Drumuri MINIME când costurile sunt MICI

Mircea Digulescu, Andrei Matei

Problema determinării celui mai scurt drum de la un nod la altul într-un graf este una fundamentală în algoritmică. În continuare vom prezenta un algoritm al cărui timp de execuție depinde de costurile asociate muchiilor. Acesta este aplicabil atunci când costurile sunt mici și sunt numere întregi nenegative.

Notații

Considerăm un graf $G = (V, E)$ și o funcție de cost $w: E \to N$ (astfel încât $w(e) \geq 0$, $\forall\ e \in E$) care asociază un întreg nenegativ fiecărei muchii a grafului. Costul unui drum p în G este suma costurilor tuturor muchiilor din p. Fie nodul sursă s.

Definim $\delta(u,v) = \min(Cost(u \xrightarrow{p} v))$ ca fiind costul minim dintre toate căile p de la u la v, unde $u, v \in V$. De asemenea, fie $C = \max(w(e))$ costul celei mai "scumpe" muchii din graf și fie $C^* = \max(\delta(s, u))$ cel mai mare cost al unui drum minim în G care pleacă din s.

Pentru toți $v \in V$, $d[v]$ reprezintă *o estimare a costului unui drum minim* ($d[v] \geq \delta(s, v)$) pe tot parcursul execuției algoritmilor prezentați aici).

Obiectivul este găsirea costurilor $\delta(s, v)$ pentru orice $v \in V$. De asemenea, un drum p de cost minim de la s la v trebuie găsit, dar algoritmul folosit pentru determinarea $\delta(s, v)$ este simplu de modificat pentru a găsi un astfel de drum p.

Algoritmul prezentat va folosi liste dublu-înlănțuite. Pentru o listă dublu-înlănțuită l operația $v \Leftarrow l$ extrage primul element. După extracție, v va avea valoarea primului element din listă. Operația $l \Leftarrow v$ introduce un element v în lista l. Totodată, operația $l \mapsto v$ are ca efect ștergerea elementului v din lista l. Valoarea din v rămâne neschimbată în urma acestei operații.

Un algoritm de ordinul O(V+E+C*)

În continuare vom prezenta un algoritm care determină costul drumului minim atunci când muchiile au costuri întregi nenegative. El reprezintă, de fapt, o modificare a algoritmului lui *Dijsktra* în care este folosită o coadă de priorități neobișnuită.

Fie $l[0]$, $l[1]$, …, $l[C^*]$ și $l[\infty]$ liste dublu-înlănțuite de noduri din V.

În total avem $C^* + 2$ liste. Pe parcursul execuției algoritmului, fiecare nod va aparține unei singure liste. Pentru un nod v, $d[v]$ va fi o estimare a costului drumului minim la v. Când algoritmul se încheie, vom avea $d[v] = \delta(s, v)$ pentru toate nodurile $v \in V$. Algoritmul poate fi ușor modificat pentru a determina și un drum de cost minim și lăsăm acest lucru pe seama cititorului.

În continuare vom prezenta varianta în pseudocod a algoritmului:

```
 1: l[i] ← ∅, 0 ≤ i ≤ C*
 2: l[∞] ← ∅
 3: l[0] ⇐ s
 4: pentru v ∈ V execută
 5:    l[∞] ⇐ v
 6:    d[v] ← ∞
 7: sfârşit pentru
 8: d[s] ← 0
 9: p ← 0
10: cât timp p < C* execută
11:    cât timp l[p] ≠ ∅ execută
12:       v ⇐ l[p]
13:       pentru (v, u) cu d[u] > d[v] + w(v, u)
          execută
14:          l[d[u]] ↦ v
15:          d[u] ← d[v] + w(v, u)
16:          l[d[u] ⇐ u
17:       sfârşit pentru
18:    sfârşit cât timp
19: p ← p + 1
20: sfârşit cât timp
```

Algoritmul anterior este asemănător cu algoritmul lui *Dijsktra* prin faptul că procesează nodurile în ordine crescătoare a costului căii minime la ele. Liniile 1-8 realizează inițializarea valorilor l, S și d. Pe parcursul execuției algoritmului lista $l[i]$ va conține toate nodurile v pentru care $d[v] = i$. Algoritmul parcurge în ordine $l[0]$, $l[1]$, $l[2]$, … și la pasul p relaxează toate muchiile adiacente cu un nod din $l[p]$. Liniile 14 și 16 vor muta un nod u pentru care estimarea costului drumului minim până la el, $d[u]$, s-a schimbat (ca urmare a relaxării unei muchii) în lista corespunzătoare noii estimări (mai întâi este șters din lista veche și apoi inserat în lista corespunzătoare).

