Algoritmi probabilişti

Capitolul 14

- **Metode probabilistice**
- **Utilizarea nedeterminismului**
- **Noţiuni "avansate"**
- **Rezumat**
- **Implementări sugerate**

Numerele generate aleator reprezintă una dintre cele mai interesante facilităţi oferite de calculatoare. Chiar dacă numerele generate nu sunt complet aleatoare, repartizarea lor este suficient de "haotică" pentru a considera că nu există nici o regulă folosită pentru generarea lor. Există o mulţime de algoritmi care permit generarea de numere pseudoaleatoare, dar nu contează metoda folosită dacă se obţin secvenţe de numere cu perioada de repetitivitate suficient de mare.

14.1. Metode probabilistice

În foarte multe situaţii în care nu cunoaştem metoda de rezolvare a unei probleme, ne-am dori să putem ghici rezultatul. Datorită existenţei numerelor aleatoare, acest lucru este uneori posibil. În cele ce urmează vom prezenta câteva situaţii în care acestea sunt utilizate pentru a rezolva (cel puţin parţial) diferite probleme.

14.1.1. Soluţii "la întâmplare"

Cea mai simplă metodă este aceea de a încerca să ghicim soluţia generând-o aleatoriu. De exemplu, dacă ni se cere să determinăm dimensiunea maximă a unei clici într-un graf, ştim cu siguranţă că aceasta conţine cel puţin un nod şi cel mult N (numărul nodurilor grafului). Aşadar, putem genera un număr aleatoriu între 1 şi N şi "spera" că am ghicit rezultatul.

Putem îmbunătăţi această "soluţie" observând, că dacă graful conţine cel puţin o muchie, atunci dimensiunea clicii va fi cel puţin 2.

În plus, putem presupune că există mai multe şanse să avem o clică mai mică, deci putem genera soluția folosind această observație. Clica va fi cu atât mai mare cu cât vom avea mai multe muchii în graf.

De asemenea, putem calcula dimensiunea maximă a clicii pe baza numărului de muchii din graf. Pentru a putea avea o clică de dimensiune k, trebuie să existe cel puțin $k \cdot (k - 1) / 2$ muchii în graf. Aşadar, vom reduce din nou domeniul soluțiilor.

Evident, o astfel de abordare are şanse de reuşită foarte mici. Practic, dacă există S soluții posibile, şansa de a o ghici pe cea corectă este $1 / S$. Cu toate acestea, în cazuri extreme poate fi utilizată şi această metodă.

Metoda poate fi aplicată şi dacă datele de ieşire au o structură mai complicată. Evident, şansa de a găsi o soluție corectă în această situație este minimă.

14.1.2. Soluții verificate

O îmbunătățire a metodei anterioare constă în generarea unei soluții aleatoare şi verificarea acesteia. Aşa cum am arătat în capitolul 11, uneori este simplu să verificăm dacă o anumită soluție este sau nu corectă. În cazul în care aceasta este corectă, o furnizăm la ieşire, iar în caz contrar generăm alta.

De exemplu, pentru problema celor N dame putem genera permutări aleatoare ale mulțimii $\{1, 2, \ldots, N\}$ şi verifica apoi dacă permutarea generată reprezintă o soluție corectă.

14.1.3. Îmbunătățirea soluției

O variantă a abordării anterioare, care poate fi utilizată în cazul problemelor de optim, este generarea de soluții până în momentul în care expiră timpul de execuție alocat programului. După expirarea acestuia, vom genera cea mai bună soluție găsită până în acel moment şi vom "spera" că este şi cea corectă.

Această metodă va funcționa în cazul în care numărul soluțiilor posibile este relativ mic sau dacă există foarte multe soluții corecte.

14.2. Utilizarea nedeterminismului

În cadrul acestei secțiuni vom prezenta câteva posibilități de utilizare a numerelor aleatoare care permit găsirea soluției corecte cu şanse relativ mari. Practic, vor fi combinate metodele cunoscute cu metoda generării de numere aleatoare.

14.2.1. Alegerea continuării

În cadrul rezolvărilor bazate pe explorarea spațiului soluțiilor, există o mulțime de situații în care "nu ştim cum e mai bine să continuăm". Aşa cum am arătat în capitolul 13, este utilă folosirea funcțiilor euristice pentru a estima "calitatea" unei continuări. Totuşi, există unele situații în care astfel de funcții sunt greu de găsit sau de calculat,

motiv pentru care este mult mai comod să alegem continuarea la întâmplare. Aşadar, vom genera un număr cuprins între 1 şi numărul continuărilor posibile şi vom alege continuarea indicată de numărul generat.

