DERANDOMIZARE

Cornelius Croitoru

Primul algoritm probabilist a fost publicat în Gazeta de Informatică (nr. 2/1997). Domnul profesor Horia Georgescu a "îndrăznit" să atragă atenția rezolvitorilor de probleme asupra acestor algoritmi tocmai din motivul că numerele generate aleator au fost "excluse" de la concursuri mult timp.

Introducere

Se spune că în anii '80 s-a descoperit "aruncarea banului" ca metodă de proiectare a algoritmilor eficienți. Ideea de bază este simplă: se construiește un algoritm probabilist pentru rezolvarea unei probleme, se analizează complexitatea acestuia, după care se face *derandomizarea* - se înlocuiesc generările de numere aleatoare cu apeluri ale unor proceduri de selecție, astfel încât să nu fie afectate concluziile asupra complexității și corectitudinii.

În această notă vom exemplifica metoda de mai sus, considerând unul dintre primii algoritmi de acest tip: ***algoritmul lui Luby*** *pentru determinarea unei mulțimi stabile maximale într-un graf.*

Dacă $G = (V, E)$ este un graf, o *mulțime stabilă* în G este o mulțime de vârfuri $S \subseteq V$ astfel încât $\forall\ v, w \in S$, $(v,w) \notin E$. O mulțime stabilă este *maximală* dacă nu este inclusă într-o altă mulțime stabilă. Altfel spus, $S \subseteq V$ este mulțime stabilă maximală dacă:

1. $\forall v,w \in S$: $(v,w) \notin E$ (S este stabilă);

2. $\forall v \in V\text{-}S$, $\exists w \in S$: $(v,w) \in E$ (maximală în raport cu incluziunea).

Considerăm următoarea problemă (***SM***):

Dat fiind graful $G = (V, E)$, să se determine o mulțime stabilă maximală.

Evident, dacă $|V| = n$ și $|E| = m$ atunci un algoritm secvențial de rezolvare a problemei ***SM*** cu complexitatea $O(n^2)$ este: se parcurge lista vârfurilor și se construiește mulțimea S (inițial vidă), testând la fiecare pas dacă vârful curent poate fi adăugat la S (pentru a forma o mulțime stabilă). Dacă G este reprezentat cu ajutorul listelor de adiacență, atunci o ordonare tip "*min-adjacency*" a mulțimii vârfurilor oferă o soluție de complexitate $O(n+m)$.

O întrebare naturală, care s-a pus la începutul anilor '80, a fost aceea dacă există un algoritm paralel eficient pentru rezolvarea problemei ***SM***. Mai precis, considerăm că dispunem de, un număr polinomial în n, procesoare și vrem să le punem să *coopereze* pentru construirea mulțimii stabile maximale în timp polinomial în $\log n$. Modelul de calculator paralel este mai puțin important pentru discuția care urmează, ceea ce urmărim este să punem în evidență dificultatea proiectării unui algoritm paralel.

Vom considera modelul ***PRAM*** (***P**arallel **R**andom **A**ccess **M**achines*): o colecție de procesoare (RAM-uri) independente, sincrone și cu acces la o memorie partajată. Deși nerealist, acest model este util în studiul asimptotic al complexității algoritmilor paraleli.

Clasa de probleme ***NC*** (***N**ick **C**lass*) este clasa problemelor pentru care există un algoritm ***PRAM***, care utilizând un număr polinomial de procesoare (în dimensiunea structurală a problemei) rezolvă problema în timp paralel polilogaritmic (polinom în logaritmul dimensiunii structurale).

Întrebarea asupra existenței unui algoritm eficient paralel pentru ***SM*** se poate formula acum simplu: ***SM*** $\in$ ***NC***?

Un prim răspuns pozitiv la această întrebare a fost dat de *Karp* și *Wigderson* [1]. O altă demonstrație pentru apartenența lui ***SM*** la ***NC*** a fost dată de *Luby* [2]. Soluția lui Luby reprezintă un prototip pentru metodele de derandomizare și merită a fi discutată.

