ROCZNIKI POLSKIEGO TOWARZYSTWA MATEMATYCZNEGO
Seria III: MATEMATYKA STOSOWANA XXI (1982)

STANISŁAW L. KRYŃSKI (Warszawa)

Minimalizacja wklęsłej funkcji na wielościanie wypukłym*

(*Praca wpłynęła do Redakcji 20. 11. 1979*)

1. Wstęp.

1.1. Rozważamy problem wyznaczenia globalnego minimum wklęsłej funkcji na wielościanie wypukłym: dla danego wielościanu $D \subset R^n$ i wklęsłej funkcji $f: R^n \to R$ znaleźć $z \in D$ takie, że:

$$\text{(PW)} \qquad f(z) = \min_{x \in D} f(x).$$

Problem ten krótko nazwiemy *zadaniem programowania wklęsłego* (PW).

1.2. Wprost z definicji funkcji wklęsłej i podstawowych własności wielościanu wypukłego wynika, że dla każdego zadania (PW) istnieje rozwiązanie $z \in D$ będące wierzchołkiem wielościanu D. Spostrzeżenie to pozwala sprowadzić rozwiązywanie zadania do przeglądu wszystkich wierzchołków wielościanu i badania wartości funkcji f w tych wierzchołkach. Jednak podobnie, jak na przykład w programowaniu liniowym, pożądane jest znalezienie metody nie wymagającej wyznaczania całego zbioru wierzchołków, który, aczkolwiek skończony, może mieć przecież dowolnie dużą moc. Podstawowym problemem, który zjawia się przy konstruowaniu takiego algorytmu dla zadania (PW), podobnie zresztą jak dla innych nie wypukłych zadań programowania matematycznego, jest to, że funkcja f może mieć na D minima lokalne nie będące minimami globalnymi. Wobec tego błędna byłaby np. metoda, która ograniczałaby się jedynie do badania wartości funkcji w wierzchołkach sąsiednich (zob. definicja w § 2.1) dla danego wierzchołka i wyznaczająca ciąg wierzchołków $v^1, v^2, \ldots, v^k, v^{k+1}, \ldots, v^r$ taki, że 1° v^k, v^{k+1} są sąsiednie dla $k = 1, \ldots, r-1$; 2° $f(v^{k+1}) < f(v^k)$, $k = 1, \ldots, r-1$; 3° dla każdego wierzchołka v sąsiedniego dla v^r jest $f(v) \geqslant f(v^r)$. Zauważmy, że ten schemat jest wykorzystany w znanej metodzie „sympleks" dla programowania liniowego, przy czym, o ile tam wiadomo, że v_r jest istotnie rozwiązaniem optymalnym, to w przypadku programowania wklęsłego jedyną dostępną informacją jest, że v^r jest minimum lokalnym.

* Praca wykonana w ramach problemu 49/13.3.

1.3. Tuy [5] zaproponował metodę, która wychodząc z wierzchołka będącego minimum lokalnym f na D, konstruuje ciąg pomocniczych zadań programowania liniowego. W pracy [5] nie podano jednak dowodu zbieżności tego algorytmu. Co więcej, okazało się później (Zwart [6]), że algorytm Tuy'a w ogólnym przypadku nie jest skończony i nie wyznacza rozwiązania zadania (PW).

W niniejszej pracy opisany jest algorytm będący modyfikacją oryginalnej koncepcji Tuy'a. Algorytm ten zawsze, tj. dla każdej pary D, f, wyznacza minimum globalne funkcji f na wielościanie D. Dowód zbieżności podany jest w pkcie 8 artykułu.

2. Podstawowe założenia, definicje i oznaczenia.

2.1. Wprowadzimy i przypomnimy tu pewne pojęcia i fakty niezbędne w dalszych częściach artykułu. Opieramy się przy tym głównie na monografii [4].

