C A P I T O L U L I

PROGRAMARE ÎN NUMERE ÎNTREGI

§ 1. Specificul programării în numere întregi

Este binecunoscut faptul că **modelarea prin programare liniară** reprezintă un mijloc puternic şi eficace pentru studiul proceselor economice în vederea îmbunătăţirii performanţelor acestora.

O proprietate foarte importantă a variabilelor de decizie, dintr-un program liniar era aceea că, o dată cu două valori **permise**, puteau lua orice altă valoare **intermediară**; această proprietate „**de a putea varia continuu**" era esenţială în fundamentarea metodelor de determinare a soluţiilor optime.

Nu puţine sunt situaţiile practice, modelate cu ajutorul programării liniare în care unele variabile de decizie nu au sens economic decât dacă au numai valori **întregi**.

De exemplu, dacă outputul unei activităţi este măsurat în unităţi **indivizibile** este clar că valorile permise ale variabilei care indică nivelul activităţii respective trebuie să fie numere întregi. La fel, repartizarea personalului muncitor, al utilajelor sau al mijloacelor de transport pe diverse activităţi trebuie exprimată tot prin cantităţi întregi.

Exemplul 1.1 [4] Un fermier are nevoie de 107 funţi de îngrăşământ pe care îl poate procura fie în saci de 35 funţi la preţul de 14 $ sacul, fie în saci de 24 funţi a 12 $ fiecare. Obiectivul său este de a cumpăra cantitatea necesară de îngrăşământ la cel mai mic cost.

Notând cu x_1, x_2 numărul sacilor de 35 funţi, respectiv 24 funţi cumpăraţi de fermier, modelul situaţiei descrise arată astfel:

$$\begin{cases} \min f = 14\,x_1 + x_2 \quad \leftarrow \text{costul sacilor cumpăraţi} \\ 35\,x_1 + 24\,x_2 \geq 107 \quad \leftarrow \text{asigurarea achiziţionării cantităţii de îngrăşământ dorite} \\ x_1, x_2 \geq 0, \text{ întregi} \end{cases}$$

În modelul rezultat, condiţia „x_1, x_2 întregi" înseamnă pur şi simplu că fermierul nu poate cumpăra o jumătate sau o treime de sac; ori cumpără un sac întreg ori nu. Fără această condiţie avem de a face cu o problemă uzuală de programare liniară.

Poate mai importante sunt situaţiile în care trebuie luate decizii de tip „**da sau nu**" ca de exemplu:

- se poate aproba realizarea unui anumit proiect de dezvoltare ?
- se poate iniţia o activitate implicând un anumit cost fix de pregătire ?
- se poate amplasa o facilitate (unitate productivă, depozit, magazin) într-un anumit loc ?

Existând doar două posibilităţi, de a amplasa sau de a respinge, asemenea decizii pot fi reprezentate prin variabile cu numai două valori permise, 0 sau 1:

1, dacă decizia este **afirmativă**;
0, dacă decizia este **negativă**.

Exemplul 1.2 [7] În cadrul unui proiect mai general de extindere şi dezvoltare, conducerea unei firme studiază oportunitatea construirii unei noi fabrici fie în oraşul A, fie în oraşul B, poate chiar în amândouă şi a cel mult un depozit într-unul din cele două oraşe, alegerea amplasamentului fiind însă condiționată de construirea unei fabrici în localitatea respectivă. În tabelul 1.1. sunt indicate: valoarea prezentă netă a diferitelor alternative, capitalul necesar acestor investiții şi capitalul disponibil pentru întregul proiect de dezvoltare.

milioane $

Nr. crt.	Alternativa decizională	Variabila de decizie	Valoarea prezentă netă	Capitalul necesar
1	Construirea unei fabrici în A	x_1	9	6
2	Construirea unei fabrici în B	x_2	5	3
3	Construirea unui depozit în A	x_3	6	5
4	Construirea unui depozit în B	x_4	4	2
Capital disponibil				10

Tabelul 1.1.

unde $x_j = \begin{cases} 1 \text{ dacă alternativa decizională j se aprobă} \\ 0 \text{ dacă alternativa decizională j se respinge} \end{cases}$ $j = 1, 2, 3, 4.$

Condițiile specificate în proiectul de dezvoltare se formalizează astfel:

- cel mult un depozit poate fi construit: $x_3 + x_4 \leq 1$

(deoarece x_3, x_4 nu pot lua decât valorile 0 sau 1, se elimină posibilitatea construirii a două depozite în ambele oraşe deoarece $x_3 = 1$, $x_4 = 1$ nu satisfac inegalitatea !)

- depozitul nu poate fi construit în lipsa unității productive: $x_3 \leq x_1$, $x_4 \leq x_2$

(este clar că $x_1 = 0$ implică cu necesitate $x_3 = 0$, etc.)

- încadrarea în capitalul disponibil: $6 x_1 + 3 x_2 + 5 x_3 + 2 x_4 \leq 10$
- maximizarea valorii prezente nete totale a obiectivelor care se vor realiza:

$$9 x_1 + 5 x_2 + 6 x_3 + 4 x_4 \rightarrow \max$$

Rezultă următorul model matematic:

$$\begin{cases} \max f = 9 x_1 + 5 x_2 + 6 x_3 + 4 x_4 \\ : 6 x_1 + 3 x_2 + 5 x_3 + 2 x_4 \leq 10 \\ x_3 + x_4 \leq 1 \\ - x_1 + x_3 \leq 0 \\ - x_2 + x_4 \leq 0 \\ \begin{cases} x_j \leq 1 \\ \text{întregi} \Leftrightarrow x_J \in \{0, 1\}\ j = 1, 2, 3, 4 \\ x_j \geq 0 \end{cases} \end{cases}$$

Observație: Dacă s-ar fi pus condiția construirii unui singur depozit într-unul din cele două oraşe, atunci inegalitatea $x_3 + x_4 \leq 1$ trebuie schimbată în $x_3 + x_4 = 1$.

Pentru cele ce urmează este necesar să precizăm câțiva termeni.

• variabilă **continuă** ≡ variabilă care, o dată cu două valori permise poate lua orice altă valoare **intermediară**;

• variabilă **întreagă** ≡ variabilă care nu poate lua decât valori numere **întregi**;

• variabilă **bivalentă (booleană)**≡variabilă **întreagă** care nu poate lua decât valorile 0 sau 1;

• problemă de **programare în numere întregi** (pe scurt **program întreg**) ≡ problemă de programare **liniară** care utilizează una sau mai multe variabile întregi. Problema se zice **totală** dacă toate variabilele sale sunt întregi şi **mixtă** dacă utilizează simultan şi variabile continue şi variabile întregi;

• problemă de **programare bivalentă** (sau **program bivalent**) – totală sau mixtă ≡ problemă care utilizează variabile bivalente.

• Exemplul 1.2. arată că utilizarea variabilelor întregi aduce un plus de **flexibilitate** în modelarea unor situaţii practice. Această flexibilitate „costă" însă destul de mult, deoarece programele întregi sunt mult mai greu de rezolvat decât programele liniare uzuale (adică în variabile continue). Fără a intra deocamdată în amănunte este suficient să amintim că dacă în prezent a devenit o obişnuinţă rezolvarea unor programe liniare cu mii de variabile continue (fireşte utilizând programe comerciale de calculator), un program cu mai puţin de 100 variabile întregi poate cauza mari dificultăţi !

Pentru evitarea complicaţiilor se poate aplica următoarea schemă de **rezolvare aproximativă** a programelor întregi.

Se ignoră condiţia ca variabilele să ia numai valori întregi şi se rezolvă programul „**relaxat**" care este un program liniar uzual. Două situaţii sunt posibile:

• soluţia optimă a problemei relaxate are toate componentele întregi; aceasta va fi, evident şi soluţia optimă a programului întreg original;

• unele componente ale soluţiei optime a programului relaxat sunt **fracţionare**. În această situaţie componentele fracţionare vor fi rotunjite – inferior sau superior – la valori întregi, în ideea că soluţia optimă „întreagă" este situată „în apropierea soluţiei optime fracţionare".

