CAPITOLUL II

PROGRAMARE COMBINATORIALĂ

2.1 Elemente introductive

O problemă de optimizare (adică o funcție de mai multe variabile care trebuie maximizată sau minimizată cu satisfacerea unui set finit de restricții) ***are caracter combinatorial*** dacă fiecare variabilă poate lua independent un număr de valori numerice apriori cunoscute de exemplu: valori întregi, nenegative inferioare unui prag dat sau numai valori 0 sau 1.

Analiza noastră se va restrânge la acele probleme cuantificabile matematic prin datele amintite: *funcția obiectiv, variabilele, restricțiile, valori permise variabilelor*, deoarece clasa de probleme combinatoriale este cu mult mai largă.

Caracteristica unei probleme combinatoriale este că *numărul combinațiilor posibile de valori este finit*. Aceste combinații se numesc **Soluții** ale problemei şi mulțimea lor se va nota cu **Ω**. Combinațiile care, în plus, satisfac restricțiile problemei, şi ele în număr finit, se vor numi **Soluții Admisibile** şi mulțimea lor se va nota cu A. Cu aceste notații, o problemă combinatorială se formalizează astfel:

dată fiind o funcție *definită* în toate elementele din Ω să se determine $x^* \in A$ cu proprietatea:

$$f(x^*) = \max_{x \in A} f(x)$$

La prima vedere o problemă de programare liniară este o problemă combinatorială deoarece soluția sa optimă se poate găsi inspectând doar mulțimea A_{SAB} care este finită. Există totuşi o mare deosebire: o soluție bazică nu poate fi construită prin acordarea de valori variabilelor întrucât din start plaja de valori admisibile este infinită. În notațiile de mai sus, A poate fi socotită finită, însă mulțimea Ω a combinațiilor posibile de valori date variabilelor este R_+^n care este infinit. Şansa de a da peste un element din A considerând o combinație de valori sau alta este practic nulă.

Exemplul tipic de problemă combinatorială pe care îl avem în vedere în cadrul acestui curs este **Problema de Optimizare cu Variabile Bivalente** (vezi problema de afectare, problema alegerii proiectelor de investiții). Aici numărul total de combinații posibile, respectiv numărul de elemente din Ω este $\mathbf{2^n}$ unde *n* este numărul variabilelor.

În general o problemă combinatorială se rezolvă prin *Enumerarea Totală* sau *Parțială* a mulțimii Ω a soluțiilor sale. Vorbim de *enumerare totală* dacă determinarea elementului optimal $x^* \in A$ necesită generarea tuturor combinațiilor posibile de valori date variabilelor deci a tuturor elementelor din Ω. *Enumerarea parțială* reprezintă determinarea lui x^* prin generarea efectivă a unei părți din Ω partea negenerată fiind recunoscută ca neconținând elemente optimale. Indiferent de schema de enumerare, o dată generat un element $x \in \Omega$ se efectuează următoarele operații:

1. Se cercetează dacă $x \in A$; dacă NU se trece la generarea altui element din Ω. Dacă DA:

2. Se compară f(x) cu valoarea obiectivului *f* în cel mai bun element din A găsit anterior; dacă se îmbunătăţeşte valoarea obiectivului (în sensul optimului), x se reţine ca cel mai bun element din A, găsit. În caz contrar x se abandonează şi se trece la generarea unui nou element din Ω.

Este important de reţinut faptul că generarea mulţimii Ω sau chiar a unei părţi nu înseamnă în nici un caz memorarea elementelor generate şi aceasta pentru două motive: sunt foarte multe şi apoi nu sunt necesare (exceptând cel mai bun element din A găsit la un stadiu sau altul al enumerării).

Schema logică din figura 1 formalizează rezolvarea problemelor de optimizare combinatorială (P). Ea foloseşte o "locaţie" x_{CMB} realizată de obicei sub forma unui vector în care se depoziteză "**c**el **m**ai **b**un" element din A găsit până la un anumit moment şi o variabilă z_{CMB} care reţine valoarea obţinută în x_{CMB}. În cazul unui obiectiv de maxim, iniţial $z_{CMB} = -\infty$, $x_{CMB} = 0$. În cuprinsul schemei, punctul delicat îl constituie modul de generare efectivă a elementului din Ω, mod care va fi discutat ulterior.

Neajunsul evident al *enumerării explicite a tuturor soluţiilor* problemei (P) rezidă în *numărul mare* al acestora. Generarea unei soluţii prin atribuire de valori variabilelor este o operaţie practic instantanee pe un calculator (sunt necesare anumite precauţii pentru evitarea generării unei soluţii de mai multe ori); chiar şi verificarea restricţiilor de către o combinaţie generală nu durează prea mult şi totuşi, verificarea atâtor soluţii face ca timpul total să depăşească lesne limita rezonabilului.

Alte trei clase de probleme formalizabile matematic pentru care numărul de soluţii admisibile este foarte mare sunt următoarele:

a) *Problema ordonanţării* prelucrării unor repere pe mai multe utilaje. Se cunosc timpii de prelucrare ai reperelor pe fiecare utilaj în parte, precum şi ordinea în care utilajele trebuie parcurse. Problema constă în stabilirea ordinii de lansare în fabricaţie a reperelor astfel încât timpul total de aşteptare al utilajului să fie minim. Este clar că numărul total de soluţii este $n!$, unde n = numărul de repere ce trebuie prelucrate. Ori pentru n = 20 avem: 20!=243.290.200.816.664.000 număr uşor de scris dar greu de imaginat ca mărime.
b) *Problema comis voiajorului.* Un comis voiajor trebuie să viziteze n oraşe $c_1,...,c_n$. Se cunoaşte matricea timpilor de deplasare de la un oraş la altul, bineînţeles acolo unde există legături directe. Comis voiajorul pleacă dintr-un anumit oraş, să zicem c_1, şi trebuie să treacă prin fiecare oraş, o singură dată, întorcându-se în final în oraşul de plecare. Problema este ca din cele (n-1)! succesiuni posibile să se aleagă acel drum căruia îi corespunde timpul total de deplasare minim.
c) Nu în ultimul rând în clasa problemelor de optimizare combinatorială putem include şi problemele de programare liniară în numere întregi (PLI). Într-adevăr, pentru această problemă şi în special pentru cele practice se pot şti de la bun început nivelele maxime admise pentru valorile întregi ce le pot lua variabilele şi în consecinţă numărul combinaţiilor posibile de valori acordate acestora este finit.

