# 3. Algorytm simpleks

## 3.1. Wyprowadzenie zrewidowanej metody simpleks

W celu rozwiązania zadania LP należy badać tylko rozwiązania w punktach wierzchołkowych, czyli tylko te rozwiązania dopuszczalne, które wyrażają się przez układy $m$ liniowo niezależnych wektorów. Ponieważ istnieje co najwyżej $\binom{n}{m}$ zbiorów liniowo niezależnych wektorów z danego zbioru $n$ wektorów, liczba $\binom{n}{m}$ określa górną granicę ilości możliwych rozwiązań zagadnienia. Dla dużych $n$ i $m$ niemożliwe jest obliczenie wszystkich rozwiązań i wybranie tego, które minimalizuje funkcję celu. Potrzebna jest metoda obliczeniowa, pozwalająca wybrać mały podzbiór możliwych rozwiązań, zbieżny do rozwiązania minimalnego. Taką metodą jest metoda simpleksów podana przez G. B. Dantziga. Polega ona na znalezieniu punktu wierzchołkowego i zbadaniu, czy w punkcie tym realizuje się minimum. Jeśli nie, znajduje się sąsiedni punkt wierzchołkowy, w którym wartość funkcji celu jest mniejsza lub równa poprzedniej wartości. Dla znalezienia minimalnego rozwiązania dopuszczalnego wystarcza skończona ilość takich kroków (zwykle $m$ do $2m$). Pozwala ona również rozstrzygnąć, czy minimalne rozwiązania są skończone albo czy istnieją rozwiązania. Jest to narzędzie pozwalające rozwiązać dowolne zagadnienie programowania liniowego.

Jeśli dane jest zagadnienie LP w postaci standardowej 2.2 - 2.4 i z założeniem, że macierz $A$ jest rzędu $m$. Jeśli dodatkowo $B$ jest dowolną nieosobliwą podmacierzą stopnia $m$ macierzy $A$, a $N$ jest pozostałą podmacierzą $A$, to można zapisać w postaci

$$[B, N] \begin{bmatrix} x_B \\ x_N \end{bmatrix} = b \tag{3.1}$$

gdzie $x_B$ i $x_N$ mają odpowiednio $m$ i $n - m$ składowych. Wygodnie jest założyć, że $B$ składa się z pierwszych $m$ kolumn macierzy $A$.

Jeśli $x_N = 0$, to rozwiązanie 3.1 nazywa się rozwiązaniem bazowym $x_B = B^{-1} b$, a nieosobliwą macierz $B$ nazywa się bazą. Jeśli ponadto $x \geqslant 0$, to wiadomo, że $x$ jest *bazowym rozwiązaniem dopuszczalnym*
Zostało udowodnione, że jeśli istnieje rozwiązanie dopuszczalne (spełniające 2.2 - 2.3) to istnieje bazowe rozwiązanie dopuszczalne. Co więcej, jeśli istnieje rozwiązanie optymalne minimalizujące $z$, to istnieje optymalne rozwiązanie bazowe.

To stwierdzenie dało klucz do algorytmu simpleks.

Istnieje kilka różnych wersji tej metody i wiele implementacji numerycznych. Opisano prymalną metodę sympleks w wersji zrewidowanej, która jest najpopularniejszą metodą rozwiązywania LP.

Dzieląc wektor kosztów $c$ na dwa zbiory elemetów związanych z $x_B$ i $x_N$, można wyrazić funkcję celu $z$ następująco :

$$z = c_B^T x_B + c_N^T x_N$$

Po podstawieniu:

