7. Aplicație la programarea în numere întregi

De regulă, variabilele unei probleme de programare matematică *variază continuu* singura condiție impusă fiind aceea ca valorile lor să fie nenegative, cerință justificată de obicei de contextul practic modelat. O mulțime de situații, cele mai multe din domeniul economic, impun utilizarea unor variabile care pot lua numai valori întregi: dacă, de exemplu, o variabilă "este măsurată" în unități indivizibile (număr de bucăți) atunci este clar că în orice soluție admisibilă şi în particular în soluția optimă, variabila în cauză nu poate lua valori fracționare.

O problemă care utilizează variabile întregi se va numi *problemă de programare în numere întregi* sau *problemă de programare discretă* (*totală* sau *mixtă* după cum toate variabilele trebuie să ia în exclusivitate valori întregi sau numai o parte din ele).

După cum va rezulta şi din exemplele concrete, utilizarea variabilelor întregi duce la creşterea flexibilității modelării în sensul obținerii unor descrieri "mai exacte" a situațiilor practice modelate. Pe de altă parte, condiția de integritate impusă variabilelor complică enorm rezolvarea, mai cu seamă din cauza faptului că mijloacele teoriei clasice nu sunt direct aplicabile.Astfel, la ora actuală, o problemă de programare liniară uzuală, cu câteva mii de variabile "continue" poate fi rezolvată lejer , utilizând unul sau altul dintre programele comerciale de calculator existente pe piață, în timp ce o problemă

de optimizare discretă cu mai puțin de 100 variabile poate cauza serioase dificultăți.

Nu este în intenția noastră ca, în cuprinsul acestei lucrări, să dezvoltăm teoria şi metodele specifice programării în numere întregi. În secțiunile următoare ne propunem să ilustrăm utilitatea acestui tip de programare în modelarea proceselor economice şi să indicăm unele posibilități de rezolvare ale programelor discrete folosind postoptimizarea.

7.1 O problemă de optimizare discretă: problema croirii

Problema debitării sau croirii unor repere din suporți de dimensiune mai mare este frecvent întâlnită în cele mai diverse domenii cum ar fi industria mobilei, a sticlei, a hârtiei, a confecțiilor, industria metalurgică sau cea

constructoare de maşini. Importanţa acestei probleme rezidă în dependenţa nemijlocită a reducerii consumului de materiale de utilizarea unor scheme eficiente de croire. Formele concrete sub care ea se prezintă diferă foarte mult de la un context la altul dar chestiunea care se pune de fiecare dată este aceeaşi:

cum trebuie să se desfăşoare procesul efectiv de croire astfel încât producerea unor cantităţi date de repere să se facă cu un consum cât mai mic de materiale.

De obicei, clasificarea problemelor de croire se face după numărul dimensiunilor relevante ale reperelor croite: avem probleme unidimensionale, bidimensionale sau tridimensionale. În *croirea unidimensională* reperele se diferenţiază printr-o singură dimensiune chiar dacă ele au mai multe. Cazul tipic este cel al debitării unor bare de diferite lungimi din bare mai mari. Iată un alt exemplu: o firmă produce un carton special în rulouri avînd o anumită lăţime. Clienţii solicită însă rulouri cu aceeaşi lungime ca şi rulourile standard dar de lăţimi mai mici. În acest caz, reperele ca şi suporţii sunt bidimensionali dar relevantă este o singură dimensiune - lăţimea (vezi exemplul 7.1.1).

Este interesant de menţionat cu acest prilej faptul că există contexte care nu implică croirea unor obiecte mai mici din suporţi similari mai mari în sensul uzual al cuvântului “croire”,dar care sunt reductibile la acest gen de probleme. Un exemplu îl constituie umplerea unor spaţii publicitare la televiziune sau radio cu diferite reclame. Aici suporţii sunt spaţiile publicitare iar reperele sunt reclamele, toate diferenţiate printr-o singură dimensiune - timpul.

După cum vom vedea problemele de croire unidimensionale sunt relativ simplu de formalizat (ca de altfel şi celelalte cu mai multe dimensiuni). Principala dificultate rezidă în numărul de posibilităţi de “tăiere” a unui suport care, în cele mai multe situaţii concrete, poate fi uluitor de mare. De exemplu, pentru o problemă cu 40 de repere ale căror lungimi sunt cuprinse între 20 şi 80 inches şi care se taie din suporţi cu lungimea de 200 inches, numărul reţetelor de croire posibile este situat între 10 şi 100 milioane!

Croirea bidimensională este cu mult mai complicată. Chiar şi în cazul cel mai simplu în care reperele şi suporţii sunt dreptunghiulari dificultăţile sunt enorme. Ele sunt cauzate în primul rând de numărul mare de posibilităţi de “aşezare” a reperelor pe suport; deşi finit acest număr este practic de necuprins.

Dacă în cazul unidimensional acestea pot fi construite algoritmic, existând siguranța - cel puțin teoretică - a generării tuturora, în cazul bidimensional nu există la ora actuală algoritmi care să garanteze producerea oricărei "aşezări" posibile. Nu mai puțin importante sunt restricțiile privind aşezarea efectivă a reperelor pe suport, aşezare care trebuie să țină seama de particularitățile instrumentului de tăiere ca şi de alte cerințe, unele imposibil de formalizat. De exemplu, în multe situații practice se poate întâmpla ca o "rețetă" de croire, acceptabilă dintr-un anumit punct de vedere - de pildă al acoperirii cât mai bune a suportului - să nu fie acceptabilă din alt punct de vedere - de pildă al "productivității" ei în procesul efectiv de debitare! Există şi alte restricții impuse de condițiile concrete, restricții care diferă de la un caz la altul. S-a conturat deja opinia că fiecare problemă de croire reală trebuie tratată de sine stătător, "speculându-se" pe cât posibil toate particularitățile ei.

