2. PROGRAMARE LINIARĂ ÎN NUMERE ÎNTREGI

2.1 Specificul programării în numere întregi

Una dintre proprietățile esențiale pentru fundamentarea metodelor de determinare a soluției optime a unei probleme de programare liniară o constituie posibilitatea variabilelor de decizie de a varia continuu: ele puteau lua orice altă valoare intermediară dintre două valori permise.

Nu puține sunt situațiile practice, modelate cu ajutorul programării liniare în care unele variabile de decizie nu au sens decât dacă acestea au valori întregi.

De exemplu, dacă variabila care exprimă o anumită mărime asociată unei activități este măsurată în unități indivizibile este clar că valorile permise ale variabilei care indică valoarea variabilei respective trebuie să fie număr întreg. La fel, repartizarea personalului muncitor, al utilajelor sau al mijloacelor de transport pe diverse activități trebuie exprimată tot prin numere întregi.

În continuare se vor considera următoarele definiții:

- 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 conține 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.

Utilizarea variabilelor întregi poate aduce un plus de flexibilitate în modelarea unor situații practice. Acest plus flexibilitate are un cost destul de ridicat în ceea ce priveşte efortul depus în rezolvarea problemei respective deoarece programele întregi sunt mult mai greu de rezolvat decât programele liniare uzuale (adică în variabile continue).

Aplicație: Să se analizeze oportunitatea construirii a două centrale şi a unei stații electrice în două locații A şi B. Nu se poate construi o stație electrică într-o anumită locație decât dacă s-a construit şi o centrală în locația respectivă. Deoarece există şi alte soluții de racordare nu este obligatorie construirea unei stații în cazul construirii unei centrale. În tabelul 2.1. sunt indicate: beneficiul anual obținut în urma realizării fiecărei investiții, costurile de realizare şi capitalul disponibil.

Tabelul 2.1.

Nr. crt.	Varianta	Variabila de decizie	Beneficiu [mil€]	Costuri [mil€]
1	Construire centrală în A	x_1	90	500
2	Construire centrală în B	x_2	60	300
3	Construire stație în A	x_3	30	300
4	Construire stație în B	x_4	40	400
			Capital disponibil	1.000

Variabilelor $x_1 \div x_4$ li se impune să fie variabile bivalente: valoarea 1 a variabilei x_j corespunde unei decizii de realizare a investiției respective iar o valoare nulă corespunde unei decizii de respingere a acesteia. Pe lângă condiția ca variabilele să fie bivalente se mai formulează următoarele restricții:

-condiția de a nu construi decât cel mult o centrală: $x_3 + x_4 \leq 1$

-condiția de a nu construi o stație într-o locație în care nu s-a construit o centrală: $x_3 \leq x_1$, $x_4 \leq x_2$

Ca funcție obiectiv se impune maximizarea beneficiului anual, rezultând următorul model matematic:

$$\max F(x_1, x_2, x_3, x_4) = 90x_1 + 60x_2 + 30x_3 + 40x_4$$
$$\begin{cases} 500x_1 + 300x_2 + 300x_3 + 400x_4 \leq 1000 \\ x_3 + x_4 \leq 1 \\ -x_1 + x_3 \leq 0 \\ -x_2 + x_4 \leq 0 \end{cases}$$
$$x_j \in \{0,1\} \quad j = \overline{1,4}$$

Soluția acestei probleme de programare liniară în numere întregi (uşor de determinat prin încercări, ținând cont că nu sunt decât 7 combinații posibile) este: $F(1,1,0,0)=150$

Dacă se încearcă obținerea soluției prin rotunjirea soluției problemei relaxate (obținută prin eliminarea condiției ca variabilele să fie numere întregi), de forma:

$$\max F(x_1,x_2,x_3,x_4)=90x_1+60x_2+30x_3+40x_4$$
$$\begin{cases}500x_1+300x_2+300x_3+400x_4\le 1000\\ x_3+x_4\le 1\\ -x_1+x_3\le 0\\ -x_2+x_4\le 0\\ x_j\le 1 \quad j=\overline{1,4}\end{cases}$$
$$x_j\ge 0 \quad j=\overline{1,4}$$

rezultând soluția: $F(1;\ 1;\ 0{,}2;\ 0{,}35)=170$ care, prin rotunjire la valoarea cea mai apropiată, conduce la soluția problemei în numere întregi.