Operaţia de rotunjire trebuie astfel făcută încât rezultatul să fie o soluţie a problemei, adică să verifice restricţiile. **este firesc să acceptăm soluţia continuă prin rotunjire, cel puţin ca soluţie suboptimală**; în multe contexte practice acest lucru este justificat prin faptul că valorile permise variabilelor sunt suficient de mari astfel încât efectul rotunjirii să fie neglijabil.

Nu puţine sunt situaţiile în care strategia de rezolvare fie şi aproximativă prin rotunjirea soluţiei optime fracţionare nu este recomandabilă. Uneori se întâmplă ca numărul alternativelor de rotunjire să fie foarte mare implicând un volum apreciabil de calcule suplimentare pentru verificarea şi sortarea acestora. Pe de altă parte, pentru unele programe întregi în care variabilele iau valori destul de mici, ca de exemplu 0 sau 1, este posibil ca „distanţa" dintre optimul „întreg" şi cel „fracţionar" să fie atât de mare încât simpla rotunjire să nu ducă la o soluţie acceptabilă.

Exemplul 1.3 Pentru programul întreg din exemplul 1.1. soluţia optimă a programului relaxat:

$$\begin{cases} \min f = 14\,x_1 + 12\,x_2 \\ 35\,x_1 + 24\,x_2 \geq 107 \\ x_1 \geq 0,\ x_2 \geq 0 \end{cases}$$

are componentele:

$$x_1^* = 3\frac{2}{35}, \quad x_2^* = 0; \qquad f\left(x^*\right) = 42\frac{2}{5}$$

Sunt evidente următoarele rotunjiri care conduc la soluții admisibile:

$$x_1' = 4, \quad x_2' = 0 \quad ; f\left(x'\right) = 56; \quad 4 \times 35 = 140 \text{ funti}$$
$$x_1'' = 3, \quad x_1'' = 1 \quad ; f\left(x''\right) = 54; \qquad 3 \times 35 + 1 \cdot 24 = 129 \text{ funti}$$

A doua soluție - mai bună – este totuşi „departe” de soluția optimă întreagă:

$$x_1^0 = 1, \quad x_2^0 = 3 \quad ; f\left(x^0\right) = 50; \quad 1 \cdot 35 + 3 \cdot 24 = 107 \text{ funti}$$

§ 2. Domenii de aplicare ale programării în numere întregi. Exemple

Situațiile practice a căror modelare necesită utilizarea variabilelor întregi sunt extrem de numeroase şi exemplele următoare nu acoperă nici pe departe această varietate. După cum vom vedea în următorul capitol, **situațiile combinatoriale** pot fi modelate – cel puțin în principiu ca probleme de programări în numere întregi.

2.1. Problema monezilor Cum poate fi plătită o sumă de bani astfel încât:

- numărul tipurilor valorice de monezi utilizate la plată să nu depăşească o limită dată;
- numărul total al monezilor necesare plății să fie minim.

Pentru formalizare avem nevoie de următoarele notații:

S = suma de plată,
n = numărul total al tipurilor valorice de monezi disponibile pentru plată;
p = numărul maxim de tipuri valorice de monezi ce pot fi utilizate la plata sumei S;
$a_j \equiv$ valoarea monezii de tip j;
$m_j \equiv$ numărul monezilor de tip j disponibile în casă.

Introducem variabilele:

x_j = numărul monezilor de tip j utilizate la plata sumei S;

Rezultă următorul program întreg

$$\begin{cases} (\min) f = \sum_{j=1}^{n} x_j \\ \sum_{j=1}^{n} a_j x_j = S \\ 0 \le x_j \le m_j y_j \qquad j=1, \ldots n \\ \sum_{j=1}^{n} y_j \le p \\ x_j \ge 0 \text{ întregi; } y_j \in \{0, 1\} \qquad j = 1, \ldots n \end{cases}$$

2.2. Alegerea proiectelor de investiții O firmă este interesată în mai multe proiecte de investiții pe care ar putea să le realizeze în câțiva ani dar, din cauza bugetului limitat, va trebui să se limiteze la o parte din ele. Proiectul j aduce firmei – în caz de finalizare – un profit estimat la c_j dolari j = 1, ... n şi necesită investiții anuale în valoare de a_{ij} dolari i = 1, ... m. Capitalul disponibil pentru anul i este b_i. Se pune problema alegerii acelor proiecte care să aducă firmei un profit total maxim cu condiția nedepăşirii capitalului disponibil anual.

Pentru formalizare introducem variabile bivalente

$$x_j = \begin{cases} 1 \text{ dacă proiectul j este acceptat} \\ 0 \text{ dacă proiectul j este respins} \end{cases}$$

Obținem programul bivalent:

$$\begin{cases} (\max) f = \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad i = 1,...m \\ x_j \in \{0, 1\} \qquad j = 1, ... n \end{cases}$$

Observație Utilizarea variabilelor bivalente ne permite să modelăm o serie de situații speciale. De exemplu:

- cel mult unul din primele trei proiecte poate fi acceptat: $x_1 + x_2 + x_3 \le 1$;
- numai unul din primele trei proiecte poate fi acceptat: $x_1 + x_2 + x_3 = 1$;
- nu poate fi acceptat proiectul 2 dacă proiectele 1 şi 4 sunt respinse: $x_2 \le x_1 + x_4$

etc.

2.3. Elaborarea programelor de producție cu costuri de pregătire

Să considerăm programul liniar

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad i = 1,...m \\ x_j \ge 0 \qquad j = 1, ... n \\ (\min) f = \sum_{j=1}^{n} c_j x_j \end{cases}$$

în care $x_1, x_2, ... x_n$ reprezintă nivelul unor activități productive iar $c_1, c_2, ...c_n$ sunt costurile unitare aferente. Ipoteza uzuală de liniaritate presupune **proporționalitatea directă** între costul unei activități şi nivelul la care este operată activitatea respectivă. Există însă situații în care demararea unei activități necesită un cost de pregătire, independent de nivelul la care activitatea va fi operată. Costul activității j va avea forma:

$$c_j(x_j) = \begin{cases} 0 \text{ dacă } x_j = 0 \\ q_j + c_j x_j \text{ dacă } x_j > 0 \end{cases}$$

Pentru a încorpora în modelul matematic anterior aceste elemente introducem următoarele bivalente

$$y_j = \begin{cases} 1 \text{ dacă activitatea j va fi operativă la un nivel } x_j > 0 \\ 0 \text{ în caz contrar} \end{cases}$$

Rezultă programul **mixt:**

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j = b_i & i = 1,...m \\ x_j \le m_j y_j & j = 1, \dots n \\ x_j \ge 0,\ y_j \in \{0,1\} & j = 1, \dots n \\ (\min) f = \sum_{j=1}^{n} q_j y_j + \sum_{j=1}^{n} c_j x_j \end{cases}$$

unde m_j reprezintă o limită superioară a nivelului x_j al activității j

2.4 Stabilirea programului de zbor al avioanelor unei companii de transporturi aeriene

O companie de transporturi aeriene are în dotare mai multe avioane de diferite tipuri pe care îşi impune să le utilizeze cât mai bine pe rutele de transport utilizate de această companie.

Datele problemei:

m = numărul tipurilor de avioane;
n = numărul rutelor servite de companie;
c_i = capacitatea maximă de transport a unui avion de tipul i (nr. locurilor disponibile);
d_i = numărul avioanelor de tipul i disponibile;
b_j = numărul clienților potențiali pe ruta j;
p_j = profitul companiei pe călător transportat pe ruta j;
a_{ij} = numărul de zboruri pe care un avion de tipul i îl poate face pe ruta j într-o perioadă (zi, decadă, lună);
q_{ij} = costul operării unui zbor al unui avion de tipul i pe ruta j

Obiectiv:
repartizarea avioanelor pe rute şi stabilirea numărului de zboruri ce trebuie efectuate la nivelul unei perioade astfel încât cererile de transport să fie rațional făcute la cel mai mic cost.