Zbiorem wielościennym nazywamy podzbiór przestrzeni R^n będący przecięciem skończonej liczby półprzestrzeni domkniętych. *Wielościan* jest ograniczonym zbiorem wielościennym. Nie będziemy definiować takich obiektów, jak *wierzchołek zbioru wielościennego*, *krawędź*, *promień ekstremalny*, *kombinacja wypukła punktów*, *powłoka wypukła zbioru*, czy *stożek wypukły*, zakładając, że są one znane czytelnikowi. Dla zbiorów $P_1, P_2 \subset R^n$ ich *suma algebraiczna* jest zdefiniowana jako zbiór $P_1+P_2 = \{x_1+x_2 |\ x_1 \in P_1, x_2 \in P_2\}$. Symbole $\operatorname{conv} P$ oraz $\operatorname{cone} P$ oznaczają, odpowiednio, powłokę wypukłą zbioru $P \subset R^n$ oraz najmniejszy stożek wypukły zawierający P. Dla zbioru wielościennego $W \subset R^n$ przez $V(W)$ oznaczymy zbiór jego wierzchołków, natomiast przez $R(W)$ zbiór jego promieni ekstremalnych. Przypomnijmy, że dla każdego zbioru wielościennego $W \subset R^n$ zachodzi

$$W = \operatorname{conv} V(W) + \operatorname{cone} R(W).$$

Jeśli W jest wielościanem, to $R(W) = \emptyset$ i powyższa równość redukuje się do $W = \operatorname{conv} V(W)$.

Jeśli x jest wierzchołkiem wielościanu W, to *wierzchołkiem sąsiednim dla* x nazywamy każdy wierzchołek w taki, że odcinek o końcach x, w jest krawędzią wielościanu. Zbiór wierzchołków sąsiednich dla x (przy ustalonym wielościanie) oznaczymy $V_s(x)$.

Niech $W \subset R^n$ będzie wielościanem mającym niepuste wnętrze. Powiemy, że jest on *niezdegenerowany*(1), jeśli każdy jego wierzchołek ma dokładnie n sąsiednich wierzchołków.

Przypomnijmy ponadto, że $f: P \to R$ jest funkcją *wklęsłą* na zbiorze wypukłym $P \subset R^n$, jeśli dla dowolnych $x, y \in P$ oraz każdej liczby λ takiej, że $0 \leqslant \lambda \leqslant 1$ zachodzi

$$f(\lambda x + (1-\lambda)y) \geqslant \lambda f(x) + (1-\lambda) f(y).$$

2.2. Zakładamy, że wielościan D występujący w sformułowaniu zadania (PW) jest dany za pomocą układu nierówności liniowych, to jest

(1) Podana tu definicja wielościanu niezdegenerowanego jest innym sformułowaniem warunku niezdegenerowania występującego w programowaniu liniowym.

$$D = \{x \in R^n | Ax \leqslant b\},$$

gdzie A jest macierzą $m \times n$, natomiast $b \in R^m$. (Nierówność $w' \leqslant w''$ dla wektorów w', w'' przestrzeni R^m jest równoważna temu, że $w_i' \leqslant w_i''$ dla $i = 1, \ldots, m$).

Bez straty ogólności możemy przyjąć, że $\operatorname{int} D \neq \emptyset$. W celu uproszczenia rozważań zakładamy([2]) ponadto, że D jest wielościanem niezdegenerowanym (podobne założenie występuje w pracy [5]).

3. Pewne własności funkcji wklęsłej. Podamy tu kilka prostych lematów dotyczących funkcji wklęsłej określonej na zbiorze wielościennym. Będą one wykorzystane przy konstrukcji algorytmu i dowodzie jego zbieżności.

Wprost z definicji funkcji wklęsłej i równości $W = \operatorname{conv} V(W)$ dla wielościanu wynika poniższy lemat:

LEMAT 1. *Jeśli W jest wielościanem w R^n oraz f: $W \to R$ jest funkcją wklęsłą, to istnieje wierzchołek w taki, że*

$$(\forall x \in W) f(x) \geqslant f(w),$$

Inaczej mówiąc, funkcja wklęsła osiąga minimum globalne na wielościanie w jednym z jego wierzchołków.