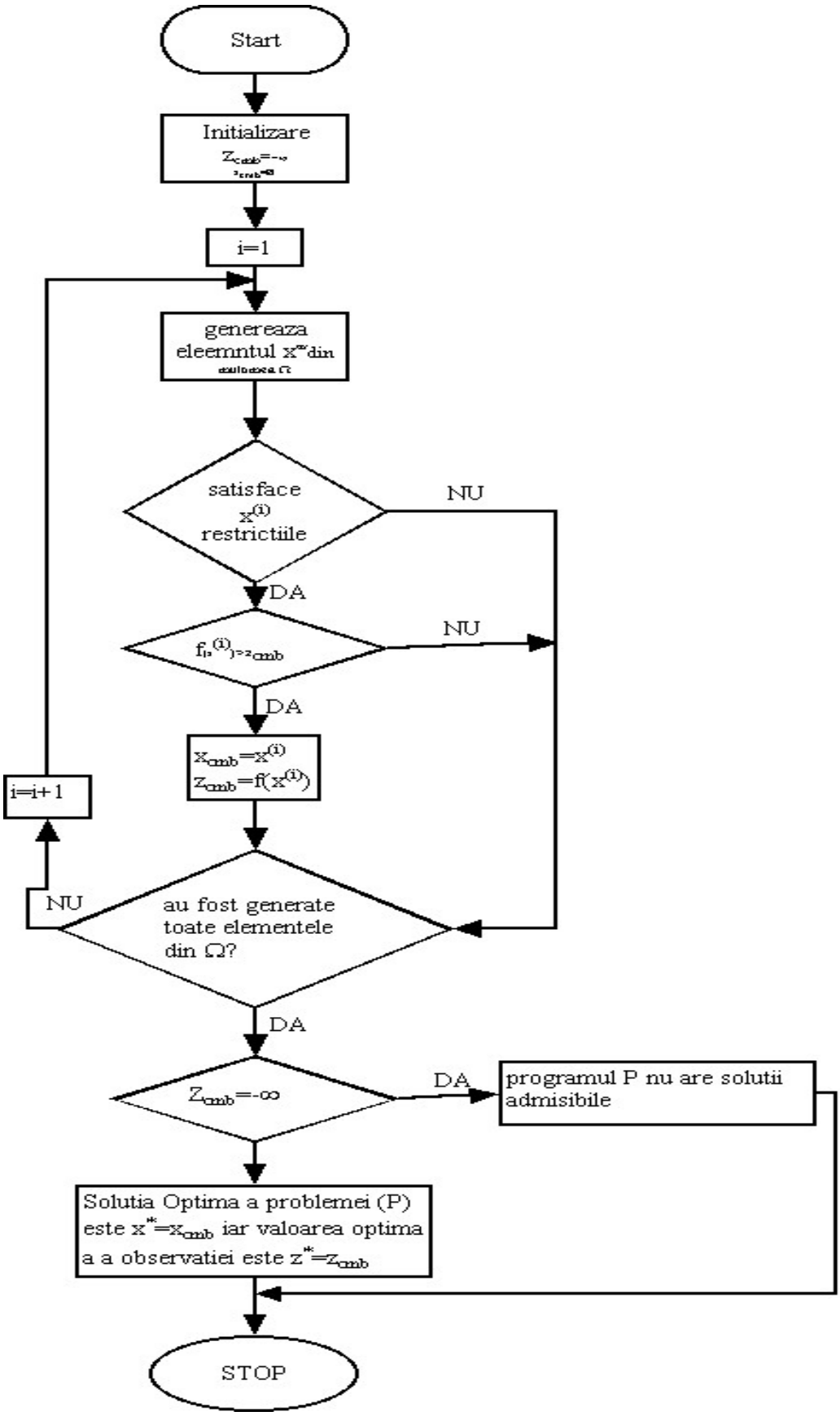

Start
Initializare
i=1
genereaza
satisface
restrictiile
NU
DA
NU
DA
i=i+1
NU
au fost generate
toate elementele
din Ω?
DA
DA
programul P nu are solutii
admisibile
Solutia Optima a problemei (P)
STOP

Figura 1

După cum am specificat deja, unul din punctele delicate ale oricărei scheme de enumerare îl constituie *generarea combinațiilor de valori* date variabilelor problemei. Acest proces trebuie să satisfacă două condiții:

1. nici o combinație nu trebuie generată de mai multe ori;
2. nici o combinație posibilă nu trebuie omisă.

2.2 Formalizarea problemelor de generare a soluțiilor unui program combinatorial

Considerăm date: un număr de variabile: $x_1,\ldots,x_n$ şi pentru fiecare variabilă x_i o listă (vector) de valori numerice L_i a cărei lungime depinde de variabila în cauză. Se cere generarea (nu neapărat memorarea) tuturor combinațiilor posibile (pe parcurs vom folosi alternativ şi termenul de soluție în locul celui de combinație)

$$\bar{x} = (\bar{x}_1,\ldots,\bar{x}_n) \text{ cu } \bar{x}_i \in L_i,\ i = 1, ..., n$$

respectând condițiile 1 şi 2 de mai sus.

Introducem termenul de *soluție parțială* sau *pseudosoluție* ca fiind un vector $\sigma=(\bar{x}_1,\ldots,\bar{x}_m)$ în care $1 \le m \le n$ şi $\bar{x}_i \in L_i$, i = 1,...,m.

Relativ la pseudosoluția σ adoptăm următoarea terminologie: vom spune că variabilele $x_1,\ldots,x_m$ *au fost fixate* la valorile $\bar{x}_1,\ldots,\bar{x}_n$ din listele corespunzătoare în timp ce restul variabilelor, adică $\bar{x}_{m+1},\ldots,\bar{x}_n$ se vor numi *libere*. Este clar că dacă m=n, adică toate variabilele au fost fixate la diferite valori, pseudosoluția respectivă este chiar o soluție (combinație). La pseudosoluțiile definite mai sus adăugăm şi pseudosoluția vidă ∅ în care toate variabilele sunt libere.

Pentru mai buna înțelegere a procedeului de enumerare construim un graf **T** ale cărui noduri sunt pseudosoluțiile mai sus definite. O muchie în graful **T** va lega pseudosoluția $\sigma=(\bar{x}_1,\ldots,\bar{x}_m)$ de o alta de forma $\tau=(\bar{x}_1,\ldots,\bar{x}_{m+1})$. Se vede că τ menține valorile variabilelor $x_1,\ldots,x_m$ fixate în σ şi în plus fixează variabila următoare x_{m+1}. Vom spune că τ este un *succesor* al lui σ în timp ce σ este un *predecesor* al lui τ. Este clar că:

1. O pseudosoluție $\sigma=(\bar{x}_1,\ldots,\bar{x}_m)$ va avea succesori numai dacă m<n şi în acest caz numărul succesorilor va fi egal cu numărul valorilor numerice din lista L_{m+1}.
2. Pseudosoluțiile fără succesori sunt exact *soluțiile căutate*.
3. Orice pseudosoluție are un unic predecesor, excepție făcând pseudosoluția ∅.
4. Graful **T** este prin construcție un *arbore* adică un graf conex şi fără cicluri, a cărui rădăcină este ∅. În consecință există un *unic lanț de muchii* care uneşte rădăcina ∅ de o pseudosoluție dată.