Obținerea soluției problemei în numere întregi prin rotunjirea soluției nu garantează întotdeauna obținerea soluției corecte. De exemplu, dacă se modifică problema inițială considerând un capital disponibil de 1.150, sub forma:

$$\max F(x_1,x_2,x_3,x_4)=90x_1+60x_2+30x_3+40x_4$$
$$\begin{cases}500x_1+300x_2+300x_3+400x_4\le 1150\\ x_3+x_4\le 1\\ -x_1+x_3\le 0\\ -x_2+x_4\le 0\end{cases}$$
$$x_j\in\{0,1\} \quad j=\overline{1,4}$$

se obține soluția $F(1,1,1,0)=180$.

Problema relaxată asociată are soluția F(1; 1; 0,34; 0,619)=185.

Prin rotunjire la valoarea cea mai apropiată se obține (1; 1; 0; 1) care nu numai că nu este soluția problemei în numere întregi, dar nu este nici soluție admisibilă pentru problema respectivă.

2.2 Particularităţile soluţiilor admisibile ale unui program întreg

Se consideră următorul **program întreg total** (P):

$\max F(x_1,x_2)=8x_1+3x_2$
$5x_1+3x_2\leq 15$
$-3x_1-8x_2\leq -12$
$x_1-2x_2\leq -1$
$x_1\geq 0,\ x_2\geq 0,\ x_1,x_2$ - întregi

din care rezultă **programul relaxat** (PL):

$\max F(x_1,x_2)=8x_1+3x_2$
$5x_1+3x_2\leq 15$
$-3x_1-8x_2\leq -12$
$x_1-2x_2\leq -1$
$x_1\geq 0,\ x_2\geq 0$

În continuare vor fi folosite următoarele definiţii şi notaţii:

- soluţie **admisibilă** - ansamblu de valori numerice (nu neapărat întregi) care verifică restricţiile şi condiţiile de nenegativitate;
- soluţie **admisibilă întreagă** - soluţie admisibilă cu toate componentele întregi;
- A_{PL} - mulţimea sluţiilor admisibile ale programului relaxat (PL);
- A_P - mulţimea soluţiilor admisibile întregi ale programului (P);
- soluţia optimă **fracţionară** - soluţia optimă a programului relaxat (PL), notată constant cu x^*;
- optim **fracţionar** - valoarea $F(x^*)$ a funcţiei obiectiv în soluţia optimă fracţionară x^*;
- soluţia optimă **întreagă** - soluţia optimă a programului întreg (P), notată constant cu x^0;
- optim **întreg** - valoarea $F(x^0)$ a funcţiei obiectiv în soluţia optimă întreagă x^0.

Pentru reprezentarea grafică a mulţimii A_{PL} se reprezintă dreptele:
d1: $5x_1+3x_2=15$
d2: $3x_1+8x_2=12$
d3: $x_1-2x_2=-1$

şi se identifică semiplanele în care sunt verificate cele trei restricţii şi cele două condiţii de nenegativitate.

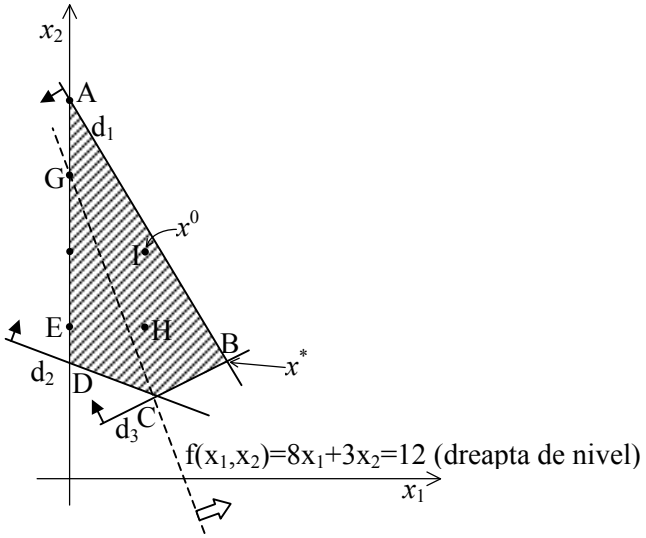

Figura 2.1. *Domeniul soluţiilor admisibile*

În figură a fost reprezentată grafic o dreaptă de nivel a funcţiei obiectiv şi direcţia de translatare a acesteia corespunzătoare creşterii valorii funcţiei obiectiv.