Variabile modelului*:*

x_{ij} = numărul avioanelor de tipul i rezultate pe ruta j;
y_{ij} = numărul de zboruri planificat a se realiza cu avioane de tipul i pe ruta j într-o perioadă.

Funcția obiectiv. Avem două cazuri:

1) Se au în vedere numai costurile de operare ale diferitelor zboruri:

$$(\min) f_1 = \sum_{j=1}^{m} \sum_{j=i}^{n} q_{ij} y_{ij} \qquad (2.1.)$$

2) La costurile de operare se adaugă şi costurile de „penalizare” pentru locurile „libere”. Pe ruta j numărul locurilor libere este dat de diferența $\sum_{i=1}^{m} c_i y_{ij} - b_i$; astfel că expresia cheltuielilor de penalizare – identificat cu profitul pierdut – este:

$$\sum_{j=1}^{n} p_j \left(\sum_{i=1}^{m} c_i y_{ij} - b_i \right) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_i p_j y_{ij} - \sum_{j=1}^{n} p_j b_j$$

Cheltuielile totale vor avea expresia:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} q_{ij} y_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{n} c_i p_j y_{ij} - \sum_{j=1}^{n} p_j b_j = \sum_{i=1}^{m} \sum_{j=1}^{n} (q_{ij} + c_i p_j) y_{ij} - \sum_{j=1}^{n} p_j b_j$$

astfel că funcția obiectiv devine:

$$(\min) f_2 = \sum_{i=1}^{m} \sum_{j=1}^{n} (q_{ij} + c_i p_j) y_{ij} \qquad (2.2.)$$

expresia $\sum_{j=1}^{n} p_j b_j$ fiind o constantă ce nu afectează optimizarea.

Restricții:

- încadrarea aparatelor repartizate în numărul disponibil de avioane

$$\sum_{j=1}^{n} x_{ij} \le d_i \qquad i = 1, \ldots m \qquad (2.3.)$$

- condiția de realizare a zborurilor planificate în funcție de numărul avioanelor repartizate

$$y_{ij} \le a_{ij} x_{ij} \qquad i = 1, \ldots m; j = 1, \ldots m \qquad (2.4.)$$

- satisfacerea cererilor de transport:

$$\sum_{i=1}^{m} c_i y_{ij} \ge b_j \quad j = 1, \ldots m \qquad (2.5.)$$

Condiții impuse explicit variabilelor:

$$x_{ij} \ge 0, y_{ij} \ge 0 \quad \text{întregi } i = 1, \ldots m; j = 1, \ldots n \qquad (2.6.)$$

Ataşând condițiilor 2.3. – 2.6. funcțiile obiectiv (2.1.) şi (2.2.) se obțin două programe întregi totale.

§ 3. Particularitățile mulțimii soluțiilor admisibile ale unui program întreg

Să considerăm următorul **program întreg total** (P) împreună cu **programul relaxat** (PL) – obținut din (P) prin eliminarea cerinței ca variabilele să ia numai valori întregi:

Programul întreg (P)

$$\begin{cases} \max f = 3x_1 - x_2 \\ 3x_1 - 2x_2 \le 3 \\ 5x_1 + 4x_2 \ge 10 \\ 2x_1 + x_2 \le 5 \\ x_1, x_2 \ge 0 \\ x_1, x_2 \text{ întregi} \end{cases}$$

Programul relaxat (PL)

$$\begin{cases} (\max) f = 3x_1 - x_2 \\ 3x_1 - 2x_2 \le 3 \\ 5x_1 + 4x_2 \ge 10 \\ 2x_1 + 2x_2 \le 5 \\ x_1, x_2 \ge 0 \end{cases}$$

Ne propunem:

- să rezolvăm grafic cele două programe;
- să punem în evidență principalele proprietăți ale mulțimilor de soluții admisibile ale celor două programe. Analiza comparativă a acestor proprietăți ne va permite degajarea – cel puțin la nivel de principiu – a unor metode de rezolvare (exactă) a programelor întregi.

Terminologie, notații:

- soluție **admisibilă** ≡ ansamblu de valori numerice (nu neapărat întregi) care verifică restricțiile şi condițiile de nenegativitate;
- soluție **admisibilă întreagă** ≡ soluție admisibilă cu toate componentele întregi;
- A_{PL} ≡ mulțimea sluțiilor admisibile ale programului relaxat (PL);
- A_P ≡ mulțimea soluțiilor admisibile întregi ale programului (P);
- soluția optimă **fracționară** ≡soluția optimă a programului relaxat (PL), notată constant cu x*;
- optim **fracționar** ≡ valoarea f (x*) a funcției obiectiv în soluția optimă fracționară x*;
- soluția optimă **întreagă** ≡ soluția optimă a programului întreg (P), notată constant cu x^0;
- optim **întreg** ≡ valoarea f (x^0) a funcției obiectiv în soluția optimă întreagă x^0.

Vizualizăm mulțimile A_P şi A_{PL}

Reamintim că pentru reprezentarea grafică a mulțimii A_{PL} se fac următoarele operații:

- se reprezintă grafic dreptele: $d_1 \equiv 3\,x_1 - 2\,x_2 = 3$
  $d_2 \equiv 5\,x_1 + 4\,x_2 = 10$
  $d_3 \equiv 2\,x_1 + x_2 = 5$

şi se identifică semiplanele în care au loc cele trei restricții şi cele două condiții de nenegativitate.

A_{PL} va fi intersecția celor cinci semiplane puse în evidență mai sus adică poligonul simplu haşurat DFGH

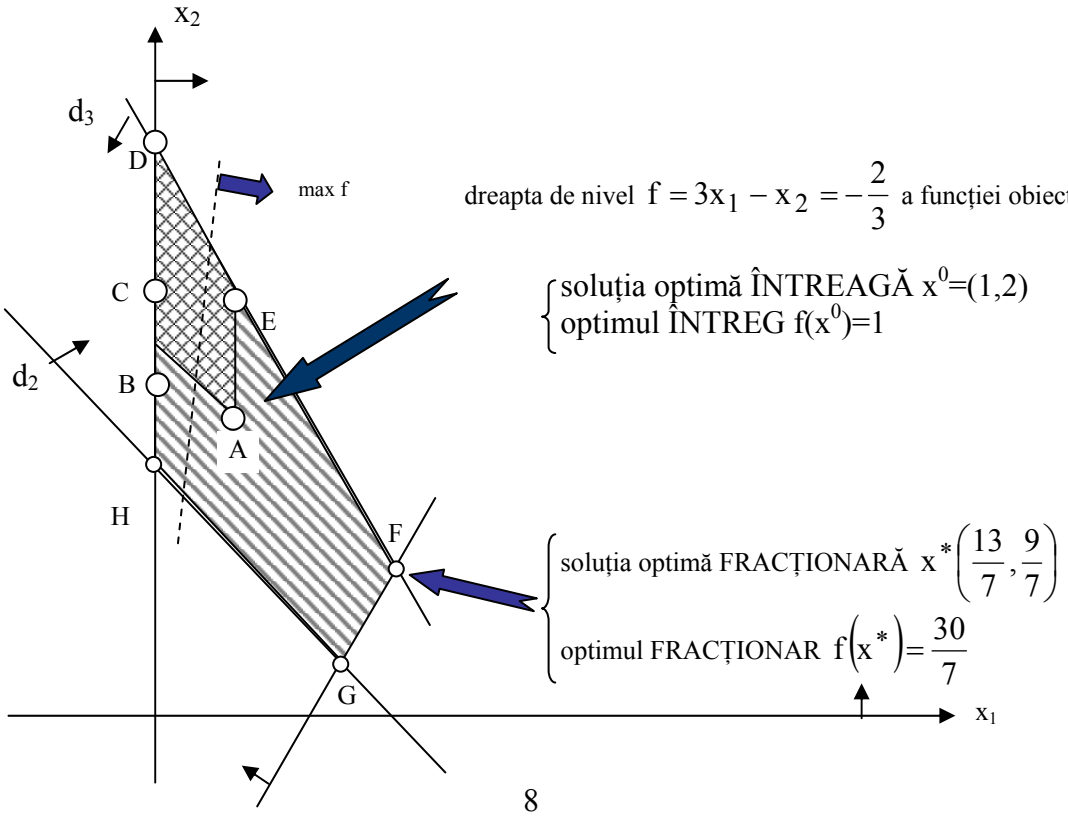

Fig. 3.1.