Pentru o problemă cu două variabile x_1, x_2 luând valori din listele $L_1=\{0,1\}$, $L_2=\{2,5,6\}$arborele **T** este prezentat în figura 2:

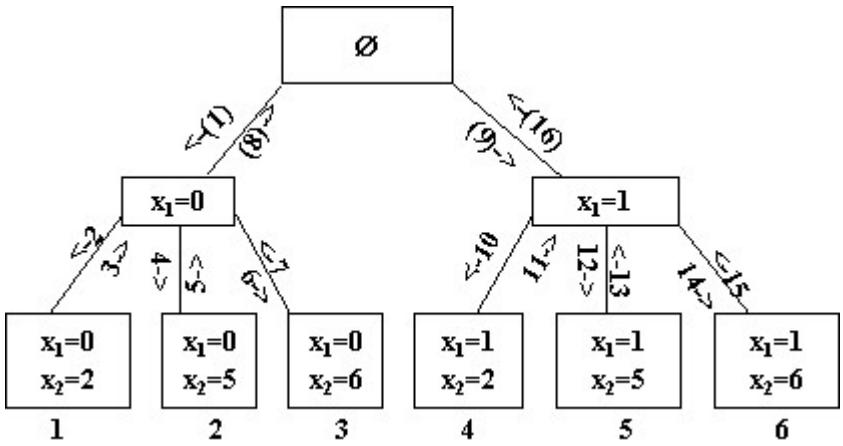


Figura 2

În principiu, generarea combinațiilor se identifică cu o explorare a nodurilor grafului **T** care trebuie să respecte următoarele reguli:

a) ∀ muchie va fi parcursă exact de două ori: o dată într-un sens şi a două oară în sensul opus;

b) Explorarea începe din rădăcina ∅ şi sfârşeşte tot în ∅.

În figura 2, exploararea arborelui este indicată prin săgeți însoțite de un număr de ordine.

Vom utiliza termenul de ***pseudosoluție curentă*** (psc) pentru a marca nodul din **T** în care a ajuns procesul de enumerare. Dacă psc este $\sigma=(\bar{x}_1,\ldots,\bar{x}_m)$ un *succesor* al său (dacă există, adică în cazul m<n), se va nota $\tau=(\bar{x}_1,\ldots,\bar{x}_m,\bar{x}_{m+1})=(\sigma,\bar{x}_{m+1})$. Un *predecesor* al psc (în caz că există, adică dacă avem m>0) se va nota cu $\eta=(\bar{x}_1,\ldots,\bar{x}_{m-1})$ adică $\sigma=(\eta,\bar{x}_m)$.

2.3 Principiul general Branch and Bound (B-B) de rezolvare a problemelor combinatoriale

Reluăm problema de optimizare combinatorială (P) constând în următoarele date:

- o mulțime finită Ω ale cărei elemente sunt soluțiile problemei (P);
- o submulțime $A \subset \Omega$ zisă *a soluțiilor admisibile*;
- o funcție numerică *f* definită în toate soluțiile problemei (P).

Problema cere aflarea unui $x^* \in A$ cu proprietatea $f(x^*) = \max(f(x)), x \in A$.

Metoda B-B este o schemă de enumerare parțială a cărei aplicare reclamă satisfacerea prealabilă a următoarelor condiții:

1. Pentru (∀) submulțime nevidă $S \subset \Omega$ se poate determina relativ uşor elementul x_s cu proprietatea

$$f(x_s)=\max(f(x)),\ x\in S \qquad (1)$$

2. Există o regulă de ramificare R care aplicată unei submulțimi $S\subset\Omega$ cu cel puțin două elemete produce o partiție a mulțimii S astfel:

$$S-\{x_s\}=S_1\cup S_2\cup\ldots\cup S_p \qquad (S_i\cap S_j=\varnothing \text{ pentru } i\neq j) \qquad (2)$$

De notat că unele din mulțimile S_i ar putea fi vide.

Definiția 1: Prin ramificarea unei mulțimi $S\subset\Omega$ înțeleg aplicarea regulii R asupra lui S şi obținerea partiției (2).

Definiția 2: Mărginirea unei mulțimi $S\subset\Omega$ reprezintă determinarea elementului optimal x_s cu proprietatea (1).

În linii generale principiul B-B se aplică astfel:

- se determină x_Ω adică elementul cu propritatea: $f(x_\Omega)=\max(f(x)),\ x\in\Omega$;
- dacă $x_\Omega\in A$ atunci în mod evident $x^*=x_\Omega \Rightarrow$ STOP
- în caz contrar ramificăm Ω căutând elementul optimal x^* într-una din submulțimile rezultate prin partiționare.

Vom introduce şi aici parametrul z_{CMB} şi locația x_{CMB} descrise în cadrul schemei generale de rezolvare a problemelor combinatoriale. Rolul lui z_{CMB} este acela de a opri procesul de exploatare a unor „zone" din Ω în care în mod cert nu se află elementul x^* căutat.

În orice stadiu al aplicării metodei mulțimii Ω se prezintă sub forma unei reuniuni de părți disjuncte:

$$\Omega=\Omega' \cup\{x_{CMB}\}\cup\Omega'' \qquad (3)$$

cu proprietatea că elementul optimal x^* nu se găseşte în mod cert în Ω''.

La rândul său submulțimea Ω' (cât timp este nevidă) se prezintă sub forma unei reuniuni de submulțimi disjuncte:

$$\Omega'=S_\alpha\cup S_\beta\ldots\cup S_\omega \qquad (4)$$

obținut printr-un proces de ramificare succesivă început prin ramificarea mulțimii totale Ω.

La start $\Omega'=\Omega$, $x_{cmb}=\varnothing$, $\Omega''=\varnothing$.

Derularea metodei B-B impune submulțimilor din partiția (4) următoarele proprietăți:

a) Fiecare din ele are cel puțin două elemente;
b) Fiecare din ele a fost mărginită; cu alte cuvinte se cunosc elementele $x_{S\alpha}$, $x_{S\beta},\ldots$, $x_{S\omega}$ care maximizează obiectivul f pe $S_\alpha,S_\beta\ldots,S_\omega$;
c) Nici una din soluțiile $x_{S\alpha}$, $x_{S\beta},\ldots$, $x_{S\omega}$ nu este soluție admisibilă;
d) Există inegalitățiile stricte: $f(x_{S\alpha})>z_{CMB}$, $f(x_{S\beta})>z_{CMB},\ldots,f(x_{S\omega})>z_{CMB}$.

Cu aceste pregătiri **o iterație a metodei B-B** constă în:

Operația 1: Dintre $S_\alpha,S_\beta\ldots,S_\omega$ care compun partiția (L_1) se alege acea submulțime corespunzătoare maximului $\underline{z} = \max\{f(x_{S\alpha}), f(x_{S\beta}),\ldots,f(x_{S\omega})\}$. Pentru simplitate notăm această submulțime cu S. Ramificăm S obținând partiția: $S=S_1\cup S_2\ldots\cup S_P$

Operația 2: Mărginim submulțimea S_1 (în caz că este nevidă, altfel vom trece la S_2). Trei cazuri sunt posibile relativ la elementul optimal din S_1 obținut în urma mărginirii:

a) $f(x_{S1}) \le z_{CMB}$ (indiferent de faptul că x_{S1} este sau nu în A). Considerăm că S_1 nu conține soluții admisibile strict mai bune decât cea mai bună găsită până acum şi ca atare S_1 se abandonează, adică se transferă în zona Ω'':

$$\Omega' \leftarrow (\Omega' - S_1),\ \Omega'' \leftarrow (\Omega'' \cup S_1).$$

Trecem la mărginea lui S_2 şamd.

b) $f(x_{S1}) > z_{cmb}$ şi $x_{S1} \in A$:
În acest caz facem actualizările: $x_{CMB}=x_{S1},\ z_{CMB}=f(x_{S1})$

Două situații sunt posibile:

$b_1)$ $\bar{z} = z_{CMB}$ => $x^* = x_{CMB}$ $\Rightarrow$ STOP

$b_2)$ $\bar{z} > z_{CMB}$ => abandonăm S_1 şi trecem la mărginirea lui S_2 şamd.

c) $f(x_{s1}) > z_{CMB}$ şi $x_{S1} \notin A$
Cu excepția cazului extrem când S_1 are un singur element şi prin urmare se abandonează, vom include S_1 printre mulțimile care realizează partiția (4) a mulțimii Ω'. Trecem la mărginirea lui S_2 şamd.

