ROZDZIAŁ 2

Wstęp do programowania dyskretnego

Motto:

Dobry Bóg stworzył liczby całkowite, resztę wymyślili ludzie

L. Kronecker

2.1 Definicje i przykłady wprowadzające

Definicja 2.1.1 Przez *zadanie programowania (liniowego) całkowitoliczbowego* rozumiemy zadanie maksymalizacji lub minimalizacji (liniowej) funkcji celu przy ograniczeniach (liniowych) równościowych lub nierównościowych określonej w obszarze $\mathbb{Z}^p \times \mathbb{R}^{n-p}$, gdzie $1 \leq p \leq n$, $n, p \in \mathbb{N}$, czyli w obszarze wektorów o pierwszych p współrzędnych całkowitych. W postaci klasycznej zadanie to ma postać:

$$\begin{array}{ll} \text{maksymalizować} & c^\top x \\ \text{przy ograniczeniach} & Ax \leq b, \\ & x \geq 0, \\ & x_i \in \mathbb{Z} \text{ dla } i = 1, ..., p. \end{array}$$

gdzie $1 \leq p \leq n$. Jeśli $p = n$, to zadanie nazywa się *czysto całkowitoliczbowe*, w przeciwnym wypadku nazywa się ono zadaniem *mieszanym całkowitoliczbowym*.

Definicja 2.1.2 Przez zadanie *programowania (liniowego) dyskretnego* lub *binarnego* rozumiemy zadanie maksymalizacji lub minimalizacji (liniowej) funkcji celu przy ograniczeniach (liniowych) równościowych lub nierównościowych określonej w obszarze $\{0,1\}^n$, czyli w obszarze wektorów o współrzędnych binarnych. W postaci klasycznej zadanie to ma postać:

$$\begin{array}{ll} \text{maksymalizować} & c^\top x \\ \text{przy ograniczeniach} & Ax \leq b, \\ & x \geq 0, \\ & x_i \in \{0,1\} \text{ dla } i = 1, ..., n. \end{array}$$

Podamy teraz kilka przykładów problemów, które można sprowadzić do zadań programowania całkowitoliczbowego lub dyskretnego. Niektóre z tych problemów od razu przedstawimy w postaci takich zadań, inne przedstawimy w takiej postaci w kolejnych rozdziałach.

Przykład 2.1.3 (*Zagadnienie rozkroju*) Z bel papieru o szerokości 210 cm należy wykroić co najmniej:

- 30 rolek o szerokości $a = 62$ cm,
- 60 rolek o szerokości $b = 55$ cm,
- 60 rolek o szerokości $c = 40$ cm.

Wyznaczymy plan rozkroju minimalizujący liczbę bel użytych do rozkroju.

Wśród wszystkich rozkrojów beli papieru można wyróżnić rozkroje efektywne, tzn. takie, dla których pozostałości po rozkroju nie da się dalej rozkroić na rolki o żądanej szerokości. Przykładowo, niech *abcc* oznacza rozkrój beli na 4 rolki o szerokościach a, b, c, c. Nietrudno sprawdzić, że wszystkie możliwe rozkroje efektywne to: *aaa*, *aab*, *aacc*, *abb*, *abcc*, *accc*, *bbbc*, *bbcc*, *bccc*, *ccccc* (ustawione w porządku leksykograficznym). Oznaczmy przez x_i liczbę bel krojonych według i-tego z wyżej wymienionych rozkrojów efektywnych. Wówczas zadanie można sformułować jako zadanie programowania liniowego całkowitoliczbowego w następujący sposób:

$$\begin{array}{ll} \text{minimalizować} & x_1 + ... + x_{10} \\ \text{przy ograniczeniach} & 3x_1 + 2x_2 + 2x_3 + x_4 + x_5 + x_6 \geq 30, \\ & x_2 + 2x_4 + x_5 + 3x_7 + 2x_8 + x_9 \geq 60, \\ & 2x_3 + 2x_5 + 3x_6 + x_7 + 2x_8 + 3x_9 + 5x_{10} \geq 60, \\ & x_1, ..., x_{10} \geq 0, \\ & x_1, ..., x_{10} \in \mathbb{Z}. \end{array}$$

Przykład 2.1.4 (*Całkowitoliczbowe zadanie analizy działalności gospodarczej*) Zakłady lotnicze produkują dwa typy samolotów A i B. W celu wyprodukowania jednego samolotu typu A względnie B trzeba zainwestować 7 względnie 6 mln zł. Z uwagi na popyt, liczba wyprodukowanych samolotów typu B nie może przekroczyć 125% liczby wyprodukowanych samolotów typu A. Zakłady mogą zainwestować maksymalnie 42 mln zł. Zysk ze sprzedaży jednego samolotu typu A względnie B wynosi 10 względnie 9 mln zł. Ile samolotów typu A i B powinny produkować zakłady, aby zapewnić sobie największy zysk?

Niech x_1 względnie x_2 będzie liczbą produkowanych samolotów typu A względnie B. Zadanie to można wówczas sformułować jako zadanie programowania liniowego całkowitoliczbowego w następujący sposób:

$$\begin{array}{ll} \text{maksymalizować} & 10x_1 + 9x_2 \\ \text{przy ograniczeniach} & 7x_1 + 6x_2 \leq 42, \\ & -5x_1 + 4x_2 \leq 0, \\ & x_1, x_2 \geq 0, \\ & x_1, x_2 \in \mathbb{Z}. \end{array}$$

Przykład 2.1.5 (*zagadnienie transportowe*)
W sieci m magazynów A_1,...,A_m składuje się pewien towar. Należy go dostarczyć do sieci n sklepów B_1,...,B_n. Zapas magazynu A_i wynosi a_i jednostek towaru, $i = 1, ..., m$. Sklep B_j potrzebuje b_j jednostek towaru, $j = 1, .., n$. Zakładamy, że

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \tag{2.1}$$

(łączna podaż jest równa łącznemu popytowi). Koszty transportu jednostki towaru z magazynu A_i do sklepu B_j wynoszą c_{ij} jednostek pieniężnych, $i = 1, ..., m$, $j = 1, ..., n$. Należy określić plan transportowy o minimalnych kosztach zaspokajający zapotrzebowanie wszystkich sklepów (czyli, przy podanym założeniu, wyczerpujący łączne zapasy magazynów).

