Proiectarea algoritmilor

Lucrare de laborator nr. 7

Paradigma programării dinamice

Problema rucsacului, varianta discretă

Cuprins

1 Problema rucsacului

1.1 Descrierea problemei

Se consideră un rucsac de capacitate $M \in \mathbb{Z}_+$ şi n obiecte $1, \ldots, n$ de dimensiuni (greutăţi) $w_1, \ldots, w_n \in \mathbb{Z}_+$.

Un obiect i este introdus în totalitate în rucsac, $x_i = 1$, sau nu este introdus deloc, $x_i = 0$, astfel că o umplere a rucsacului constă dintr-o secvenţă $x_1, \ldots, x_n$ cu $x_i \in \{0, 1\}$ şi $\sum\limits_{i=1}^{n} x_i \cdot w_i \leq M$.

Introducerea obiectului i în rucsac aduce profitul $p_i \in \mathbb{Z}$, iar profitul total este $\sum\limits_{i=1}^{n} x_i p_i$.

Problema constă în a determina o alegere $(x_1, \ldots, x_n)$ *care să aducă un profit maxim.*

Singura deosebire faţă de varianta continuă studiată la metoda greedy constă în condiţia $x_i \in \{0, 1\}$, în loc de $x_i \in [0, 1]$.

1.2 Modelul matematic

Problema iniţială (starea $\mathrm{RUCSAC}(n, M)$)

Funcţia obiectiv:

$$\max \sum_{i=1}^{n} x_i \cdot p_i$$

Restricţii:

$$\begin{array}{l} \sum\limits_{i=1}^{n} x_i \cdot w_i \leq M \\ x_i \in \{0, 1\}, i = 1, \ldots, n \\ w_i \in \mathbb{Z}_+, p_i \in \mathbb{Z}, i = 1, \ldots, n \\ M \in \mathbb{Z}_+ \end{array}$$

Generalizarea problemei iniţiale (starea RUCSAC(j,X))
Funcţia obiectiv:

$$\max \sum_{i=1}^{j} x_i \cdot p_i$$

Restricţii:

$$\begin{array}{l} \sum\limits_{i=1}^{j} x_i \cdot w_i \leq X \\ x_i \in \{0,1\}, i = 1,\ldots,j \\ w_i \in \mathbb{Z}_+, p_i \in \mathbb{Z}, i = 1,\ldots,j \\ X \in \mathbb{Z}_+ \end{array}$$

Notăm cu $f_j(X)$ valoarea optimă pentru instanţa RUCSAC(j,X). Dacă $j = 0$ şi $X \geq 0$, atunci $f_j(X) = 0$. Presupunem $j > 0$. Notăm cu $(x_1,\ldots,x_j)$ alegerea care dă valoarea optimă $f_j(X)$.

Dacă $x_j = 0$ (obiectul j nu este pus în rucsac), atunci, conform principiului de optim, $f_j(X)$ este valoarea optimă pentru starea RUCSAC$(j-1,X)$ şi de aici $f_j(X) = f_{j-1}(X)$.

Dacă $x_j = 1$ (obiectul j este pus în rucsac), atunci, din nou conform principiului de optim, $f_j(X)$ este valoarea optimă pentru starea RUCSAC$(j-1,X-w_j)$ plus p_j şi, de aici, $f_j(X) = f_{j-1}(X - w_j) + p_j$.

Combinând relaţiile de mai sus obţinem:

$$f_j(X) = \begin{cases} -\infty, & \text{dacă } X < 0 \\ 0, & \text{dacă } j = 0 \text{ şi } X \geq 0 \\ \max\{f_{j-1}(X), f_{j-1}(X-w_j)+p_j\}, & \text{dacă } j > 0 \text{ şi } X \geq 0 \end{cases} \quad (1)$$

Am considerat $f_j(X) = -\infty$, dacă $X < 0$.

Din relaţia (1) rezultă că proprietatea de substructură optimă se caracterizează astfel:

> Soluţia optimă $(x_1,\ldots,x_j)$ a problemei RUCSAC(j,X) include soluţia optimă $(x_1,\ldots,x_{j-1})$ a subproblemei RUCSAC$(j-1,X-x_jw_j)$.

Soluţia optimă pentru RUCSAC(j,X) se poate obţine utilizând soluţiile optime pentru subproblemele RUCSAC(i,Y) cu $1 \leq i < j, 0 \leq Y \leq X$.