Operația 3: Mărginirea şi toate considerațiile precedente se aplică apoi mulțimii S_2 apoi lui S_3 şamd. La sfârşitul operației submulțimea S dispare din partiția (4), ea fiind înlocuită cu o reuniune de părți disjuncte ale sale. Reuniunea acestor părți este, de regulă, mai mică decât mulțimea S înlocuită, diferența fiind abandonată, adică transferată în zona Ω''. În consecință cu fiecare iterație mulțimea Ω' se micşorează.

Se revine la Operația 1 în caz că Ω este nevidă, altfel STOP.

Observații:

1. Pe parcursul aplicării procedurii descrise anterior, valorile succesive ale lui $\bar{z}$ formează un şir descrescător în timp ce valorile lui z_{CMB} formează un şir crescător. În orice moment al derulării algoritmului avem:

$$z_{CMB} \le f(x^*) \le \bar{z}$$

2. Algoritmul se opreşte fie atunci când $\bar{z} = z_{CMB}$ (cazul b_1), fie când $\Omega' = \varnothing$.
3. Dacă $z_{CMB} = -\infty$ ($\Leftrightarrow x_{CMB} = \varnothing$) => (P) nu are soluții admisibile. Altminteri $x^* = x_{CMB}$.
4. Submulțimile care realizează partiția curentă (4) a mulțimii Ω' se numesc **active** şi ele se înscriu într-o listă pe măsura apariției lor. Din descrierea algoritmului rezultă că după ramificare o submulțime încetează a mai fi activă fiind înlocuită cu alte părți ale sale mai mici.
5. Ceea ce s-a prezentat mai înainte constituie esența metodei B-B. Algoritmii de tip B-B specializați în rezolvarea unei probleme combinatoriale concrete vor diferi atât prin forma de prezentare specifică cât şi prin prezența unor teste suplimentare care permit recunoaşterea

inexistenței elementului optimal x^* într-o submulțime rezultată din ramificare şi drept urmare abandonarea acesteia.

2.4 Algoritmul aditiv pentru rezolvarea problemelor de programare liniară bivalentă

Programarea bivalentă este un mijloc adecvat de modelare pentru multe probleme practice importante cum ar fi cele de afectare, de alegere a proiectelor de investiții, ş.a.m.d.

Considerăm problema de programare liniară bivalentă în forma:

$$(P)\begin{cases} \text{Să se determine } x^*=(x^*_1, x^*_2,\ldots, x^*_n) \text{ care maximizează} \\ \text{funcția obiectiv } f=\sum_{j=1}^{n} c_j x_j, & (1) \\ \text{cu restricțiile } \sum_{j=1}^{n} a_{ij}x_j \le b_i \ ; \ i=1,\ldots, m & (2) \\ \text{şi condițiile explicite } x_j \in \{0,1\}, \ j=1,\ldots n & (3) \end{cases}$$

Pentru (P) există 2^n combinații de valori 0 sau 1 atribuite celor *n* variabile. Aceste combinații se vor numi **soluții** ale problemei (P) şi mulțimea lor se va nota cu Ω. Soluțiile care satisfac restricțiile (2) se vor numi **admisibile** şi mulțimea lor se va nota cu A. pentru valori mici ale lui *n* problema (P) se rezolvă simplu prin enumerare explicită a tuturor soluțiilor. Fiecare soluție generată este introdusă în restricțiile (2) pentru a putea vedea dacă este admisibilă sau nu. Pe parcursul derulării procedurii se reține cea mai bună soluție admisibilă generată astfel că în final aceasta va fi soluția optimă x^* căutată.

Deşi soluțiile generate nu trebuie şi memorate, excepție făcând cea mai bună soluție admisibilă găsită pe parcurs, enumerarea totală încetează a mai fi practică atunci când numărul *n* al variabilelor este mare. Se pune în acest caz problema de a găsi x^* generând efectiv numai o parte din mulțimea Ω a soluțiilor (această parte dorindu-se a fi cât mai mică).

Următorul algoritm răspunde problemei formulate anterior şi, deoarece aplicarea lui nu necesită decât adunări si scăderi, el se numeşte ***algoritmul aditiv***. Versiunea originală se datorează lui Egon Balaş (1965) (profesor universitar la Carnegie Mellon University, originar din Cluj, Romania).

2.4.1 Notații. Terminologie

Putem presupune în funcția obiectiv (1) că toți $c_j \le 0$. În cazul în care un coeficient $c_k>0$ efectuăm transformarea $x'_k = 1-x_k$.

- Introducem variabilele de abatere $s_1, s_2,\ldots, s_m$ aducând (P) la forma STAS. Spre deosebire de $x_1,x_2,\ldots,x_n$ care nu pot lua decât valorile 0 şi 1, variabilele $s_1, s_2,\ldots, s_m$ sunt supuse numai condiției de nenegativitate.
- Rescriem matriceal problema:

Să se determine $x^*=(x^*_1, x^*_2,\ldots, x^*_n)$ care maximizează
funcția obiectiv $f = cx$
cu restricțiile $Ax + s = b$
şi condițiile explicite $x_j \in \{0,1\}$, $j = 1,...n$, $s_i \geq 0$, $i = 1,...m$

- Notăm cu $N=\{1,2,\ldots,n\}$ indicii variabilelor, $M=\{1,2,\ldots,m\}$ indicii restricțiilor modelului.
  Pentru fiecare soluție $x \in \Omega$ fie $I(x) = \{ j \in N \mid x_j = 1\}$.
  Pentru soluția în care $x_j = 0$, $j \in N$ vom prefera notația θ. Este clar că $I_{(\theta)} = \varnothing$.
- Construim un graf G ale cărui noduri sunt cele 2^n soluții ale problemei (P). În G există un arc orientat de la nodul *x* la nodul *y* numai dacă $I_{(x)} \subset I_{(y)}$ şi $I_{(y)}$ are exact un element în plus.

Pentru n=3 graful se prezintă ca în figura 3. Toate soluțiile *x* pentru care mulțimiile $I_{(x)}$ au acelaşi număr de elemente au fost reprezentate la acelaşi „nivel".

În principiu soluțiile problemei (P) vor fi examinate astfel:
- se va începe cu soluția θ în care $x_j=0$, $(\forall)$ $j \in N$ pentru motivul că ea realizează maximul funcției obiectiv pe întreg Ω (pentru că $c_j \leq 0$, $j \in N$);
- odată examinată o soluție *x* următoarea soluție care se generează se deduce din *x* acordând valoarea 1 unei variabile x_j care în *x* are valoarea 0.