În ceeace priveşte *croirea tridimensională,* termenul de "croire" este impropriu, el fiind folosit mai degrabă pentru a sublinia asemănarea cu problemele anterioare.Pentru a nu intra în amănunte vom da următorul exemplu: mai multe containere paralelipipedice trebuiesc încărcate în vagoane identice. Cum trebuie făcută încărcarea acestor containere astfel încât numărul de vagoane utilizate să fie minim? În mod curent, problemele de acest tip poartă numele de probleme de *împachetare.*

În continuare ne vom ocupa de *modelarea problemei de croire unidimensională.*

Un număr de repere cu lungimile $l_1 > l_2 > ... > l_m$ trebuiesc executate în cantitățile $b_1, b_2, ..., b_m$. Aceste repere se obțin prin tăiere din suporți identici cu lungimea L. *În ce mod trebuie făcută croirea reperelor astfel încât cantitățile planificate să fie realizate cu un consum minim de suporți?*

O modalitate de tăiere a unui suport în repere se va numi *rețetă de croire.* Evident, o rețetă este complet determinată de un vector cu m componente *întregi, nenegative* $a = (a_1, a_2, \ldots, a_m)$ în care a_i reprezintă numărul reperelor de lungime l_i rezultate prin tăiere. Deoarece suma lungimilor reperelor tăiate dintr-un suport nu depăşeşte lungimea acestuia, urmează că mulțimea rețetelor de croire se identifică cu mulțimea soluțiilor întregi,nenegative şi nenule ale inecuației:

$$l_1 a_1 + l_2 a_2 + \ldots + l_m a_m \leq L \qquad (7.1.1)$$

Să numim *rest* al rețetei $a = (a_1, a_2, \ldots, a_m)$ diferența:

$$r(a) = L - l_1 a_1 - l_2 a_2 - \ldots - l_m a_m$$

Din punct de vedere practic, importante sunt rețetele din al căror rest nu se mai pot croi alte repere; aceste rețete se vor numi *maximale*. Este clar că rețeta *a* va fi maximală numai dacă :

$$r(a) < l_m \equiv \text{lungimea celui mai mic reper.}$$

În continuare vom avea în vedere numai rețetele maximale. Fie A^1, A^2, ..., A^n lista lor, ordonată într-un fel oarecare, de exemplu *lexicografic*, unde:

$$A^j = (a_{1j}, a_{2j}, \ldots, a_{mj})^T$$

(pentru nevoi ulterioare rețetele vor fi scrise în "coloană"). Notând cu x_j *numărul de aplicări* ale rețetei A^j (sau, cum se mai spune, *multiplicitatea* rețetei A^j) problema de croire unidimensională se modelează astfel:

(C) *Să se determine x_1, x_2, ..., x_n întregi nenegativi, astfel încât:*

$$\sum_{j=1}^{n} A^j x_j \geq b \quad \Leftrightarrow \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i = 1, \ldots, m \qquad (7.1.2)$$

şi care minimizează funcţia obiectiv:

$$f = \sum_{j=1}^{n} x_j \qquad (7.1.3)$$

Obsevație: Deşi în enunț se specifică realizarea reperelor "exact" în cantitățile b_1, b_2, ..., b_m nu putem impune în (7.1.2) satisfacerea restricțiilor cu egalitate deoarece sistemul $\sum_{j=1}^{n} A^j x_j = b$ s-ar putea să nu aibe soluții întregi nenegative (acest lucru ar avea loc dacă am considera *toate* rețetele, maximale şi nemaximale!). Iată de ce, pentru a asigura compatibilitatea programului (C), suntem nevoiți să admitem că anumite repere pot fi croite "în exces".

În modelarea problemei de croire am putea considera şi un alt criteriu de performanță şi anume *minimizarea restului inutilizabil* reprezentat prin expresia:

$$f' = r(A^1)x_1 + r(A^2)x_2 + \ldots + r(A^n)x_n \qquad (7.1.4)$$

După cum vom vedea în exemplul 7.1.1 cele două funcţii obiectiv (7.1.3) şi (7.1.4) pot conduce la soluţii optime diferite.

În aplicaţiile practice, pe lângă restul inutilizabil trebuie avut în vedere şi aşa numitul *rest utilizabil* reprezentat de suma lungimilor reperelor croite "peste" cantităţile planificate (acesta se mai numeşte şi *supraplan*).

Exemplul 7.1.1 O sucursală a unei firme producătoare de hîrtie produce un carton izolator în rulouri avînd lăţimea de 70 inches (un inch = 2,54 cm). Toate rulourile au aceeaşi lungime. Clienţii firmei solicită rulouri având însă o lăţime mai mică, dar de aceeaşi lungime ca şi ruloul standard. Cererea zilnică este de 100 rulouri de 22 inches, 125 rulouri de 20 inches şi 80 rulouri

de 12 inches lăţime. Rulourile de lăţime mai mică se obţin prin tăiere din
rulourile standard. Firma doreşte să acopere aceste cereri de aşa manieră încât pierderile datorate tăierii să fie minime.