Evident, dacă nu avem la dispoziţie o funcţie care aproximează costul transformării, o anumită continuare este la fel de bună ca oricare alta. În aceste condiţii s-ar putea spune că putem alege continuările în ordinea în care generăm configuraţiile corespunzătoare. Deşi această afirmaţie este adevărată din punct de vedere teoretic, practica arată că, de obicei, nu se obţin soluţiile dorite. De aceea, este mai indicat să alegem continuări folosindu-ne de nedeterminism.

14.2.2. Generarea primilor paşi

Metoda pereche celei prezentate anterior constă în generarea nedeterministă a primilor paşi ai soluţiei. În momentul în care ne apropiem de găsirea ei, putem aplica o altă metodă pentru a o găsi.

De exemplu, pentru problema celor *N* dame, dacă se cere o singură soluţie putem genera aleatoriu primele elemente ale permutării iar apoi, folosind metoda backtracking, să le generăm şi pe celelalte.

Evident, este de dorit ca primii paşi să fie valizi. Cu alte cuvinte, dacă amplasăm primele *k* dame, ar fi indicat să fim siguri că acestea sunt amplasate corect.

Metoda duce la obţinerea de rezultate surprinzătoare în cazul în care numărul soluţiilor posibile este relativ mare.

Practic, prima parte a algoritmului este o euristică în care folosim şi generarea de numere aleatoare, iar a doua parte este o metodă de explorare a spaţiului soluţiilor.

Evident, este posibil ca modul în care au fost generaţi primii paşi să nu permită găsirea soluţiei. În această situaţie, după ce am constatat că nu am găsit soluţia căutată, repetăm algoritmul: generăm din nou primii paşi şi apoi aplicăm din nou metoda backtracking sau o altă metodă de explorare a spaţiului soluţiilor.

14.3. Noţiuni "avansate"

Se pune în mod evident problema momentului în care ar trebui să ne bazăm pe şansă. Există mai multe situaţii în care cea mai bună continuare este clară şi deducem destul de uşor faptul că oricare alta are mult mai puţine şanse de reuşită. În această situaţie pare a nu fi recomandabil să lăsăm totul pe seama şansei.

Totuşi, cea mai bună continuare se poate dovedi, până la urmă, a fi o continuare care nu duce la soluţie. În acest caz s-ar părea că ar trebui să permitem norocului să determine programul să nu aleagă o astfel de continuare.

Argumentele pro şi contra pot continua practic la nesfârşit şi concluzia este destul de clară: nu există o "reţetă" care să indice când şi cum trebuie utilizat generatorul de numere aleatoare pentru rezolvarea unei probleme.

Abordarea "pro-deterministă" susține că vom folosi numerele aleatoare doar atunci când nu putem decide care continuare este mai bună. Aşadar, vom alege aleator o continuare doar atunci când avem mai multe continuări posibile şi toate sunt "la fel de bune".

Abordarea "pro-nedeterministă" susține că vom folosi numerele aleatoare în orice situație, eventual stabilind o probabilitate mai mare de alegere a continuărilor mai promițătoare.

Evident, va trebui să ajungem la un compromis... În principiu, va depinde de problemă. Totuşi, se pare că rezultatele sunt mai bune dacă se stabileşte o probabilitate de alegere a continuărilor.

14.5. Rezumat

În cadrul acestui capitol am prezentat pe scurt modul în care putem profita de şansă pentru a "rezolva" anumite probleme (cel puțin în anumite cazuri). După ce am descris metodele prin care putem încerca să ghicim soluția, am prezentat şi câteva modalități prin care nu ne lăsăm complet "pe mâna" norocului, ci să folosim distribuțiile aleatoare pentru a ne alege "drumul" spre găsirea soluțiilor.

14.6. Implementări sugerate

Pentru a vă obişnui să folosiți generarea numerelor aleatoare atunci când este cazul, vă sugerăm să implementați algoritmi pentru:

1. generarea aleatoare a unei permutări a mulțimii $\{1, 2, \ldots, N\}$; algoritmul trebuie să fie liniar;
2. alegerea unui număr cuprins între 1 şi 100; şansa de apariție a unui anumit număr va trebui să depindă de valoarea sa; un număr i va avea probabilitatea de apariție egală cu $0{,}019801980198... \cdot i$;
3. rezolvarea problemei celor N dame folosind generarea aleatoare de permutări;
4. rezolvarea problemei celor N dame alegând aleator dama care va fi amplasată la fiecare pas;
5. rezolvarea problemei celor N dame generând aleator primele elemente ale permutării şi aplicarea metodei backtracking pentru generarea celorlalte;
6. sortarea unui şir de numere utilizând generarea de numere aleatoare;
7. implementarea algoritmului de sortare rapidă alegând pivotul în mod aleator.