Relaţia (1) implică o recursie în cascadă şi deci numărul de subprobleme de rezolvat este $O(2^n)$, fapt pentru care calculul şi memorarea eficientă a valorilor optime pentru subprobleme devine un task foarte important.

Fie $M = 10, n = 3$ şi greutăţile şi profiturile date de următorul tabel:

i	1	2	3
w_i	3	5	6
p_i	10	30	20

Valorile optime pentru subprobleme sunt calculate cu ajutorul relaţiei (1)≡(2)

$$f_j(X) = \begin{cases} -\infty, & \text{dacă } X < 0 \\ 0, & \text{dacă } j = 0 \text{ şi } X \geq 0 \\ \max\{f_{j-1}(X), f_{j-1}(X-w_j)+p_j\}, & \text{dacă } j > 0 \text{ şi } X \geq 0 \end{cases} \quad (2)$$

Valorile optime pot fi memorate într-un tablou bidimensional astfel:

X	0	1	2	3	4	5	6	7	8	9	10
f_0	0	0	0	0	0	0	0	0	0	0	0
f_1	0	0	0	10	10	10	10	10	10	10	10
f_2	0	0	0	10	10	30	30	30	40	40	40
f_3	0	0	0	10	10	30	30	30	40	40	40

Tabloul de mai sus este calculat linie cu linie. Exemplu: $f_2(8) = \max\{f_1(8), f_1(8-5)+30\} = \max\{10, 40\} = 40$.

Tabloul valorilor optime are dimensiunea $n \cdot M$ (au fost ignorate prima linie şi prima coloană). Dacă $M = O(2^n)$ rezultă că atât complexitatea spaţiu, cât şi cea timp sunt exponenţiale.

Privind tabloul de mai sus observăm că există multe valori care se repetă. *Cum putem memora mai compact tabloul valorilor optime*?

Soluţie: Construim graficele funcţiilor $f_0, f_1, f_2 \cdots$

$$f_0(X) = \begin{cases} -\infty & ,X < 0 \\ 0 & ,X \geq 0 \end{cases}$$

$$g_0(X) = f_0(X - w_1) + p_1 = \begin{cases} -\infty & ,X < 3 \\ 10 & ,3 \leq X \end{cases}$$

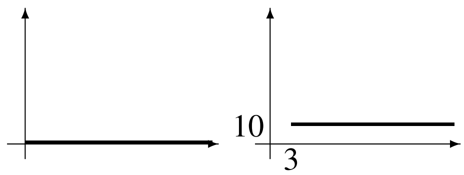

Figura 1: Funcţiile f_0 şi g_0

$$f_1(X) = \max\{f_0(X), g_0(X)\} = \begin{cases} -\infty & ,X < 0 \\ 0 & ,0 \leq X < 3 \\ 10 & ,3 \leq X \end{cases}$$

$$g_1(X) = f_1(X - w_2) + p_2 = \begin{cases} -\infty & ,X < 5 \\ 30 & ,5 \leq X < 8 \\ 40 & ,8 \leq X \end{cases}$$

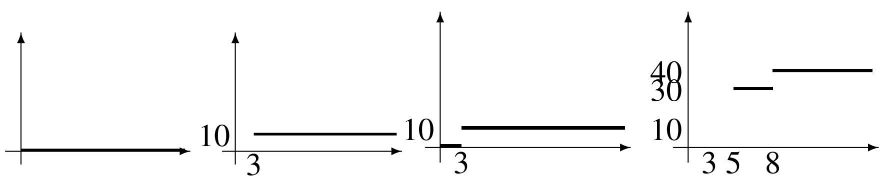

Figura 2: Funcţiile f_0 şi g_0; Funcţiile f_1 şi g_1

$$f_2(X) = \max\{f_1(X), g_1(X)\} = \begin{cases} -\infty & ,X < 0 \\ 0 & ,0 \leq X < 3 \\ 10 & ,3 \leq X < 5 \\ 30 & ,5 \leq X < 8 \\ 40 & ,8 \leq X \end{cases}$$

$$g_2(X) = f_2(X - w_3) + p_3 = \begin{cases} -\infty & ,X < 6 \\ 20 & ,6 \le X < 9 \\ 30 & ,9 \le X < 11 \\ 50 & ,11 \le X < 14 \\ 60 & ,14 \le X \end{cases}$$