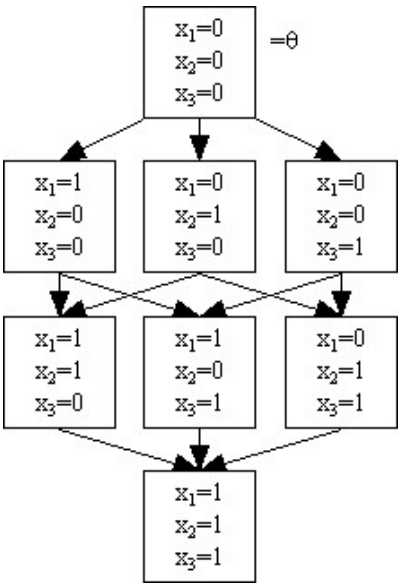


Figura 3

Dată fiind corespondeța biunivocă între soluțiile problemei (P) şi nodurile grafului G, trecerea de la *x* la *y* echivalează cu o deplasare de la nodul *x* la nodul *y* de-a lungul arcului care le uneşte.

Dezvoltând analogia putem spune că derularea algoritmului ce va fi prezentat se identifică cu o deplasare în graful G, deplasare care satisface următoarele cerințe:

1. începe din nodul corespunzător soluției θ;

2. o muchie a grafului G poate fi parcursă în sensul orientării sale cel mult o dată. Dacă acest lucru se întâmplă la un moment dat, într-o fază ulterioară a derulării algoritmului are loc în mod necesar şi deplasarea în sens invers;
3. deplasarea se încheie în nodul de plecare θ.

Notăm cu T subgraful nodurilor şi al muchiilor pe care s-a făcut deplasarea. Condiţiile în care loc această deplasare fac ca T să fie un arbore (un graf convex şi fără cicluri) cu rădăcina θ.

Relativ la două noduri *x* şi *y* din T între care există un arc (aceasta înseamnă că *x* şi *y* sunt soluţii efectiv generate, *y* obţinându-se din *x* dând valoarea 1 unei variabile cu valoarea 0 în *x*), vom spune că *y* este un **succesor** al lui *x* în timp ce *x* este un **predecesor** al lui *y*. T fiind un arbore, orice nod exceptând rădăcina are un singur predecesor şi poate avea mai mulţi succesori sau nici unul.

Bineînţeles că arborele T nu există de la început. La start el se reduce la rădăcina θ şi apoi, pe măsură ce algoritmul înaintează, câştigă noi noduri corespunzătoare deplasărilor care au loc în graful G. O dată ajunşi într-un nod x, numai 2 mişcări sunt permise:

- o mişcare „înainte", către un succesor *y* al lui *x*. Dacă această înaintare este recomandată de algoritm şi se execută, atunci *y* şi arcul (*x, y*) devin elemente ale arborelui T.
- o mişcare "înapoi" către unicul predecesor *z* al lui *x* în arborele T. Dacă această manevră este executată, nodul *x* devine nod **terminal** al arborelui T (adică fără succesori). Vom spune că *x* devine un **nod mort**. Nodurile lui T care nu sunt declarate 'moarte' sunt **noduri vii**.

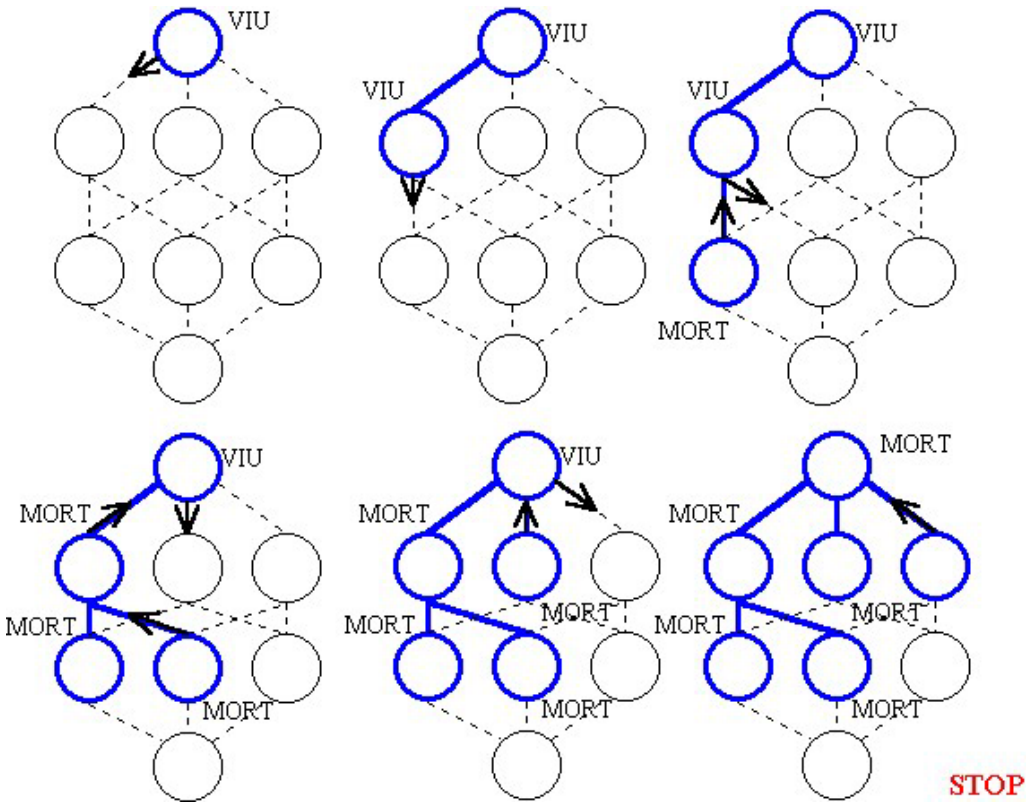


Figura 4 Formarea arborelui T în graful soluţiilor unei probleme cu 3 variabile.

Legendă: ——— elementele arborelui T


graful suport G
întoarcerile şi noile intenţii de înaintare

Prin urmare, un nod poate fi vizitat de mai multe ori dar o dată declarat mort prin el nu se mai trece. Arborele T nu cuprinde în mod necesar toate nodurile lui G decât în cazul unei enumerări complete.

2.4.2 Prezentarea algoritmului

Iniţializăm o variabilă z_{CMB} şi o locaţie x_{CMB} (sub forma unui vector cu *n* componente) care reţin, în orice moment al derulării algoritmului, maximul valorii funcţiei obiectiv pe mulţimea soluţiilor admisibile efective generate până în acel moment, respectiv soluţia admisibilă care realizează acest maxim.

Să presupunem că în procesul de examinare a nodurilor grafului G am ajuns pentru prima dată în nodul corespunzător unei soluţii x. Fie Ω(x) mulţimea formată din *x* şi toţi descendenţii lui *x* în G. Altfel spus: $\mathbf{\Omega(x) = \{y \in \Omega \mid y_i = x_i = 1, (\forall)\ i \in I(x)\}}$.

Deoarece cj≤0, (∀) j∈N este clar că $\mathbf{f(x) = max(f(y)),\ y \in \Omega(x)}$.

Sunt posibile două cazuri:

1. $\mathbf{f(x) = \sum_{i \in I(x)} c_i \le z_{CMB}}$

   În această situaţie este inutil să mai inspectăm soluţiile din Ω(x), deoarece nici una (şi în particular cele admisibile) nu oferă funcţiei *f* o valoare maximă mai bună decât cea oferită de ce mai bună soluţie admisibilă găsită până acum. În termenii grafului G este inutil să ne deplasăm din *x* către oricare din descendenţii săi. În consecinţă nodul *x* se declară MORT (se abandonează) şi ne întoarcem în unicul predecesor al lui x în arborele T.

