Algoritmi de ORDONANȚARE

Ilie Vieru

Ordonanțarea este o problemă de optimizare specifică sistemelor dinamice în care apar o serie de dereglări care determină un comportament necorespunzător al sistemului față de obiectivul acestuia. În cele ce urmează vom prezenta trei probleme de ordonanțare împreună cu algoritmii de rezolvare.

În prima problemă de ordonanțare tratată în acest articol se folosește *diagrama Gantt* și se pune în evidență algoritmul corespunzător. Apoi vom rezolva problema minimizării timpului total pentru prelucrarea a n piese pe două mașini (algoritmul lui *Johnson*). În ultima problemă se minimizează suma penalizărilor pentru neîncadrarea în grafic (algoritmul lui *Moore*).

Ordonanțarea este o noțiune întâlnită în diverse domenii de activitate, cu precădere în procesele de organizare și planificare a fluxului tehnologic (operațiilor) dintr-o unitate de producție, de planificare și corelare a activităților la o societate de construcții-montaj sau în oricare altă unitate în care se dorește organizarea muncii, disciplină, încadrare în termene, îmbunătățire a performanțelor etc.

Prin ordonanțare se urmărește programarea activităților critice (care nu pot fi întârziate și nici devansate), precum și determinarea rezervelor de timp pentru activitățile mai puțin critice.

Folosind teoria grafurilor, aceste probleme se rezolvă, în principiu, cu ajutorul algoritmul *Bellman-Kalaba*. Pentru procese mai puțin complexe, pentru care ordinea de desfășurare a activităților poate fi reprezentată și urmărită într-un grafic de timp care se mai numește și ***diagrama Gantt***, se pot utiliza algoritmi elementari, dar foarte eficienți (care au ordinul de complexitate $O(n \cdot \ln n)$).

În cele ce urmează vom identifica algoritmii pentru trei probleme de ordonanțare.

Problema 1

Într-un atelier se execută n piese la două mașini M_1 și M_2; fiecare piesă se prelucrează mai întâi pe prima mașina M_1, apoi pe mașina M_2. O mașină nu poate prelucra simultan mai multe piese. Timpii necesari prelucrării pieselor pentru fiecare mașină sunt specificați într-un tabel de forma:

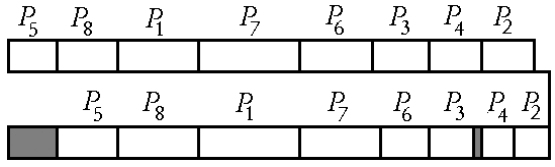
	P_1	P_2	P_3	…	P_n
M_1	a_1	a_2	a_3	…	a_n
M_2	b_1	b_2	b_3	…	b_n

unde a_i reprezintă timpul necesar pentru a realiza operația de prelucrare a piesei i pe mașina M_1, iar b_i reprezintă timpul necesar pentru a realiza operația de prelucrare a piesei i pe mașina M_2, $i \in \{1, 2, \ldots, n\}$.

Presupunând că piesele se prelucrează în această ordine, să se determine timpul total de lucru pentru fiecare mașină, precum și timpul de așteptare pentru mașina M_2.

Rezolvare

Vom începe descrierea rezolvării acestei probleme prezentând *diagrama Gantt*:



pentru exemplul:

P_5	P_8	P_1	P_7	P_6	P_3	P_4	P_2
12	15	20	25	18	14	12	10
15	20	25	20	12	11	10	8

Observăm că timpul total de lucru pentru mașina M_1 nu conține timpi de așteptare, deci toate operațiile sunt cri-

tice; mașina M_2 așteaptă 12 unități de timp, necesare și suficiente pentru terminarea prelucrării primei piese pe mașina M_1, iar după realizarea operației pentru piesa P_3, mașina M_2 rămâne din nou fără piese pentru lucru și trebuie să mai aștepte o unitate de timp, deci $T_S = 13$. Notăm cu t_1 și t_2 timpul necesar prelucrării pieselor pe mașina M_1, respectiv mașina M_2, iar cu t_3 timpul de așteptare (stagnare) pentru mașina M_2.