Corectitudinea algoritmului

Vom arata că atunci când algoritmul se încheie, vom avea $d[v] = \delta(s, v)$ pentru orice $v \in V$. Vom demonstra mai întâi câteva rezultate intermediare.

Lema 1: *Dacă un nod v este extras în linia 12, atunci toate nodurile u care sunt legate de v (direct sau indirect) prin muchii de cost 0 vor fi procesate la aceeași iterație (a ciclului care începe la linia 10), ca și v, și vor avea* $d[u] = d[v]$.

Demonstrația se realizează prin inducție după numărul de muchii de cost 0 dintre u și v. Toate nodurile adiacente cu v vor fi descoperite de codul din linia 13, vor avea $d[u] = d[v]$ datorită liniei 15 și vor fi adăugate în lista curentă la linia 16. Prin inducție, dacă nodurile aflate la distanța k de v au fost procesate, atunci nodurile aflate la distanța $k + 1$ de v vor fi descoperite de codul din linia 13 și procesate. Deci, toate nodurile legate de v prin muchii de cost 0 vor fi procesate și vor avea $d[u] = d[v]$.

Lema 2: *La sfârșitul fiecărei iterații a ciclului care începe la linia 10 toate nodurile v pentru care* $\delta(s, v) \leq p$ *au fost procesate*.

Demonstrația se realizează tot prin inducție. Primul nod extras este s. Din moment ce costurile muchiilor sunt nenegative, singurele noduri pentru care $\delta(s, v) \leq 0$ sunt cele pentru care $\delta(s, v) = 0$. Orice nod pentru care $\delta(s, v) = 0$ trebuie să fie conectat de sursă prin muchii de cost 0. Conform lemei 1, toate nodurile de acest tip vor fi procesate.

La pasul p, datorită ipotezei de inducție, toate nodurile x pentru care $\delta(s, v) < p$ au fost procesate. Fie v un nod astfel încât $\delta(s, v) = p$. Fie u predecesorul direct al lui v pe un drum de cost minim s:

$$\delta(s, v) = \delta(s, u) + w(u, v) \Rightarrow p = \delta(s, u) + w(u, v).$$

Dacă $w(u, v) > 0$, atunci $\delta(s, u) < p$, deci u a fost procesat la o iterație precedentă. Deoarece u a fost procesat atunci când muchia (u, v) a fost relaxată, nodul v a fost inserat în lista $l[p]$ (dacă nu era inserat deja în listă). Deoarece toate nodurile din $l[p]$ sunt procesate la pasul p, atunci nodul v va fi procesat.

Cazul în care $w(u, v) = 0$ nu prezintă interes, deoarece, conform lemei 1, toate nodurile legate prin muchii de cost 0 de nodul u sunt procesate la aceeași iterație ca și u. Deci, dacă am dovedit că toate nodurile pentru care $\delta(s, v) = p$ și care sunt legate de predecesorii direcți prin muchii de cost pozitiv sunt procesați la pasul p, atunci toate nodurile pentru care $\delta(s, v) = p$ sunt procesate la pasul p.

Teoremă: *Atunci când un nod este extras în linia* 12 *el va avea* $d[v] = \delta(s, v)$.

Demonstrația se realizează tot prin inducție. Primul nod extras este s și va avea $d[s] = \delta(s, s) = 0$.

Fie v un nod arbitrar extras din $l[p]$ cu operația de la linia 12. Prin inducție avem calculate costurile drumurilor minime pentru toate nodurile care au fost extrase înaintea lui. Datorită lemei 2 toate nodurile pentru care costul drumului minim este mai mic decât p au fost procesate. Astfel avem:

$d[u] = \delta(s, u)$, pentru toți u astfel încât $\delta(s, u) < p$.