2. $\mathbf{f(x) > z_{CMB}}$

   Avem de examinat două situaţii:

   2.1 $\boldsymbol{x \in A}$. Acest lucru poate fi probat calculând S(x)=b-Ax; dacă S(x)≥0 ⇒ x∈*A*; actualizăm:

   $\mathbf{x_{CMB} = x,\quad z_{CMB} = f(x).}$

   Nodul *x* se declară MORT; ne întoarcem în unicul predecesor al lui *x* în arborele T.

   2.2 $\boldsymbol{x \notin A}$ (echivalent cu S(x)≤0).

   Pentru moment nu avem nici un motiv să credem că elementul optimal x^* nu s-ar afla printre descendenţii lui *x* în G, adică în Ω(x). Deoarece x∉*A*, x^*, dacă ar fi în Ω(x), ar diferi de *x* prin cel puţin o valoare 1 corespunzătoare unei variabile care în *x* are valoarea 0 (corespunzătoare unei variabile x_j cu j∈N-I(x)). Vom încerca să dăm peste x^* în mod progresiv, examinând mai întâi succesorii direcţi ai lui *x*, apoi succesorii direcţi ai acestora ş.a.m.d.

Se impune deci schimbarea în 1 a valorii unei variabile x_k care are în *x* valoarea 0. În termenii grafului G, schimbarea din 0 în 1 a valorii variabilei x_k corespunde unei deplasări din *x* către un succesor al său, "pe direcţia k".

Dar cum alegem "direcţia de deplasare k"? Vom stabili mai întâi mulţimea R(x)⊂N a **direcţiilor recomandabile** de deplasare din soluţia *x*. Pentru aceasta identificăm *direcţiile nerecomandabile* de deplasare din *x*, care sunt de mai multe tipuri:

1. direcţiile *i* corespunzătoare variabilelor x_i cu valorea 1 în *x*. Acestea formează mulţimea $\mathbf{I_{(x)}}$.
2. direcţiile $j \in N$ anterior examinate. Acestea formează mulţimea $\mathbf{J_{(x)}}$. Deoarece suntem pentru prima dată în nodul *x*, $J_{(x)} = \varnothing$.
3. direcţiile j neexaminate ($\Leftrightarrow j \in N\text{-}I_{(x)} \cup J_{(x)}$)care conduc la valori ale obiectivului *f* ce nu depăşesc z_{CMB}: $f(x) + c_j \le z_{CMB}$. Mulţimea lor se va nota $\mathbf{K_{(x)}}$.
4. direcţiile *j* diferite de cele descrise anterior care conduc la soluţii *y* neadmisibile ca şi *x* dar mai proaste decât *x* în sensul că pentru toţi $i \in M$ pentru care $s_i(x)<0$ avem $\mathbf{s_i(y) \le s_i(x)}$.

   Pentru a caracteriza o asemenea direcţie *j* să observăm că soluţia *y* corespunzătoare se obţine din soluţia *x* acordând valorea 1 variabilei x_j. Prin urmare, $\mathbf{s(y)=s(x)-A_j}$ unde s-a notat cu A_j coloana *j* a matricii A. Rezultă de aici că $s_i(y) \le s_i(x) \Leftrightarrow a_{ij} \ge 0$. Deci direcţiile *j* din categoria 4 se caracterizează prin $a_{ij} \ge 0$ pentru toţi acei $i \in M$ pentru care $s_i(x)<0$. Mulţimea lor se va nota cu $\mathbf{L_{(x)}}$.

În final obţinem:

$$\mathbf{R_{(x)}=N\text{-}I_{(x)} \cup J_{(x)} \cup K_{(x)} \cup L_{(x)}}$$

- Dacă $R_{(x)}=\varnothing$ nodul *x* se declară MORT şi ne întoarcem în predecesorul lui *x* în arborele T.
- Dacă $R_{(x)} \neq \varnothing$ vom aplica un nou test care (dacă nu este verificat) arată că $\Omega(x)$ nu conţine nici o soluţie admisibilă a lui (P). Aceasta nu înseamnă că dacă testul este trecut atunci $\Omega(x) \cap A \neq \varnothing$ (testul este numai necesar, nu şi suficient)!

Pentru formularea testului sunt necesare câteva pregătiri:
Evaluăm ecartul S(y)=b-Ay pentru y variind în $\Omega(x)$:

$$s_i(y) = b_i - \sum_{j \in N} a_{ij}y_j = b_i - \sum_{i \notin R(x)} a_{ij}y_j - \sum_{j \in R(x)} a_{ij}y_j = s_i(x) - \sum_{j \in R(x)} a_{ij}y_j .$$

$s_i(y)$ va fi **maxim** dacă fiecărei variabile y_j, $j \in R_{(x)}$ îi atribuim valoarea $y_j = \begin{cases} 1 \text{ dacă } a_{ij}<0 \\ 0 \text{ dacă } a_{ij} \ge 0 \end{cases}$

Rezultă că:

$$\max_{y \in \Omega(x)} s_i(y) = s_i(x) - \sum_{j \in R(x)} (a_{ij})^- .$$

Reamintim că pentru un număr real *a* s-a notat cu $(a)^- = \begin{cases} a, \text{ dacă } a < 0 \\ 0, \text{ dacă } a \ge 0 \end{cases}$, $(a)^-$ fiind partea negativă a numărului *a*.

Este clar că dacă $i \in M$ este un indice pentru care $s_i(x)<0$ şi $\Sigma(a_{ij})^- > s_i(x)$ atunci $s_i(y)<0$, $(\forall)y \in \Omega(x)$ altfel spus Ω nu conţine soluţii admisibile.

Cu aceste pregătiri, suntem în măsură să formulăm testul anunţat:

TEST: Pentru indicii $i \in M$ pentru care $s_i(x)<0$ considerăm inegalitatea

$$\sum_{j \in R(x)} (a_{ij})^- \le s_i(x).$$

Dacă cel puțin una din aceste inegalități nu este adecvată atunci nodul *x* se declară MORT şi se revine la predecesorul său în arborele T.

În cazul în care testul T este verificat şi $R_{(x)}$ are cel puțin două elemente direcția de deplasare k (adică variabila x_k a cărei valoare se schimbă din 0 în 1) se determină astfel:

Fie $j \in R_{(x)}$. Atribuind lui x_j valoarea 1 obținem din *x* o nouă soluție *y* pentru care:

$$s_i(y) = s_i(x) - a_{ij}, \ (\forall)\ i \in M$$

Dacă $s_i(y) \geq 0$, $(\forall)\ i \in M$ atunci *y* va fi o **soluție admisibilă**. Altfel, numărul negativ

$$s_i(x)-a_{ij}$$

reprezintă o măsură a nesatisfacerii restricției *i* de către noua soluție *y*. Introducem o măsură a nesatisfacerii tuturor restricțiilor de către *y* prin formula:

$$v_j(x) = \sum_{i \in M} (s_i(x) - a_{ij})^-$$

Este clar că întotdeauna $v_j(x) \leq 0$ şi că $v_j(x)=0$ dacă şi numai dacă $y \in A$.

Este normal atunci să alegem ca direcție de deplasare indicele $k \in R_{(x)}$ pentru care avem:

$$v_k(x) = \max [v_j(x)], \quad j \in R_{(x)}$$

În cazul în care că maximul din această formulă nu este unic are prioritate indicele *k* pentru care coeficientul c_k este cel mai mare.