Pentru a găsi soluția optimă fracționară se reprezintă grafic o **dreaptă de nivel** a funcției obiectiv şi se stabileşte direcția de translatare a acesteia corespunzătoare maximizării. Rezultă $x^* = \left(\frac{13}{7}, \frac{9}{7}\right) \equiv$ vârful F din desen optimul fracționar având valoarea $f\left(x^*\right) = \frac{30}{7}$.

Prin simplă inspecție se constată că mulțimea soluțiilor admisibile întregi ale programului (P) se compune din cinci puncte:

$$A_P = \{A(1,2), B(0,3), C(0,4), D(0,5), E(1,3)\}$$

Funcția obiectiv f are valoarea maximă în A(1,2); prin urmare soluția optimă întreagă este $x^0=(1,2)$ iar optimul întreg are valoarea $f(x^0) = 1$.

Comparând mulțimile A_P şi A_{PL} se pot formula următoarele concluzii şi în cazul general:

1) A_{PL} este o mulțime în care fiecare punct este „înconjurat" de alte puncte din mulțimi **oricât de apropiate**. Ca urmare, teoria clasică a optimizării, bazată după cum se ştie pe posibilitatea efectuării unor deplasări infinitesimale în jurul unui punct, este **direct** aplicabilă.

Prin contrast, A_P este o mulțime **rară**, adică orice punct din ea posedă o vecinătate suficient de mică în care nu se mai află alte puncte din mulțime. Aplicarea teoriei amintite este în acest caz lipsită de sens.

2) A_{PL} este o mulțime **convexă** şi mai mult **poliedrală** având un număr **finit** de **vârfuri.** Aceste proprietăți, plus liniaritatea funcției obiectiv, ne asigură că soluția optimă fracționară se găseşte într-unul din vârfuri (conform teoremei centrale a programării liniare). Cercetarea sistematică a mulțimii vârfurilor lui A_{PL} cu ajutorul **algoritmului simplex** conduce la optimul fracționar într-un număr finit de paşi.

Cu rare excepții, soluția optimă întreagă – care este la urma urmei o soluție admisibilă problemei relaxate – se găseşte **în interiorul** mulțimii A_{PL} şi ca urmare nu este nici măcar generată de către algoritmul simplex.

3) Să considerăm **anvelopa convexă** a mulțimii A_P, adică cea mai mică mulțime convexă care conține A_P. În fig. 3.1. această anvelopă este reprezentată de poligonul dublu haşurat ABDE. Prin construcție vârfurile anvelopei sunt soluții admisibile întregi ale programului (P).

Dacă vom maximiza funcția obiectiv f pe anvelopa convexă a soluțiilor admisibile întregi vom regăsi soluția optimă întreagă x^0.

În concluzie, putem rezolva o problemă de programare în numere întregi ca o problemă de programare liniară uzuală cu condiția să ştim să descriem în limbaj de inecuații liniare anvelopa convexă a soluțiilor admisibile întregi.

În cazul studiat, această descriere este uşor de făcut: anvelopa ABDE se compune din soluțiile sistemului de inecuații

$$\begin{cases} 2x_1 + x_2 \le 5 \\ x_1 + x_2 \ge 3 \\ x_1 \le 1 \\ x_1 \ge 0,\ x_2 \ge 0 \end{cases}$$

În cazul general descrierea este practic imposibil de făcut. Totuşi am putea recupera ceva din această idee: adăugând la problema relaxată PL un număr de restricţii suplimentare judicios alese, numite **tăieturi**, se pot îndepărta din A_{PL} o serie de porţiuni astfel încât soluţia optimă întreagă să devină vârf în mulţimea rămasă – vezi fig. 3.2.

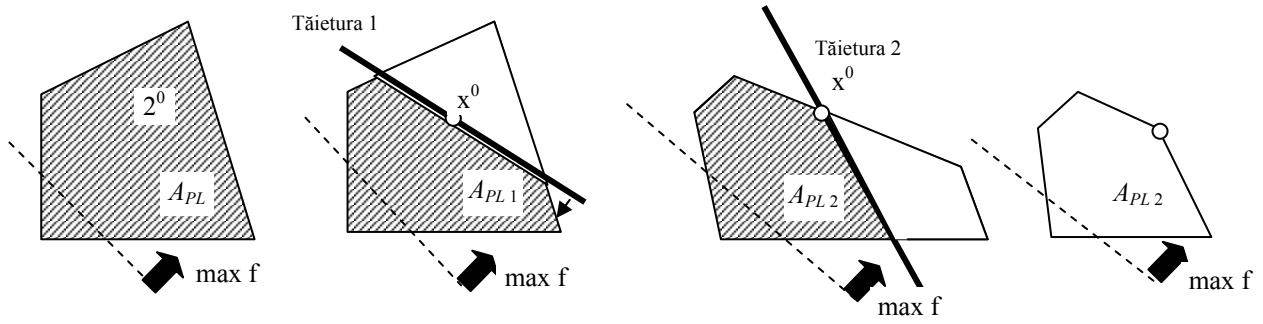

Fig. 3.2

În acest fel soluţia optimă întreagă va putea fi detectată de către algoritmul simplex.

4) În cazul studiat A_P era o mulţime finită. Acest lucru se întâmplă şi în situaţii mai generale; de exemplu o problemă de programare cu n variabile bivalente nu poate avea mai mult de 2^n soluţii admisibile întregi. Fără a influenţa generalitatea concluziilor, se poate presupune că orice program întreg are un număr finit de soluţii întregi.

Aceste constatări ne vor permite să înţelegem mai bune fundamentele, performanţele şi limitele metodelor de rezolvare a programelor întregi ce vor fi prezentate în următoarea secţiune.

§ 4. Metode de rezolvare ale programelor întregi

Următoarele consideraţii vizează programul întreg în formă standard cu m ecuaţii şi n variabile:

$$(P)\begin{cases} \sum_{j=1}^{n} a_{ij}x_j = b_i \qquad i = 1,\dots m \\ x_j \geq 0 \qquad j = 1,\dots n \\ x_j \text{ întregi} \\ (\max) f = \sum_{j=1}^{n} c_j x_1 \end{cases} \Leftrightarrow \begin{cases} A\,x = b \\ x \geq 0 \\ x \text{ întreg } (\equiv \text{cu componente întregi}) \\ (\max)\ f = cx \end{cases}$$

în care toate constantele a_{ij}, b_i, c_j sunt numere întregi.

Forma generală de mai sus acoperă şi cazul particular important al programelor bivalente prin introducerea restricţiilor $x_j \leq 1$, j = 1, ... n Ca şi până acum (PL) va desemna programul relaxat dedus din (P) prin eliminarea condiţiei de integritate impusă variabilelor.

Vom presupune în mod constant că mulţimea soluţiilor admisibile ale problemei (PL) este **mărginită**; în problemele practice această cerinţă poate fi întotdeauna asigurată. Rezultă imediat că (P) are un număr **finit** de soluţii admisibile întregi.