Fie un drum de cost minim de la s la v: $s \xrightarrow{t} u \rightarrow v$. Vom avea $\delta(s, v) = \delta(s, u) + w(u, v)$.

Presupunem, fără a reduce generalitatea, că $w(u, v) > 0$ (deoarece, conform lemei 1, toate nodurile care sunt conectate prin muchii de cost 0 de un nod x vor avea în cele din urmă $d[u] = d[x]$ care va fi $\delta(s, u)$ dacă $d[x] = \delta(s, x)$). Prin urmare: $\delta(s, u) < \delta(s, v) \Rightarrow \delta(s, u) < p$.

Conform lemei 2, toate nodurile de acest tip au fost procesate. Fie u primul asemenea nod procesat de algoritm. Prin ipoteza de inducție avem $d[u] = \delta(s, u)$. Atunci când muchia (u, v) este relaxată avem $d[v] = \delta(s, u) + w(u, v)$ și, datorită alegerii făcute, pentru u (ca predecesor al lui v pe un drum de cost minim), vom avea $d[v] = \delta(s, v)$.

În sfârșit, deoarece fiecare nod aparține unei liste (prin inițializare), și toate listele sunt procesate, toate nodurile vor fi procesate. Astfel, conform teoremei 1, vom avea $d[v] = \delta(s, v)$ pentru toți $v \in V$ când algoritmul se încheie.

Analiza complexității

Linia 1 se execută într-un timp de ordinul $O(C^*)$; ordinul de complexitate al operațiilor efectuate în liniile 4-7 este $O(V)$, iar cel al operațiilor efectuate în liniile 2, 3, 8 și 9 este $O(1)$.

Ciclul care începe la linia 10 necesită un timp $O(C^*)$, dacă nu luăm în considerare ciclul care începe la linia 11.

Se observă că dacă un nod este extras în linia 12 (și procesat) el nu va mai fi procesat ulterior. Prin urmare, timpul total consumat de execuțiile liniei 12 este $O(V)$, deoarece fiecare nod este extras exact o dată. Așadar, timpul necesar pentru toate execuțiile ciclului care începe la linia 11 este $O(V)$, dacă facem abstracție de timpul necesar execuției ciclului care începe la linia 13.

Se observă că fiecare muchie (v, u) a grafului este luată în considerare exact o dată în cadrul ciclului care începe la linia 13 (sau de două ori dacă graful este neorientat), mai exact atunci când nodul v este procesat (dacă graful este neorientat este considerată și atunci când este procesat nodul u; condiția $d[u] > d[v] + w(v, u)$ va fi adevărată cel mult o dată: la prima procesare a unui nod adiacent acesteia). Operațiile de inserare și eliminare pentru liste dublu-înlănțuite pot fi executate în timp constant; așadar ordinul de complexitate al tuturor execuțiilor acestui ciclu este $O(E)$.

În concluzie, ordinul de complexitate al algoritmului este $O(C^*) + O(V) + O(C^*) + (O(V) + O(E)) = \boldsymbol{O(V + E + C^*)}$.

Datorită faptului că $C^* < V \cdot C$ ordinul de complexitate este cu siguranță mai mic decât $\boldsymbol{O(V \cdot C + E)}$.

Bibliografie

1. T. H. Cormen, C. E. Leiserson, R. R. Rivest, *Introducere în algoritmi*, Computer Libris Agora, 2000
2. Ahuja, Mehlhorn, Orlin, Tarjan, *Faster algorithms for the shortest path problem*, Tehnical Report 193, MIT Operations Research Center, 1998
3. Gabow, Tarjan, *Faster scaling algorithms for network problems*, SIAM Journal on Computing, 1989

Mircea Digulescu și Andrei Matei sunt elevi în clasa a XI-a la Colegiul Național de Informatică Tudor Vianu din București și pot fi contactați prin e-mail la mircea85@yahoo.com, *respectiv* andreimatei@home.ro.