Jeśli przez x_{ij} oznaczymy ilość towaru transportowanego z magazynu A_i do sklepu B_j, $i = 1, ..., m$, $j = 1, ..., n$, to powyższe zagadnienie będzie miało następujący model matematyczny:

$$\begin{array}{rll} \text{minimalizować} & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} & \\ \text{przy ograniczeniach} & \sum_{j=1}^{n} x_{ij} = a_i & i = 1, .., m, \\ & \sum_{i=1}^{m} x_{ij} = b_j & j = 1, ..., n, \\ & x_{ij} \geq 0, & \\ & x_{ij} \in \mathbb{Z}, \quad i = 1, .., m, j = 1, ..., n. & \end{array} \tag{2.2}$$

Przykład 2.1.6 (*Zagadnienie przydziału*) n pracownikom należy przydzielić n czynności o znanych kosztach ich wykonania $c_{ij}, i = 1, ..., n$, $j = 1, ..., n$, przez każdego z pracowników. Przydziału należy dokonać tak, aby:

a) każdy z pracowników wykonywał dokładnie jedną czynność i każda czynność była przydzielona dokładnie jednemu pracownikowi,

b) plan przydziałów miał minimalny koszt.

Niech

$$x_{ij} = \begin{cases} 1 & \text{gdy } i\text{-temu pracownikowi zostanie przydzielona } j\text{-ta czynność,} \\ 0 & \text{poza tym.} \end{cases}$$

Zadanie można wówczas sformułować jako zadanie dyskretnego programowania liniowego w następujący sposób:

$$\begin{array}{ll} \text{minimalizować} & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{przy ograniczeniach} & \sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, ..., n, \\ & \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, ..., n, \\ & x_{ij} \in \{0, 1\}, \qquad i, j = 1, .., n. \end{array}$$

Zwróćmy przy okazji uwagę na fakt, że zadanie to jest szczególnym przypadkiem zadania transportowego w obszarze zmiennych binarnych.

Przykład 2.1.7 (*Zagadnienie plecakowe*) Turysta może wziąć do plecaka 4 różne przedmioty o użyteczności 3, 4, 2 względnie 3 i o ciężarze 3, 2, 4 względnie 1. Maksymalny ciężar wziętych przedmiotów wynosi 9. Które przedmioty powinien zabrać ze sobą turysta, aby przy zachowaniu podanego maksymalnego ciężaru osiągnąć maksymalną użyteczność?

Niech

$$x_j = \begin{cases} 1 & \text{jeśli turysta weźmie ze sobą } j\text{-ty przedmiot,} \\ 0 & \text{poza tym.} \end{cases}$$

Zadanie można wówczas sformułować jako zadanie dyskretnego programowania liniowego w następujący sposób:

$$\begin{array}{ll} \text{maksymalizować} & 3x_1 + 4x_2 + 2x_3 + 3x_4 \\ \text{przy ograniczeniu} & 3x_1 + 2x_2 + 4x_3 + x_4 \leq 9, \\ & x_j \in \{0, 1\}, \qquad j = 1, 2, 3, 4. \end{array}$$

Przykład 2.1.8 (*Minimalne drzewo rozpinające*) Obóz składa się z kilkunastu namiotów. Jak połączyć namioty ścieżkami tak, żeby można było przejść od każdego do każdego po ścieżkach i żeby przy tym łączna długość ścieżek była możliwie najmniejsza?

Przykład 2.1.9 (*Najkrótsza droga*) Należy znaleźć najkrótszą (najszybszą, najtańszą) drogę z Campusu A do Campusu B Uniwersytetu Zielonogórskiego.

Przykład 2.1.10 (*Zagadnienie komiwojażera*) Handlowiec chce odwiedzić n miejscowości (klientów) startując ze swojej miejscowości i wracając tam po skończonej podróży. W jakiej kolejności powinien odwiedzać te miejscowości, aby łączna przebyta trasa (łączne koszty podróży) była(y) minimalna(e).

Niech

$$x_{ij} = \begin{cases} 1 & \text{jeśli bezpośrednio po } i\text{-tej odwiedzana będzie } j\text{-ta miejscowość,} \\ 0 & \text{poza tym.} \end{cases}$$

Ponadto niech c_{ij} oznacza odległość (czas podróży, koszt podróży) od i-tej do j-tej miejscowości, $i, j = 1, ...n$. Zadanie można wówczas sformułować jako zadanie dyskretnego programowania liniowego w następujący sposób:

$$\begin{array}{ll} \text{minimalizować} & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{przy ograniczeniach} & \sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, ..., n, \\ & \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, ..., n, \\ & x_{ij} \in \{0, 1\}, \qquad i, j = 1, .., n \\ & \text{i krótkie cykle są zabronione.} \end{array}$$

Ostatnie ograniczenie gwarantuje, że trasa nie rozłoży się na kilka rozłącznych cykli. W jednym z kolejnych rozdziałów zapiszemy to ograniczenie w innej postaci.

Uwaga 2.1.11 Każde dyskretne zadanie programowania (liniowego) można sprowadzić do (liniowego) zadania całkowitoliczbowego zastępując każde ograniczenie $x_i \in \{0, 1\}$ ograniczeniem $0 \leq x_i \leq 1$, $x_i \in \mathbb{Z}$, $i = 1, ..., n$.

Uwaga 2.1.12 Każde zadanie (liniowego) programowania całkowitoliczbowego z ograniczonymi zmiennymi można sprowadzić do (liniowego) zadania programowania dyskretnego zastępując każdą ograniczoną zmienną całkowitoliczbową $0 \leq x_i \leq d_i$, $x_i \in \mathbb{Z}$, jej rozwinięciem dwójkowym

$$x_i = 2^0 y_1 + 2^1 y_2 + ...2^{r_j - 1} y_{r_j},\ y_j \in \{0, 1\},\ j = 1, ..., r_i,$$

$i = 1, ..., n$. W wielu zadaniach optymalizacji zmienne są albo z definicji ograniczone albo wartości je ograniczające można wyznaczyć. Tak więc ograniczoność zmiennych nie jest istotnym zawężeniem rozważań. Bardziej istotnym problemem jest fakt, że omawiane sprowadzenie do zadania programowania dyskretnego nie jest najczęściej efektywne.