Algoritmul PRAM probabilist al lui Luby pentru rezolvarea problemei *SM*

Input: $G = (V, E)$ graf cu $n = |V|$ vârfuri și $m = |E|$ muchii.
Output: $\Im \subseteq V$, $\Im$ mulțime stabilă maximală în G.

```
Begin
  ℑ:=∅
  while V<>∅ do
  begin
    S:=∅;
    forall v∈V in parallel do
    if d(v)=0
    then S:=S∪{v}
    else se introduce v în S cu probabilitatea 1/(2d(v));
```

```
    forall e∈E in parallel do
    if ambele extremități ale lui e sunt în S
    then elimină din S pe cea cu gradul mai mic (în caz
         de egalitate a gradelor, se elimină una din ele)
    I:=S; { I este o mulțime stabilă în graful curent.}
    ℑ:=ℑ∪I;
    G:=G-(I∪N(I)) { se șterg vârfurile din I∪N(I), }
{ muchiile care au cel puțin o extremitate în I sau N(I) 4 }
  end
End.
```

Este clar că mulțimea I construită într-o iterație **while** este o mulțime stabilă în graful curent G. Mulțimea finală $\Im$ este stabilă, deoarece în fiecare iterație se șterg vârfurile din $N(I)$.

Timpul paralel de executare al fiecărei iterații este $O(1)$. Evident, operațiile cu mulțimi, precum și construirea implicită a grafurilor curente se fac cu ajutorul unor instrucțiuni **forall** cu timp constant de executare.

Dacă se demonstrează că numărul mediu de iterații **while** este de $O(\log n)$ atunci rezultă că întreg algoritmul are timpul mediu paralel $O(\log n)$ (și deci, avem o demonstrație pentru ***SM*∈*RNC***). Nu avem nevoie de o demonstrație completă pentru această afirmație (deși ea se poate deduce cu ușurință), singurul fapt ce trebuie demonstrat este acela că la fiecare iterație numărul mediu de muchii ce se șterg din graful curent este cel puțin o fracție constantă din numărul de muchii ale acestuia. Atunci când vom face derandomizarea, vom obține aceeași proprietate pentru numărul de muchii șterse, și deci, în acest ultim caz, numărul iterațiilor va fi $O(\log n)$.

În cele ce urmează vom demonstra că în fiecare iterație se șterg în medie măcar 1/72 din numărul total de muchii ale grafului curent (cel cu care se începe iterația) $G=(V, E)$.

Lema 1. Pentru orice vârf $v \in V$, $\Pr(v \in I) \geq \frac{1}{4d(v)}$.

Demonstrație:

$$Pr(c\in I)=Pr(v\in I \mid v\in S)\cdot Pr(v\in S)=\frac{1}{2d(v)}\cdot Pr(v\in I \mid v\in S)=$$

$$=\frac{1}{2d(v)}.[I-Pr(v\notin I \mid v\notin S)]$$

Un vârf $v\in S$ nu va aparține lui I dacă $\exists u\in N(v)$ cu $d(u) \geq d(v)$ și $u\in S$.

Fie $L(v)=\{u \mid u\in N(v),\ d(u) \geq d(v)\}$

Atunci $Pr(v\notin I \mid v\in S) \leq Pr(\exists\ u\in L(v)\cap S \mid v\in S) \leq$

$$\leq \sum_{u\in L(v)} Pr(u\in S \mid v\in S)$$

"aruncări independente 2 câte 2"

$$\leq \sum_{u\in L(v)} Pr(u\in S)=\sum_{u\in L(v)}\frac{1}{2d(u)}\leq \sum_{u\in L(v)}\frac{1}{2d(v)}\leq \frac{d(v)}{2d(v)}=\frac{1}{2}$$

Deci:

$$Pr(v\in I)\frac{1}{2d(v)}\left[I-Pr(v\notin I \mid v\in S)\right]\geq \frac{1}{2d(v)}\left(1-\frac{1}{2}\right)=\frac{1}{2d(v)}$$

Definiție:

Un vârf $v\in V$ se numește *bun* dacă $\sum_{u\in N(v)}\frac{1}{2d(u)}\geq\frac{1}{36}$.

O muchie este *bună* dacă cel puțin unul din vârfurile pe care le unește este *bun*. Vârfurile sau muchiile care nu sunt bune se numesc *rele*.

Lema 2. Dacă v este un vârf bun atunci $Pr(v\in N(I))\geq\frac{1}{36}$.

Consecință:

Probabilitatea de a șterge o muchie bună este cel puțin 1/36 (este cel puțin egală cu probabilitatea ca una din extremitățile sale - care este vârf bun - să aparțină lui $N(I)$).

Demonstrație:

Fie v un vârf bun.

Cazul 1. $\exists u\in N(v)$ astfel încât $d(u)\in\{1,2\}$.

Atunci $Pr(v\in N(I))\geq Pr(u\in I)\geq \frac{1}{4d(u)}\geq\frac{1}{8}>\frac{1}{36}$

Cazul 2. $u\in N(v)$, $d(u)\geq 3$.