Prin acest exemplu practic nu intenţionăm să ilustrăm aspectele teoretice şi de calcul din domeniul croirii (foarte importante, dar depăşind obiectivele pe care ni le-am propus la începutul paragrafului...). De altfel, problemele de optimizare liniară ce vor apare pe parcurs sunt rezolvate cu ajutorul unor pachete de programe utilitare. Dorim să realizăm o analiză economică a diferitelor variante rezultate din studiu în vederea formulării unei decizii finale cât mai corecte.

Pentru început vom lista toate reţetele maximale de croire; ne putem permite aceasta întrucît numărul lor este, în cazul de faţă, mic.

Reţeta	A_1	A^2	A_3	A_4	A_5	A^6	A^7	A_8	A_9	A^{10}
$l_1 = 22$	3	2	2	1	1	1	0	0	0	0
$l_2 = 20$	0	1	0	2	1	0	3	2	1	0
$l_3 = 12$	0	0	2	0	2	4	0	2	4	5
Rest inutilizabil	4	6	2	8	4	0	10	6	2	10

Tabelul 7.1.1

Acoperirea cererii zilnice conduce la restricțiile:

$$\begin{cases} 3x_1 + 2x_2 + 2x_3 + x_4 + x_5 + x_6 \geq 100 \\ x_2 + 2x_4 + x_5 + 3x_7 + 2x_8 + x_9 \geq 125 \\ 2x_3 + 2x_5 + 4x_6 + 2x_8 + 4x_9 + 5x_{10} \geq 80 \\ x_j \geq 0 \quad j = 1,....,10 \text{ intregi} \end{cases} \qquad (7.1.5)$$

Dacă se urmăreşte minimizarea restului total inutilizabil, la sistemul (7.1.5) ataşăm funcția obiectiv:

$$(\min) f' = 4x_1 + 6x_2 + 2x_3 + 8x_4 + 4x_5 + 10x_7 + 6x_8 + 2x_9 + 10x_{10}$$

În vederea rezolvării problemei relaxate, vom introduce variabilele de abatere x_{11}, x_{12}, x_{13} care vor indica numărul de rulouri de 22, 20, respectiv de 12 inches tăiate peste cantitățile cerute.

			4	6	2	8	4	0	10	6	2	10	0	0	0
CB	B	VVB	$\bar{A}^1$	$\bar{A}^2$	$\bar{A}^3$	$\bar{A}^4$	$\bar{A}^5$	$\bar{A}^6$	$\bar{A}^7$	$\bar{A}^8$	$\bar{A}^9$	$\bar{A}^{10}$	$\bar{A}^{11}$	$\bar{A}^{12}$	$\bar{A}^{13}$
0	A^{13}	820	12	12	6	12	6	0	12	6	0	-5	-4	-4	1
2	A^9	125	0	1	0	2	1	0	3	2	1	0	0	-1	0
0	A^6	100	3	2	2	1	1	1	0	0	0	0	-1	0	0
	f	250	-4	-4	-2	-4	-2	*	-4	-2	*	*	0	-1	*

Tabelul 7.1.2

Din tabelul simplex 7.1.2 rezultă următoarea soluție optimă:

$$x_6^0 = 100\,,\ x_9^0 = 125\,,\ x_{13}^0 = 820\,,\ x_j^0 = 0 \quad j = 1,\ldots,13 \quad j \neq 6,9,13 \quad (\min) f' = 250$$

Ea prevede tăierea a 225 rulouri standard:100 de rulouri după rețeta A^6 şi 125, după rețeta A^9. Resturile inutilizabile însumează 250 inches. Rulourile de 22 şi 20 inches se produc în cantitățile planificate, în schimb apare un surplus de 820 rulouri de 12 inches, cu mult mai mare decât cantitatea cerută. Restul total, utilizabil şi inutilizabil măsoară 820 × 12 + 250 =10090 inches şi reprezintă

$\frac{10090}{225 \times 70} = 64{,}1\%$ din lățimea tuturor rulourilor tăiate!! Este clar că această soluție nu poate fi acceptată.

Dacă se urmărește minimizarea numărului de rulouri standard tăiate, funcția obiectiv *f'* trebuie schimbată în:

$$f = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10}$$

Prin reoptimizare se obține tabelul 7.1.3. În raport cu noul criteriu problema relaxată are o infinitate de soluții optime: nu mai puțin de şase costuri reduse sunt nule! - vezi observația 5) din secțiunea 4.2 şi exemplul 6.2.1.