Prin aplicarea algoritmului simplex sau grafic se obţine soluţia problemei relaxate la intersecţia dreptelor d_1 şi d_3, care reprezintă vârful D al poligonului soluţiilor admisibile $x^* = \begin{pmatrix} 20/13 \\ 27/13 \end{pmatrix}$ căruia îi corespunde valoarea optimă $F(x^*) = \frac{276}{13}$.

Prin observare directă se găseşte mulţimea soluţiilor admisibile întregi ale programului (P) care este alcătuită din următoarele puncte:

A_P = {E(0,2), F(0,3), G(0,4), A(0,5), H(1,2),I(1,3)}

Funcția obiectiv F are valoarea maximă în I(1,3), deci soluția optimă întreagă este x^0=(1,3) iar optimul întreg are valoarea $F(x^0) = 17$.

Comparând mulțimile A_P şi A_{PL} se pot formula următoarele concluzii valabile pentru toate problemele de programare în numere întregi:

1) Spre deosebire de A_{PL}, A_P este o mulțime rară, adică orice punct din ea posedă o vecinătate în care nu se mai află alte puncte din mulțime. Aplicarea teoriei teoria clasice a optimizării, bazată pe posibilitatea efectuării unor deplasări infinitesimale în jurul unui punct este în acest caz lipsită de sens.

2) A_{PL} fiind o mulțime **poliedrală, convexă** cu un număr **finit** de **vârfuri,** în general soluția optimă întreagă – care este în același timp o soluție admisibilă problemei relaxate – se găseşte în interiorul mulțimii A_{PL} şi, ca urmare, nu este generată de către algoritmul simplex care conduce la determinarea soluției probleme relaxate ce se găseşte într-unul din vârfurile poliedrului soluțiilor admisibile.

3) Dacă se consideră **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 AIHE) se constată că vârfurile anvelopei sunt soluții admisibile întregi ale programului (P).

Dacă se maximizează funcția obiectiv F pe anvelopa convexă a soluțiilor admisibile întregi se poate regăsi soluția optimă întreagă x^0.

În concluzie, se poate 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 AIHE se compune din soluțiile sistemului de inecuații:

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

la care se adaugă condițiile de nenegativitate: $x_1 \ge 0$, $x_2 \ge 0$.

În cazul general descrierea matematică a anvelopei convexe a mulțimii A_P este foarte dificil de realizat.

Totuşi din aceste constatări se poate obține o metodă de rezolvare a programului întreg: adăugând la problema relaxată PL un număr de restricții suplimentare judicios alese, numite **tăieturi** (reprezentate de inecuații liniare), se pot îndepărta din A_{PL} o serie de porțiuni astfel încât soluția optimă întreagă să devină vârf în mulțimea rămasă – fig. 2.2. În acest fel, chiar dacă nu s-a obținut anvelopa convexă a mulțimii A_P, soluția problemei relaxate asociată programului întreg modificat (la care s-au adăugat restricțiile suplimentare reprezentând inecuațiile tăieturilor) reprezintă şi soluția programului întreg inițial.

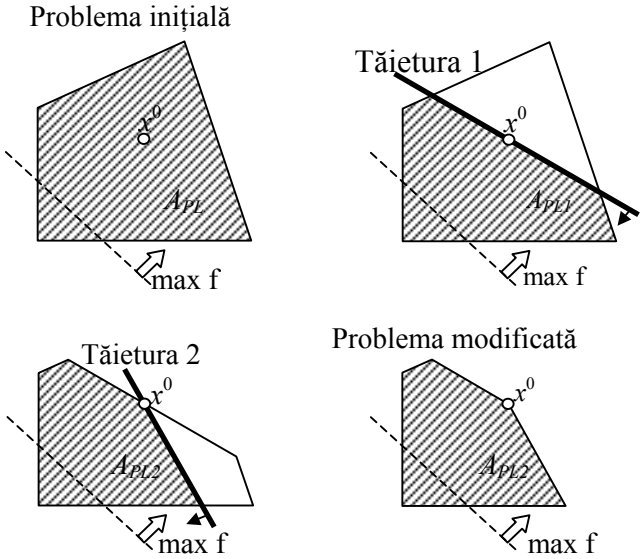

Figura 2.2 *Principiul metodei tăieturilor*

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

2.3 Metode de rezolvare ale programelor întregi

Se consideră un program întreg în formă standard:

$$(P)\ \begin{cases} \max F = \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i = 1,...,m \\ x_j \geq 0 \quad j = 1,...,n \\ x_j \ \text{întregi} \end{cases} \tag{2.1}$$

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 $x_j \leq 1$, $j = 1, \ldots n$.

Se presupune 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 coeficienţii funcţiei obiectiv sunt întregi şi optimul întreg va fi un număr întreg. Mai mult, optimul întreg este cel mult egal cu rotunjirea întreagă inferioară a optimului fracţionar.

Pentru rezolvarea programului întreg general (P) pot fi utilizaţi următorii algoritmi de rezolvare:

(1) Transformarea rezolvării programului întreg (P) în rezolvarea unei secvenţe finite de programe liniare uzuale:

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

ultimul având drept soluţie optimă chiar soluţia optimă întreagă a programului original (P).

Pentru fiecare $k = 1, 2, \ldots t$ programul (PL_k) se obţine din cel anterior prin adăugarea unei anumite restricţii suplimentare a cărei construcţie diferă de la metodă la metodă. Restricţiile 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 ≡ PL_0 până când soluția optimă întreagă devine vârf în mulțimea rămasă, putând fi astfel cunoscută de algoritmul simplex.

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

2) Faptul că un program întreg are (sau poate fi făcut să aibă) un număr finit de soluții sugerează posibilitatea de rezolvare a programelor întregi folosind metode bazate pe enumerarea totală sau parțială a acestor soluții. 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 dar pot fi adaptate şi pentru programe întregi generale.

Astfel, presupunând că a fost rezolvată problema relaxată PL, cu ajutorul algoritmului simplex obținând soluția optimă fracționară x^*, în cazul în care una sau mai multe din componentele $x_1^*, x_2^*, \ldots$ ale soluției x^* nu sunt întregi soluția programului întreg diferă de soluția programului relaxat; se presupune că x_1^* este fracționar. Este clar atunci că soluția optimă întreagă căutată va verifica una din următoarele inegalități mutual exclusive:

$$x_1 \le [x_1^*] \quad \text{sau} \quad x_1 \ge [x_1^*] + 1$$

($[a]$ = partea întreagă a numărului real a).

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

$$(PL_1)\begin{cases} PL \\ x_1 \le [x_1^*] \end{cases} \qquad (PL_2)\begin{cases} PL \\ x_1 \ge [x_1^*] + 1 \end{cases}$$

Se spune că problema (PL) a fost ramificată după variabila x_1.

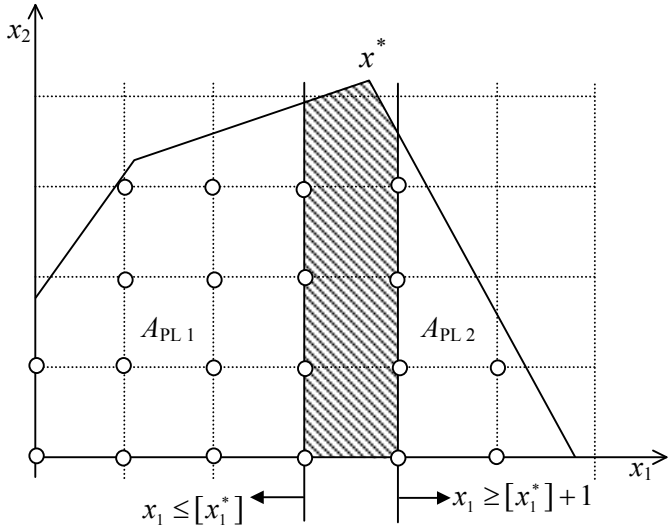

Figura 2.3 *Împărţirea domeniului soluţiilor admisibile la ramificarea problemei*