Deoarece coeficienții funcției obiectiv sunt întregi şi optimul întreg va fi un număr întreg. Mai mult:

$$\text{optimul întreg} \leq \lfloor \text{optimul fracționar} \rfloor$$

unde $\lfloor \alpha \rfloor$ înseamnă **rotunjirea întreagă inferioară** a numărului real α.

Ne propunem în continuare să dăm o clasificare a metodelor şi algoritmilor de rezolvare a programului întreg general (P); observațiile şi concluziile din secțiunea prezentată vor juca un rol esențial.

1) O mare parte din aceste metode reduc rezolvarea programului întreg (P) la rezolvarea unei secvențe **finite** şi programe liniare **uzuale**:

$$(PL_0 = PL), PL_1, PL_2, \ldots, PL_t$$

ultimul având drept soluție optimă chiar soluția optimă întreagă a programului original (P).

Pentru fiecare k = 1, 2, ... t programul (PL_k) se obține din cel anterior prin adăugarea unei anumite restricții suplimentare a cărei construcție diferă de la metodă la metodă. Restricțiile suplimentare sunt astfel alese încât:

- să fie verificate de orice soluție admisibilă întreagă a programului original (P);

- prin introducerea lor să îndepărteze porțiuni din mulțimea A_{PL} a tuturor soluțiilor admisibile ale relaxatei $PL \equiv PL_0$ până când soluția optimă întreagă devine **vârf** în mulțimea rămasă, putând fi astfel cunoscută de algoritmul simplex - vezi fig. 3.2.

Din acest motiv aceste restricții suplimentare se mai numesc şi **tăieturi** sau **plane de secțiune.**

Din categoria algoritmilor bazați pe tăieturi fac parte algoritmii **discret** şi **ciclic** datorați lui **Gomory** (1960) şi algoritmul **primal** al lui **Young** şi **Glover** (1972).

2) Faptul că un program întreg are (sau poate fi făcut să aibă) un număr finit de soluții sugerează un alt mod de rezolvare a programelor întregi bazate pe **enumerarea totală** sau **parțială** a acestor soluții. Enumerarea totală este evocată doar ca posibilitate pentru că numărul soluțiilor admisibile întregi, deşi finit este de regulă foarte mare.

Schemele de enumerare parțială determină soluția optimă întreagă generând efectiv doar o parte a mulțimii soluțiilor admisibile întregi, soluțiile negenerate fiind recunoscute implicit ca neoptimale.

Domeniul predilect de aplicare a metodelor de enumerare îl constituie programarea bivalentă. Un exemplu reprezentativ îl constituie algoritmul **aditiv** al lui **Balaş** (1965).

3) Punerea în evidență a soluției optime întregi x^0 – situată de regulă în interiorul mulțimii soluțiilor admisibile ale problemei relaxate – se poate face şi în următorul mod (**Dakin, Driebeck**, 1964).

Să presupunem că am rezolvat problema relaxată PL, cu ajutorul algoritmului simplex obținând soluția optimă fracționară x*. Dacă x* are toate componentele întregi, am terminat:

$$x^* \equiv x^0 \equiv \text{soluția optimă întreagă a programului original (P)}$$

În caz contrar, una sau mai multe din componentele $x_1^*, x_2^*, \ldots$ ale soluției x* nu sunt întregi; să presupunem de exemplu că x_1^* este fracționar. Este clar atunci că soluția optimă întreagă căutată va verifica una din următoarele inegalități **mutual exclusive**:

$$x_1 \le \lfloor x_1^* \rfloor \quad \text{sau} x_1 \ge \lfloor x_1^* \rfloor + 1$$

(unde s-a notat cu $\lfloor a \rfloor \equiv$ partea întreagă a numărului real a).

Considerăm programele liniare obținute prin extinderea relaxatei PL cu fiecare din cele două inegalități:

$$PL_1 \equiv \begin{cases} PL \\ x_1 \le \lfloor x_1^* \rfloor \end{cases} \qquad PL_2 \equiv \begin{cases} PL \\ x_1 \ge \lfloor x_1^* \rfloor + 1 \end{cases}$$

Vom spune că **am ramificat** problema (PL) după variabila x_1.

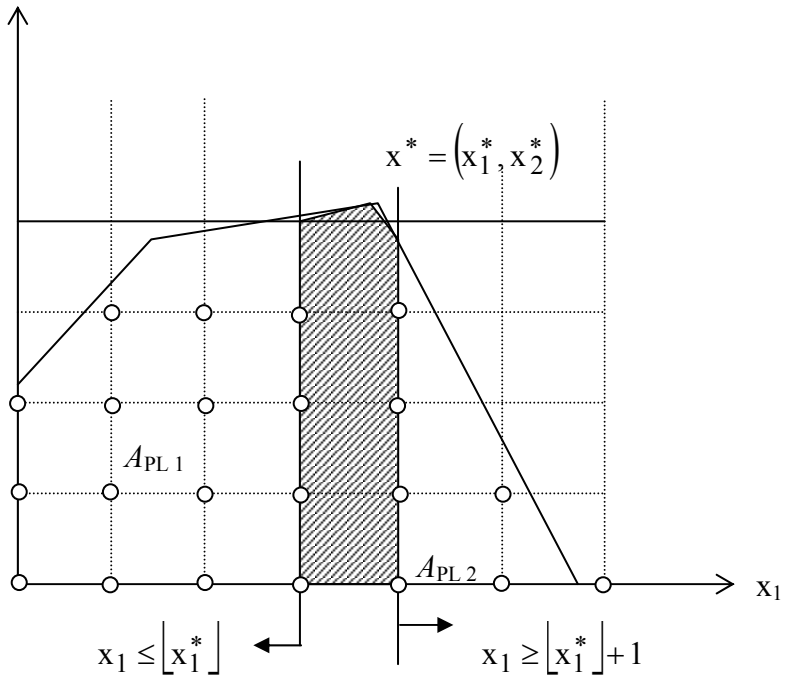

Fig. 4.1.

Este uşor de arătat că:

- Mulțimile de soluții admisibile $A_{PL\,1}$ şi $A_{PL\,2}$ ale celor două programe noi sunt submulțimi stricte în A_{PL}.
- $A_1 \cup A_2 = A_P$ şi $A_1 \cap A_2 = \emptyset$

unde A_1, A_2 sunt mulțimile de soluții admisibile întregi ale programelor PL_1 şi PL_2. În particular soluția optimă întreagă a programului original (P) se găseşte într-una (şi numai în una) din mulțimile mai mici $A_{PL\,1}$ sau $A_{PL\,2}$ vezi figura 4.1.

Rezolvăm în continuare PL_1; putem face acest lucru prin **reoptimizare**. Să presupunem că PL_1 este compatibilă; în soluția sa optimă, notată x^{*1} prima componentă va fi cu siguranță întreagă: $x_1^{*1} = \lfloor x_1^* \rfloor$.

În ipoteza ca a doua componentă x_2^{*1} este fracționară vom deriva alte două noi programe liniare după modelul de mai sus:

$$PL_{11} \equiv \begin{cases} PL_1 \\ x_2 \le \lfloor x_2^{*1} \rfloor \end{cases} \qquad PL_{12} \equiv \begin{cases} PL_1 \\ x_2 \ge \lfloor x_2^{*1} \rfloor + 1 \end{cases}$$

Dacă soluția optimă întreagă căutată este soluție admisibilă pentru programul PL_1 atunci cu siguranță ea se va găsi în una din mulțimile de soluții admisibile ale celor două programe rezultate prin ramificare.

În principiu, reunificarea poate continua de la oricare din problemele PL_{11}, PL_{12} sau PL_2, **condițiile de ramificare** fiind acelea ca programul în cauză:

- să fie compatibil (adică să aibă soluții admisibile);
- soluția sa optimă să aibă cel puțin o componentă fracționară.