$$z_B = B^{-1}b - B^{-1}Nx_N \tag{3.2}$$

otrzymano

$$z = c^T B^{-1} b + (c_N^T - c_N^T B^{-1} N) x_N = z_0 + p^T x_N \tag{3.3}$$

gdzie

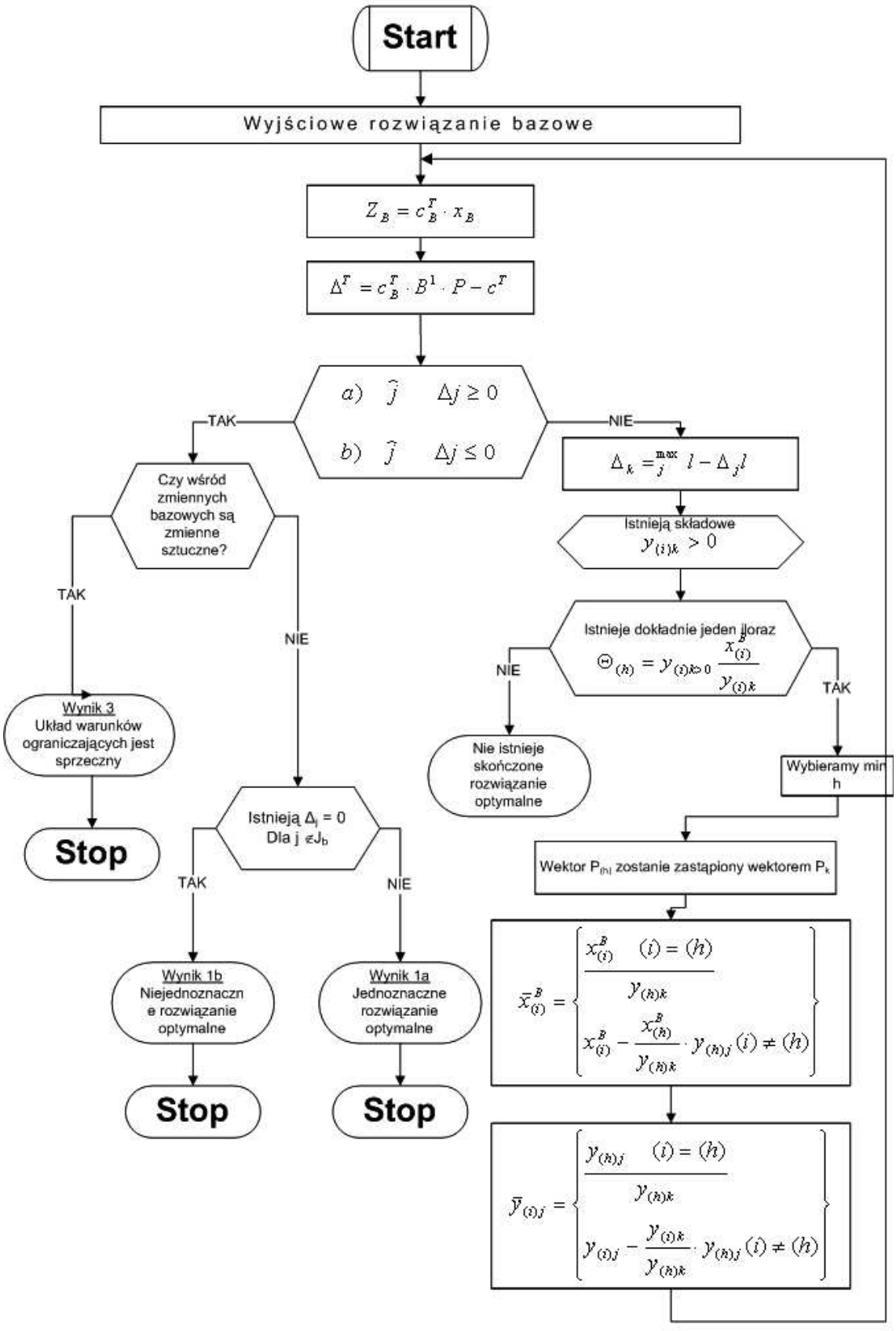
$$x_0 = c_N^T B^{-1} b \tag{3.4}$$

$$p^T = c_N^T - c_B^T B^{-1} N \tag{3.5}$$

Wartość funkcji celu $z$ można polepszyć w następnej iteracji, jeśli znajdzie się ujemną składową w $p$ i wprowadzi odpowiadającą jej niebazową zmienną do bazy. Wektor $p$ może być wygodnie obliczony w dwóch krokach:

$$B^T \lambda = c_B \text{ oraz } p^T = c_N^T - \lambda^T N$$

gdzie $\lambda$ nazywa się *wektorem mnożników simpleksowych*, a $p$ - *wektorem względnych kosztów.* Jeżeli




Rysunek 3.1. Schemat blokowy algorytmu simpleks

$$p_k \; - \; C_{Nk} \; - \; \lambda^T a_k \; < \; 0$$

dla pewnego $m \; + \; 1 \; \leqslant \; k \; \leqslant \; n$, to $a_k$ jest niebazową kolumną macierzy $A$, która może wejść do bazy $B$ w następnej iteracji, a wartość funkcji celu zostanie zmniejszona. Należy znaleźć kolumnę macierzy $B$, którą trzeba usunąć z bazy. Oznaczając bieżące bazowe rozwiązanie dopuszczalne przez $x_0$, żąda się by:

$$x_B \; = \; x_0 \; - \; B^{-1} a_k \; x_{Nk} \; \geqslant \; 0, \tag{3.6}$$

czyli aby

$$x_B \; = \; x_0 \; - \; y x_{Nk} \; \geqslant \; 0,$$

gdzie $y$ otrzymuje się jako rozwiązanie układu $By \; = \; a_k$. Nierówność 3.6 musi być spełniona, aby zachować dopuszczalność następnego rozwiązania $x_B$. Kolumnę opuszczającą bazę można znaleźć rozpatrując

$$x_{0i}/y_i \text{ dla } y_i > 0, \quad i = 1, 2, \ldots, m.$$

Jeśli

$$\Theta \; = \; \frac{x_{0l}}{y_l} \; = \; min \; \left[ \frac{x_{0i}}{y_i} : \; y_i \; > \; 0, \qquad 1 \; \leqslant \; i \; \leqslant \; m \right]$$

to zmienna $x_{Bl}$, staje się zerem i jest przesuwana do zbioru niebazowego. Odpowiednio, kolumna $a_i$ przechodzi do macierzy niebazowej $N$.

Jak wiadomo, jeżeli zadanie programowania liniowego ma skończone rozwiązanie optymalne to jest nim punkt wierzchołkowy tego obszaru. Ogólna idea metody simpleks polega na przechodzeniu pomiędzy sąsiednimi punktami wierzchołkowymi obszaru rozwiązań dopuszczalnych w celu polepszenia funkcji celu. W przypadku niemożności polepszenia wartości funkcji celu, dane rozwiązanie jest rozwiązaniem optymalnym.

Z matematycznego punktu widzenia każdemu punktowi wierzchołkowemu odpowiada bazowe rozwiązanie dopuszczalne. Przejściu pomiędzy sąsiednimi wierzchołkami odpowiada zaś zamiana danej bazy (danego dopuszczalnego rozwiązania bazowego) na sąsiednią (sąsiednie dopuszczalne rozwiązanie bazowe). Sąsiednią bazę konstruuje się przez zamianę jednego wektora bazowego innym - niebazowym. Wektory te są dobierane w taki sposób, by polepszyć wartość funkcji bazowego rozwiązania dopuszczalnego. Problemem pozostaje jedynie dobór początkowego (startowego) bazowego rozwiązania dopuszczalnego.

Rysunek 3.2. Interpretacja graficzna metody Simpleks

### Zrewidowany algorytm simpleks

Jedna iteracja zrewidowanego algorytmu sympleks przebiega następująco:

**Krok 1.** *Dana jest taka baza* $B$, *że* $x_B = B^{-1}b \geqslant 0$

**Krok 2.** *rozwiązać* $B^T \lambda = c_B$ *względem wektora mnożników simpleksowych* $\lambda$

**Krok 3.** *Wybrać z* $N$ *taką kolumnę* $a_k$, *że* $p_k = C_{Nk} - \lambda^T a_k < 0$. *Można na przykład wybrać takie* $a_k$, *które daje najbardziej ujemną wartość* $p_k$. *Jeżeli* $p^T = c_N^T - \lambda^T N \geqslant 0$, *to zatrzymać się : bieżące rozwiązanie jest optymalne.*

**Krok 4.** *Rozwiązać układ równań* $B_y = a_k$ *względem* $y$.

**Krok 5.** *Znaleźć*

$$\Theta = \frac{x_{0l}}{y_l} = min \left[ \frac{x_{0i}}{y_i} : y_i > 0, \qquad 1 \leqslant i \leqslant m \right]$$

*Jeśli żadne* $y_i$ *nie jest dodatnie, to zbiór rozwiązań układu* $Ax = b$, $x \geqslant 0$, *jest nieograniczony i z może przyjąć dowolnie dużą wartość ujemną. Zakończyć obliczenia.*

**Krok 6.** *Uaktualnić rozwiązanie bazowe, korzystając z 3.6:*

$$\begin{aligned} \bar{x}_i &= x_i - \Theta\, y_i, \quad i \neq k \\ \bar{x}_k &= \Theta \end{aligned}$$

*gdzie* $\bar{x}_i$ *jest nową zmienną bazową.*

**Krok 7.** *Uaktualnić bazę* $B$ *i wrócić do kroku 2.*

w kroku 1 zakłada się, że znane jest bazowe rozwiązanie dopuszczalne. Aby je znaleźć dla układu