Przykład 2.1.13 Rozważmy zadanie programowania całkowitoliczbowego omawiane w przykładzie 2.1.4:

$$\begin{array}{rr} \text{maksymalizować} & 10x_1 + 9x_2 \\ \text{przy ograniczeniach} & 7x_1 + 6x_2 \leq 42 \\ & -5x_1 + 4x_2 \leq 0 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}. \end{array}$$

Rozwiązanie optymalne tego zadania bez uwzględnienia całkowitoliczbowości zmiennych wynosi $x^* = (2\frac{26}{29}, 3\frac{18}{29}) \simeq (2.90, 3.62)$ z optymalną wartością funkcji celu $z^* \simeq 61.55$. Rozwiązanie to nie jest jednak całkowitoliczbowe. Najbliższa od x^* para o współrzędnych całkowitych $x' = (3, 4)$ nie spełnia warunków zadania. Najbliższe od x^* rozwiązanie dopuszczalne wynosi $x'' = (3, 3)$ z wartością funkcji celu $z'' = 57$. Nie jest to jednak rozwiązanie optymalne. Jest nim bowiem $\overline{x} = (6, 0)$ z optymalną wartością funkcji celu $\overline{z} = 60$. Zauważmy przy okazji, że w omawianym przykładzie $\overline{x}$ jest najbardziej odległym od x^* rozwiązaniem dopuszczalnym.

2.2 Relaksacje zadań programowania dyskretnego

Zadania programowania dyskretnego są znacznie trudniejsze do rozwiązania aniżeli zadania programowania liniowego. Dlatego często pomocniczo rozwiązuje się pewne zastępcze, prostsze zadania, które są w pewnym sensie powiązane z danym zadaniem programowania dyskretnego. Takie pomocnicze zadania nazywa się relaksacją (osłabieniem) danego zadania, o ile spełnione są pewne warunki. Zanim podamy te warunki wprowadzimy pewne oznaczenia. Niech D będzie pewnym zbiorem skończonym, na którym określona jest pewna funkcja f. Niech ponadto dany będzie pewien zbiór D' (niekoniecznie skończony) i funkcja g określona na zbiorze D'.

Definicja 2.2.1 Zadanie

$$\begin{array}{ll} \text{maksymalizować} & g(x) \\ \text{przy ograniczeniach} & x \in D' \end{array} \tag{2.3}$$

nazywamy *relaksacją* zadania

$$\begin{array}{ll} \text{maksymalizować} & f(x) \\ \text{przy ograniczeniach} & x \in D, \end{array} \tag{2.4}$$

jeśli

(i) $D \subset D'$,

(ii) $f(x) \leq g(x)$ dla każdego $x \in D$.

Uwaga 2.2.2 Przy odpowiednim doborze funkcji g i zbioru D' zadanie (2.3) jest łatwiej rozwiązać niż zadanie (2.4). Jeśli stwierdzimy, że dla rozwiązania optymalnego zadania (2.3) zachodzi $f(x^*) = g(x^*)$ i $x^* \in D$, to x^* będzie rozwiązaniem optymalnym zadania (2.4).

Przykład 2.2.3 Zadanie programowania liniowego

$$\begin{array}{rl} \text{maksymalizować} & c^\top x \\ \text{przy ograniczeniach} & Ax \leq b \\ & x \geq 0 \end{array}$$

jest relaksacją zadania programowania liniowego całkowitoliczbowego

$$\begin{array}{rl} \text{maksymalizować} & c^\top x \\ \text{przy ograniczeniach} & Ax \leq b \\ & x \geq 0 \\ & x_i \in \mathbb{Z} \text{ dla } i = 1, ..., n. \end{array}$$

Relaksacja ta nazywa się *relaksacją LP*.

Przykład 2.2.4 Zadanie programowania liniowego

$$\begin{array}{rl} \text{maksymalizować} & c^\top x \\ \text{przy ograniczeniach} & Ax \leq b \\ & 0 \leq x \leq 1 \end{array}$$

jest relaksacją zadania programowania liniowego dyskretnego

$$\begin{array}{rl} \text{maksymalizować} & c^\top x \\ \text{przy ograniczeniach} & Ax \leq b \\ & x \geq 0 \\ & x_i \in \{0, 1\} \text{ dla } i = 1, ..., n. \end{array}$$

Przykład 2.2.5 Zadanie przydziału

$$\begin{array}{rll} \text{minimalizować} & & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{przy ograniczeniach} & \sum_{j=1}^{n} x_{ij} = 1, & i = 1,...,n, \\ & \sum_{i=1}^{n} x_{ij} = 1, & j = 1,...,n, \\ & x_{ij} \in \{0,1\}, & i,j = 1,..,n \end{array}$$

jest relaksacją zadania komiwojażera

$$\begin{array}{rll} \text{minimalizować} & & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{przy ograniczeniach} & \sum_{j=1}^{n} x_{ij} = 1, & i = 1,...,n, \\ & \sum_{i=1}^{n} x_{ij} = 1, & j = 1,...,n, \\ & x_{ij} \in \{0,1\}, & i,j = 1,..,n \\ & \text{i krótkie cykle są zabronione.} & \end{array}$$

Przykład 2.2.6 Rozważmy teraz zadanie programowania liniowego całkowitoliczbowego

$$\begin{array}{rr} \text{maksymalizować} & c^\top x \\ \text{przy ograniczeniach} & Ax \le b \\ & x \ge 0 \\ & x_i \in \mathbb{Z} \text{ dla } i = 1,...,n. \end{array}$$

i niech $u \ge 0$ będzie pewnym ustalonym wektorem, $u \in \mathbb{R}^m$. Zadanie

$$\begin{array}{rr} \text{maksymalizować} & c^\top x \\ \text{przy ograniczeniach} & u^\top Ax \le u^\top b \\ & x \ge 0 \\ & x_i \in \mathbb{Z} \text{ dla } i = 1,...,n. \end{array}$$

jest relaksacją rozpatrywanego zadania.