			1	1	1	1	1	1	1	1	1	1	1	1	1
CB	B	VVB	A^1	A^2	A^3	A^4	A^5	A^6	A^7	A^8	A^9	A^{10}	A^{11}	A^{12}	A^{13}
1	A^7	55/3	-1/2	0	-1/2	1/2	0	-1/2	1	1/2	0	-5/12	1/6	-1/3	1/12
1	A^9	20	0	0	1/2	0	1/2	1	0	1/2	1	5/4	0	0	-1/4
1	A^2	50	3/2	1	1	1/2	1/ 2	1/2	0	0	0	0	-1/2	0	0
	f	265/3	0	*	0	0	0	0	*	0	*	-1/6	-1/6	-1/3	-1/6

Tabelul 7.1.3

Soluția din tabelul 7.1.3 nu este acceptabilă deoarece valoarea variabilei x_7 este fracționară. Aplicând un algoritm adecvat de rezolvare a problemelor de optimizare liniară discretă au fost determinate soluțiile optime întregi din tabelul 7.1.4. Oricare din cele şase soluții listate , deşi prevăd o dublare a restului inutilizabil, sunt mult mai realiste deoarece:

- se taie un număr mult mai mic de rulouri standard: 89 față de 225!
- supraplanul este nesemnificativ ca valoare;
- procentul pierderilor rezultate din tăiere este de numai : $\frac{570}{89 \times 70} \times 100 = 9{,}15\%$

	Retete utilizate	Nr. rulouri standard	Supraplan	Rest utilizabil	Rest inutilizabil	Rest

SOLUTIA	multiplicități			utilizate	22	20	12	(lungime)	(lungime)	total
I	A^2	A^7	A^9							
	51	18	20	89	2	-	-	44	526	570
II	A^2	A^6	A^7							
	41	20	28	89	2	-	-	44	526	570
III	A^2	A^5	A^7							
	31	40	18	89	2	-	-	44	526	570
IV	A^2	A^7	A^9							
	50	18	21	89	-	-	4	48	522	570
V	A^1	A^6	A^7							
	27	20	42	89	1	1	-	42	528	570
VI	A^1	A^4	A^6							
	6	63	20	89	1	1	-	42	528	570

Tabelul 7.1.4

7.2 Probleme cu variabile bivalente

A) Alegerea proiectelor de investiții. O firmă este interesată în mai multe proiecte de investiții pe care ar putea să le realizeze într-o perioadă de 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. Problema constă în alegerea acelor proiecte care să aducă firmei un profit total maxim cu condiția nedepăşirii capitalului disponibil anual.

Introducem variabilele bivalente:

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

Comment:

În ipotezele de liniaritate uzuale obținem problema de programare bivalentă:

$$(\max) f = \sum_{j=1}^{n} c_j x_j$$

cu restricțiile:

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1,\dots,m$$

$$x_j \in \{0,1\} \quad j = 1,\dots,n$$

Utilizarea variabilelor bivalente ne permite să modelăm o serie de situații speciale.Dacă, de exemplu, în problema de mai sus din primele trei proiecte cel mult unul (respectiv exact unul) trebuie realizat, introducem restricția:

$$x_1 + x_2 + x_3 \le 1 \qquad \text{(respectiv } x_1 + x_2 + x_3 = 1\text{)}$$

Este posibil ca un proiect să fie acceptat numai dacă este realizat un altul.Pentru a arăta, de exemplu, că proiectul 7 nu poate fi acceptat dacă proiectul 9 nu se realizează, introducem restricția: $x_7 \le x_9$.

D) Problema monezilor. Această problemă îşi are originea în studiul unei situații în aparență banale, care are loc în orice magazin. S-a observat că numai în puține cazuri clienții plătesc la casă o sumă egală cu contravaloarea mărfurilor cumpărate; de regulă, ei dau o sumă acoperitoare în bancnote şi monezi de valoare mare şi aşteaptă restul. Pentru casier, operația de dare a restului nu este aşa de simplă cum s-ar crede; el trebuie s-o execute rapid şi
corect, mai cu seamă atunci când la casă sunt mai mulți clienți care îşi aşteaptă rândul să plătească. După cum este ştiut, în foarte multe magazine s-a trecut la automatizarea înregistrării mărfurilor cumpărate de clienți şi a evaluării resturilor de plată, modul în care aceste resturi sunt înapoiate clienților rămânând în sarcina casierilor.

Însă, un rest format din multe monezi, de cele mai diverse tipuri valorice (în special mici...) nu este de natură să satisfacă pe client care ar putea să plece nemulțumit. Pe de altă parte, este greu să dai un rest exact utilizând numai monezi de valoare mare. Chestiunea se complică dacă se are în vedere şi faptul că diferitele tipuri valorice de monezi, disponibile pentru plata restului se pot afla la un moment sau altul în cantități neîndestulătoare.Se naşte firesc întrebarea: cum se poate da un rest de plată astfel încât:

- numărul tipurilor valorice de monezi utilizate la plata restului să nu depăşească o limită dată;

-numărul total al monezilor necesare plății să fie minim;

Rezolvarea acestei chestiuni a permis automatizarea operației - cel puțin în marile magazine -contribuind astfel la creşterea gradului de servire al clienților.

În continuare vom arăta cum se formalizează problema descrisă; rezultatul va fi un nou program liniar în numere întregi. Vom folosi următoarele notații:

S = restul 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 pentru plata sumei S;
a_j = valoarea monezii de tip j;
m_j = numărul monezilor de tip j disponibile în casă;

Pentru fiecare $j = 1,..., n$ introducem variabilele:

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

$$y_j = \begin{cases} 1 \text{ daca tipul de moneda } j \text{ este utilizat efectiv la plata sumei S;} \\ 0 \text{ in caz contrar;} \end{cases}$$

Obținem programul:

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

7.3 O problemă de programare mixtă: Program de producție cu costuri de pregătire

Reluăm problema de programare liniară:

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j = b_i \quad i = 1,\dots,m \\ x_j \geq 0 \quad j = 1,\dots,n \\ (\min) f = \sum_{j=1}^{n} c_j x_j \end{cases}$$