$Ax = b, \quad x \geqslant 0$ , rozpatrzono pomocnicze zagadnienie minimalizacji:

$$min \sum_{i=1}^{n} y_i^* \tag{3.7}$$

przy warunkcach

$$Ax + Iy^* = b, \qquad x \geqslant 0, \qquad y^* \geqslant 0$$

gdzie $y^*$ jest wektorem sztucznych zmiennych. Jeśli tylko znajdzie się takie optymalne rozwiązanie bazowe zagadnienia 3.7, że $\sum_{i=1}^{n} y_i^* = 0$, to otrzyma się także bazę dającą rozwiązanie $x_B$. Jeżeli 3.7 ma dodatnie minimum, to nie ma rozwiązania dopuszczalnego dla $Ax = b, \quad x \geqslant 0$.
Zagadnienie 3.7 można łatwo rozwiązać za pomocą tej samej metody simpleks, ponieważ ma ono oczywiste początkowe rozwiązanie dopuszczalne $x = 0, \quad y^* = b$ dla $B = I$. Ta metoda dwufazowa została zaimplementowana do rozwiązania ogólnych zagadnień programowania liniowego. Faza I jest użyta do znalezienia rozwiązania dopuszczalnego dla $Ax = b, \quad x \geqslant 0$, lub do stwierdzenia, że nie istnieje rozwiązanie dopuszczalne. Faza II używa bazowego rozwiązania dopuszczalnego, znalezionego w fazie I, do rozwiązania zagadnienia 2.2 - 2.4.

Główny wysiłek obliczeniowy, w zrewidowanej metodzie simpleks, jest związany z wielokrotnym rozwiązywaniem układów równań $B^T \lambda = c_B$ i $By = a_k$, gdzie w kolejnych iteracjach macierze $B$ różnią się tylko jedną kolumną. Prostym sposobem rozwiązania tych równań jest obliczenie $B^{-1}$ oraz wyznaczenie $\lambda$ i $y$, mnożąc macierz przez wektor. Uaktualnienie $B^{-1}$ w kolejnych iteracjach może być wykonane za pomocą transformacji (przekształceń elementarnych) zastosowanych do $B^{-1}$. Jeśli $B^{-1}$ jest bieżącą macierzą odwrotną, a $B^{-1}$ jest uaktualnioną macierzą odwrotną, to $B^{-1} = EB^{-1}$, gdzie $E$ jest macierzą elementarną postaci

$$E = \begin{bmatrix} 1 & & 0 & & \eta_1 & & \\ & \ddots & & & \vdots & 0 & \\ & & 1 & & \vdots & & \\ & & & \ddots & & & \\ & & & & \eta_l & & \\ & 0 & & & \vdots & \ddots & \\ & & & & \eta_m & & 1 \end{bmatrix}$$

przy czym

$$\eta^T = \left( -\frac{y_1}{y_l}, \ldots, -\frac{y_{l-1}}{y_l}, \frac{1}{y_l}, -\frac{y_{l+1}}{y_l}, \ldots, -\frac{y_m}{y_l} \right).$$

Można teraz albo uaktualnić elemetny $B$ bezpośrednio, albo użyć postaci iloczynowej macierzy odwrotnej (w skrócie PFI od angielskeij nazwy product form of the inverse). Pierwszy sposób ma sens praktyczny, dopóki macierz $B^{-1}$, składająca się z $m^2$ elementow, może być pamiętana w pamięci głównej kompuera. Dla zagadnień o wielkich rozmiarach, które wymagają już pamięci pomoczniczych, metoda PFI jest znacznie korzystniejsza. W metodzie tej oblicza się ciąg macierzy $E$ i przedstawia $B^{-1}$ po p - tej iteracji jako iloczyn:

$$B^{-1} = E_p \ldots E_3 E_2 E_1. \tag{3.8}$$

założono w 3.8, że macierz jednostkowa była I bazą początkową. Taka postać odwrotnej macierzy bazowej jest bardzo wygodna do implementacji z użyciem sekwencyjnych pamięci pomoczniczych. Wektor $\lambda^T$ oblicza sie ze wzoru

$$\lambda^T = ((c_b^T E_p) E_{p-1}) \ldots E_1,$$

a $y$ - zgodnie ze wzorem

$$y = E_p \ldots (E_2(E_1 a_k)).$$

Każda iteracja metody simpleks daje dodatkowo jeden wektor $\eta$. W praktyce, pamięta się jedynie niezerowe elementy $E_j$ i na ogół dzięki temu używa się mniejszej pamięci niż dla całej macierzy B. Metoda PFI została wykorzystana z powodzeniem do rozwiązywania wielkich zagadnień z macierzami rzadkimi.
Należy jednak powiedzieć, że w kolejnych iteracjach simpleksowych metoda PFI wymaga pamiętania coraz to nowych macierzy elementarnych. Aby ograniczyć zapotrzebowanie na pamięć i zapewnić dokładność numeryczną, trzeba od czasu do czasu zatrzymać proces iteracyjny i odwrocić macierz $B$ używając kolumn macierzy $A$. W tym kroku reinwersji należy także ponownie obliczyć $x_B$ z równania $x_B = B^{-1}b$; potem następuje ciąg iteracji, aż do ponownej reinwersji.

Aby zilustrować metodę simpleksową, wykonano jedną iterację zrewidowanego algorytmu simpleks na następującym przykładzie:
znaleźć minimum

$$z = -0.5\, x_1 - 0.4 x_2$$

przy warunkach

$$\begin{aligned} x_1 &+ 2x_2 + x_3 &&&&= 24,\\ 1.5x_1 &+ x_2 + &+ x_4 &&&= 18,\\ x_1 &+ &&+ x_5 &&= 11 \end{aligned}$$

$$x_1, x_2, \ldots, x_5 \geqslant 0.$$

Występują tutaj następujące wektory i macierze:

$$A = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 \\ 1.5 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 24 \\ 18 \\ 11 \end{bmatrix}, \qquad c = [-0.5, -0.4, 0, 0, 0]^T.$$

Wygodnie jest rozpocząć obliczenia dla

$$x_B = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix}, \qquad B = B^{-1} = I, \qquad c = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

co daje

$$x_B = B^{-1}b = \begin{bmatrix} 24 \\ 18 \\ 11 \end{bmatrix}$$

Oblicza się $\lambda$ i $p$ $\lambda^T = c_N^T B^{-1} = (0, 0, 0)$,

$p = c_N^T - \lambda^T N = (-0.5\ ,\ -0.4)$

i wybiera $x_1$ do bazy. Rozwiązując układ równań $B_y = a_1$ względem $y$ otrzymano:

$$y = \begin{bmatrix} 1 \\ 1.5 \\ 1 \end{bmatrix}$$

Natomiast

$\Theta = min\left\{\frac{24}{1}, \frac{18}{1.5}, \frac{11}{1}\right\}$

zatem $x_5$ opuszcza bazę. Nowa macierz odwrotna fazy i rozwiązanie bazowe mają postać:

$$x_B = B^{-1}b = \begin{bmatrix} 1 & 0 & \text{-1} \\ 0 & 1 & \text{-1.5} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \text{-1} \\ 0 & 1 & \text{-1.5} \\ 0 & 0 & 1 \end{bmatrix}$$

$$x_B = B^{-1}b = \begin{bmatrix} 13 \\ 1.5 \\ 11 \end{bmatrix}$$

Pierwsza iteracja jest zakończona.