O dată aleasă direcția de deplasare k transferăm indicele *k* din $R_{(x)}$ în $J_{(x)}$: *k* este o direcție examinată şi la reîntoarcerea în nodul *x* într-o fază ulterioară nu trebuie să ne mai deplasăm pe această direcție.

În continuare trecem la examinarea soluției *y* dedusă din *x* prin atribuirea $x_k = 1$. Arborele T se completează cu nodul *y* şi cu arcul (*x, y*). Astfel pentru *y*, nodul *x* apare în arborele T ca predecesor. Examinarea nodului *y* se face după aceleaşi instrucțiuni folosite la examinarea nodului *x*.

Din descrierea algoritmului a rezultat că un nod poate fi „vizitat" de mai multe ori. La fiecare revenire în nodul *x* este necesar să actualizăm lista $K_{(x)}$ deoarece este posibil ca între timp z_{CMB} să fi crescut. Într-adevăr pentru un indice *j*, anterior „recomandabil", pentru care inegalitatea $f(x) + x_j > (z_{CMB})_{vechi}$ era adecvată, este posibil ca $f(x) + x_j \leq (z_{CMB})_{nou}$ deoarece $(z_{CMB})_{nou} > (z_{CMB})_{vechi}$. Prin urmare indicele *j* nu mai este recomandabil în momentul în care se pune problema deplasării din *x* către un nou succesor diferit de toți cei deja examinați.

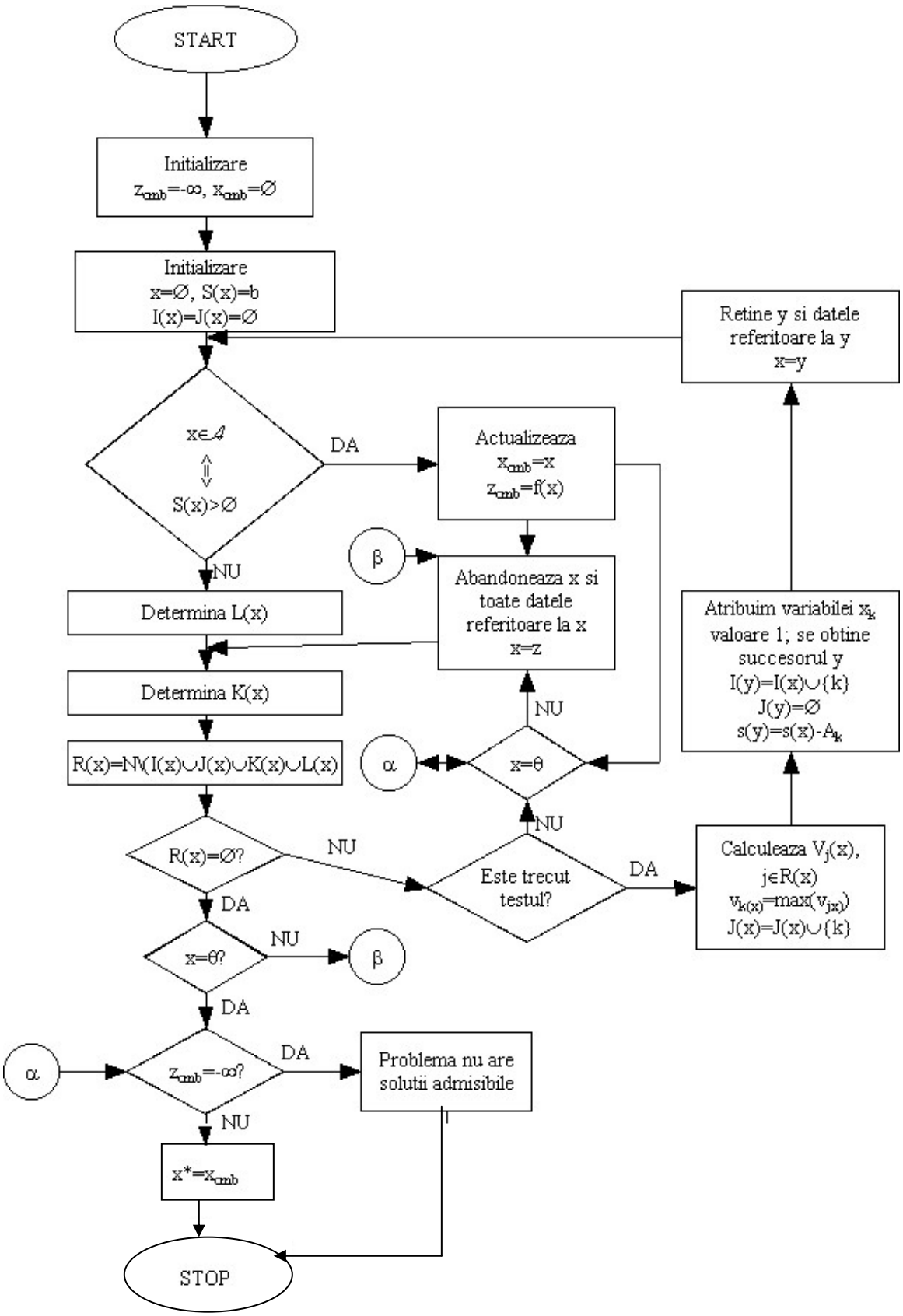
Recapitulând la fiecare revenire în nodul *x* mulțimile de direcții nerecomandabile J(x) şi K(x) au tendința de „creştere". Având în vedere că I(x) şi L(x) nu se modifică, la fiecare revenire în *x*, R(x) descreşte cu cel puțin un element. În mod clar rezultă că după un număr finit de asemenea reveniri R(x) devine vidă şi nodul *x* (împreună cu toți descendenții săi examinați şi neexaminați) este abandonat.

Procesul se încheie în momentul în care s-a revenit în rădăcina θ a arborelui T din care s-a plecat şi mulțimea R(θ) actualizată a devenit vidă. În acel moment toate soluțiile din Ω au fost testate fie explicit (adică efectiv generate) fie implicit.

Dacă, în final, $z_{CMB} = -\infty$ atunci (P) nu are soluții admisibile $A = \varnothing$. În caz contrar, $x^* = x_{CMB}$.

Schema logică a algoritmului expus foloseşte pe lângă variabilele x_{CMB}, z_{CMB} şi variabilele:

- *x* cu semnificația de nod curent (în care a ajuns procesul de enumerare);
- *y* succesorul lui *x* rezultat după alegerea direcției de deplasare din *x*;
- *z* predecesorul lui *x* din arborele T.