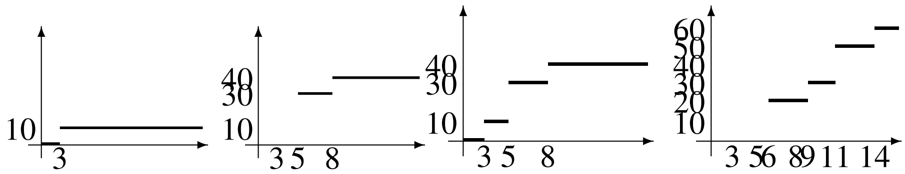

Figura 3: Funcţiile f_1 şi g_1; Funcţiile f_2 şi g_2

$$f_3(X) = \max\{f_2(X), g_2(X)\} = \begin{cases} -\infty & ,X < 0 \\ 0 & ,0 \le X < 3 \\ 10 & ,3 \le X < 5 \\ 30 & ,5 \le X < 8 \\ 40 & ,8 < X \le 11 \\ 50 & ,11 \le X < 14 \\ 60 & ,14 \le X \end{cases}$$

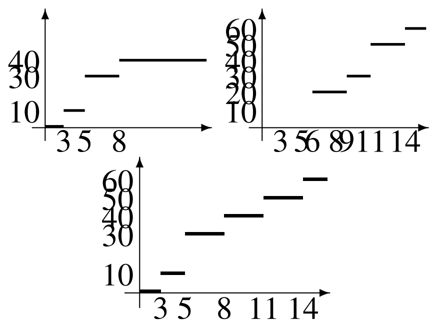

Figura 4: Funcţiile f_2 şi g_2; Funcţia f_3

Se remarcă faptul că funcţiile f_i şi g_i sunt funcţii în scară. Graficele acestor funcţii pot fi reprezentate prin mulţimi finite din puncte din plan. De exemplu, graficul funcţiei f_2 este reprezentat prin mulţimea $\{(0,0),(3,10),(5,30),(8,40)\}$.

O mulţime care reprezintă o funcţie în scară conţine acele puncte în care funcţia face salturi.

Graficul funcţiei g_i se obţine din graficul funcţiei f_i printr-o translaţie. Graficul funcţiei f_{i+1} se obţine prin interclasarea graficelor funcţiilor f_i şi g_i.

1.3 Algoritm

În general, fiecare f_i este complet specificat de o mulţime $S_i = \{(X_j, Y_j) \mid j = 0, \ldots, r\}$, unde $Y_j = f_i(X_j)$.

Presupunem $X_1 < \cdots < X_r$.

Analog, funcţiile g_i sunt reprezentate prin mulţimile $T_i = \{(X + w_{i+1}, Y + p_{i+1}) \mid (X,Y) \in S_i\}$.

Notăm $T_i = \tau(S_i)$ şi $S_{i+1} = \mu(S_i, T_i)$.

Mulţimea S_{i+1} se obţine din S_i şi T_i prin interclasare.

Operaţia de interclasare se realizează într-un mod asemănător cu cel de la interclasarea a două linii ale orizontului.

Se consideră o variabilă L care ia valoarea 1 dacă graficul lui f_{i+1} coincide cu cel al lui f_i şi cu 2 dacă el coincide cu cel al lui g_i. Deoarece $(0,0)$ aparţine graficului rezultat, considerăm $L=1, j=1$ şi $k=1$. Presupunând că la un pas al interclasării se compară $(X_j,Y_j) \in S_i$ cu $(X_k,Y_k) \in T_i$, atunci:

- dacă $L=1$:
  - dacă $X_j < X_k$, atunci se adaugă (X_j,Y_j) în S_{i+1} şi se incrementează j;
  - dacă $X_j = X_k$:
    - dacă $Y_j > Y_k$, atunci se adaugă (X_j,Y_j) în S_{i+1} şi se incrementează j şi k;
    - dacă $Y_j < Y_k$, atunci se adaugă (X_k,Y_k) în S_{i+1}, $L=2$ şi se incrementează j şi k;
  - dacă $X_j > X_k$, atunci, dacă $Y_k > Y_j$, se adaugă (X_k,Y_k) în S_{i+1}, $L=2$ şi se incrementează k;
- dacă $L=2$:
  - dacă $X_j < X_k$, atunci, dacă $Y_j > Y_k$, se adaugă (X_j,Y_j) în S_{i+1}, $L=1$ şi se incrementează j;
  - dacă $X_j = X_k$:
    - dacă $Y_j < Y_k$, atunci se adaugă (X_k,Y_k) în S_{i+1} şi se incrementează j şi k;
    - dacă $Y_j > Y_k$, atunci se adaugă (X_j,Y_j) în S_{i+1}, $L=1$ şi se incrementează j şi k;
  - dacă $X_j > X_k$, atunci se adaugă (X_k,Y_k) în S_{i+1} şi se incrementează k;

Notăm cu `interclGrafice(Si,Ti)` funcţia care determină S_{i+1} conform algoritmului de mai sus.