Zwróćmy uwagę na następującą własność funkcji wklęsłej:

LEMAT 2. *Niech f: $R^n \to R$ będzie wklęsła oraz załóżmy, że dla pewnych $a \in R^n$ i $\alpha \in R$ zachodzi*

$$(\forall t \geqslant 1) f(ta) \geqslant \alpha.$$

Wówczas dla każdego $b \in R^n$ i każdego $t \geqslant 0$ spełnione jest

$$f(b + ta) \geqslant f(b).$$

Dowód. Ustalmy $b \in R^n$, $t \geqslant 1$. Dla każdego $\varepsilon > 1$ mamy z definicji funkcji wklęsłej

$$f(b+ta) = f\left(\frac{1}{\varepsilon}(\varepsilon b) + \frac{\varepsilon - 1}{\varepsilon}\left(\frac{\varepsilon t}{\varepsilon - 1} a\right)\right) \geqslant \frac{1}{\varepsilon} f(\varepsilon b) + \frac{\varepsilon - 1}{\varepsilon} f\left(\frac{\varepsilon t}{\varepsilon - 1} a\right).$$

Ale $\dfrac{\varepsilon t}{\varepsilon - 1} \geqslant 1$, z założenia wynika więc, że $f\left(\dfrac{\varepsilon t}{\varepsilon - 1} a\right) \geqslant \alpha$ i w konsekwencji

$$f(b+ta) \geqslant \frac{1}{\varepsilon} f(\varepsilon b) + \frac{\varepsilon - 1}{\varepsilon} \alpha.$$

f, jako funkcja wklęsła na całej przestrzeni R^n, jest ciągła. Stąd

$$\lim_{\varepsilon \to 1} \left(\frac{1}{\varepsilon} f(\varepsilon b) + \frac{\varepsilon - 1}{\varepsilon} \alpha\right) = f(b)$$

([2]) Przypadek ogólny, tj. wielościanu zdegenerowanego, można sprowadzić do rozpatrywanego tu za pomocą np. techniki analogicznej do metody perturbacji znanej z programowania liniowego. Możliwe jest również uogólnienie opisanego algorytmu, w którym nie korzysta się z metody perturbacji (zob. notki 3 i 4).

i ostatecznie otrzymujemy $f(b+ta) \geqslant f(b)$. Dla $t \in [0, 1)$ nierówność ta jest również prawdziwa, z definicji bowiem funkcji wklęsłej i pierwszej części dowodu mamy

$$f(b+ta) \geqslant (1-t)f(b)+tf(b+a) \geqslant f(b). \blacksquare$$

Poniższy lemat 3 można uważać za ogólniejszą wersję lematu 1.

LEMAT 3. *Niech W będzie zbiorem wielościennym o wierzchołkach $w^1, \ldots, w^m$ i promieniach ekstremalnych $p^1, \ldots, p^r$ oraz $f\colon W \to R^n$ funkcją wklęsłą. Wówczas dla każdego $x \in W$ zachodzi:*

$$f(x) \geqslant \inf\{f(w^i+tp^j) \mid i = 1, \ldots, m, j = 1, \ldots, r, t \geqslant 0\}.$$

D o w ó d. Jak wiemy, zbiór wielościenny W można przedstawić w postaci $W = \operatorname{conv}\{w^1, \ldots, w^m\}+\operatorname{cone}\{p^1, \ldots, p^r\}$. Jeśli więc $x \in W$, to $x = \sum_{i=1}^{m} \lambda_i w^i + \sum_{j=1}^{r} \mu_j p^j$ dla pewnych $\lambda_i \geqslant 0, \mu_j \geqslant 0$ $(i = 1, \ldots, m, j = 1, \ldots, r)$, przy czym $\sum_{i=1}^{m} \lambda_i = 1$.