în care x_1, x_2,..., x_n reprezintă nivelele unor activități productive iar c_1, c_2,..., c_n costuri unitare acestor activități. Ipoteza uzuală de liniaritate presupune proporționalitatea directă între costul unei activități şi nivelul la care se operează activitatea respectivă. Există situații în care demararea unei activități necesită un cost de pregătire, independent de nivelul la care se va opera ea. Costul activității *j* va avea deci forma:

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

Pentru a încorpora în modelul matematic anterior aceste noi elemente introducem variabilele bivalente:

$$y_j = \begin{cases} 1 \text{ daca activitatea } j \text{ se va opera la un nivel } x_j > 0 \\ 0 \text{ in caz contrar} \end{cases}$$

Rezultă programul:

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

în care m_j reprezintă o margine superioară pentru nivelul x_j al activității j.

Se obține astfel o problemă de optimizare *mixtă* în care o parte din variabile continuă să ia orice valoare nenegativă în timp ce altele nu pot lua decât valori întregi.

7.4 Principiul Branch & Bound de rezolvare a problemelor de programare în numere întregi

Să considerăm un program liniar în numere întregi cu m restricții şi n variabile în formă standard:

$$(P)\begin{cases} Ax = b \\ x \geq 0\ ,\ x \text{ intreg} \\ (\max) f = cx \end{cases}$$

Vom presupune că toate componentele masivelor *A, b, c* sunt numere întregi. Fie $\mathcal{A}$ mulțimea soluțiilor admisibile întregi ale programului (P). Alături de (P) vom considera şi programul *relaxat* (adică fără condiția de integritate impusă variabilelor):

$$(PL)\begin{cases} Ax = b \\ x \geq 0 \\ (\min) f = cx \end{cases}$$

despre care vom presupune că are optim finit. Fie $x^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$ soluția sa optimă. Dacă x^* are toate componentele întregi atunci x^* este şi o soluție optimă a programului (P). Altminteri, soluția optimă întreagă x^0 se va afla undeva în interiorul mulțimii soluțiilor admisibile ale relaxatei (PL) şi pentru a o pune în evidență prin algoritmul simplex vom "sparge" această mulțime în "bucăți" din ce în ce mai "mici" până când x^0 devine vârf pentru unul din "cioburi". Desigur, fiecare "bucată" va fi mulțimea soluțiilor admisibile a unui anumit program liniar.

Pentru a fi mai preciși, să presupunem că în x^* componenta x_1^* este fracționară. Este clar atunci că soluția optimă întreagă x^0 va satisface una din următoarele restricții mutual exclusive:

$$x_1 \le \lfloor x_1^* \rfloor \qquad \text{sau} \qquad x_1 \ge \lfloor x_1^* \rfloor + 1 \qquad (7.4.1)$$

Considerăm programele liniare:

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

Spunem că am "*ramificat*" problema (PL) în raport cu variabila x_1 .Dacă $\mathcal{A}_1$ şi $\mathcal{A}_2$ sunt mulțimile de soluții admisibile întregi ale celor două probleme este clar că:

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

Să considerăm acum soluția optimă x^{*1} a programului (PL_1), în caz că el este compatibil şi să presupunem că a doua componentă a sa x_2^{*1} este fracționară (prima componentă este sigur întreagă: $x_1^{*1} = \lfloor x_1^* \rfloor$!) .Ramificăm ca mai sus problema (PL_1) în raport cu variabila x_2 obținând problemele:

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

Dacă $\mathcal{A}_{11}$, $\mathcal{A}_{12}$ sunt mulțimile de soluții admisibile întregi ale celor două probleme rezultate prin ramificare atunci:

$$A_{11} \cap A_{12} = \varnothing \qquad \text{şi} \qquad A = A_{11} \cup A_{12} \cup A_2$$

În principiu, ramificarea poate continua de la oricare din problemele PL_{11}, PL_{12} sau PL_2, condiția de ramificare fiind aceea ca programul în cauză să fie compatibil iar soluția sa optimă să fie fracționară. De notat că ramificarea unei probleme se poate face în mai multe moduri în funcție de alegerea variabilei a cărei valoare este fracționară. Iată două din ele mai des folosite:

- ramificarea se face după *prima* variabilă cu valoare *fracționară* în soluția optimă curentă.

- ramificarea se face după variabila care, în soluția optimă curentă, are *cea mai mare parte fracționară.*

Am putea vizualiza acest proces de ramificare printr-un *arbore* T ale cărui noduri sunt diferitele probleme rezultate, ca în figura 7.4.1. Nodurile terminale ale acestui arbore sunt fie probleme incompatibile, fie probleme cu soluții optime întregi. Fiecare nod are un unic "predecesor" şi - dacă nu este nod terminal - doi "succesori".

Chestiunea fundamentală este cum conducem acest proces de ramificare pentru a găsi soluția optimă x^0. Pentru aceasta introducem o variabilă z_{CMB} şi o "locație" x_{CMB}; în orice moment al derulării procesului de ramificare x_{CMB} va reține **C**ea **M**ai **B**ună soluție admisibilă întreagă găsită până în acel moment iar z_{CMB} va fi valoarea obiectivului problemei (P) în x_{CMB}.