$S_3 = \{(0,0),(3,10),(5,30),(8,40),(11,50),(14,60)\}$.

$S_2 = \{(0,0),(3,10),(5,30),(8,40)\}$.

$S_1 = \{(0,0),(3,10)\}$.

$S_0 = \{(0,0)\}$.

Se caută în $S_n = S_3$ perechea (X_j,Y_j) cu cel mai mare X_j pentru care $X_j \le M$. Obţinem $(X_j,Y_j) = (8,40)$. Deoarece $(8,40) \in S_3$ şi $(8,40) \in S_2$ rezultă $f_{optim}(M) = f_{optim}(8) = f_3(8) = f_2(8)$ şi deci $x_3 = 0$. Perechea (X_j,Y_j) rămâne neschimbată.

Pentru că $(X_j,Y_j) = (8,40)$ este în S_2 şi nu este în S_1, rezultă că $f_{optim}(8) = f_1(8-w_2)+p_2$ şi deci $x_2 = 1$. În continuare se ia $(X_j,Y_j) = (X_j - w_2, Y_j - p_2) = (8-5, 40-30) = (3,10)$.

Pentru că $(X_j,Y_j) = (3,10)$ este în S_1 şi nu este în S_0, rezultă că $f_{optim}(3) = f_1(3-w_1)+p_1$ şi deci $x_1 = 1$.

Iniţial se determină perechea $(X_j,Y_j) \in S_n$ cu cel mai mare X_j pentru care $X_j \le M$. Valoarea Y_j constituie încărcarea optimă a rucsacului, *i.e.*, valoarea funcţiei obiectiv din problema iniţială.

Pentru $i = n-1, \ldots, 0$:

- dacă (X_j,Y_j) este în S_i, atunci $f_{i+1}(X_j) = f_i(X_j) = Y_j$ şi se consideră $x_{i+1} = 0$ (obiectul $i+1$ nu este ales);
- dacă (X_j,Y_j) nu este în S_i, atunci $f_{i+1}(X_j) = f_i(X_j - w_{i+1}) + p_{i+1} = Y_j$ şi se consideră $x_{i+1} = 1$ (obiectul $i+1$ este ales), $X_j = X_j - w_{i+1}$ şi $Y_j = Y_j - p_{i+1}$.

```
procedure rucsac(M, n, w, p, x)
    S0 ← {(0,0)}
    T0 ← {(w1,p1)}
    for i ← 1 to n
        Si ← interclGrafice(Si−1,Ti−1)
```

```
        T_i ← {(X + w_{i+1}, Y + p_{i+1}) | (X,Y) ∈ S_i}
    determină (X_j, Y_j) cu X_j = max{X_i | (X_i, Y_i) ∈ S_n, X_i ≤ M}
    for i ← n-1 downto 0 do
        if (X_j, Y_j) ∈ S_i
           then x_{i+1} ← 0
           else x_{i+1} ← 1
                X_j ← X_j − w_{i+1}
                Y_j ← Y_j − p_{i+1}
end
```

2 Sarcini de lucru

1. Scrieţi o funcţie C/C++ care implementează algoritmul `rucsac`.
2. Se consideră un rucsac de capacitate $M \in \mathbb{Z}_+$ şi n obiecte $1,\ldots,n$ de dimensiuni (greutăţi) $w_1,\ldots,w_n \in \mathbb{Z}_+$.. Scrieţi un program care să afişeze soluţia optimă.

Bibliografie

[1] Lucanu, D. şi Craus, M., *Proiectarea algoritmilor*, Editura Polirom, 2008.

[2] R.E. Bellman şi S.E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, 1962.

[3] Moret, B.M.E.şi Shapiro, H.D. , *Algorithms from P to NP: Design and Efficiency*, The Benjamin/Cummings Publishing Company, Inc., 1991.