De regulă, ramificarea se face după **prima variabilă cu valoare fracționară** sau după variabila cu **cea mai mare parte fracționară**.

Pentru înțelegerea metodei vom vizualiza procesul de ramificare printr-un graf **arbore** T ale cărui moduri sunt diferitele probleme rezultate din ramificare. Nodurile **terminale** (adică nodurile din care nu s-a mai efectuat ramificarea) sunt fie probleme incompatibile fie probleme cu soluții optime întregi. Cu excepția **rădăcinii** (PL) fiecare nod din T are un unic **predecesor**. Orice nod care nu este nod terminal are doi **succesori**.

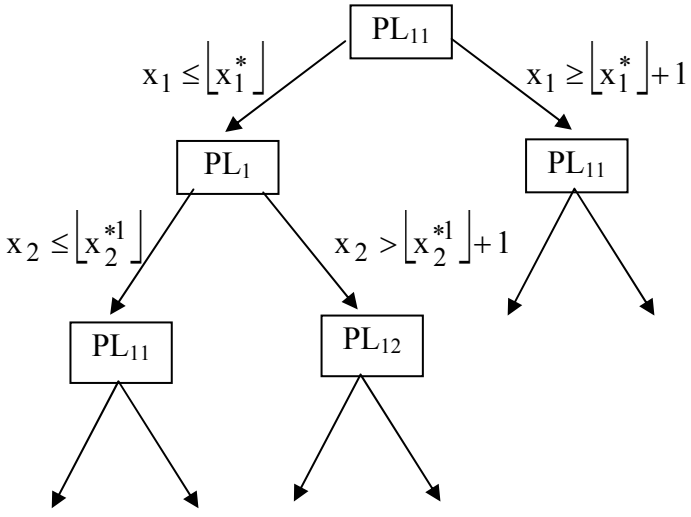

Figura 4.1

Introducem următoarele variabile:

x_{CMB} (vector) care reține **cea mai bună** soluție admisibilă întreagă la un moment sau altul al derulării procesului de ramificare

z_{CMB}, care reține valoarea funcției obiectiv în x_{CMB}.

La start: $x_{CMB} = Ø$ $z_{CMB} = -\infty$

Fiecare nod PL_α al arborelui T – unde α este o succesiune de 1 şi 2 – va avea ataşată o **margine superioară**:

$z_\alpha \equiv$ **rotunjirea întreagă inferioară** a optimului problemei PL_α în cazul în care aceasta este

compatibilă.

Este clar că dacă soluția optimă întreagă x^0 se află printre soluțiile admisibile ale problemei PL_α atunci:

$$z_{CMB} \leq f(x^0) \equiv \text{optimul întreg} \leq z_\alpha$$

Prin urmare, dacă pentru o problemă PL_α rezultată din ramificare, avem:

$$z_\alpha \leq z_{CMB}$$

nu mai are nici un rost să ramificăm PL_α, deoarece printre eventualele sale soluții admisibil întregi nu există nici una **mai bună** decât actuala x_{CMB} !

Arborele T nu există de la început. La start el se reduce la rădăcina (PL) şi în continuare primeşte noi noduri şi arce de legătură în funcție de problemele rezultate din ramificare şi efectiv rezolvate.

Metoda de rezolvare a unui program întreg succint expusă mai sus va fi explicată mai în detaliu pe următorul exemplu:

$$(P)\begin{cases}(\max) f = 5x_1 + 3x_2\\ -x_1 + 2x_2 \geq 2\\ x_1 - x_2 \leq 2\\ 2x_1 + 2x_2 \leq 7\\ x_1, x_2 \geq 0 \text{, intregi}\end{cases}$$

START

- Se inițializează: $z_{CMB} = -\infty$ şi $x_{CMB} = Ø$ (locația vidă)
- Se rezolvă cu algoritmul simplex relaxata problemei (P):

$$(PL)\begin{cases}(\max)\ f = 5\ x_1 + 3\ x_2\\ -\ x_1 + 2\ x_2 \geq 2\\ x_1 - x_2 \leq 2\\ 2\ x_1 + 2\ x_2 \leq 7\\ x_1, x_2 \geq 0\end{cases}$$

- Se găsește soluția optimă fracționară: $x_1^* = 1\frac{2}{3}, x_2 1\frac{5}{6}; f(x^*) = 13\frac{5}{6}$

- Se conchide că optimul întreg nu poate depăşi marginea superioară:

$$z = \left\lfloor 13\frac{5}{6} \right\rfloor = 13$$

- Variabila după care se face ramificarea va fi aleasă după criteriul „părții fracționare mai mari”; în cazul de față este vorba de variabila x_2.

Iterația 1 Se rezolvă cu algoritmul simplex programul liniar (PL_1) dedus din (PL) prin adăugarea restricției $x_2 \leq 1$.

- Se găsește soluția întreagă $x_1 = 0$, $x_2 = 1$; $f = 3$ pe care o reținem:

$x_{CMB} = (0, 1)$ $z_{CMB} = 3$

Ştim acum că optimul întreg este cel puțin 3. Revenim la problema (PL).

Iterația 2 Se rezolvă cu algoritmul simplex programul liniar (PL_2) dedus din (PL) prin adăugarea restricției $x_2 \geq 2$.

- Se găseşte soluția fracționară $x_1 = 1\frac{1}{2} \quad x_2 = 2; \quad f = 13\frac{1}{2}$
- Se conchide că soluțiile admisibile întregi ale problemei (PL_2) care sunt şi soluții ale problemei inițiale (P) nu pot oferi funcției obiectiv o valoare mai mare decât marginea:

$$z_2 = \left\lfloor 13\frac{1}{2} \right\rfloor = 13$$

- Ramificăm după variabila x_1.

Iterația 3 Se rezolvă problema (PL_{21}) obținută din PL_2 adăugând restricția $x_2 \leq 1$.

- Rezultă soluția fracționară $x_1 = 1, \quad x_2 = 2\frac{1}{2}; \quad f = 12\frac{1}{2}$
- Dacă soluția optimă a programului (P) s-ar găsi printre soluțiile întregi ale problemei PL_{21}, optimul întreg n-ar depăşi marginea $z_{21} = \left\lfloor 12\frac{1}{2} \right\rfloor = 12$.
- Ramificăm după variabila x_2.

Iterația 4 adăugăm la (PL) restricția $x_2 \leq 2$, obținând un nou program liniar PL_{211}. Rezolvarea lui conduce la soluția întreagă $x_1 = 1$, $x_2 = 2$, $f = 11$ evident mai bună decât soluția întreagă „depozitată" în x_{CMB}. În consecință actualizăm:

$x_{CMB} = (1,2)$ $z_{CMB} = 11$

Conchidem că optimul întreg al problemei originale (P) este mai mare sau egal cu 11.Revenim la problema PL_{21}.

Iterația 5 De această dată adăugăm la PL_{21} restricția $x_2 \geq 3$. Programul PL_{212} rezultat are soluția fracționară $x_1 = \frac{1}{2}, \quad x_2 = 3; \quad f = 11\frac{1}{2}$

Să observăm că eventualele soluții întregi ale acestei probleme nu pot oferi funcției obiectiv o valoare superioară marginii $z_{212} = \left\lfloor 11\frac{1}{2} \right\rfloor = 11;$ cum valoarea funcției obiectiv în cea mai bună soluție întreagă găsită până acum este chiar 11, conchidem că ramificând PL_{212} nu vom găsi soluții întregi mai bune decât actuala x_{CMB}. În consecință ne întoarcem din nou la problema PL_{21}.

Recapitulând, studiul problemei PL_{21} a produs o soluția întreagă care a fost reținută precum şi concluzia că PL_{21} nu are soluții întregi mai bune decât cea găsită. Ne întoarcem la problema PL_2 din care am derivat PL_{21}.