## 3.2. Implementacja algorytmu

Program dołączony do pracy oparty jest na zrewidowanej metodzie simpleks. Opisano w pełni procedurą obliczeniową, opuszczając pewne szczegóły odnoszące się do tego, jak ta implementacja ma się do zarysu zrewidowanej metody simpleks, przedstawionego w paragrafie 3.1.

Rozpatrując ogólne zagadnienie programowania liniowego w postaci standardowej i zakładając, że $b > 0$, dla zrewidowanej metody simpleks zagadanienie to zapisać można w następującej postaci:

$$\text{znaleźć maksimum} \quad x_{x+m+1} \tag{3.9}$$

przy warunkach:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + x_{n+1} &= b1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n + x_{n+m} &= b_m, \\ a_{m+1,1}x_1 + a_{m+1,2}x_2 + \ldots + a_{m+1,n}x_n + x_{n+m+1} &= 0, \\ a_{m+2,1}x_1 + a_{m+2,2}x_2 + \ldots + a_{m+2,n}x_n + x_{n+m+2} &= b_{m+2}, \\ x_i \geqslant 0, \qquad i &= 1, \ldots, n+m, \end{aligned}$$

gdzie

$$a_{m\,+\,1,j} = c_j, \qquad a_{m\,+\,2,j} = -\textstyle\sum_{i\,=\,1}^{m} a_{ij}, \qquad j = 1, 2, \ldots, n,$$

$$b_{m+2} = -\textstyle\sum_{i=1}^{m} b_i,$$

Faza I służy do stwierdzenia, czy rozwiązanie jest dopuszczalne, natomiast w fazie II wyznacza się rozwiązanie optymalne. Chociaż ostatnie równanie w 3.9

jest zbędne, wykorzystuje się je w fazie I algorytmu. Nieujemne sztuczne zmienne $x_{n+i}$ , $i = 1, \ldots, m$ zostały dodane do zmiennych strukturalnych $x_1$ , $x_2$ , $\ldots$ , $x_n$, aby utworzyły prostą macierz bazową $B = I$ . Ponieważ $x_{n+m+2} = -(x_{n+1} + \ldots + x_{n+m})$, więc zmienna $x_{n+m+2}$ jest zanegowaną sumą zmiennych sztucznych.

Jest oczywiste, że $x_{n+m+2} \leqslant 0$.

Zrewidowane zagadnienie ma $m + 2$ równania i $n + m + 2$ zmiennych. Bazowe rozwiązanie dopuszczalne, jeśli tylko istnieje, zawiera $m + 2$ zmiennych ze zbioru $\{ x_1, \ldots, x_n, x_{n+m+1}, x_{n+m+2} \}$, przy czym $- x_{n+m+1}$ podaje optymalną wartość funkcji celu, natomiast $x_{n+m+2} = 0$.

W fazie I maksymalizuje się $x_{n+m+2}$ przy warunkach 3.9, Jeśli $max\ x_{n+m+2}$ to rozpoczyna się fazę II z funkcją celu $x_{n+m+2} = -(c_1x_1 + \ldots, c_nx_n)$ , zachowując $x_{n+m+2} = 0$.

Dodatkowo macierz $A$ postaci:

$$\bar{A} = \left[ \begin{array}{c} \text{A} \\ \hline a_{m+1,1} \ldots a_{m+1,n} \\ a_{m+2,1} \ldots a_{m+2,n} \end{array} \right];$$

jest macierzą powiększoną o 2 wiersze z dołu. W wierszu $m + 2$ przechowuje się wektor względnych kosztów p w fazie I. Wiersz $m + 1$ pełni tę samą rolę w fazie 2. Wprowadzono także macierz o rozmiarach $(m + 2)$ x $(m + 2)$. Początkowa postać tej macierzy:

$$U = \left[ \begin{array}{c|c} I_{mxm} & 0 \\ \hline 0 & I_{2x2} \end{array} \right]$$

Początkowe $m$ wierszy i kolumn macierzy $U$ zawiera macierz odwrotną do $B$. Ostatnie 2 wiersze $U$ służą do określenia wektora, który ma wejść do bazy (wiersz $m + 2$ w fazie I i wiersz $m + 1$ w fazie II).

Gdy przyjmie się teraz podane oznaczenia oraz wartości początkowe $x_{n+i} = b + i$, $i = 1, 2, \ldots, m$, wtedy obliczenia w fazach I i II przebiegają następująco:

***Faza I***

**Krok 1.** *Jeśli* $x_{n+m+2} < 0$*, to obliczyć*

$$\delta_j = wiersz_{m+2}(U) \cdot kolumna_j(\bar{A}) = \sum_{p=1}^{m+2} u_{m+2,p}a_{pj}, j = 1, 2, \ldots, n.$$

*i przejśc do kroku 2.*
*Jeśli* $x_{n+m+2} = 0$*, przejść do kroku 1 fazy II .*
*W fazie I* $x_{n+m+2}$ *jest maksymalizowaną funkcją celu i wszystkie jej współczynniki, poza* $c_{n+m+2} = 1$*, są równe zeru. Wartości* $\delta_j$ *są składowymi wektora względnych kosztów p, określonego równością 3.5*

**Krok 2.** *Jeśli wszystkie* $\delta_j \geqslant 0$*, to zmienna* $x_{n+m+2}$ *osiągnęła maksymalną wartość i nie istnieje rozwiązanie dopuszczalne zagadnienia LP. Jeśli co najmniej jedno* $\delta_j \geqslant 0$ *to wówczas zmienną wprowadzaną do bazy jest takie* $x_k$*, że*

$$\delta_k = min\{\delta_j : \delta_j < 0, \quad 1 \leqslant j \leqslant n\}.$$

**Krok 3.** *Obliczyć:*

$$y_i = wiersz_i(U) \cdot kolumna_k(\bar{A}) = \sum_{p=1}^{m+2} u_{ip} a_{pk}, \quad j = 1, 2, \ldots, m+2.$$

**Krok 4.** *Obliczyć:*

$$min\left[\frac{x_i}{y_i} : y_i > 0, \quad 1 \leqslant i \leqslant m\right] = \frac{x_l}{y_l} = \Theta.$$

*Jeśli* $u_i \leqslant 0$ *dla wszystkich* $i = 1, 2, \ldots, m$*, to nie ma wtedy rozwiązania dopuszczalnego. W przeciwnym przypadku zmienną* $x_l$ *należy usunąć z bazy.*