Niech J będzie zbiorem indeksów, $J = \{i \mid \lambda_i > 0\}$ oraz m' jego mocą. Wówczas możemy zapisać $x = \sum_{i \in J} \lambda_i w^i + \sum_{j=1}^{r} \mu_j p^j$, co z kolei przekształca się następująco:

$$x = r \sum_{i \in J} \frac{\lambda_i}{r} w^i + m' \sum_{j=1}^{r} \frac{\mu_j}{m'} p^j = \sum_{i \in J} \sum_{j=1}^{r} \frac{\lambda_i}{r} \left(w^i + \frac{r\mu_j}{m'\lambda_i} p^j \right).$$

Ponieważ $\sum_{i \in J} \sum_{j=1}^{r} \lambda_i/r = 1$ oraz $\lambda_i/r \geqslant 0$ dla $i \in J$, to

$$x \in \operatorname{conv}\left\{ w^i + \frac{r\mu_j}{m'\lambda_i} p^j \;\middle|\; i \in J, j = 1, \ldots, r \right\}.$$

Z lematu 1 otrzymujemy

$$f(x) \geqslant \min\left\{ f\left(w^i + \frac{r\mu_j}{m'\lambda_i} p^j \right) \;\middle|\; i \in J, j = 1, \ldots, r \right\}$$

i stąd natychmiast

$$f(x) \geqslant \inf\{f(w^i+tp^j) \mid i = 1, \ldots, m, j = 1, \ldots, r, t \geqslant 0\}. \blacksquare$$

Bezpośrednio z lematów 2 i 3 otrzymujemy

WNIOSEK. *Niech W będzie zbiorem wielościennym o wierzchołkach $w^1, \ldots, w^m$ i promieniach ekstremalnych $p^1, \ldots, p^r$ oraz $f\colon R^n \to R$ funkcją wklęsłą. Ponadto załóżmy, że istnieje liczba rzeczywista α taka, że dla $j = 1, \ldots, r$ oraz $t \geqslant 1$ zachodzi $f(tp^j) \geqslant \alpha$. Wówczas $(\exists w^i)(\forall x \in W) f(x) \geqslant f(w^i)$.*

4. Idea algorytmu.

4.1. Algorytm opisywany w niniejszym artykule w pierwszej fazie wyznacza wierzchołek wielościanu D będący minimum lokalnym funkcji f na D. Bez straty ogólności można przyjąć, że ten wierzchołek jest początkiem układu współrzędnych 0.

Ideę algorytmu można przedstawić następująco:

Niech $\hat{x}$ będzie wierzchołkiem o najmniejszej wartości funkcji celu f wśród wierzchołków zbadanych dotychczas oraz $\hat{\alpha} = f(\hat{x})$. Mamy daną rodzinę Z n-elementowych afinicznie niezależnych podzbiorów R^n, $Z = \{U_1, \ldots, U_p\}$, o której wiemy, że jeśli $x \in V(D) \setminus \bigcup_{U \in Z} \operatorname{cone} U$, to $f(x) \geqslant \hat{\alpha}$. Ponadto znane są dolne oszacowania $\alpha(U)$ wartości funkcji f na zbiorach $V(D) \cap \operatorname{cone} U$ dla $U \in Z$. Jeśli $Z = \emptyset$ lub $\min_{U \in Z} \alpha(U) \geqslant \hat{\alpha}$, to $\hat{x}$ jest oczywiście rozwiązaniem optymalnym zadania (PW). W przeciwnym przypadku wybieramy $\bar{U} \in Z$ taki, że $\alpha(\bar{U}) = \min_{U \in Z} \alpha(U)$ i badamy, czy w zbiorze $V(D) \cap \operatorname{cone} \bar{U}$ mogą istnieć wierzchołki mające mniejszą niż $\hat{\alpha}$ wartość funkcji f. W tym celu rozwiązujemy pewne zadanie pomocnicze, prostsze niż (PW), opisane dokładniej w następnym punkcie. Po jego rozwiązaniu aktualizujemy rodzinę Z eliminując z niej zbiór $\bar{U}$ (jeśli odpowiedź jest negatywna) lub wprowadzając do niej zamiast $\bar{U}$ pewne nowe podzbiory $U_{p+1}, \ldots, U_{p+k}$ takie, że $\bigcup_{i=1}^{k} \operatorname{cone} U_{p+i} = \operatorname{cone} \bar{U}$ (jeśli odpowiedź jest pozytywna); konstrukcja zbiorów U_{p+i} opisana jest w pkcie 7.