Atunci, cum v este bun,

$$\sum_{u\in N(r)}\frac{1}{2d(u)}\geq\frac{1}{6}\ \text{ și }\ \forall u\in N(v)\frac{1}{2d(u)}\leq\frac{1}{6}.$$

Există $M(v)\subseteq N(v)$ astfel încât $\frac{1}{6}\leq\sum_{u\in M(v)}\frac{1}{2d(u)}\leq\frac{1}{3}$.

$Pr(v\in N(I))\geq Pr(\exists u\in M(v)\cap I)\geq$

$$\geq \sum_{u\in M(v)} Pr(u\in I)-\sum_{u,w\in M(v)} Pr(u\in I \text{ și } w\in I)\geq$$

$$(lema1)\ \sum_{u\in M(v)}\frac{1}{4d(u)}-\sum_{\substack{u,w\in M(v)\\ u\neq v}} Pr(u\in S \text{ și } w\in S)=(\text{“independente 2 câte 2”})$$

$$\sum_{u\in M(v)}\frac{1}{4d(u)}-\sum_{u,w\in M(v)} Pr(u\in S)Pr(w\in S)=$$

$$\sum_{u\in M(v)}\frac{1}{4d(u)}-\sum_{\substack{u,w\in M(v)\\ u\neq v}}\frac{1}{2d(u)}\frac{1}{2d(w)}$$

$$\left(\sum_{u\in M(v)}\frac{1}{2d(u)}\right)\left[\frac{1}{2}-\sum_{w\in M(v)}\frac{1}{2d(w)}\right]\geq\frac{1}{6}\cdot\left(\frac{1}{2}-\frac{1}{3}\right)=\frac{1}{36}$$

Lema 3. *Cel puțin jumătate din muchii sunt bune.*

Demonstrație:

Orientăm muchiile grafului astfel: dacă $(u,v)\in E$ o orientăm de la u la v dacă și numai dacă $d_G(u)\leq d_G(v)$. În graful orientat obținut este valabilă următoarea proprietate: numărul arcelor care ies dintr-un vârf rău este cel puțin de două ori mai mare decât al celor care ies.

Fie v rău: $\sum_{u\in N(v)}\frac{1}{2d(u)}<\frac{1}{6}$.

După orientare, $N(v)=A\cup B$

$u\in A \Rightarrow uv$ intră în $v(d(u)\leq d(v))$

$u \in B \Rightarrow vu$ iese din $u(d(v) \le d(w))$

Trebuie să demonstrăm că $|B| > 2|A|$.

Presupunem că $|B| \le 2|A|$ și atunci $3|A| \ge d(v)$, iar

$$\sum_{u \in N(v)} \frac{1}{2d(u)} = \sum_{u \in A} \frac{1}{2d(u)} + \sum_{u \in B} \frac{1}{2d(u)} \ge \sum_{u \in A} \frac{1}{2d(u)} \ge$$

$$\ge \sum_{u \in A} \frac{1}{2d(v)} \ge \frac{|A|}{2d(v)} \ge \frac{d(v)}{3 \cdot 2d(v)} = \frac{1}{6}.$$

Am obținut că v este bun, contradicție.

Folosind proprietatea pusă în evidență putem asocia fiecărui arc rău (provenit din orientarea unei muchii rele) o mulțime de arce ce ies din extremitatea sa finală, asocierea fiind astfel încât la arce distincte rele corespund mulțimi disjuncte. Rezultă că există de două ori mai multe arce în total decât cele rele, deci numărul arcelor rele este cel mult jumătate din numărul total de arce, deci numărul muchiilor bune este măcar jumătate din numărul total de muchii.

Teoremă:

Numărul mediu de muchii care se șterg la fiecare iterație a algoritmului 1 este mai mare sau egal cu $\frac{|E|}{72}$ (*$|E|$ este numărul de muchii ale grafului curent*).

Demonstrație:

Fie X variabilă aleatoare reprezentând numărul muchiilor șterse într-o iterație. Pentru fiecare $e \in E$ considerăm X_e variabilă aleatoare

$$X_e = \begin{cases} 1, \text{dacă } e \text{ este stearsă} \\ 0, \text{altfel} \end{cases}$$

Atunci $X = \sum_{e \in E} X_e$ și media lui X este suma mediilor variabilelor aleatoare X_e și $MX = \sum_{e \in E} MX_e \ge \sum_{e \in E} MX_e =$

$$= \sum_{e \in E} Pr(e \text{ stearsă}) \underset{Consec.}{\ge} \sum_{\substack{e \in E \\ e \text{ bună}}} \frac{1}{36} \underset{lema3}{\ge} \frac{|E|}{72}$$

Observație:

Se poate demonstra că din această teoremă rezultă că numărul mediu de iterații necesare până se șterg toate muchiile (când algoritmul se termină) este $O(\log m)$ unde m este numărul de muchii ale grafului inițial (acesta este un exercițiu standard în teoria probabilităților discrete). Se obține deci că ***SM*** $\in$ ***RNC*** (un certificat al acestei apartenențe este algoritmul 1, pentru că $m = O(n^2)$).