**Krok 5.** *Obliczyć nowe wartości zmiennych w rozwiązaniu bazowym*
$\bar{x_k} = \Theta$,
$\bar{x_i} = x_i - \Theta y_i, \quad i \neq k, \quad i = 1, 2, \ldots, m+2$
*oraz*
$\bar{u_{ij}} = \frac{u_{lj}}{y_l}$,
$\bar{u_{ij}} - y_i \frac{u_{lj}}{y_l}, \quad i \neq l.i = 1, 2, \ldots, m+2 \quad j = 1, 2, \ldots, m$*. Wrócić do kroku 1.*
*Kolumny* $m+1$ *i* $m+2$*macierzy I nie zmieniają się. Iteracje fazy I są wykonywane aż do momentu, gdy albo* $x_{n+m+2} = 0$*, albo też stwierdzi się, że nie istnieje rozwiązanie dopuszczalne. W pierwszym z tych przypadków przechodzi się do fazy II.*

***Faza II***

**Krok 1.** *Teraz zachowuje się* $x_{n+m+2} = 0$.
*Obliczyć*

$$\gamma_j = wiersz_{m+1}(U) \cdot kolumna_j(\bar{A}) = \sum_{p=1}^{m+2} u_{m+1,p} a_{pj}, \quad j = 1, 2, \ldots, n.$$

**Krok 2.** *Obliczyć*

$$\gamma_k = min\{\gamma_j : \gamma_j < 0, \quad 1 \leqslant j \leqslant n\}.$$

*Zmienna $x_k$ wchodzi do bazy. Jeżeli wszystkie $\gamma_j \geqslant 0$, to zmienna $x_{n+m+1}$ osiągnęła maksimum i otrzymano rozwiązanie podanego zagadnienia.*

**Krok 3.** *Obliczyć*

$$y_i = \sum_{p=1}^{m+2} u_{ip} a_{pk}, \quad i = 1, 2, \ldots, m+2,$$

**Krok 4.** *Znaleźć*

$$min \left[ \frac{x_i}{y_i} : y_i > 0, \quad 1 \leqslant i \leqslant m \right] = \frac{x_l}{y_l} = \Theta.$$

*Jeżeli wszystkie $y_i \leqslant 0$, to funkcję celu $x_{n+m+1}$ można powiększyć nieskończenie. Obliczenia się kończą. W przeciwnym razie przejść do kroku 5.*

**Krok 5.** *Obliczyć*

$\bar{x_k} = \Theta$,

$\bar{x_i} = x_i - \Theta y_i, \quad i \neq k, \quad i = 1, 2, \ldots, m+1$

*oraz*

$\bar{u_{ij}} = \frac{u_{lj}}{y_l}$,

$\bar{u_{ij}} - y_i \frac{ulj}{y_l}, \quad i \neq l.i = 1, 2, \ldots, m+2 \quad j = 1, 2, \ldots, m.$

*Wrócić do kroku 1.*

Kroki 2 do 5 faz I i II są bardzo podobne. Zagadnienie rozpatrywane w fazach I i II mają różne funkcje celu i różne warunki zakończenia. Funkcja SIMPLEX automatycznie wprowadza sztuczne zmienne $x_{n+1}$ do $x_{n+m}$, oraz zmienne bilansujące $x_{n+m}$ do maksymalnie $x_{n+2m}$ (jeśli takie są potrzebne). W innych wersjach metody, jako jednostkowych wektorów bazy można użyć tych z warunków $Ax = b$, których współczynniki kosztu są równe zeru. Jeśli zdarzy się, że jest ich $m$, faza I może być opuszczona. Dalszym udoskonaleniem techniki znajdowania wstępnego rozwiązania bazowego jest wpychanie, w którym usiłuje się wprowadzić do bazy możliwie dużo zmiennych strukturalnych. Obliczenia rozpoczyna się od bazy jednostkowej, utworzonej poprzez wprowadzenie zmiennych uzupełniających dla wszystkich nierówności i zmniennych sztucznych dla wszystkich równości. Potem testuje się kolejno każdą zmienną strukturalną $x_i$, czy da się ją wprowadzić do bazy bez tworzenia nowych niedopuszczalnych zmiennych, pod warunkiem, że dzięki temu usuwa się z bazy zmienną sztuczną i otrzymuje dodatni koszt zredukowany lub zmniejsza liczbę zmiennych dopuszczalnych. W implementacji komputerowej testuje się wszystkie zmienne niebazowe, czy mogą wejśc do bazy. Ponieważ w przypadku dużych zagadnień zajmuje to wiele czasu, więc aby polepszyć wartości funkcji, w każdym kroku

iteracyjnym może być zastosowana technika znana jako wielokrotny wybór kolumn. W technice tej wybiera się $k > 1$ niebazowych kolumn macierzy $A$ o najbardziej ujemnych kosztach zredukowanych, oblicza się odpowiednie wektory $y$ i wykonuje testy ilorazowe kroku 4 fazy II. Zmienna $x_i$ która najbardziej redukuje wartość funkcji celu, jest wprowadzana do bazy. Baza jest uatkualniana i można szukać nowej zmiennej, która mogłaby wejśc do bazy.
Inną techniką jest wybranie $k > 1$ niebazowych kolumn o najbardziej ujemnym zredukowanym koszcie i utworzenie podproblemu składającego się z bieżącej bazy i - tych $k$ - kolumn. W tym zagadnienu suboptymalizacyjnym szuka się rozwiązania optymalnego.
Programy komputerowe dla dużych zagadnień nie obliczają całego wektora kosztów zredukowanych znalezienia zmiennej $x_k$, odpowiadającej najbardziej ujemnej składowej tego wektora. Zamiast tego obliczane są po kolei elementy $\gamma_j$, aż do znalezienia dostatecznie dużego ujemnego zredukowanego kosztu. Odpowadająca jemu zmienna niebazowa jest wprowadzana do bazy. Czas wykonania każdej iteracji jest znacznie krótszy, ale być może trzeba będzie wykonać znacznie więcej iteracji, aby rozwiązać zagadanienie.