4.2. Wspomnieliśmy powyżej, że w początkowej fazie algorytm wyznacza wierzchołek $x^0 \in V(D)$ będący minimum lokalnym funkcji f na D, a dokładniej taki, że $f(x) \geqslant f(x^0)$ dla $x \in V_s(x^0)$. Dokonujemy tego w sposób naszkicowany już w § 1.2. Tu ograniczymy się więc jedynie do zapisania ścisłego schematu tej prostej procedury.

1° Znajdź dowolny wierzchołek $v^1 \in V(D)$, $i = 1$.

2° Oblicz $f(v)$ dla $v \in V_s(v^i)$. Wyznacz $f^i = \min\{f(v) \mid v \in V_s(v^i)\} = f(\bar{v})$.

3° Jeśli $f^i < f(v^i)$, to idź do 4°. W przeciwnym przypadku $x^0 = v^i$. Stop.

4° $v^{i+1} = \bar{v}$. Zwiększ i o 1; idź do 2°.

Zgodnie z uwagą w § 4.1 przyjmijmy dalej, że $x^0 = 0$.

Wobec założenia o niezdegenerowaniu wielościanu D (zob. § 2.2), wyznaczenie zbioru $V_s(v^i)$ wierzchołków sąsiednich dla v^i, wymagane w kroku 2°, jest proste([3]) i sprowadza się do wykonania n przekształceń sympleksowych macierzy bazowej układu $Ax + Iy = b, y \geqslant 0$, odpowiadającej rozwiązaniu bazowemu, w którym $x = v^i$.

([3]) Gdy pominiemy założenie o niezdegenerowaniu, wówczas do wyznaczania zbioru wierzchołków sąsiednich dla v^i wskazane jest użycie metody lepszej niż zwykły przegląd wszystkich rozwiązań bazowych, sąsiednich dla rozwiązania układu $Ax + Iy = b, y \geqslant 0$, w którym $x = v^i$. Metoda taka może być oparta np. na algorytmie Czernikowej (Żurn. Wyczysl. Mat. i Mat. Fiz. 5, 2 (1965); por. też [2]).

Uzupełnieniem początkowej iteracji jest ustalenie startowej rodziny Z. Ma ona jeden element(4), mianowicie $Z = \{V_s(x^0)\}$.

5. Zadanie pomocnicze.

5.1. Zadanie pomocnicze jest jednoznacznie określone za pomocą pewnego liniowo niezależnego układu $U = \{u^1, \ldots, u^n\} \subset R^n$ i liczby rzeczywistej α spełniających poniższe warunki (a) i (b):

(a) $(\forall u^i \in U) f(u^i) \geqslant \alpha$,

(b) $f(0) \geqslant \alpha$.

Zadanie pomocnicze dla danej pary (U, α) oznaczać będziemy przez $ZP(U, \alpha)$ a otrzymujemy je w następujący sposób:

Dla każdego $i = 1, \ldots, n$ znajdujemy

$$\bar{\Theta}_i = \sup\{\Theta \in R |\ f(\Theta u^i) \geqslant \alpha\}.$$

Niech I oznacza zbiór indeksów, $I = \{i \in N |\ \bar{\Theta}_i < +\infty\}$, gdzie $N = \{1, \ldots, n\}$. Jeżeli $I = \emptyset$, to powiemy, że zadanie pomocnicze nie ma rozwiązań. Załóżmy, że $I \neq \emptyset$. Połóżmy $\Theta_i = \bar{\Theta}_i$ dla $i \in I$ oraz $\Theta_i = 1$ dla $i \in N \setminus I$. Określamy macierz $B = (B_1, \ldots, B_n)$, której kolumny są wektorami $B_i = \Theta_i \cdot u^i$ dla $i = 1, \ldots, n$. Jest ona nieosobliwa. Za pomocą macierzy B definiujemy następujące obiekty:

stożek $K(B) = \{x \in R^n |\ B^{-1}x \geqslant 0\}$,

hiperpłaszczyznę $H(B) = \left\{x \in R^n \,\middle|\, \sum_{i \in I} (B^{-1}x)_i = 1\right\}$,

półprzestrzeń $P(B) = \left\{x \in R^n \,\middle|\, \sum_{i \in I} (B^{-1}x)_i > 1\right\}$,

gdzie $(B^{-1}x)_i$ oznacza i-tą współrzędną wektora $B^{-1}x$.