Cu aceste notații avem următorul algoritm:

```
t1 ← a1                                          (1)
t2 ← a1
i ← 1
j ← 0
repetă
   dacă t1 = t2 atunci                           (2)
   i ← i + 1
   j ← j + 1
   t1 ← t1 + ai
   t2 ← t2 + bj
   altfel
      dacă t1 < t2 atunci                        (3)
         i ← i + 1
         t1 ← t1 + ai
      altfel
         dacă j < i - 1 atunci                   (4)
            j ← j + 1
            t2 ← t2 + bj
         altfel                                  (5)
            t3 ← t3 + t1 - t2
            t2 ← t1
         sfârşit dacă
      sfârşit dacă
   sfârşit dacă
până când i = n                                  (6)
```

Observații:

(1) La început M_1 lucrează la prima piesă, iar M_2 așteaptă terminarea acestei operații.

(2) Dacă cele două mașini termină simultan prelucrarea piesei i, respectiv j ($i \neq j$), ambele trec la prelucrarea pieselor următoare.

(3) Dacă M_1 termină prelucrarea piesei i, iar M_2 lucrează încă la piesa j, atunci M_1 trece la prelucrarea piesei următoare.

(4) Dacă M_2 termină piesa j și mai are piese (prelucrate de M_1) trece la prelucrarea piesei următoare.

(5) Dacă M_2 nu mai are piese, așteaptă să termine M_1 piesa i la care lucrează.

(6) În acest moment M_1 a terminat lucrul, iar M_2 mai are de prelucrat piesele: $j + 1$, $j + 2$, …, n.

Problema 2

În condițiile problemei 1, să se ordonanțeze operațiile de prelucrare a pieselor pe mașinile M_1 și M_2, astfel încât timpul total de lucru pe mașina M_2 să fie minim.

Rezolvare

Această problemă se rezolvă cu ajutorul algoritmului lui *Johnson* care conține următorii trei pași:

- alegem valoarea minimă dintre vectorii a și b care presupunem că se găsește în poziția s;
- dacă minimul este a_s, vom începe prelucrarea pe mașina M_1 cu piesa având numărul de ordine s, în caz contrar, vom termina lucrarea cu piesa de pe poziția s;
- eliminăm a_s și b_s și repetăm acești trei pași până la construirea succesiunii optime de prelucrare a celor n piese.

Dacă notăm cu l_s și l_d capetele curente ale intervalului de sortat și $r = 1$ sau $r = 2$, după cum minimul s-a găsit în șirul a, respectiv b, algoritmul se poate reprezenta astfel:

```
ls ← 1
ld ← n
pentru k = 1, n - 1 execută:
   dacă a_ls <b_ls atunci
      min ← a_ls
      r ← 1
   altfel
      min ← b_ls
      r ← 2
   sfârşit dacă
   pmin ← ls
   pentru i = ls + 1, ld execută:
      dacă ai < min atunci
         min ← ai
         prim = i
         r = 1
      sfârşit dacă
      dacă bi < min atunci
         min = bi
         prim = i
         r = 2
      sfârşit dacă
   sfârşit pentru
   dacă r = 1 atunci
      aux ← a_pmin
      a_pmin ← a_ls
      a_ls ← aux
      aux ← b_pmin
      b_pmin ← b_ls
      b_ls ← aux
      ls ← ls + 1
   altfel
      aux ← a_pmin
      a_pmin ← a_ld
      a_ld ← aux
      aux ← b_pmin
      b_pmin ← b_ld
      b_ld ← aux
      ld ← ld - 1
   sfârşit dacă
sfârşit pentru
```

Cunoscând ordinea de prelucrare, putem aplica algoritmul de la problema anterioară și vom obține rezultatul căutat. Exemplul prezentat la problema 1 reprezintă ordinea optimizată a lucrărilor pe cele două mașini.

Problema 3

Să presupunem că o mașină *M* trebuie să prelucreze *n* piese. Pentru fiecare piesă se cunosc: timpul de execuție a_j, timpul limită de predare b_j (raportat la momentul de începere a lucrului $t = 0$), precum și penalizarea c_j în cazul în care predarea nu se face la termen, toate fiind numere reale strict pozitive. Să se determine o ordine de prelucrare a pieselor care să minimizeze suma penalizărilor corespunzătoare pieselor realizate după termenele limită stabilite.