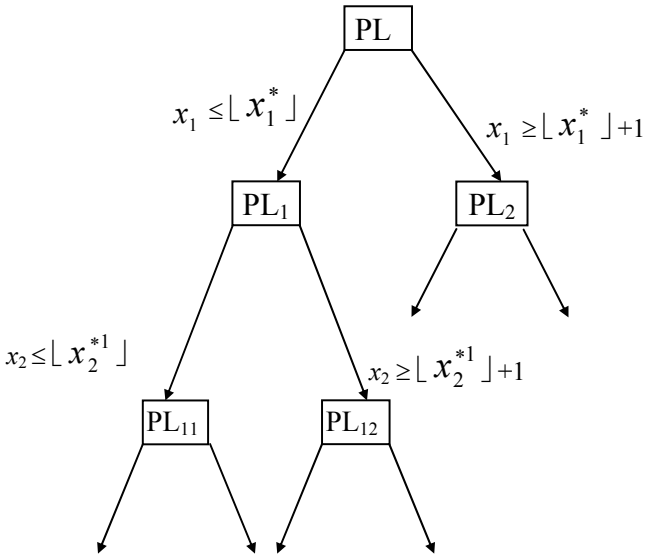

Figura 7.4.1

La start $z_{CMB} = -\infty$ şi $x_{CMB} = \varnothing$ (locația vidă). Fiecare nod PL_α - unde α este o succesiune de 1 şi 2 ! - va avea ataşată o *"margine superioară"* $\bar{z}_\alpha$ reprezentată prin *rotunjirea întreagă inferioară* a optimului problemei PL_α. Este clar că dacă x^0 este o soluție admisibilă pentru PL_α atunci:

$$z_{CMB} \le f(x^0) \le \bar{z}_\alpha$$

Să notăm că în procesul efectiv de ramificare arborele T nu există de la bun început! La start el se reduce la "rădăcina" (PL) şi în continuare primeşte noi noduri şi muchii de legătură în funcţie de problemele rezultate prin ramificare şi efectiv rezolvate. Să presupunem că suntem în nodul (PL_α) al arborelui T. Rezolvăm problema (PL_α). Exceptând problema iniţială (PL), celelalte vor fi rezolvate prin postoptimizare, adăugând la problema "predecesor" una din restricţiile (7.4.1). Dacă (PL_α) este compatibilă fie $x^{*\alpha}$ soluţia sa optimă şi $\bar{z}_\alpha$

marginea superioară definită mai sus. Dacă (PL_α) este incompatibilă vom pune $\bar{z}_\alpha = -\infty$. Sunt posibile mai multe situaţii:

1) $\bar{z}_\alpha \le z_{CMB}$.În acest caz este inutil să continuăm procesul de ramificare din nodul (PL_α) deoarece orice soluţie admisibilă întreagă a problemei (PL_α) nu este "mai bună" decât x_{CMB}. Nodul (PL_α) se declară *"mort"* şi ne întoarcem în unicul predecesor al acestuia din arborele T.

2) $\bar{z}_\alpha > z_{CMB}$. Două cazuri se pot întâmpla:

2.1) *Soluţia optimă $x^{*\alpha}$ a problemei (PL_α) este întreagă.*Actualizăm:

$$x_{CMB} = x^{*\alpha} \ , \ z_{CMB} = \bar{z}_\alpha$$

după care revenim în unicul predecesor al nodului (PL_α) din arborele T.

2.2) *Soluţia optimă $x^{*\alpha}$ are componente fracţionare*. Putem fi într-una din următoarele situaţii:

2.2.1) *Problema (PL_α) este cercetată pentru prima oară.* În acest caz alegem o variabilă, fie ea x_j, a cărei valoare optimă $x_j^{*\alpha}$ este fracţionară. Adăugăm la (PL_α) restricţia $x_j \le \lfloor x_j^{*\alpha} \rfloor$ obţinând problema ($PL_{\alpha 1}$) şi reoptimizăm. Arborele T primeşte un nou nod ($PL_{\alpha 1}$) şi o muchie de legătură care uneşte (PL_α) cu ($PL_{\alpha 1}$). Spunem că ne-am deplasat din (PL_α) "pe ramura din stânga".

2.2.2) *Problema (PL_α) a fost rezolvată într-o fază anterioară* şi ramificată deja după o anumită variabilă, să zicem x_j .Dacă numai problema ($PL_{\alpha 1}$) a fost rezolvată atunci se trece la rezolvarea problemei ($PL_{\alpha 2}$), obţinută

din (PL_α) prin adăugarea restricției $x_j \le \lfloor x_j^{*\alpha} \rfloor + 1$. Arborele T primeşte un nou nod ($PL_{\alpha 2}$) şi o muchie de legătură între (PL_α) şi ($PL_{\alpha 2}$). Spunem că am înaintat din nodul (PL_α) "pe ramura din dreapta".Dacă ambele probleme ($PL_{\alpha 1}$) şi ($PL_{\alpha 2}$) rezultate din ramificarea lui (PL_α) sunt deja rezolvate declarăm nodul (PL_α) "mort" şi ne întoarcem în unicul său predecesor din arborele T.

Este clar acum că procedura se încheie în momentul în care rădăcina (PL) a arborelui T este declarată nod mort. Să mai observăm că de fiecare dată

când se "înaintează" în T dintr-un anumit nod se merge mai întâi pe ramura "din stânga" şi apoi pe cea "din dreapta".