5.2. Zauważmy, że hiperpłaszczyzna $H(B)$ zawiera punkty B_i dla $i \in I$ oraz jest równoległa do kierunków wektorów B_j dla $j \in N \setminus I$. Wobec liniowej niezależności zbioru $\{B_1, \ldots, B_n\}$ jest to jedyna hiperpłaszczyzna o tej własności. $P(B)$ jest otwartą półprzestrzenią, wyznaczoną przez $H(B)$ i nie zawierającą początku układu współrzędnych. Ponadto zachodzi równość $K(B) = \operatorname{cone} U$.

Istotną własnością macierzy B, wykorzystywaną w dowodzie zbieżności algorytmu, jest to, że jeśli $x \in K(B) \setminus P(B)$, to $f(x) \geqslant \alpha$ (zob. lemat 4). Można też pokazać, że jeśli P' jest inną półprzestrzenią otwartą taką, że dla $x \in K(B) \setminus P'$ zachodzi $f(x) \geqslant \alpha$, to $K(B) \setminus P(B) \supset K(B) \setminus P'$. Inaczej mówiąc, hiperpłaszczyzna $H(B)$

---

(4) W wielościanie zdegenerowanym zbiór $V_s(x^0)$ może mieć więcej niż n elementów i dlatego inaczej należy konstruować startową rodzinę Z. W pewnych przypadkach łatwo znaleźć taki n-elementowy liniowo niezależny zbiór U, że $\operatorname{cone} U \supset D$ oraz $f(u) \geqslant f(x^0)$ dla $u \in U$. Wtedy przyjmiemy startowe $Z = \{U\}$ (por. [2]). Natomiast zawsze można wyznaczyć taką rodzinę Z n-elementowych liniowo niezależnych zbiorów, że dla każdego $U \in Z$ zachodzi $U \subset V_s(x^0)$ oraz $\bigcup_{U \in Z} \operatorname{cone} U \supset D$. W pracy [1] opisano procedurę „P" generującą taką rodzinę, która ponadto spełnia warunek $\operatorname{int} \operatorname{cone} U' \cap \operatorname{int} \operatorname{cone} U'' = \emptyset$ dla $U' \neq U''$, $U', U'' \in Z$.

„odcina" największą możliwą (przy użyciu hiperpłaszczyzny) część stożka $K(B) = \operatorname{cone} U$ nie zawierającą punktów o mniejszej niż α wartości funkcji f.

5.3. Niech $\varrho\,(x, H(B))$ oznacza odległość punktu $x \in R^n$ od hiperpłaszczyzny $H(B)$. Rozwiązując zadanie pomocnicze ZP(U, α) poszukujemy takiego punktu $x^* \in V(D) \cap K(B) \cap P(B)$, że

$$\varrho(x^*, H(B)) = \max \varrho(x, H(B)) \quad \text{dla} \quad x \in V(D) \cap K(B) \cap P(B).$$

Rozwiązaniem dopuszczalnym dla ZP(U, α) nazywamy każdy punkt x należący do zbioru $V(D) \cap K(B) \cap P(B)$.

Dla sformułowania procedury rozwiązującej zadanie pomocnicze wygodniejsza będzie inna, ale równoważna powyższej, jego postać. Wobec liniowej niezależności zbioru $\{B_1, \ldots, B_n\}$ dla każdego $x \in R^n$ istnieje dokładnie jeden wektor $\lambda \in R^n$ taki, że $x = B\lambda$. Zauważmy, że zachodzą następujące związki:

$$\text{(c)} \quad x \in K(B) \Leftrightarrow \lambda \geqslant 0, \quad x \in P(B) \Leftrightarrow \sum_{i \in I} \lambda_i > 1, \quad x \in V(D) \Leftrightarrow \lambda \in V(D'),$$

gdzie D' jest wielościanem, $D' = \{\lambda \in R^n \mid AB\lambda \leqslant b\}$.