Derandomizarea

În fiecare iterație a algoritmului 1 se execută n aruncări, independente două câte două, ale unor monezi "strâmbe".

(Pentru vârful $u \in V$ se consideră o monedă având Pr (de apariție a celeilalte fețe) $= 1 - \frac{1}{2d(n)}$).

Vom demonstra că aceste n aruncări se pot genera cu ajutorul unui șir de $O(\log n)$ biți aleatori. Fie p un număr prim astfel încât $n \le p < 2n$. Vârfurile grafului se vor eticheta cu elemente din Z_p. Pentru fiecare $u \in V$ se determină $0 < a_u < p$ astfel încât:

$$\left| \frac{a_u}{p} - \frac{1}{2d(u)} \right| = \min \left\{ \left| \frac{x}{p} - \frac{1}{2d(u)} \right| ; x \in Z_p \right\}$$

Nu se vor genera aruncări cu probabilitățile

$$\left(\frac{1}{2d(u)}, 1 - \frac{1}{2d(u)} \right) \text{ ci cu probabilitățile } \left(\frac{a_u}{p}, 1 - \frac{a_u}{p} \right).$$

Aceasta nu afectează concluzia din teoremă).

Pentru fiecare $u \in V$ se consideră $A_u \subset \{0,1,...,p-1\}$ cu $|A_u| = a_u$.

Se aleg uniform la întâmplare două elemente $x, y \in Z_p$; aceasta necesită $2p = O(\log n)$ biți aleatori. Pentru fiecare $u \in V$ se calculează $x + uy$ în Z_p.

Dacă $x + uy \in A_u$ se consideră că moneda lui u a căzut cu fața în sus și în algoritmul 1 se va introduce u în S.

Trebuie să demonstrăm că $Pr(x + uy \in A_u) = \frac{a_u}{p}$.

Să observăm că pentru orice $z, y \in Z_p$ există un unic $x \in Z_p$ astfel încât $x + uy = z$. Folosind această observație avem:

$$Pr(x + uy \in A_u) = \frac{1}{p^2} \cdot \left| \{(x,y); x + uy \in A_u\} \right| =$$

$$= \frac{1}{p^2} \sum_{z \in A_u} \left| \{(x,y) | x + uy = z\} \right| = \frac{1}{p^2} \cdot \sum_{z \in A_u} p = \frac{1}{p^2} \cdot |A_u| \cdot p = \frac{a_u}{p}.$$

De asemenea, trebuie să demonstrăm independența 2 câte 2 a aruncărilor, adică $\forall u \ne v$, $Pr(x + uy \in A_u \text{ și } x + vy \in A_v) = Pr(x + uy \in A_u) \cdot Pr(x + vy \in A_v)$.

Într-adevăr, $Pr(x + uy \in A_u \text{ și } x + vy \in A_v) =$

$$= \frac{1}{p^2} \left| \{(x,y) \mid x + uy \in A_u \text{ și } x + vy \in A_v\} \right| =$$

$$= \frac{1}{p^2} \sum_{z \in A_u} \sum_{w \in A_v} \left| \{(x,y) | x + uy = z \text{ si } x + wy = w\} \right|.$$

Să observăm că în corpul Z_p sistemul $\begin{cases} x + uy = z \\ x + vy = w \end{cases}$ are determinantul $v - u \ne 0$ și deci are soluție unică. Rezultă că se poate scrie în continuare $Pr(x + uy \in A_u \text{ și } x + vy \in A_v) =$

$$= \frac{1}{p^2} \sum_{z \in A_u} \sum_{w \in A_v} 1 = \frac{a_u \cdot a_v}{p^2} = \frac{a_u}{p} \cdot \frac{a_v}{p} = Pr(x + uy \in A_u) \cdot Pr(x + vy \in A_v)$$

Rezultă că dacă se dispune de șirul s de lungime $O(\log n)$ de biți aleatori, algoritmul de generare a celor n aruncări este:

Algoritmul GenA:

Input: *s* șir de lungime $O(\log n)$ de biți aleatori;
mulțimea $S \subseteq V$ a vârfurilor de grad 0.