Este evident că:

- Mulţimile de soluţii admisibile A_{PL1} şi A_{PL2} ale celor două programe noi sunt submulţimi stricte în A_{PL}.
- Mulţimile de soluţii admisibile întregi ale programelor relaxate PL_1 şi PL_2 verifică condiţiile

$$A_1 \cup A_2 = A \text{ şi } A_1 \cap A_2 = \emptyset$$

În particular soluţia optimă întreagă a programului original (P) se găseşte într-una (şi numai în una) din mulţimile mai mici $A_{PL\,1}$ sau $A_{PL\,2}$.

În continuare se rezolvă PL_1. Presupunând că PL_1 este compatibilă, în soluţia sa optimă, notată x^{*1} prima componentă va fi cu siguranţă întreagă: $x_1^{*1} = [x_1^*]$.

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

$$(PL_{11})\begin{cases} PL_1 \\ x_1 \le [x_2^{*1}] \end{cases} \qquad (PL_{12})\begin{cases} PL_1 \\ x_1 \ge [x_2^{*1}]+1 \end{cases}$$

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, ramificarea 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 procesul de ramificare poate fi reprezentat printr-un graf arbore T ale cărui noduri 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 predecesor unic. Orice nod care nu este nod terminal are doi succesori.

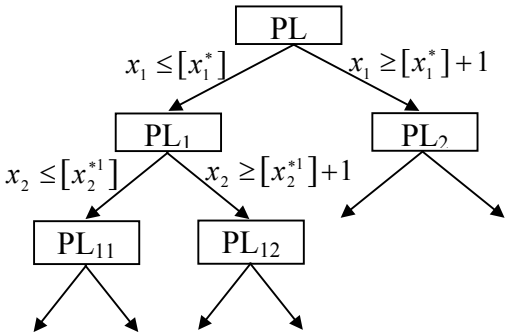

Figura 2.4. *Arbore de ramificare*

Se pot introduce următoarele variabile:

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

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

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

Fiecărui 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:

$$z_{CMB} \leq f(x^0) \leq z_\alpha$$

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}. Această constatare reduce numărul de ramificații care vor fi analizate.

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:

$$(P)\begin{cases}(\max)F = 4x_1 + 3x_2\\ 3x_1 + 4x_2 \leq 12\\ 4x_1 + 2x_2 \leq 9\\ x_1, x_2 \in N\end{cases}$$

START

- Se iniţializează: $z_{CMB} = - \infty$ şi $x_{CMB} = \varnothing$
- Se rezolvă cu algoritmul simplex relaxata problemei (P):

$$(PL)\begin{cases}(\max) f = 4x_1 + 3x_2 \\ 3x_1 + 4x_2 \le 12 \\ 4x_1 + 2x_2 \le 9 \\ x_1, x_2 \ge 0\end{cases}$$

şi se găseşte soluţia optimă fracţionară: $x_1^* = 1{,}2;\ x_2 = 2{,}1;\ f(x^*) = 11{,}1$

- Rezultă că optimul întreg nu poate depăşi marginea superioară:

$$z = [11{,}1] = 11$$

- Variabila după care se face ramificarea va fi aleasă după criteriul „părţii fracţionare mai mari"; în cazul de faţă este vorba de variabila x_2.

Iteraţia 1 Se rezolvă cu algoritmul simplex programul liniar (PL_1) dedus din (PL) prin adăugarea restricţiei $x_1 \le 1$.

- Se găseşte soluţia fracţionară $x_1 = 1 \quad x_2 = 1{,}25; \quad f = 10{,}75$
- Se conchide că soluţiile admisibile întregi ale problemei (PL_1) care sunt şi soluţii ale problemei iniţiale (P) nu pot oferi funcţiei obiectiv o valoare mai mare decât marginea:

$$z_1 = [10{,}75] = 10$$

- Se ramifică după variabila x_2.

Iteraţia 2 Se rezolvă problema (PL_{11}) obţinută din PL_1 adăugând restricţia $x_2 \le 2$.