Dacă mulțimea soluțiilor admisibile ale problemei relaxate (PL) este mărginită (fapt care poate fi asigurat cel puțin în aplicațiile practice) atunci algoritmul este convergent în sensul că într-un număr finit de paşi - *înaintări şi retrageri în arborele T* - se obține fie soluția optimă întreagă a problemei originale fie concluzia că ea este incompatibilă, adică nu are soluții admisibile întregi.

Principalul dezavantaj al algoritmului constă în faptul că volumul de calcule creşte *exponențial* o dată cu dimensiunile problemei; ca urmare, el nu poate fi aplicat decât în cazul unor probleme de "talie" mică. Pentru problemele practice, caracterizate în general prin dimensiunile lor impresionante, se utilizează proceduri *euristice* care, fără a determina soluția optimă, furnizează soluții acceptabile cu un efort de calcul rezonabil.

Algoritmul descris este o specializare a unei metode mai generale denumită *Branch and Bound* (ramifică şi mărgineşte) şi folosită la rezolvarea *problemelor de optimizare combinatorială.* Specific problemelor combinatoriale este numărul *finit* de soluții admisibile, *relativ uşor de construit*.În ciuda acestui fapt, ele fac parte din categoria problemelor *grele* deoarece, de regulă, numărul soluțiilor admisibile este *imens*, practic *impredictibil***,** chiar şi în cazul unor probleme de dimensiuni modeste.

În principiu, metoda Branch & Bound "ramifică", adică *partiţionează* mulţimea finită a soluţiilor admisibile ale unei probleme combinatoriale în părţi mai mici pe care "mărgineşte", aceasta însemnând *optimizarea* funcţiei obiectiv pe fiecare din părţile rezultate. Unele din aceste părţi sunt ramificate şi mărginite în continuare ; nu sunt ramificate acele părţi care în mod sigur nu conţin soluţia optimă a problemei! Ideea metodei este deci de a găsi soluţia optimă fără a inspecta toate soluţiile admisibile.

Exemplul 7.4.1 Vom aplica procedura descrisă următorului program liniar în numere întregi:

$$(P)\begin{cases} x_1 + 2x_2 + x_3 \le 130 \\ x_1 + x_2 + 2x_3 \le 100 \\ 2x_1 + x_2 + 3x_3 \le 140 \\ 6x_1 + 9x_2 - 92x_3 \le 0 \\ x_j \ge 0\ ,\ j = 1,2,3 \text{ intregi} \\ (\max) f = 3x_1 + 4x_2 + 2x_3 \end{cases}$$

Ne amintim că relaxata acestui program a fost studiată în exemplul 6.5.2 unde se punea problema maximizării profitului firmei X cu condiţia ca producţia bunului A^3 să reprezinte cel puţin 8% din valoarea întregii producţii. Soluţia optimă găsită acolo nu era acceptată deoarece avea componente fracţionare. Ne propunem acum să determinăm combinaţia optimă cu componente întregi utilizând algoritmul descris mai sus cu precizarea că *ramificarea se va face după variabila cu cea mai mare parte fracţionară în soluţia curentă*.

Iniţializare: Rezolvăm programul relaxat (PL) obţinut din (P) prin omiterea condiţiei de integritate impusă varibilelor x_1, x_2, x_3.(Operaţia a fost deja făcută în exemplul 6.5.2, prin postoptimizare). Rezultă soluţia optimă:

$$x_1 = 3{,}889 \quad x_2 = 42{,}222 \quad x_3 = 6{,}667 \quad (\max) f = 298{,}889$$

Iniţializăm arborele T prin rădăcina sa (PL) căreia îi ataşăm marginea superioară $\bar{z} = 298 = \lfloor 298{,}889 \rfloor$ (nici o soluţie admisibilă întreagă nu poate oferi obiectivului f o valoare mai mare!). Punem $z_{CMB} = -\infty$ şi $x_{CMB} = \varnothing$.

Notă: În forma sa finală, arborele T este vizualizat în fig. 7.4.2. Cititorul este invitat să urmărească etapele ce vor fi parcurse pe această figură

şi mai mult, este îndemnat să reconstituie "în dinamică" construcția arborelui, urmărind indicațiile date în cuprinsul iterațiilor.

Iterația 1. Avem $\bar{z} > z_{CMB}$. Ramificăm (PL) după variabila x_1 a cărei valoare în soluția optimă curentă are cea mai mare parte fracționară. Rezolvăm problema (PL_1), rezultată din (PL) prin adăugarea restricției $x_1 \leq 38$. Se obține soluția, deasemeni fracționară:

$$x_1 = 38 \quad x_2 = 42{,}400 \quad x_3 = 7{,}200 \quad (\max) f = 298{,}000$$

Arborele T primeşte nodul (PL_1) şi o muchie de legătură cu predecesorul (PL). Ataşăm nodului (PL_1) marginea superioară $\bar{z}_1 = 298$.

Iterația 2. Deoarece $\bar{z}_1 > z_{CMB}$, ramificăm (PL_1) după variabila x_2. Adăugăm la (PL_1) restricția $x_2 \leq 42$ şi rezolvăm prin postoptimizare problema (PL_{11}) astfel construită. Se găseşte soluția:

$$x_1 = 38 \quad x_2 = 42 \quad x_3 = 7{,}333 \quad (\max) f = 296{,}667$$

Introducem în T nodul (PL_{11}), muchia de legătură cu predecesorul (PL_1) şi marginea superioară $\bar{z}_{11} = 296$.