Iterația 6 Adăugăm acum la PL_2 restricția $x_1 \geq 2$. Noul program PL_{22} se dovedeşte a fi incompatibil. Ne întoarcem din nou la problema PL_2 cu concluzia că actuala x_{CMB} este cea mai bună soluție întreagă a sa. Mai departe, revenim la problema PL din care am derivat PL_2.

În acest moment putem spune că am examinat – direct sau indirect - toate soluțiile întregi ale

problemei (P), deoarece acestea erau, fie soluții întregi ale problemei PL_1 deja studiate, fie ale problemei PL_2 de asemeni examinate.

Soluția întreagă reținută în x_{CMB} este din cea mai bună soluție întreagă adică este soluția optimă a problemei originale (P).

Comentariile de mai sus sunt sintetizate în arborele T din fig. 4.2.

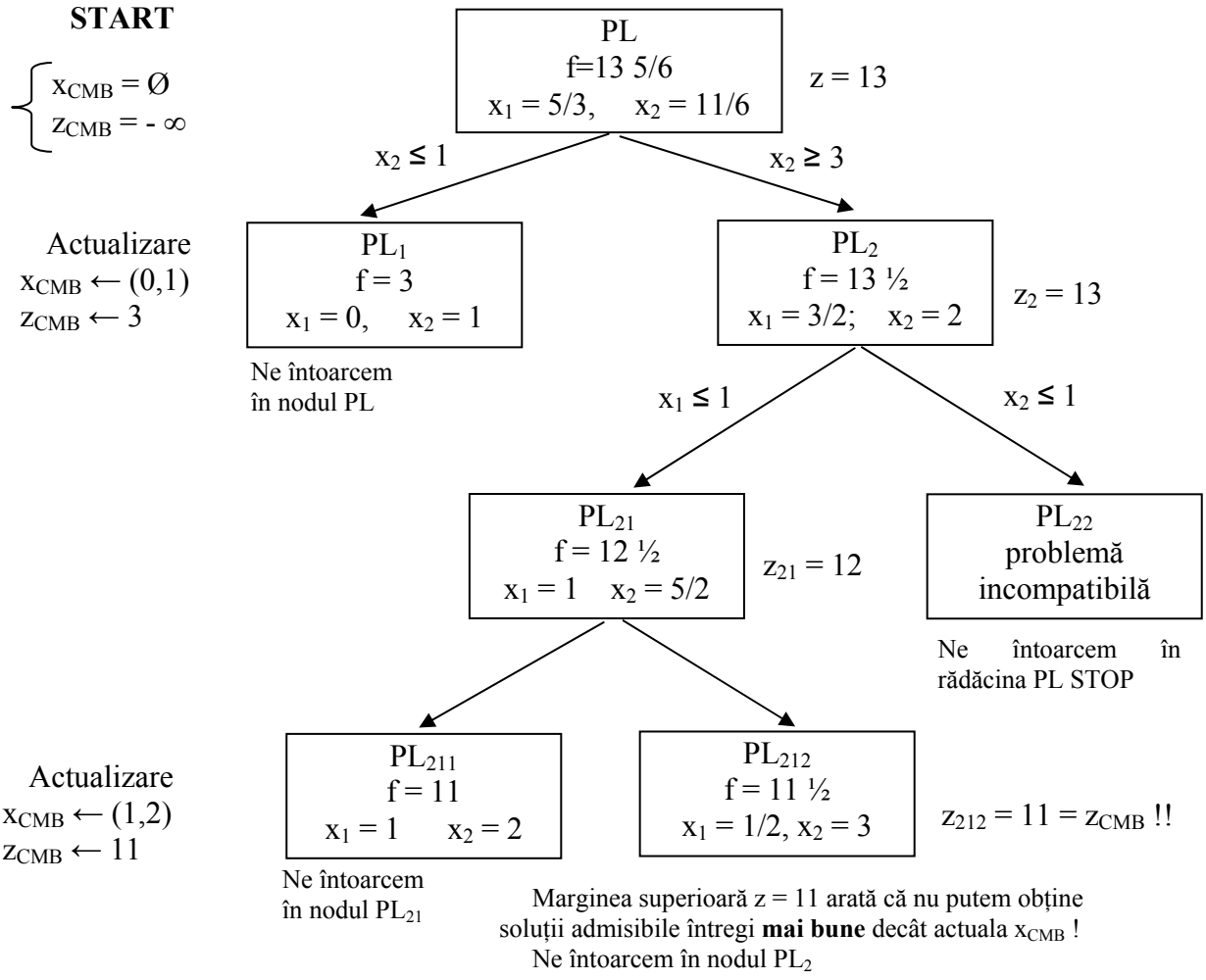

Fig. 4.2.

Ordinea de rezolvare a problemelor:

$$PL \rightarrow PL_1 \rightarrow PL_2 \rightarrow PL_{21} \rightarrow PL_{211} \rightarrow PL_{212} \rightarrow PL_{22}$$

Soluția optimă întreagă: $x^0 = x_{CMB} = (1, 2)$; Optimul întreg $f(x^0) = 11$

Observații 1) Utilitarul QM ramifică o problemă după variabila cu **cea mai mare parte fracționară** !

2) Utilitarul QM opreşte ramificarea unui nod PL_α numai dacă $z_\alpha < z_{CMB}$ în ideea găsirii tuturor soluțiilor admisibile întregi. În cazul de față şi nodul PL_{212} a fost ramificat ! vezi fig.4.3

Un nod al arborelui T din care ramificarea poate continua se numeşte **nod activ**; altminteri el se va

numi nod mort. Din denumire rezultă că algoritmul se opreşte în momentul în care rădăcina PL este declarată nod mort.

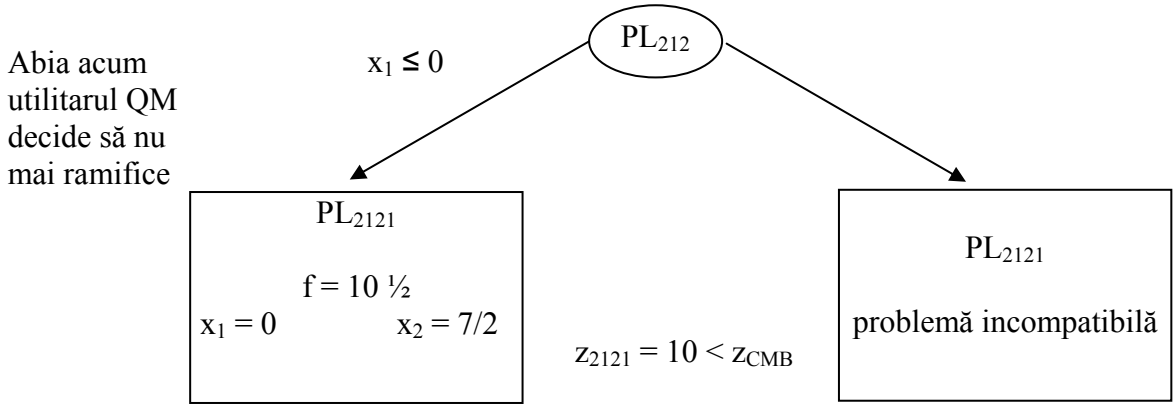

Fig 4.3

Principalele dezavantaje ale metodei descrise sunt:

- **creşterea foarte rapidă**, chiar explozivă a numărului problemelor de rezolvat şi de aici a volumului de calcule o dată cu creşterea dimensiunilor programului original şi mai cu seamă a numărului de variabile întregi;
- comportare **impredictibilă** pe probleme de dimensiuni apropiate: arborele problemelor rezolvate poate fi extrem de „stufos" pentru o problemă şi foarte simplu pentru o alta.

În practică, metoda se conbină cu alte procedee (cum ar fi metoda planelor de secțiune) care pot oferi rapid soluții admisibile întregi suficient de bune micşorându-se astfel drastic numărul ramificărilor.