Observație: Dacă toate valorile a_j și c_j sunt numere întregi, iar valorile b_j sunt egale, problema este *NP- completă* [1]. Totuși, în cazul în care toate valorile c_j sunt egale, *J. M. Moore* a propus un algoritm cu timp de lucru polinomial. Acest algoritm se poate extinde la un caz puțin mai general, și anume pentru oricare două piese *i* și *j* să se verifice condiția: $a_i < a_j \Rightarrow c_i \geq c_j$, adică, penalizările sunt considerate invers proporționale cu timpii de execuție.

Rezolvare

În rezolvarea acestei probleme se pornește de la observația simplă că întotdeauna există o succesiune optimă pentru care toate lucrările neîntârziate se execută primele, în ordinea crescătoare a timpilor limită, urmată de cele întârziate care se pot executa într-o ordine arbitrară. Va trebui, deci, să determinăm un algoritm pentru selectarea optimă a lucrărilor (pieselor) care se vor realiza fără întârzieri. Conform [1] se introduce relația de ordine totală pe mulțimea lucrărilor:

$$i \prec j \Leftrightarrow \begin{cases} a_i > a_j, \text{ sau} \\ a_i = a_j \text{ și } c_i < c_j, \text{ sau} \\ a_i = a_j, c_i < c_j \text{ și } i < j \end{cases}.$$

Algoritmul folosit constă din următorii doi pași:

- se sortează lucrările în ordinea crescătoare a timpilor limită *b*;
- se construiesc recurent mulțimile lucrărilor *j*-optime astfel:

$$S_1 = \{1\}$$

$$S_{j+1} = \begin{cases} S_j \cup \{j+1\}, \text{ dacă } p(S_j) + a_{j+1} \leq b_{j+1}, \\ (S_j \cup \{j+1\}) - \{k\}, \text{ în caz contrar} \end{cases}$$

unde *k* este elementul minim relativ la relația $\prec$ din mulțimea $S_j \cup \{j+1\}$, iar $p(S)$ reprezintă suma timpilor de efectuare a lucrărilor din *S*. Implementarea relației de ordine totală "$\prec$" pe care o vom nota *ordn*(*i*, *j*) presupune:

```
dacă a_si > a_sj atunci
   returnează i
sfârşit dacă
dacă a_si < a_sj atunci
   returnează j
sfârşit dacă
dacă c_si < c_sj atunci
   returnează i
sfârşit dacă
dacă c_si > c_sj atunci
   returnează j
sfârşit dacă
dacă i < j atunci
   returnează i
sfârşit dacă
returnează j
```

Cu aceasta, algoritmul pentru construirea mulțimilor *j*-optime devine:

```
p ← 1
s_p ← 1
S_a ← a_sp
S_b ← b_p
pentru j = 2, n execută:
   p = p + 1
   s_p = j
   S_a = S_a + a_sp
   S_b = b_j
   dacă S_a > S_b atunci
      p_m = p
      k = p - 1
      cât timp k > 0 execută:
         p_m = ordn(p_m, k)
         k = k - 1
      sfârşit cât timp
      S_a = S_a - a_spm
      pentru i = p_m, p-1 execută:
         s_i = s_i+1
      sfârşit pentru
      p = p - 1
   sfârşit dacă
sfârşit pentru
```

Lucrările care se vor executa fără întârziere sunt în număr de *p*, iar ordinea optimă se obține prin afișarea elementelor tabloului *S*.

Algoritmul lui *Johnson* se poate extinde la o problemă de minimizare a timpului total de lucru a *n* piese pe trei mașini păstrând în continuare aceeași ordine de prelucrare (mai întâi pe mașina M_1, apoi pe M_2 și în final pe M_3); de asemenea, există numeroase alte tipuri de ordonanțare, însă pentru multe dintre ele nu au fost propuși algoritmi eficienți de rezolvare.

Bibliografie

1. **Tomescu Ioan**, *Combinatorică și teoria grafurilor*, Universitatea București, 1978
2. **Tomescu Ioan**, *Probleme de combinatorică și teoria grafurilor*, Editura Didactică și Pedagogică București, 1981
3. **Rancea Doina**, *Algoritmi fundamentali*, Editura Computer Libris Agora, Cluj-Napoca, 1999

Dl. prof. Ilie Vieru este cadru didactic la Colegiul Național "Gh. Vrănceanu" din Bacău și poate fi contactat prin e-mail la adresa **ivieru@xnet.ro.**