2.2.1 Relaksacja Lagrange'a

Opiszemy teraz pewną metodę, tzw. metodę *relaksacji Lagrange'a*, stosowaną czasem w zadaniach programowania liniowego całkowitoliczbowego. W zadaniach tych często występują ograniczenia, które są niewygodne przy bezpośrednim rozwiązaniu zadania. Przykładem tego może być zagadnienie komiwojażera, dla którego takim niewygodnym ograniczeniem jest ograniczenie wyrażone słownie "krótkie cykle są zabronione", a które można przedstawić również w postaci odpowiednich nierówności. Ze względu na zastosowania przedstawimy więc relaksację Lagrange'a dla zadania programowania liniowego całkowitoliczbowego postaci

$$\begin{array}{ll} \text{maksymalizować} & f(x) = c^\top x \\ \text{względem} & x \in \mathbb{Z}^n, \\ \text{przy ograniczeniach} & Ax \le b. \end{array} \tag{2.5}$$

Przypuśćmy, że istnieje rozwiązanie optymalne tego zadania i oznaczmy jego wartość optymalną symbolem f^*. Przypuśćmy ponadto, że ograniczenia nierównościowe tego zadania złożone są z dwóch części: $A_1x \le b_1$ i $A_2x \le b_2$, gdzie $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ i $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, przy czym ograniczenia $A_1x \le b_1$ są w pewnym sensie niewygodne. Przykładem takich niewygodnych ograniczeń są

ograniczenia "krótkie cykle są zabronione" w zagadnieniu komiwojażera, które jak wcześniej powiedzieliśmy można zapisać w postaci nierówności liniowych. Zdefiniujmy zbiór X w postaci $X = \{x \in \mathbb{Z}^n : A_2 x \leq b_2\}$. Rozważane zadanie przyjmuje zatem postać

$$\begin{array}{ll} \text{maksymalizować} & f(x) = c^\top x \\ \text{względem} & x \in X \\ \text{przy ograniczeniach} & A_1 x \leq b_1. \end{array} \tag{2.6}$$

Zadanie

$$\begin{array}{ll} \text{maksymalizować} & c^\top x + \lambda^\top (b_1 - A_1 x) \\ \text{względem} & x \in X, \end{array} \tag{2.7}$$

gdzie $\lambda \geq 0$ jest wektorem odpowiedniego wymiaru, jest relaksacją rozpatrywanego zadania (2.5), gdyż dla dowolnego wektora x spełniającego ograniczenia zadania (2.5) i dla dowolnego $\lambda \geq 0$ zachodzi nierówność

$$c^\top x \leq c^\top x + \lambda^\top (b_1 - A_1 x) \tag{2.8}$$

oraz zbiór D rozwiązań dopuszczalnych zadania (2.5) jest zawarty w zbiorze rozwiązań dopuszczalnych D' zadania (2.7). Możemy więc powiedzieć, że pewne (niewygodne) ograniczenia zostały dodane do funkcji celu z pewnymi nieujemnymi wagami (mnożnikami Lagrange'a). Zadanie (2.7) ma zatem prostszy zbiór rozwiązań dopuszczalnych i jest najczęściej znacznie prostsze do rozwiązania, aniżeli zadanie (2.5). Z nierówności (2.8) wynika, że dla dowolnego $\lambda \geq 0$ liczba

$$h(\lambda) = \sup_{x \in X} \{c^\top x + \lambda^\top (b_1 - A_1 x)\}$$

jest górnym ograniczeniem wartości optymalnej zadania (2.5). Chodzi nam jednak o to, aby ograniczenie to jak najmniej odbiegało od tej wartości optymalnej. Należy w tym celu dobrać wektor wag $\lambda \geq 0$ tak, aby wartość $h(\lambda)$ była możliwie najmniejsza. W ten sposób dochodzimy do zadania

$$\begin{array}{rl} \text{minimalizować} & h(\lambda) = \sup_{x \in X} \{b_1^\top \lambda + (c - A_1^\top \lambda)^\top x\} \\ \text{względem} & \lambda \geq 0. \end{array}$$

Zadanie to jest zadaniem minimalizacji wypukłej, ale najczęściej nieróżniczkowalnej. Istnieją jednak efektywne metody rozwiązania takiego zadania, o których będziemy mówić w jednym z kolejnych rozdziałów. Należy jednak podkreślić, że tzw. *luka dualności*

$$\inf_{\lambda \geq 0} h(\lambda) - f^*$$

jest najczęściej większa od zera. Niemniej jednak $\inf_{\lambda \geq 0} h(\lambda)$ jest najlepszym (z możliwych do uzyskania stosunkowo niedużym kosztem) górnym ograniczeniem wartości optymalnej rozważanego zadania.

2.3 Metoda Gomory'ego

Rozpocznijmy od następującego przykładu.

Przykład 2.3.1 Rozważmy następujące zadanie programowania liniowego całkowitoliczbowego:

$$\begin{array}{rr} \text{maksymalizować} & x_1 + x_2 \\ \text{przy ograniczeniach} & x_1 + 2x_2 \leq 32 \\ & 18x_1 + 3x_2 \leq 224 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}. \end{array} \tag{P}$$

Opuśćmy najpierw ograniczenie całkowitoliczbowości zmiennych i rozwiążmy przy pomocy metody sympleksowej powstałe w ten sposób zadanie (P') będące relaksacją LP zadania . Otrzymamy następującą początkową tablicę sympleksową:

	$-x_1$	$-x_2$	
0	-1	-1	$= z$
32	1	2	$= u_1$
224	18	3	$= u_2$

.

Po dwóch piwotyzacjach otrzymamy tablicę

	$-u_2$	$-u_1$	
$\frac{64}{3}$	$\frac{1}{33}$	$\frac{5}{11}$	$= z$
$\frac{32}{3}$	$-\frac{1}{33}$	$\frac{6}{11}$	$= x_2$
$\frac{32}{3}$	$\frac{2}{33}$	$-\frac{1}{11}$	$= x_1$

. (2.9)

Rozwiązanie optymalne zadania (P') $x^* = (10\frac{2}{3}, 10\frac{2}{3}), u^* = (0, 0)$ nie spełnia warunku całkowitoliczbowości zmiennych zadania (P). Zauważmy, że w zadaniu (P) zmienne uzupełniające u_1, u_2 są całkowite, gdyż wszystkie dane w tym zadaniu są całkowite.