Iterația 3. Din nou $\bar{z}_{11} > z_{CMB}$; ramificăm (PL_{11}) după x_3, singura variabilă cu valoare fracționară în soluția optimă curentă. Adăugăm la (PL_{11}) restricția $x_3 \leq 7$ şi rezolvăm problema extinsă (PL_{111}). Obținem prima soluție admisibilă întreagă a problemei originale (P):

$$x_1 = 38 \quad x_2 = 42 \quad x_3 = 7 \quad (\max) f = 296$$

Actualizăm:

$$x_{CMB} = (38, 42, 7) \qquad z_{CMB} = f(x_{CMB}) = 296$$

Nodul (PL_{111}) se declară mort şi ne întoarcem în predecesorul (PL_{11}). Vom adăuga acum la (PL_{11}) restricția $x_3 \geq 8$. Noua problemă (PL_{112}) furnizează o nouă soluție admisibilă întreagă a problemei originale (P):

$$x_1 = 37 \quad x_2 = 42 \quad x_3 = 8 \quad (\max) f = 295$$

dar mai "slabă" decât cea mai bună soluție întreagă găsită pînă acum şi "depozitată" în x_{CMB}. Abandonăm nodul (PL_{112}) şi ne întoarcem în predecesorul (PL_{11}). Deoarece ambele probleme rezultate din ramificarea lui (PL_{11}), după variabila x_3, au fost rezolvate ne întoarcem în nodul (PL_1) - predecesorul lui (PL_{11}). Adăugăm la (PL_1) restricția $x_2 \geq 43$ şi rezolvăm problema (PL_{12}) astfel obținută. Se găseşte soluția fracționară:

$$x_1 = 37{,}357 \quad x_2 = 43 \quad x_3 = 6{,}643 \quad (\max) f = 297{,}357$$

Completăm arborele T cu nodul (PL_{12}) şi cu muchia de legătură corespunzătoare. Ataşăm nodului proaspăt inclus marginea superioară $\bar{z}_{12} = 297$.

Iterația 4. Avem $\bar{z}_{12} = 297 > 296 = z_{CMB}$; s-ar putea ca ($PL_{12}$) să aibe o soluție admisibilă întreagă mai bună decît x_{CMB}. Ramificăm (PL_{12}) după variabila x_3. Mai întâi rezolvăm problema (PL_{121}) dedusă din (PL_{12}) prin adăugarea restricției $x_3 \leq 6$. Rezultă soluția fracționară:

$$x_1 = 27{,}500 \quad x_2 = 43 \quad x_3 = 6 \quad (\max) f = 266{,}500$$

Introducem în T nodul (PL_{121}) pe care-l legăm de predecesorul său (PL_{12}). Deoarece $\bar{z}_{121} = 266 < 296 = z_{CMB}$ orice soluție admisibilă întreagă a problemei (PL_{121}) este mai "slabă" decît x_{CMB}. Ca urmare, abandonăm nodul (PL_{121}) şi ne întoarcem în nodul (PL_{12}). Efectuăm o "înaintare spre dreapta" în arborele T rezolvând problema (PL_{122}) obținută din (PL_{12}) prin extindere cu restricția $x_3 \geq 7$. Se găseşte o soluție admisibilă întreagă mai bună decât actuala x_{CMB}:

$$x_1 = 37 \quad x_2 = 43 \quad x_3 = 7 \quad (\max) f = 297$$

Drept care, facem actualizarea:

$$x_{CMB} = (37, 43, 7) \qquad z_{CMB} = f(x_{CMB}) = 297$$

şi ne întoarcem în nodul (PL_{12}), de acolo în nodul (PL_1), ajungând în final în rădăcina (PL) a arborelui T. Construim problema (PL_2), adăugând la (PL)

restricția $x_1 \geq 39$ (vezi iterația 1). Prin reoptimizare se obține soluția fracționară:

$$x_1 = 39 \quad x_2 = 42{,}034 \quad x_3 = 6{,}655 \quad (\max) f = 298{,}445$$

Arborele T primeşte nodul (PL_2) cu marginea superioară $\bar{z}_2 = 298$.

Iterația 5. Întrucât $\bar{z}_2 = 298 > 297 = z_{CMB}$ ramificăm (PL_2) după variabila x_3. Problema (PL_{21}), obținută din (PL_2) prin completare cu restricția $x_3 \le 6$ are soluția fracționară:

$$x_1 = 45{,}500 \quad x_2 = 31 \quad x_3 = 6 \quad (\max) f = 272{,}500$$

Abandonăm nodul corespunzător (PL_{21}) deoarece $\bar{z}_{21} = 272 < 297 = z_{CMB}$. Adăugând la ($PL_2$) restricția $x_3 \ge 7$ şi rezolvând problema (PL_{22}) astfel construită obținem o soluție admisibilă întreagă mai slabă decît x_{CMB}:

$$x_1 = 39 \quad x_2 = 41 \quad x_3 = 7 \quad (\max) f = 295$$

Întorcându-ne în rădăcina (PL), constatăm că nici o ramificare nu mai este posibilă.În concluzie soluția optimă întreagă a problemei originale (P) este actuala x_{CMB}, adică:

$$x_1^0 = 37 \quad x_2^0 = 43 \quad x_3^0 = 7 \quad (\max) f = z_{CMB} = 297$$