Algoritmul descris este o specializare a unei metode mai generale denumite **branch & bound**.

În principiu această metodă **ramifică** adică partiționează mulțimea în care se caută un anumit element – numit element optimal – în părți mai mici pe care le **mărgineşte,** aceasta însemnând optimizarea funcției obiectiv pe fiecare din părțile rezultate. Unele din aceste părți sunt ramificate şi mărginite în continuare. Nu sunt ramificate acele părți care nu conțin în mod sigur soluția optimă căutată.

Ideea metodei B & B este de a găsi elementul optimal fără a inspecta toate elementele între care acesta se găseşte; de aceea se spune că B & B este o **schemă de enumerare parțială**.

§ 5. Întrebări şi probleme

1. Enumerați sursele de apariție ale modelelor de programare în numere întregi.

2. Explicați termenii: relaxata unui program întreg, soluție optimă fracționară, soluție optimă

întreagă, optim fracționar, optim întreg.

3. Explicați principiul metodelor de rezolvare a programelor întregi bazate pe tăieturi.

4. Explicați – eventual pe baza unui exemplu - metoda Branch & Bound.

5. În rezolvarea unui program liniar în numere întregi (P) în care funcția obiectiv se maximizează s-a ajuns la următorul arbore al programelor liniare efectiv rezolvate:

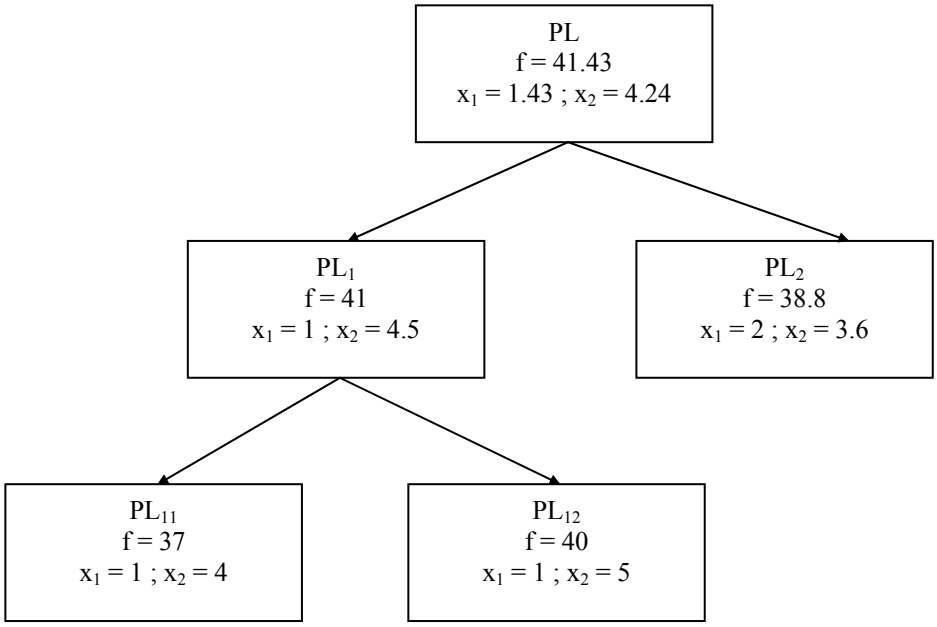

Figura 5.1

a) Precizați soluția optimă fracționară x^* şi $f(x^*)$ şi soluția optimă întreagă x^0 şi $f(x^0)$;
b) Indicați pe arcele arborelui restricțiile după care s-a făcut ramificarea;
c) În ce ordine au fost rezolvate cele cinci probleme?
d) De ce nu s-a continuat ramificarea din nodul (PL_2) ?

5. Se consideră programele întregi:

$$a)\begin{cases} 2x_1 + 2x_2 \le 9 \\ 3x_1 + x_2 \le 11 \\ x_1, x_2 \ge 0 \text{ intregi} \\ (\max) f = 5x_1 + 2x_2 \end{cases} \qquad b)\begin{cases} -x_1 + 2x_2 \le 2 \\ x_1 - x_2 \le 2 \\ 2x_1 + 2x_2 \le 7 \\ x_1, x_2 \ge 0 \text{ intregi} \\ (\max) f = 3x_1 + 5x_2 \end{cases} \qquad c)\begin{cases} -3x_1 + 8x_2 \le 19 \\ 6x_1 + 3x_2 \le 17 \\ x_1, x_2 \ge 0 \quad \text{intregi} \\ (\max) f = 3x_1 + 2x_2 \end{cases}$$

Pentru fiecare din ele:

a) reprezentați grafic mulțimea soluțiilor admisibile ale problemei relaxate;
b) puneți în evidență soluțiile admisibile întregi şi acoperirea convexă a acestora;

determinați grafic soluțiile optime fracționară şi întreagă precum şi optimele fracționar şi întreg.

6. Din produsul B sunt necesare cel puțin b unități. Produsul se poate obține pe două instalații I_1 şi I_2 dar numai în cicluri complete de fabricație (şi aceasta datorită unei operații de coacere prevăzută în tehnologia de fabricație). Pe fiecare ciclu instalația I_1 produce p_1 unități din B iar I_2 produce p_2

unități din B. Un ciclu de fabricație durează a_1 ore pe I_1 şi a_2 ore pe I_2. Fondurile de timp disponibil ale celor două instalații sunt de F_1 ore respectiv F_2 ore. Costurile de fabricație se ridică la c_1 u.m. (unități monetare) pe I_1 şi c_2 u.m. pe I_2 pe fiecare ciclu. Scrieți un program întreg pentru producerea cantității minimale B cu cheltuieli totale minime. Indicație: se vor lua ca variabile de decizie numărul ciclurilor de fabricație planificate a se realiza pe cele două instalații.

7. Un produs complex P se compune din 4 unități din componenta A şi 3 unități din componenta B. Componentele A şi B pot fi realizate prin trei procese de fabricație diferite în cadrul cărora se utilizează resursele R şi S disponibile în cantitățile de 100 unități respectiv 200 unități. Realizarea unui proces de fabricație necesită anumite cantități din resursele R şi S şi are ca rezultat producerea anumitor cantități din componentele A şi B conform datelor din următorul tabel:

Proces	Intrări (unități specifice)		Ieşiri (unități specifice)	
	din R	din S	din A	din B
1	8	6	7	5
2	5	9	6	9
3	3	8	8	4

Tabelul 5.1

Să se scrie un program întreg pentru determinarea numărului maxim de produse complexe P ce pot fi realizate din resursele date. Indicație: se va nota cu x_1 , x_2 , x_3 numărul de procese 1,2 respectiv 3 necesare şi cu y numărul produselor complexe P rezultate.

8. Folosiți utilitarul QM pentru rezolvarea programului întreg:

$$\begin{cases} (\max)\ x_4 \\ 8x_1 + 5x_2 + 3x_3 \le 100 \\ 6x_1 + 9x_2 + 8x_3 \le 200 \\ 7x_1 + 6x_2 + 8x_3 - 4x_4 \ge 0 \\ 5x_1 + 9x_2 + 4x_3 - 3x_4 \ge 0 \\ x_1, x_2, x_3, x_4 \ge 0 \ \text{intregi} \end{cases}$$

(Este vorba de modelul matematic al problemei precedente în care variabila y a fost renotată x_4) Interpretați soluția obținută.

9. Rezolvați prin metoda Branch & Bound programele întregi din ex.5. În rezolvarea programelor liniare generate de metodă se va folosi metoda grafică. Construiți arborele T al programelor liniare efectiv rezolvate.

10. Descrieți următoarele probleme:

- problema monezilor;
- problema alegerii proiectelor de investiții;
- problema stabilirii programului de producție cu costuri fixe de pregătire;
- problema stabilirii orarului de zbor al avioanelor.

Particularizați aceste modele pe date concrete plauzibile.