Z równania

$$x_1 - \frac{1}{11}u_1 + \frac{2}{33}u_2 = 10\frac{2}{3} \tag{2.10}$$

wynika nierówność

$$x_1 - u_1 \leq 10\frac{2}{3},$$

gdyż $-1 = \left[-\frac{1}{11}\right] \leq -\frac{1}{11}$ i $0 = \left[\frac{2}{33}\right] \leq \frac{2}{33}$. Ponieważ x_1, u_1 są całkowite, ostatnia nierówność jest równoważna nierówności

$$x_1 - u_1 \leq 10$$

lub inaczej – układowi

$$\begin{array}{rcl} x_1 - u_1 + u_3 & = & 10 \\ u_3 & \geq & 0. \end{array}$$

Jeśli teraz od ostatniego równania odejmiemy równanie (2.10), to otrzymamy równanie

$$-\frac{10}{11}u_1 - \frac{2}{33}u_2 + u_3 = -\frac{2}{3}.$$

Zauważmy, że rozwiązanie optymalne zadania (P'), $x^* = (10\frac{2}{3}, 10\frac{2}{3}), u^* = (0,0)$ nie spełnia ostatniego równania, gdyż u_3 musi byś nieujemne. Równanie to musi być jednak spełnione przez dowolne rozwiązanie optymalne zadania (P). Jeśli dołączymy to równanie do układu przedstawionego tablicą (2.9), to otrzymamy tablicę:

	$-u_2$	$-u_1$	
$\frac{63}{3}$	$\frac{1}{33}$	$\frac{5}{11}$	$= z$
$\frac{32}{3}$	$-\frac{1}{33}$	$\frac{6}{11}$	$= x_2$
$\frac{32}{3}$	$\frac{2}{33}$	$-\frac{1}{11}$	$= x_1$
$-\frac{2}{3}$	$-\frac{2}{33}$	$-\frac{10}{11}$	$= u_3$

.

Związane z tą tabelą zadanie maksymalizacji (P") jest równoważne zadaniu (P) i może być rozwiązane na przykład przy pomocy dualnego algorytmu sympleksowego. Po jednej jego iteracji otrzymamy tablicę:

	$-u_3$	$-u_1$	
21	$\frac{1}{2}$	0	$= z$
11	$-\frac{1}{2}$	1	$= x_2$
10	1	-1	$= x_1$
11	$-\frac{33}{2}$	15	$= u_2$

.

Tablica ta przedstawia rozwiązanie optymalne zadania (P"), $x'' = (10, 11)$. Ponieważ rozwiązanie to jest całkowitoliczbowe, więc jest ono również rozwiązaniem optymalnym zadania (P) (patrz uwaga 2.2.2).

W powyższym przykładzie zastosowaliśmy tzw. *metodę cięć Gomory'ego*, którą opiszemy teraz dla ogólnego przypadku zadania programowania liniowego czysto całkowitoliczbowego:

$$\begin{aligned} &\text{maksymalizować} && c^\top x \\ &\text{przy ograniczeniach} && Ax \leq b \\ & && x \geq 0 \\ & && x_i \in \mathbb{Z},\ i = 1, ..., n, \end{aligned} \tag{2.11}$$

przy czym elementy A i b są liczbami całkowitymi. Najpierw należy rozwiązać powyższe zadanie bez uwzględniania ograniczenia całkowitoliczbowości zmiennych, czyli relaksację LP zadania (2.11). Jeśli rozwiązanie optymalne x^* tego zadania jest całkowitoliczbowe, to jest ono równocześnie rozwiązaniem optymalnym wyjściowego zadania (2.11). W przeciwnym wypadku niech

1	$-r_1$	...	$-r_n$	
h_{00}	h_{01}	...	h_{0n}	$= z$
h_{10}	h_{11}	...	h_{1n}	$= s_1$
...	...	...	...	...
h_{i0}	h_{i1}	...	h_{in}	$= s_i$
...	...	...	...	...
h_{m0}	h_{m1}	...	h_{mn}	$= s_m$

(2.12)

będzie tablicą sympleksową odpowiadającą temu rozwiązaniu, przy czym wartość h_{i0} nie jest całkowita. i-te równanie układu przedstawionego tablicą (2.12) ma postać

$$s_i + \sum_{j=1}^{n} h_{ij} r_j = h_{i0}. \tag{2.13}$$

Ponieważ $[h_{ij}] \leq h_{ij}$ i $r_j \geq 0$, więc zachodzi nierówność

$$s_i + \sum_{j=1}^{n} [h_{ij}]\, r_j \leq h_{i0}.$$

W konsekwencji

$$s_i + \sum_{j=1}^{n} [h_{ij}]\, r_j \leq [h_{i0}]\,,$$

ponieważ wszystkie zmienne są całkowite. Ostatnia nierówność jest równoważna układowi

$$\begin{aligned} s_i + \sum_{j=1}^{n} [h_{ij}]\, r_j + u_{m+1} &= [h_{i0}] \\ u_{m+1} &\geq 0. \end{aligned} \tag{2.14}$$

Jeśli teraz odejmiemy równanie (2.13) od równania (2.14), otrzymamy równanie

$$\sum_{j=1}^{n} ([h_{ij}] - h_{ij})\, r_j + u_{m+1} = [h_{i0}] - h_{i0}, \tag{2.15}$$

przy czym $u_{m+1} \geq 0$. Równanie to "odetnie" rozwiązanie x^*, ponieważ $[h_{i0}] - h_{i0} < 0$ i wszystkie zmienne niebazowe $r_j = 0$. Równanie to musi być jednak spełnione dla każdego dopuszczalnego (czyli całkowitoliczbowego) rozwiązania zadania (2.11) i dla pewnego $u_{m+1} \geq 0$. Jeśli więc równanie to dołączymy jako dodatkowe ograniczenie do wyjściowego zadania, otrzymamy zadanie równoważne. W praktyce równanie to dołącza się do tablicy sympleksowej (2.12). W efekcie otrzymamy tablicę

1	$-r_1$	...	$-r_n$	
h_{00}	h_{01}	...	h_{0n}	$= z$
h_{10}	h_{11}	...	h_{1n}	$= s_1$
...	...	...	...	...
h_{i0}	h_{i1}	...	h_{in}	$= s_i$
...	...	...	...	...
h_{m0}	h_{m1}	...	h_{mn}	$= s_m$
$[h_{i0}] - h_{i0}$	$[h_{i1}] - h_{i1}$	...	$[h_{in}] - h_{in}$	$= u_{m+1}$

,

która nie przedstawia rozwiązania optymalnego, gdyż $[h_{i0}] - h_{i0} < 0$. Tablicę tę można dalej piwotyzować, najlepiej przy pomocy dualnego algorytmu sympleksowego, z uwagi na dualną dopuszczalność. Jeśli otrzymane rozwiązanie będzie całkowitoliczbowe, to będzie ono rozwiązaniem wyjściowego zadania. W przeciwnym wypadku opisaną procedurę należy powtarzać aż do otrzymania rozwiązania całkowitoliczbowego.