Output: mulțimea *S* la care se adaugă vârfurile de grad nenul din *V* cu o probabilitate "apropiată" de $\frac{1}{2d(u)}$, independente unele de altele.

```
Begin
  Construiește x și y din s;
  Forall u∈V in parallel do
  begin
    Construiește a_u, A_u
    if x+uy∈A_u then S:=S∪{u}
End.
```

Numărul șirurilor de lugime $O(\log n)$ cu componente 0 și 1 este $2^{O(\log n)} = n^{O(1)}$. Fie *S* mulțimea acestor șiruri.

Rezultă că în loc să generăm un șir de lungime $O(\log n)$ de biți aleatori pentru care să aplicăm algoritmul **GenA** putem proceda astfel:

```
forall (s∈S) pardo
  aplică o iterație a algoritmului 1 folosind GenA pentru
  a face selecția aleatoare.
```

Fiecare aplicare paralelă va construi o mulțime *I(s)*.

Notăm cu *X(s)* numărul muchiilor care se șterg din graful curent prin îndepărtarea lui *I(s)*. *X'(s), s∈S* reprezintă valorile posibile ale variabilei aleatoare *X* reprezentând numărul de muchii șterse într-o iterație a algoritmului.

Cum $MX \geq \frac{|E|}{72}$ rezultă că $\exists s_0 \in S$ astfel încât:

$$X(s_0) \geq \frac{|E|}{72}.$$

Rezultă că dacă procedăm astfel:

```
forall s∈S pardo
  aplică o iterație a algoritmului 1 folosind GenA și
  determină I(s) și X(s); determină în paralel
```

$$\max_{s \in S} X(s) = X(s_0)$$

atunci $I(s_0)$ va fi o mulțime stabilă construită (în mod determinist) într-o iterație și are proprietatea că dacă se șterge $I(s_0) \cup N(I(s_0))$ din graful curent, atunci numărul muchiilor șterse este ≥ 1/72 din numărul de muchii ale acestui graf.

Descrierea completă a algoritmului determinist

Input: $G=(V,E)$ graf cu n =V vârfuri și m =E muchii
Output: $\Im \subseteq V$, $\Im$ mulțime stabilă maximală

```
Begin
  ℑ:=∅
  while V<>∅ do
  begin
    S:=∅;
    forall v∈V in parallel do
      if d(v)=0 then S:=S∪{u}; V:=V-{v}
      determină p număr prim |V| ≤ p ≤ 2|V|
    forall (x,y)∈{0,1,...,p-1}×{0,1,...,p-1} in parallel
    do
      begin
        S(x,y):=∅
        forall u∈V in parallel do
        begin
          determină a_u∈{1,...,p-1} a.î.
          |a_u/p - 1/2d(u)| = min{|x/p - 1/2d(u)|; x∈Z_p}
          construiește A_u.

          if x+uy∈A_u then S(x,y):=S(x,y)∪{u}
        end;
        forall e∈E in parallel do
          if ambele extremități ale lui e sunt în S(x,y)
          then elimină din S(x,y) extremitatea cu
               gradul mai mic;
        I(x,y):=S(x,y)
        determină G(x,y):=G-(I(x,y)∪N(I(x,y))∪S)
        determină m(x,y) numărul de muchii ale lui
        G(x,y)
      end;
    Determină (x_0,y_0) astfel încât:
      m(x_0,y_0)=min{m(x,y); x,y∈Z_p}
    I:=S∪I(x_0,y_0)
    ℑ:=ℑ∪I
    G:=G(x_0,y_0)
  end
End.
```

Numărul iterațiilor **while** este $O(\log n)$, fiecare iterație necesită $O(\log n)$ timp paralel. Rezultă că întreg algoritmul se execută în timpul paralel $O(\log^2 n)$. Numărul de procesoare utilizate este $O(n^2)$.

Bibliografie

1. **R.M. Karp, A. Wigderson** - *A Fast Parallel Algorithm for the Maximal Independent Set Problem*, Proc. 16th Symp. Theory of Computing, p. 266-272, ACM, 1984
2. **Luby** - *A Simple Parallel Algorithm for the Maximal Independent Set Problem*, Proc. 17th Symp. Theory of Computing, p. 1-10, ACM, 1985

Domnul profesor Cornelius Croitoru este cadru didactic la Universitatea "Al. I. Cuza" din Iași. Poate fi contactat prin e-mail la adresa:
croitoru@infoiasi.ro