## 3.3. Stabilność numeryczna

Jest oczywiste, że w procesie uaktualniania $b^{-1}$ nie ma żadnej kontroli elementów głównych $y_i$, i jeśli pewne z nich mają małe wartości, macierz odwrotna może być obliczona bardzo niedokładnie. Bezpieczniejsze, pod względem numerycznym wersje zrewidowanej metody simpleks są oparte na faktoryzacji macierzy. Rozważono w tym celu faktoryzację trójkątną $LU$. Gdzie $L$ oraz $U$ są odpowiednio dolną i górną macierzą trójkątną.
Metoda simpleks, oparta na faktoryzacji trójkątnej została podana przez Bartelsa i Goluba. Podczas procesu reinwersji bazowa macierz $B$ jest przeprowadzana w macierz trójkątną poprzez eliminacje Gaussa, tj.
$B_{j+1} - M_j B_j, \quad j = 1, 2, \ldots, m-1$, gdzie $B_1 = B, \quad B_m = U$. Po $m-1$ eliminacjach otrzymuje się:

$$U = M_{m-1} M_{m-2} \ldots M_1 B,$$

gdzie macierze $M_j$ są postaci:

$$M_j = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & m_{ij} & \ddots & \\ & & & 1 \end{bmatrix}$$

a$m_{ij} \leqslant 1$, gdy dokonano zmian wierszy.
Po oznaczeniu

$$\Pi_{j=m-1}^{1} M_j = L^{-1}$$

otrzymano

$$B = LU = M_1^{-1} M_2^{-1} \dots M_{m-1}^{-1} U,$$

przy czym

$$M_j^{-1} = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & m_{ij} & \ddots & \\ & & & 1 \end{bmatrix}$$

Jeśli $B$ jest bieżącą bazą $B = [a_1, \dots, a_l, \dots, a_m, a_k]$.
Wyliczono teraz

$$H = L^{-1}B = \left[u_1, \dots, u_{l-1}, u_{l+1}, \dots, u_m, L^{-1}u_k\right],$$

gdzie $u_1, \dots, u_m$ są kolumnami macierzy $U$; $H$ jest górną macierzą postaci:

$$H = \begin{bmatrix} \ddots & \dots & \dots & \dots \\ & \ddots & \dots & \dots \\ & 0 & \ddots & \dots \\ & & & \ddots \end{bmatrix}$$

i może być sprowadzona do postaci trójkątnej poprzez ciąg eliminacji Gaussa $U = M_{m-1} \dots M_{l+1} M_l H$, gdzie $U$ jest macierzą trójkątną, natomiast

$$M_j^{-1} = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & m_{j+1,j} & \ddots & \\ & & & 1 \end{bmatrix}$$

Jeżeli eliminacja jest przeprowadzana z równoczesną zamianą wierszy (w każdym kroku można wybrać jeden z dwu wierszy), to ma się do czynienia ze stabilnym procesem uaktualniania, przy czym $|m_{j+1,j}| \leqslant 1$
Nową uaktualnioną bazą jest

$$B = LH = LM_1^{-1}M_{l+1}^{-1}\ldots M_{m-1}^{-1}U = LU.$$

Czynniki macierzy $L^{-1}$ mogą być pamiętane w postaci iloczynowej, tak jak w metodzie PFI. Jednak elementy macierzy $U$ muszą być jednakowo dostępne i są zwykle pamiętane w postaci listy połączonej.
Taki sposób pamiętania jest bardzo wygodny do wstawiania i usuwania elementów $U$, bez przemieszczania pozostałych elementów niezerowych.
Metoda $LU$ jest numerycznie bezpieczna, jednak w swej czystej postaci ignoruje problem rozmnażania się elementów niezerowych w procesie reinwersji i uaktualniania. Gęstość elementów niezerowych w wielu zagadnieniach LP wielkiej skali wynosi dużo poniżej jednego procenta. Jest przeto rzeczą ważną, by proces obliczania $B^{-1}$ dawał rzadką reprezentację macierzy odwrotnej. Zasugerowano małą modyfikację tej metody, która umożliwia szerszy wybór elementów głównych. Modyfikacja polega na tym, by było $|m_{ij}| \leqslant \gamma, \quad \gamma > 1$, i by wybrać taki element główny, który lokalnie minimalizowałby wzrost liczby elemetów niezerowych w czynnikach macierzy $B$. Można zatem osiągnąć kompromis między zachowaniem rzadkości macierzy i stabilnością numeryczną. Jednakże możliwość minimalizacji rozmnażania się elementów niezerowych, dzięki użyciu faktoryzacji $LU$, jest ograniczona tym, iż faktoryzacja $LU$ nie istnieje dla wszystkich permutacji wierszy i kolumn macierzy $B$, a dla niektórych z nich może być źle określona numerycznie. Z tego powodu nie ma pełnej swobody w wyborze permutacji, biorąc pod uwagę rzadkość macierzy.

## 3.4. Duże programy LP

Dotychczas zakładano, że macierz $A$ współczynników przy ograniczeniach nie ma specjalnej struktury. W wielu zastosowaniach, szczególnie w zagadnieniach o wielkch rozmiarach, macierz $A$ ma określone własności strukturalne, na przykład niezerowe elementy $A$ mogą być umieszczone w blokach przekątniowych, z wyjątkiem stosunkowo małej liczby wierszy i kolumn. Zagadnienia o takiej strukturze

muszą być rozwiązywane za pomoca specjalnych algorytmów, które wykorzystują te własności. Korzystając z tak dopasowanych algorytmów, można nie tylko osiągnąć znaczne zmniejszenie kosztów obliczeń, ale w wielu przypadkach bardzo duże zagadnienia nie mogą być w ogóle rozwiązane za pomocą ogólnych programów ponieważ wymagana byłaby zbyt wielka pamięć lub też obliczenia trwałyby zbyt długo.
Jedna z pierwszych znaczących metod, przeznaczonych do rozwiązywania zagadnień wielkich rozmiarów o blokowej strukturze diagonalnej, została podana przez Dantziga i Wolfe'a w roku 1960. Od tego czasu zbadano i zastosowano w praktyce wiele technik dostosowanych do specjalnych struktur.

## 3.5. Zbieżność i złożoność czasowa

Metoda simpleks opiera się na fakcie, że optymalna wartość programu liniowego, jeśli istnieje, jest zawsze osiągnięta w rozwiązaniu bazowym. Zakładając, że wszystkie bazy rozwiązania dopuszczalne są niezdegenerowane, metoda simpleks znajduje rozwiązanie optymalne w skończonej liczbie iteracji, ponieważ liczba możliwych baz jest skończona i żadna z nich nie powtarza się. Program liniowy nazywa się niezdegenerowanym, jeśli $C^{-1}b > 0$. Jest to równoważne ze stwierdzeniem, iż $b$ nie może być wyrażone jako kombinacja liniowa żadnego zbioru $m - 1$, lub mniej, kolumn macierzy $A$. Jeśli zagadnienie LP jest niezdegenerowane, to każde bazowe rozwiązanie dopuszczalne ma jednoznaczną bazę z nim związaną. W przypadku degeneracji możemy spotkać ciąg iteracji, generujący ciąg baz $B_i, B_{i+1}, \ldots, B_p$, które wszystkie odpowiadają temu samemu bazowemu rozwiązaniu dopuszczalnemu i tej samej wartości funkcji celu. Może się również zdarzyć, że $B_i = B_p$ i metoda simpleks wejdzie w cykl nieskończony. Powstanie cyklu nie jest poważnym problemem i właściwie nigdy nie zdarza się w praktyce. W większości programów komputerowych dla programowania liniowego nie ma żadnych sposobów unikania degenereacji. Powstanie cykli jest jednak rzeczywistym problemem w niektórych zagadnieniach całkowitoliczbowych, które są rozwiązywane metodami pochądzącymi od metody simpleks. Zatem programy całkowitoliczbowe powinny być wyposażone w zabezpieczenia przed cyklami.