Przykład 2.3.2 Wyznaczyć rozwiązanie zadania programowania liniowego całkowitoliczbowego

$$\begin{aligned} \text{maksymalizować} \quad & x_1 + 2x_2 \\ \text{przy ograniczeniach} \quad & 2x_1 + 3x_2 \leq 27 \\ & 2x_1 - 2x_2 \leq 7 \\ & -6x_1 - 2x_2 \leq -9 \\ & -2x_1 - 6x_2 \leq -11 \\ & -6x_1 + 8x_2 \leq 21 \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned}$$

2.4 Metody podziału i ograniczeń

Metody podziału i ograniczeń (ang. *branch and bound methods*) stosuje się do zadań maksymalizacji (bądź minimalizacji) funkcji f na zbiorze skończonym G:

$$\max_{x \in G} f(x)$$

Opiszemy te metody dla zadań maksymalizacji. Jest to cała klasa metod posiadających pewne wspólne cechy. Na pewnej rodzinie zbiorów $\mathcal{D} \subset 2^G$ zawierającej zbiór G oraz wszystkie jego jednoelementowe podzbiory definiuje funkcję φ (*ograniczenie górne*) posiadającą następujące własności:
1^0 $G_1 \subset G_2 \in \mathcal{D} \Rightarrow \varphi(G_1) \leq \varphi(G_2)$
2^0 $a \in G \Rightarrow \varphi(\{a\}) = f(a)$

Funkcję φ dobiera się specjalnie do rodzaju zadania.

Uwaga 2.4.1 Z własności $1^0 - 2^0$ wynika, że funkcja φ ma również własność

$$x \in G \in \mathcal{D} \Rightarrow f(x) = \varphi(\{x\}) \leq \varphi(G),$$

czyli

$$\varphi(G) \geq \max_{x \in G} f(x).$$

Z tego powodu nazywa się ona *ograniczeniem górnym*. Oczywiście własność ta przysługuje dowolnemu podzbiorowi $G_i \in \mathcal{D}$, czyli

$$\varphi(G_i) \geq \max_{x \in G_i} f(x).$$

Przykład 2.4.2 Rozważmy następujące zadanie programowania liniowego czysto całkowitoliczbowego

$$\begin{array}{rr} \text{maksymalizować} & c^\top x \\ \text{przy ograniczeniach} & Ax \leq b, \\ & 0 \leq x \leq h, \\ & x \in \mathbb{Z}^n, \end{array}$$

przy czym $h \in \mathbb{R}^n$. Niech G będzie zbiorem rozwiązań dopuszczalnych tego zadania:

$$G = \{x \in \mathbb{R}^n : Ax \leq b, 0 \leq x \leq h, x \in \mathbb{Z}^n\},$$

Oczywiście G jest zbiorem skończonym. Niech rodzina $\mathcal{D}$ składa się ze wszystkich zbiorów $G(d; g) \subset G$ postaci:

$$G(d; g) = \{x \in \mathbb{R}^n : Ax \leq b, d \leq x \leq g, x \in \mathbb{Z}^n\},$$

gdzie $d, g \in \mathbb{R}^n, 0 \leq d \leq g \leq h$. Zdefiniujmy

$$\varphi(G(d; g)) = \max\{c^\top x : Ax \leq b, d \leq x \leq g\}.$$

Przyjmujemy przy tym, że $\varphi(\emptyset) = -\infty$. Nietrudno zauważyć, że funkcja φ ma własności $1^0 - 2^0$, gdzie $f(x) = c^\top x$.

W każdej iteracji dowolnej metody podziału i ograniczeń zbiór G rozkłada się na (możliwie rozłączne) podzbiory $G_1, G_2, ..., G_{m_k} \in \mathcal{D}$:

$$G = \bigcup_{i=1}^{m_k} G_i.$$

Innymi słowy, zadanie $\max_{x \in G} f(x)$ zostaje rozłożone (lub podzielone) na zadania cząstkowe $\max_{x \in G_i} f(x)$, $i = 1, ..., m_k$ (*branching*).

Następnie wylicza się ograniczenia górne $\varphi(G_i)$ dla zadań cząstkowych

$$\max_{x \in G_i} f(x)$$

$i = 1, 2, ..., m_k$ (*bounding*). Na podstawie wartości tych ograniczeń górnych rozstrzyga się, które ze zbiorów G_i muszą by dalej rozłożone. Jeśli przy okazji stwierdzi się, że

$$\varphi(G_i) = \max_{x \in G_i} f(x),$$

to wartości te mogą służyć jako *ograniczenie dolne* optymalnej wartości funkcji celu $f^* = \max_{x \in G} f(x)$. W tym przypadku w zbiorze G_i nie ma bowiem elementu x, dla którego $f(x) > \varphi(G_i)$, a ponadto

