aleatoare într-un context sensibil (e.g. în criptografie), recomandarea este să utilizați un **generator de numerele pseudo-aleatoare sigur din punct de vedere criptografic** (CSPRNG [6]).

2.1 Algoritmi Monte Carlo

Un algoritm de tip Monte Carlo se execută pentru un anumit număr de pași și ne oferă o soluție care este doar *probabil* corectă; cu cât lăsăm algoritmul să ruleze mai mult, cu atât ne apropiem de soluția optimă sau o aproximăm mai bine [7].

Un exemplu de astfel de algoritm este un algoritm bazat pe programarea genetică; cu cât lăsăm să ruleze mai mult programul, cu atât soluțiile obținute sunt din ce în ce mai bune.

2.2 Algoritmi Las Vegas

Un algoritm de tip Las Vegas s-ar putea să nu obțină o soluție validă după o execuție, și să fie nevoie să-l rulăm de mai multe ori. Dar în momentul în care a obținut o soluție, aceasta este sigur cea corectă [8].

Un exemplu de algoritm de acest tip este cel care rezolvă problema așezării a 8 regine pe tabla de șah alegând aleator unde să pună o regină pe următorul rând; există riscul ca o soluție pe care a început să o construiască să nu fie corectă și să fie obligat să o ia de la capăt, dar în momentul în care a ajuns la final, soluția este sigur bună.

Vedere de ansamblu

Tabelul de mai jos compară cele două familii principale de algoritmi probabiliști, arătând cum una este „duala” celeilalte:

	Timp de execuție	Corectitudine rezultat
Monte Carlo	Determinist	Probabilistă
Las Vegas	Probabilist	Deterministă

Exerciții

1. Dacă aveți la dispoziție o funcție `random()` care generează un număr real în intervalul $[0, 1]$ (cu o distribuție uniformă), cum puteți obține un număr real în intervalul $[a, b]$ (cu o distribuție uniformă pe $[a, b]$)?

2. Dacă aveți la dispoziție o funcție `biased_random()` care generează aleator 0 sau 1, dar cu o probabilitate inegală p ($\neq 50\%$), puteți să definiți o funcție care să vă genereze 0 sau 1 cu probabilitate 50%–50%?

3. În acest exercițiu ne propunem să construim un algoritm probabilist care să ne ajute să determinăm valoarea lui π.

1. Care este aria cercului unitate?
2. Care este aria pătratului definit de punctele $(-1,-1)$, $(1,-1)$, $(1,1)$ și $(-1,1)$?
3. Dacă aleg aleator un punct în pătratul descris mai sus, care este probabilitatea ca el să fie în interiorul cercului unitate?
4. Dacă am un punct $(x,y) \in \mathbb{R}^2$, cum pot verifica dacă acesta se află în interiorul cercului unitate?
5. Avem următorul algoritm: luăm aleator perechi de numere din $[-1,1]$ (adică puncte din pătratul descris mai sus). Dacă punctul descris de această pereche de numere este în interiorul cercului unitate îl contorizăm, altfel îl ignorăm. La final, împărțim numărul de puncte care au căzut în interiorul cercului la numărul total de puncte luate în considerare. Ce valoare aproximează acest raport?
6. Ce fel de algoritm este cel descris anterior? Monte Carlo sau Las Vegas?

3 Programare liniară și algoritmul simplex

Unele dintre cele mai simple tipuri de probleme de optimizare (și cele mai des întâlnite în practică) sunt cele de *programare liniară*[1]. Din fericire, pentru acestea avem o interpretare vizuală destul de clară (cel puțin, în cazul 2D) și un algoritm eficient de rezolvare (metoda simplex).

Pentru a înțelege cum arată aceste probleme și cum le putem rezolva, vă recomand să vă uitați la [9], la capitolul 13 din [10] (notele de curs sunt disponibile aici), la această lecție [11] dintr-un curs de la MIT sau la acest video [12].

[1] În acest context, la fel ca în cazul „programării dinamice”, termenul de „programare” se referă la o metodă tabelară de rezolvare a unei probleme, nu neapărat la programare/dezvoltare software.

Referințe

[1] David Rydeheard, *Probabilistic (Randomized) algorithms*, URL: http://www.cs.man.ac.uk/~david/courses/advalgorithms/probabilistic.pdf.

[2] Roger Eckhardt, „Stan Ulam, John von Neumann, and the Monte Carlo method", în *Los Alamos Science* (15 1987), pp. 131–137, URL: https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-88-9068.

[3] Wikipedia contributors, *Monte Carlo method*, URL: https://en.wikipedia.org/wiki/Monte_Carlo_method.

[4] Wikipedia contributors, *Random number generation*, URL: https://en.wikipedia.org/wiki/Random_number_generation.

[5] Wikipedia contributors, *Pseudorandom number generator*, URL: https://en.wikipedia.org/wiki/Pseudorandom_number_generator.

[6] Wikipedia contributors, *Cryptographically-secure pseudorandom number generator*, URL: https://en.wikipedia.org/wiki/Cryptographically-secure_pseudorandom_number_generator.

[7] Wikipedia contributors, *Monte Carlo algorithm*, URL: https://en.wikipedia.org/wiki/Monte_Carlo_algorithm.

[8] Wikipedia contributors, *Las Vegas algorithm*, URL: https://en.wikipedia.org/wiki/Las_Vegas_algorithm.

[9] Trefor Bazett, *Intro to Linear Programming and the Simplex Method*, URL: https://www.youtube.com/watch?v=K7TL5NMlKIk.

[10] Georgia Tech, *Computability, Complexity & Algorithms*, URL: https://www.udacity.com/course/computability-complexity-algorithms--ud061.

[11] MIT OpenCourseWare, *15. Linear Programming: LP, reductions, Simplex*, URL: https://www.youtube.com/watch?v=WwMz2fJwUCg.

[12] The Organic Chemistry Tutor, *Linear Programming*, URL: https://www.youtube.com/watch?v=Bzzqx1F23a8.