Rozpoczynając od dopusczalnej bazy, metoda simpleks znajduje, po skończonej liczbie iteracji, albo rozwiązanie optymalne, albo bazę dopuszczalną, która

pozwala stwierdzić, że zagadnienie LP ma rozwiązanie nieograniczone.
Należy odpowiedzieć na pytanie: Ile iteracji jest potrzebnych do rozwiązania zagadnienia LP o $n$ zmiennych i $m$ ograniczeniach? Trudno jest podać dokładną liczbę iteracji potrzebnych do osiągnięcia rozwiązania optymalnego. Dla zagadnienia o $n$ zmiennych i $m$ ograniczeniach istnieje co najwyżej

$$\left( \begin{array}{c} \mathrm{n} \\ \mathrm{m} \end{array} \right) = \frac{n!}{m!(n-m)!}$$

rozwiązań bazowych. Stanowi to jednak bardzo słabe oszacowanie. Metoda simpleks nie bada wszystkich rozwiązań bazowych, których liczba jest bardzo duża, nawet dla zagadnienia średniej wielkości. Bada ona bardzo mały podzbiór wszystkich bazowych rozwiązań dopuszczalnych.
Doświadczenia zdobyte podczas rozwiązaywnaia wielu setek praktycznych zagadnień wskazały, że liczba iteracji waha się między $m$ i $3m$. Niektórzy matematycy twierdzą, że (dla losowo wybranego zagadnienia z ustalonym $m$) liczba iteracji jest proporcjonalna do $n$. Rozsądnym oszacowaniem liczby iteracji jest $2(n + m)$.
Inną kwestią jest największa możliwa liczba bazowych rozwiązań dopuszczalnych. W jednej transformacji algorytm simpleks przesuwa się z jednego ekstremalnego punktu zbioru rozwiązań dopuszczalnych do sąsiedniego punktu ektremalnego. Wartość funkcji celu zmniejsza się monotonicznie na tej drodze. Tę drogę bazowych rozwiązań dopuszczalnych nazywa się *ścieżką izotoniczną* względem funkcji celu $z = c^T x$. Długością tej ścieżki jest l, liczba punktów leżących na niej, bez punktu początkowego.

Metodę simpleks można zmusić do pójścia taką ścieżką, dobierając odpowiednie zmienne bazowe w krokach iteracyjnych. Skonstruowano przykład obrazujący, że największa długość ścieżki może wynosić aż $2^{k-1}$, gdzie $k = m / 2 = n$. Liczba ta nie jest ograniczona żadnym wielomianem zmiennych $m$ i $n$. Istnieje możliwość zbudowania takiego wariantu metody simpleks, który szedłby wzdłuż ścieżki izotonicznej. Zatem nie byłby to algorytm o wielomianowym czasie działania. Jeśli jednak dowolny standardowy algorytm simpleks (taki jak wersja przedstawiona w tej pracy) zastosować do tego przykładu, to dostaje się nieoczekiwany wynik. Okaże się, że potrzebna jest tylko jedna iteracja, by go rozwiązać. Przedstawiono także zagadnienie LP o $n$ wierszach i $2n$ kolumnach, które podczas rozwiązywania algorytmem zastosowanym w aplikacji - wymagałoby $2^{n-1}$ iteracji.
Tak więc z teoretycznego punktu widzenia metoda simpleks ma poważny defekt, gdyż jej złożoność czasowa nie jest ograniczona wielomianowo. Dolne ograniczenie

czasu działania jest wydkładnicze i fakt ten zwykle charakteryzuje algorytymy, których stosowanie jest ograniczone do problemów stosunkowo małych. Jednak wada ta nie ma znaczącego wpływu na stosowalność algorytmu simpleks. W przeważającej liczbie zadań algorytm rozwiązuje zagadnienie maksymalnie w $2(n + m)$ iteracjach.

## 3.6. Algorytm o wielomianowej złożoności czasowej

W zupełnie odmiennym podejściu do programowania liniowego wykorzystuje się badanie rozwiązalności zagadnienia liniowego nierówności $Ax < b$ o całkowitych współczynnikach, ale z wymiernym $x$. Punktem wyjściowym w konstrukcji nowego algorytmu jest spostrzeżenie, że jeśli obszar dopuszczalności nie jest pusty, to istnieje w tym obszarze zbiór $S$ określonego rozmiaru. Ponadto $S$ leży w skończonej odległości od początku układu współrzędnych.
Metoda wielomianowa określa, czy układ ostrych nierówności liniowych $Ax < b$ (gdzie $A$ ma rozmiar $m$ x $n$) ma rozwiązanie. W najgorszym przypadku algorytm ten zatrzymuje się po $k = 6n^2L$ krokach, a $L$ jest wielomianem względem $m$ i $n$. Z tego wynika, że algorytm ten posiada wielomianową złożoność czasową.
Geometryczną podstawą opisywanego algorytmu jest tworzenie ciągu elipsoid $(x - x_k)^T A_k^{-1}(x - x_k) \leqslant 1$ o zmniejszającej się objętości i zawierających rozwiązanie dopuszczalne (jeśli istnieje).
Algorytm rozpoczyna działanie od kuli o środku w $x_0 = 0$ i promieniu $2^L$ dostatecznie dużym, by kula zawierała jakiś zbiór $S$ rozwiązań dopuszczalnych. Jeśli środek elipsoidy $x_k$ nie jest dopuszczalny, to dzieli się tę elipsoidę na pół hiperpłaszczyzną równoległą do jakiegoś nie spełnionego ograniczenia i przechodzącą przez środek.
Ta połówka elipsoidy jest użyta do utworzenia następnej elipsoidy, zawierającej połówkę. Ta nowa elipsoida ma objętość mniejszą od objętości poprzedniej elipsoidy rys 3.3.

Przechodzi ona przez $x_1$ i $x_2$, ma środek po tej stronie cięciwy $c$, po której leży rozwiązanie, i jest styczna do poprzedniej elipsoidy w punkcie $T$. Otrzymana elipsoida jest tą, która ma najmniejszą objętość i ma pożądane własności. Wykazano, że jeśli algorytm nie znajdzie rozwiązania układu $Ax < b$ w $k = 6n^2L$ krokach, to zbiór rozwiązań dopuszczalnych jest pusty.
Udowodniono, że zadanie LP można rozwiązać w czasie wielomianowym.