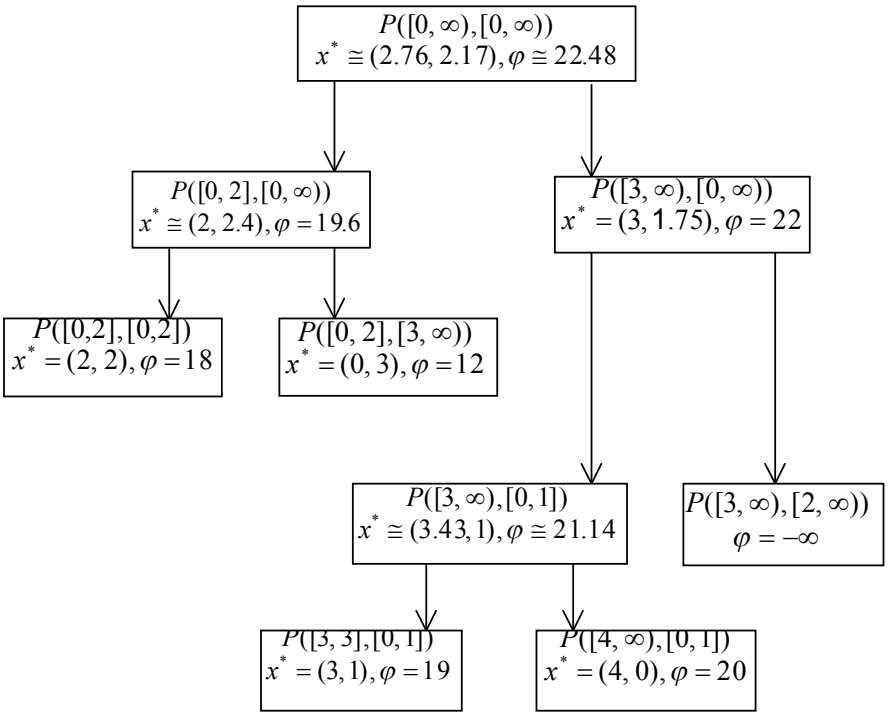
$$f^* = \max_{x \in G} f(x) \geq \max_{x \in G_i} f(x) = \varphi(G_i).$$

Przykład 2.4.3 Rozważmy następujące zadania programowania liniowego całkowitoliczbowego:

$$\begin{aligned} \text{maksymalizować} \quad & f(x) = 5x_1 + 4x_2 \\ \text{przy ograniczeniach} \quad & 7x_1 + 4x_2 \leq 28 \\ & 3x_1 + 10x_2 \leq 30 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned} \tag{2.16}$$

Oznaczmy przez $P([a_1, b_1], [a_2, b_2])$ zadanie (2.16) z dodatkowymi ograniczeniami $a_1 \leq x_1 \leq b_1, a_2 \leq x_2 \leq b_2$. Dalej, oznaczmy każdorazowo przez φ maksymalną wartość funkcji celu dla zadania cząstkowego $P([a_1, b_1], [a_2, b_2])$ z pominięciem założenia o całkowitoliczbowości zmiennych. Jako ograniczenie dolne $\underline{\varphi}$ możemy przyjąć wartość funkcji celu dla dowolnego rozwiązania dopusczalnego zadania (2.16), np $\underline{\varphi} = f(0, 0) = 0$. Rozwiązujemyu najpierw (na przykład metodą sympleksową) relaksację LP zadania (2.16), czyli zadanie $P([0, \infty), [0, \infty))$. Otrzymujemy rozwiązanie $x^* \cong (2.76, 2.17)$ z wartością optymalną $f(x^*) \cong 22.48$. Wartość tę przyjmujemy jako ograniczenie górne $\bar{\varphi} \cong 22.48$ dla zadania (2.16). Rozwiązanie x^* zadania $P([0, \infty), [0, \infty))$ nie jest całkowitoliczbowe, nie może być więc rozwiązaniem optymalnym zadania (2.16). Wiemy natomiast, że wartość optymalna zadania (2.16) mieści się w przedziale $[0, 22.48]$. Ponieważ $x_1^* \cong 2.76$ nie jest liczbą całkowitą, dzielimy zadanie $P([0, \infty), [0, \infty))$ na dwa zadania $P([0, 2], [0, \infty))$ i $P([3, \infty), [0, \infty))$, bowiem pierwsza wsołrzędna x_1 dla dowolnego rozwiązania dopuszczalnego zadania (2.16) musi być całkowita, czyli $x_1 \in [0, 2)$ albo $x_1 \in [3, \infty)$. Rozwiązujemy oba zadania $P([0, 2], [0, \infty))$ i $P([3, \infty), [0, \infty))$ otrzymując dla nich rozwiązania optymalne i wartości optymalne równe $x^* \cong (2, 2.4)$ i $\varphi = 19.6$ dla zadania $P([0, 2], [0, \infty))$ oraz $x^* = (3, 1.75)$ i $\varphi = 22$ dla zadania $P([3, \infty), [0, \infty))$. Zadne z tych rozwiązań nie jest całkowitoliczbowe. Możemy natomiast stwierdzić, że $\bar{\varphi} = \max\{19.6, 22\} = 22$. Zawęziliśmy więc przedział, w którym znajduje się wartość optymalna do $[0, 22]$. Dla drugiego z zadań

cząstkowych możemy wybrać rozwiązanie dopuszczalne, np. $x = (3, 1)$ i wyznaczyć $f(3, 1) = 19$. Wiemy zatem, że wartość optymalna nie może być mniejsza od 19. Możemy więc znowy zawęzić przedział, w którym znajduje się wartość optymalna do $[19, 22]$. Ponieważ dla problemu cząstkowego $P([3, \infty), [0, \infty))$ ograniczenie górne $\varphi = 22$ jest większe niż dla problemu cząstkowego $P([0, 2], [0, \infty))$ wynoszące $\varphi = 19.6$, więc spodziewamy się, że rozwiązanie optymalne wyjściowego zadania będzie w zbiorze $[3, \infty) \times [0, \infty)$. Dlatego dzielimy ten problem na dwa problemy $P([3, \infty), [0, 1))$ i $P([3, \infty), [2, \infty))$. Dalsze elementy procesu pozostawiamy czytelnikowi jako ćwiczenie. Skrótowo cały proces podziału jest przedstawiony w poniższym diagramie.



Rozwiązaniem optymalnym zadania (2.16) jest $x^* = (4, 0)$ i optymalna wartość funkcji celu wynosi $\varphi = 20$.

Powróćmy teraz do własności ograniczenia górnego $\varphi : \mathcal{D} \to \mathbb{R}$:

b1^0 $G_1 \subset G_2 \in \mathcal{D} \Rightarrow \varphi(G_1) \leq \varphi(G_2)$

2^0 $a \in G \Rightarrow \varphi(\{a\}) = f(a)$

Własności te nie opisują jednoznacznie funkcji φ. Aby otrzymać szybciej rozwiązanie, funkcja φ powinna być dopasowana do konkretnego zadania. Przy konstrukcji rodziny $\mathcal{D}$ i funkcji φ powinny być wzięte pod uwagę następujące kryteria:

1. Wartości funkcji φ powinny być wyznaczane możliwie prosto.

2. Wartości funkcji φ powinny możliwie dokładnie szacować z góry maksymalną wartość funkcji f, tzn. wartość

$$\Delta(G_i) = \varphi(G_i) - \max_{x \in G_i} f(x)$$

powinna być możliwie mała.