- Rezultă soluţia întreagă $x_1 = 1, \quad x_2 = 2; \quad f = 10$

evident mai bună decât soluţia întreagă „depozitată" în x_{CMB}. În consecinţă se actualizează: $x_{CMB} = (1,2)^T \qquad z_{CMB} = 10$

- Se revine la problema PL_1.

Iteraţia 3 Deoarece valoarea funcţiei pentru soluţia întreagă a problemei PL_{11} este egală cu z_1, rezultă că prin rezolvarea problemelor obţinute prin ramificarea problemei PL_{12} rezultată din adăugarea la (PL) a restricţiei $x_2 \geq 3$ nu poate conduce la o valoare mai mare decât cea deja obţinută (f=10, PL_{11}). În aceste condiţii, programul PL_{12} şi cele ce decurg din acesta se rezolvă doar în cazul în care se doreşte să se determine eventualele soluţii echivalente din punct de vedere al valorii funcţiei obiectiv. Rezolvarea programului PL_{12} conduce la soluţia întreagă $x_1 = 0$, $x_2 = 3$, f = 9 care este mai slabă decât cea stocată în x_{CMB}.

- Se revine la problema PL_1 şi apoi la PL.

Iteraţia 4 Se rezolvă cu algoritmul simplex programul liniar (PL_2) dedus din (PL) prin adăugarea restricţiei $x_1 \geq 2$.

- Se găseşte soluţia fracţionară $x_1 = 2$, $x_2 = 0{,}5$; $f = 9{,}5$

Programele care pot rezulta prin ramificare din aceasta adăugând condiţiile $x_2=0$ şi, respectiv, $x_2 \geq 1$ vor conduce la valori ale funcţiei obiectiv care nu pot depăşi:

$$z_2 = [9{,}5] = 9$$

deci care sunt inferioare soluţiei deja determinate. În aceste condiţii acesta este un nod terminal.

Recapitulând, studiul problemei PL_{11} a produs o soluţia întreagă care a fost reţinută precum şi concluzia că PL_{12} şi PL_2 nu are soluţii întregi mai bune decât cea găsită. În acest moment se poate spune că au fost examinate – direct sau indirect - toate soluţiile întregi ale problemei (P). Soluţia întreagă reţinută în x_{CMB} este din cea mai bună soluţie întreagă adică este soluţia optimă a problemei originale (P).

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.

Comentariile de mai sus sunt sintetizate în arborele din fig. 2.5.

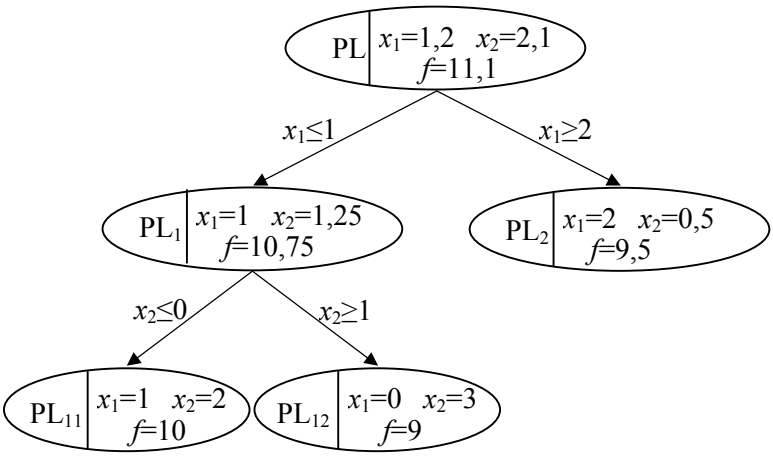

Figura 2.5 *Arborele de ramificare al problemei considerate*

Principalele dezavantaje ale metodei descrise sunt:

- creşterea foarte rapidă 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 combină cu alte procedee (cum ar fi metoda planelor de secțiune) care pot oferi rapid soluții admisibile întregi suficient de bune, micşorându-se astfel numărul ramificărilor.

Algoritmul descris este o variantă a unei metode mai generale denumite *branch & bound*, care ramifică (adică partiționează) mulțimea în care se caută un anumit element – numit element optimal – în părți mai mici pe care le mărginește, 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.