Rysunek 3.3. Graficzna interpretacja algorytmu elipsoid

Należy jednak powiedzieć, że obecnie algorytm elipsoidalny w swej oryginalnej postaci nie jest praktycznym algorytmem obliczeniowym. Zachowuje się on najgorzej wtedy, gdy dany układ nierówności nie ma rozwiązania, a liczba iteracji wynosi wówczas $k = 6n^2L$. Dla każdego zagadnienia o średnim rozmiarze wartość $L$ byłaby astronomicznej wielkości. Dokładna wersja metody musiałaby działać na wielkościach zmiennopozycyjnych z dokładnością $0(nL)$ bitów; jest to nierealistyczne. Nadzwyczajna dokładność wymagana w tej metodzie wydaje się być główną przeszkodą. Pokazano też inne wady tej metody. Oto częściowa ich lista:

— punkty $x_k$, generowane w metodzie, mają tendencję do błędnego poruszania się wokół obszaru dopuszczalnego, zanim nie znajdzie się punktu dopuszczalnego.
— macierze $A_k$ zmierzają do macierzy o pełnej gęstości. Jest to nie do przyjęcia przy rozwiązywaniu wielkich, rzadkich zagadnień LP.
— uwarunkowanie macierzy $A_k$ szybko się zwiększa; metoda ma zatem słabe własności numeryczne.

Wobec tego rozwiązywanie w czasie wielomianowym niewiele daje.

## 3.7. Dualność w programowaniu liniowym

Ważnym pojęciem w programowaniu liniowym jest dualność
Dla każdgo problemu programowania liniowego istnieje inny stowarzyszony z nim problem dualny. Ten nowy problem liniowy posiada kilka ważnych własności i może być użyty do znalezienia rozwiązania problemu oryginalnego.
Zmienne występujące w problemie dualnym dostarczają bardzo użytecznych inforamcji o rozwiązaniu optymalnym problemu prymalnego.
Przejście między zadaniami jest w dwie strony i jest sprawą umowną, które z zadań jest zadaniem pierwotnym. Dozwolone są następujące przejścia (i tylko te!):

| **zadanie pierwotne** | **zadanie dualne** |
|---|---|
| max $z_p$ | min $z_d$ |
| $\ldots \leqslant \ldots$ | $\ldots \geqslant \ldots$ |
| $\longleftrightarrow$ | |

Wykorzystując zapis macierzowy, prymalne zagadnienie programowania liniowego formułuje się w postaci:

$$\begin{aligned} max \quad & z = c^T x \\ & Ax \leqslant b \\ & x \geqslant 0 \end{aligned} \tag{3.10}$$

Symetrycznym zadaniem dualnym do zadania prymalnego nazywa się problem:

$$min \quad u = b^T \lambda \tag{3.11}$$

przy ograniczeniach:

$$\begin{aligned} A^T \lambda & \geqslant c \\ \lambda & \geqslant 0 \end{aligned} \tag{3.12}$$

gdzie $\lambda$ ma identyczny wymiar z b. Jednocześnie zadanie 3.13 jest dualne do zadania 3.11
Rozwiązanie problemu dualnego osiąga sie w taki sam sposób jak problem prymalny przy użyciu ogólnego algorytmu simpleks opisanego w poprzednim paragrafie, jednak

po wprowadzeniu pewnych modyfikacji.
Dla łatwiejszego zrozumienia sedna problemu przytoczono przykład:

$$min(60y_1 + 15y_2 + 16y_4)$$

przy ograniczeniach:

$$\begin{aligned} 3y_1 + 4y_2 + 3y_3 - y_4 \quad &= 11 \\ 10y_1 + y_2 + 2y_3 \quad - y_5 &= 7 \\ y_i \geqslant 0, \quad i = 1, \ldots, 5 & \end{aligned}$$

Porównując oba problemy stwierdzono, że w zapisie problemu programowania liniowego w postaci dualnej, zmienne błędu posiadaj współczynniki ujemne, równe -1. W związku z tym nie jest możliwe znalezienie dopuszczalengo rozwiązania początkowego identyczną metodą jak w algorytmie prymalnym. W rozważanym przykładnie nie można bowiem przyjąć, np $-y_4 = 11$, co powoduje, że $y_4 = -11$, tzn, przyjmuje wartość niedopusczalną, gdyż zgodnie z zależnością 3.13 $y_i \geqslant 0, \quad i = 1, \ldots, 5$ Trudność tę można wyeliminować wprowadzając do równań sztuczne zmienne o dodatnich współczynnikach. Dla prezentowanego przykładu otrzymano nowy system równań:

$$\begin{aligned} 3y_1 + 4y_2 + 3y_3 - y_4 \quad + y_6 \quad &= 11 \\ 10y_1 + y_2 + 2y_3 \quad - y_5 + \quad y_7 &= 7 \end{aligned}$$

Zmienne sztuczne są wprowadzane do funkcji celu ze współczynnikiem $M$ o wartości bezwzglednej bardzo dużej, dodatniej lub ujemnej, zależnie od tego czy poszukuje się minimum, czy maksimum. Po wprowadzeniu zmiennych sztucznych rozwiązanie początkowe ma postać $y_6 = 11$ oraz $y_7 = 7$, $y_i = 0, \quad i = 1, \ldots, 5$. Zmienne sztuczne zostają bardzo szybko wyeliminowane w wyniku działania algorytmu, ponieważ ich współczynniki w funkcji celu posiadają bardzo dużą wartość bezwzględną.
Po wprowadzeniu zmiennych sztucznych zagadnienie polega na minimalizacji funkcji celu:

$$min\,[60y_1 + 15y_2 + 16y_3 + M(y_6 + y_7)]$$

lub na maksymalizacji fukcji celu ze znakiem minus.

$$max\,[-60y_1 - 15y_2 - 16y_3 - M(y_6 + y_7)]$$

przy ograniczeniach występujących w 3.7 Rozwiązanie początkowe otrzymuje się przy wykorzystaniu zmiennych sztucznych o współczynnikach -M w funkcji celu. Wielkości $\delta_j$ w pierwszym etapie nie są wpółczynnikami startowymi tej funkcji, lecz

rolę tę pełnią wielkości $-c_j + (M, \ldots, M) \cdot (kolumna\ y_j)$
Wobec tego rozwiązanie startowe nie odpowiada tak jak w metodzie prymalnej początkowi układu współrzędnych.
Porównując rozwiązania problemu prymalnego i dualnego sformułowano następujące uwagi:

— wartość minimalna funkcji celu problemu dualnego jest równa wartości maksymalnej funkcji celu problemu prymalnego,
— rozwiązanie problemu dualnego odpowiada co do wartości bezwzględnej wartościom $\delta_j$ problemu prymalnego w ostatnim etapie
— w ostatnim etapie wartości $\lambda_j$ problemu dualnego są co do wartości bezwzględnej równe wartościom rozwiązania prymalnego:

Spostrzeżenia te pozwalają w ostatnim etapie przejść od tablicy prymalnej do tabeli dualnej i odwrotnie. Rozwiązania problemu dualnego oraz prymalnego są z matematycznego punktu widzenia równoważne, natomiast z punktu widzenia ekonomicznego równoważność ta nie jest spełniona.