Przy wyborze funkcji φ problem polega często na tym, że im lepiej funkcja ta spełnia pierwsze kryterium, tym gorzej spełnione jest drugie i odwrotnie. Przy konstrukcji funkcji φ należy więc znaleźć odpowiedni kompromis. Najczęściej funkcję $\varphi : \mathcal{D} \to \mathbb{R}$ definiuje się następująco:

$$\varphi(G_i) = \max_{x \in H_i} f(x),$$

przy czym $G_i \subset H_i$ i podzbiór H_i wybiera się tak, aby $\varphi(G_i)$ można było stosunkowo prosto wyznaczyć (mówi się o *relaksacji* wyjściowego zadania). Na przykład w zadaniu programowania liniowego całkowitoliczbowego lub dyskretnego można opuścić te ograniczenia, których usunięcie prowadzi do zadania programowania liniowego (patrz przykład 2.4.2). W tym przypadku relaksacja taka nazywa się *relaksacją LP*. Inną możliwością jest tak zwana *relaksacja Lagrange'a*. Powstaje ona w ten sposób, że usuwane ograniczenia dodaje się do funkcji celu z odpowiednimi wagami proporcjonalnymi do tak zwanych *mnożników Lagrange'a* (mnożniki te są w istocie równe zmiennym dualnym).

2.4.1 Ogólny opis metod podziału i ograniczeń

Rozważmy zadanie

$$\max_{x \in G} f(x). \tag{2.17}$$

Zakładamy, że rodzina $\mathcal{D} \subset 2^G$ i funkcja $\varphi : \mathcal{D} \to \mathbb{R}$ zostały zdefiniowane. Ponadto, niech $\varphi(\emptyset) = -\infty$.

Algorytm 2.4.4

Krok 0 (*inicjalizacja*)

(a) Położyć $\mathcal{G} = \{G\}$.

(b) Wyznaczyć $\varphi(G)$.

(c) Wybrać $\underline{x} \in G$ i położyć $\underline{\varphi} = f(\underline{x})$ (ewentualnie położyć $\underline{\varphi} = -\infty$).

(d) Wybrać *tolerancję optymalności* $\varepsilon \geq 0$.

Krok 1 (*ograniczenie*)

(a) Wyznaczyć

$$\overline{\varphi} = \max_{G_i \in \mathcal{G}} \varphi(G_i).$$

Jeśli $\overline{\varphi} = -\infty$, to zadanie (2.17) jest sprzeczne – algorytm zatrzymuje się.

(b) Jeśli stwierdzi się, że

$$\varphi(G_i) = f(x_i) \text{ dla pewnego } x_i \in G_i,$$

to usunąć G_i z rodziny $\mathcal{G}$ (w zbiorze G_i nie ma elementu lepszego niż x_i). Jeśli dodatkowo

$$f(x_i) > \underline{\varphi},$$

to położyć

$$\underline{\varphi} = f(x_i) \text{ i } \underline{x} = x_i$$

($f(x_i)$ jest nowym ograniczeniem dolnym i x_i jest kandydatem na rozwiązanie optymalne).

(c) Usunąć z rodziny $\mathcal{G}$ te podzbiory G_i, dla których

$$\varphi(G_i) \leq \underline{\varphi}$$

(podzbiory te nie będą więcej rozważane, gdyż nie zawierają one rozwiązania optymalnego).

(d) Jeśli $\mathcal{G} = \emptyset$, to $\underline{x}$ jest rozwiązaniem optymalnym zadania (2.17) – algorytm zatrzymuje się.

(e) Jeśli $\overline{\varphi} - \underline{\varphi} \leq \varepsilon$, to $\underline{x}$ jest rozwiązaniem ε-optymalnym zadania (2.17) – algorytm zatrzymuje się.

Krok 2 (*podział*)

(a) Wybrać zbiór $\widetilde{G} \in \mathcal{G}$ i rozłożyć go na (możliwie rozłączne) podzbiory $\widetilde{G}_i \in \mathcal{D}$, $i \in L$:

$$\widetilde{G} = \bigcup_{i \in L} \widetilde{G}_i.$$

(b) Położyć

$$\mathcal{G} = \mathcal{G} \cup \{\widetilde{G}_i : i \in L\} \setminus \{\widetilde{G}\}.$$

(c) Wyznaczyć $\varphi(G_i)$ dla wszystkich nowych $G_i \in \mathcal{G}$.

(d) Przejść do kroku 1.

Uwaga 2.4.5 Przy zastosowaniu relaksacji LP:

a) można stwierdzić, czy

$$\varphi(G_i) = f(x_i) \text{ dla pewnego } x_i \in G_i,$$

w kroku 1b), jeśli na przykład rozwiązanie optymalne dla zadania zrelaksowanego jest całkowitoliczbowe.

b) w kroku 2a) wybiera się taki zbiór $\widetilde{G} = G(d; g) \in \mathcal{G}$, dla którego wartość $\varphi(G(d; g))$ jest osiągnięta dla pewnego $x^* = (x_1^*, ..., x_n^*)^\top$ z przynajmniej jedną niecałkowitą współrzędną, powiedzmy x_i. Wówczas zbiór $\widetilde{G} = G(d; g)$ rozkłada się na dwa rozłączne podzbiory

$$\widetilde{G}_1 = G(d; g_1, ..., g_{i-1}, \lfloor x_i^* \rfloor, g_{i+1}, ..., g_n)$$

i

$$\widetilde{G}_2 = G(d_1, ..., d_{i-1}, \lceil x_i^* \rceil, d_{i+1}, ..., d_n; g),$$

gdzie $\lfloor x \rfloor$ oznacza część całkowitą liczby x, zaś $\lceil x \rceil$ oznacza najmniejszą liczbę całkowitą niemniejszą od x (w tym przypadku większą niż x).

Ćwiczenie 2.4.6 Prześledzić Algorytm 2.4.4 dla zadania programowania liniowego całkowitoliczbowego podanego w przykładzie 2.1.4.