START
Initializare
$z_{cmb}=-\infty$, $x_{cmb}=\emptyset$
Initializare
$x=\emptyset$, $S(x)=b$
$I(x)=J(x)=\emptyset$
Retine y si datele referitoare la y
$x=y$
$x\in A$
$\Updownarrow$
$S(x)>\emptyset$
DA
Actualizeaza
$x_{cmb}=x$
$z_{cmb}=f(x)$
NU
β
Abandoneaza x si toate datele referitoare la x
$x=z$
Determina L(x)
Determina K(x)
$R(x)=N\setminus(I(x)\cup J(x)\cup K(x)\cup L(x)$
α
$x=\theta$
NU
Atribuim variabilei x_k valoare 1; se obtine succesorul y
$I(y)=I(x)\cup\{k\}$
$J(y)=\emptyset$
$s(y)=s(x)-A_k$
$R(x)=\emptyset$?
NU
Este trecut testul?
NU
DA
Calculeaza $V_j(x)$, $j\in R(x)$
$v_{k(x)}=\max(v_{jx})$
$J(x)=J(x)\cup\{k\}$
DA
$x=\theta$?
NU
β
DA
α
$z_{cmb}=-\infty$?
DA
Problema nu are solutii admisibile
NU
$x^*=x_{cmb}$
STOP

Exemplu numeric

Să se rezolve problema de programare bivalentă:

$$\begin{cases} -x_1+3x_2-5x_3-x_4+4x_5 \le -2 \\ 2x_1-6x_2+3x_3+2x_4-2x_5 \le 0 \\ x_2-2x_3+x_4+x_5 \le -1 \\ (\max)f = -5x_1-7x_2-10x_3-3x_4-x_5 \\ x_j \in \{0,1\};\ j=1,\ldots,5 \end{cases}$$

Rezolvare:
Punem în evidență matricea A a coeficienților sistemului de restricții. În partea dreaptă vor fi înscrise succesiv ecarturile diferitelor soluții testate.

A_1	A_2	A_3	A_4	A_5	$S(x^1)$	$S(x^2)$	**$S(x^3)$**
-1	3	-5	-1	-4	-2	3	**8**
2	-6	3	2	-2	0	-3	**0**
0	1	-2	1	1	-1	1	**1**

Inițializare: $x_{CMB} = \emptyset$, $z_{CMB} = -\infty$
Soluția curentă: $x^1 = \theta = (0,0,0,0,0)$

Iterația 1:
$f(x^1)=0 > z_{CMB}$
$S(x^1)$ nu are componentele $\ge 0 \Rightarrow x^1 \notin A$
$I(x^1)=\emptyset$, $J(x^1)=\emptyset$, $L(x^1)=\{2,5\}$, $K(x^1)=\emptyset$, $R(x^1)=\{1,3,4\}$
Aplicăm testul de admisibilitate (T):
$S_1(x^1) = -2<0 \quad \sum_{j\in R(x^1)} (a_{1j})^- = -1-5-1=-7<-2;$
$S_3(x^1) = -1<0 \quad \sum_{j\in R(x^1)} (a_{1j})^- = 0-2+0=-2.$
Testul (T) este trecut.
Deoarece avem 3 direcții recomandabile, calculăm $v_j(x^1)$, $j\in R(x^1)$:

- $v_1(x^1)= \sum_{i\in M} (S_i(x^1)-a_{1j})^- = (-2-(-1))^- + (0-2)^- + (-1-0)^- = -1-2-1 = -4$
- $v_3(x^1)= (-2-(-5))^- + (0-3)^- + (-1-(-2))^- = 0-3+0 = -3$
- $v_4(x^1)= (-2-(-1))^- + (0-2)^- + (-1-1)^- = -1-2-2 = -5$

Maximul valorilor calculate este $v_3(x^1)$
Atribuim $x_3 = 1$ şi obținem soluția $x^2 = (0,0,1,0,0)$
Transferăm indicele 3 în $J(x^1)$: $J(x^1)=\{3\}$.

Iterația 2:
$f(x^2) = f(x^1)+c_3=-10>z_{cmb}$
$S(x^2)=S(x^1)-A_3$ nu are componentele $\ge 0 \Rightarrow x^2 \notin A$.
$I(x^2)=\{3\}$, $J(x^2)=\emptyset$, $L(x^2)=\{1,4\}$, $K(x^2)=\emptyset$, $R(x^2)=\{2,5\}$
Testul (T) este trecut (-8<-3)

Calculăm $v_2(x^2) = 0$; $v_5(x^2) = -2$; maximul este $v_2(x^2) \Rightarrow$ modificăm $x_2=1$ şi obținem soluția $x^3 = (0,1,1,0,0)$

Transferăm indicele 2 în $J(x^2)$: $J(x^2)=\{2\}$

Iterația 3:

$f(x^3)= f(x^2)+c_2=-10-7 = -17$

$S(x^3)=S(x^2)-A_2\geq 0 \Rightarrow x^3 \in A$.

Actualizăm $\mathbf{x_{CMB} = x^3, z_{CMB} = -17}$

Ne întoarcem în predecesorul lui x^3, soluția x^2.

Iterația 4:

Suntem în nodul x^2 pentru a doua oară. Actualizăm $K(x^2) = \varnothing \Rightarrow R(x^2) = \{5\}$

Testul (T) nu este trecut (-2 nu este ≤ decat -3).

Ne întoarcem în predecesorul x^1

Iterația 5:

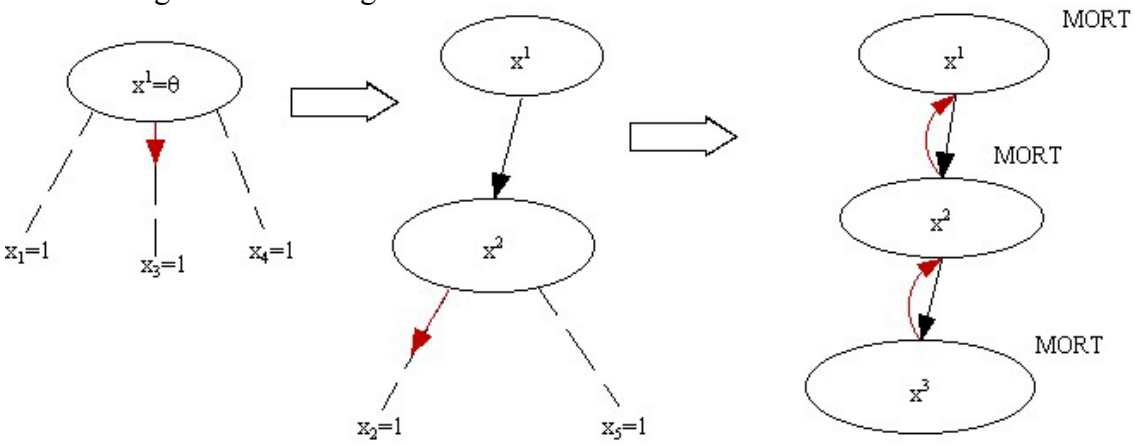
Actualizăm $K(x^1) = \varnothing \Rightarrow R(x^1) = \{1,4\}$

Testul (T) nu este trecut.

STOP deoarece suntem în rădăcina arborelui T.

Soluția optimă a problemei este $x^* = (0,1,1,0,0)$, $f(x^*) = -17$

Arborele T generat de-a lungul rezolvării:



Legendă:

Intențiile de deplasare

– – – O parte din muchiile grafului G (care are 2^5=32 de noduri) şi anume cele care corespund direcțiilor recomandabile.

În exemplul considerat, din cele 32 de soluții ale problemei au fost efectiv generate numai trei.

Observații:

Anterior a fost prezentată versiunea originală a algoritmul aditiv, care între timp a primit o serie de perfecționări sub forma unor teste suplimentare. Prin aceste teste se reduce numărul soluțiilor efectiv generate crescând în schimb efortul de calcul.