Odległość $\varrho(x, H(B))$ można wyrazić wzorem

$$\varrho(x, H(B)) = g \Big| \sum_{i \in I} \lambda_i - 1 \Big|,$$

gdzie $\lambda = (\lambda_1, \ldots, \lambda_n) = B^{-1}x$, $g = 1 / \big\| \sum_{i \in I} B_i \big\|$ jest dodatnim współczynnikiem nie zależącym od x, a $\| \cdot \|$ oznacza euklidesową normę w R^n. Wynika z tego, że dla $x \in P(B)$, a więc gdy $\sum_{i \in I} \lambda_i = \sum_{i \in I} (B^{-1}x)_i > 1$, odległość punktu x od hiperpłaszczyzny $H(B)$ można zapisać jako

$$\varrho(x, H(B)) = g \Big(\sum_{i \in I} \lambda_i - 1 \Big).$$

Z powyższych spostrzeżeń otrzymujemy, że x^* jest rozwiązaniem ZP(U, α) wtedy i tylko wtedy, kiedy $\lambda^* = B^{-1}x^*$ jest rozwiązaniem zadania

$$\text{(ZP)} \qquad \max \sum_{i \in I} \lambda_i, \quad \lambda \in V(D'), \lambda \geqslant 0, \quad \sum_{i \in I} \lambda_i > 1.$$

5.4. Zadanie (ZP) jest szczególnym przypadkiem następującego π-problemu rozpatrywanego przez Pollatschka i Avi-Itzhaka [3]:

$$(\pi) \qquad \max_{x \in \pi = P \cap C} c^T x,$$

gdzie $c \in R^n$, $P \subset R^n$ jest wielościanem wypukłym, natomiast $C \subset R^n$ jest takim zbiorem, że $\pi = P \cap C \subset V(P)$.

W pracy [3] autorzy opisali algorytm rozwiązujący zadania tego typu. W pierwszym kroku wymaga on zastosowania algorytmu „sympleks" do zadania liniowego $\max \{c^T x \mid x \in P\}$. Następnie wierzchołki wielościanu P są przeglądane w porządku

malejących wartości funkcji celu. Pierwszy napotkany wierzchołek należący do C jest rozwiązaniem π-problemu.

Przyjmijmy $P = \{\lambda \in R^n |\ AB\lambda \leqslant b,\ \lambda \geqslant 0\}$, $C = V(D') \cap \{\lambda \in R^n |\ \sum_{i \in I} \lambda_i > 1\}$ oraz $c^T = (c_1, \ldots, c_n)$, gdzie $c_i = 1$ dla $i \in I$, $c_i = 0$ dla $i \in N \setminus I$.

Przy tak określonych P, C, c wystarczy w (π) podstawić $x = \lambda$, aby otrzymać zadanie (ZP).

5.5 Schemat procedury rozwiązującej ZP(U, α) zapiszemy następująco:

1° Oblicz $\bar{\Theta}_i$ dla $i = 1, \ldots, n$. Jeśli $\min \bar{\Theta}_i < +\infty$, to idź do 2°. W przeciwnym przypadku ZP(U, α) nie ma rozwiązań. Stop.

2° Wyznacz macierz $B = (B_1, \ldots, B_n)$.

3° Rozwiąż zadanie liniowe

$$\max_{AB\lambda \leqslant b,\ \lambda \geqslant 0} \sum_{i \in I} \lambda_i .$$

Niech λ^1 będzie rozwiązaniem. $k := 1$.

4° Jeśli $\sum_{i \in I} \lambda_i^k > 1$, to idź do 5°. W przeciwnym przypadku ZP(U, α) nie ma rozwiązań. Stop.

5° Jeśli $\lambda^k \in V(D')$, to $x^* = B\lambda^k$ jest rozwiązaniem ZP(U, α). Stop.

W przeciwnym przypadku idź do 6°.

6° Jeśli $\{\lambda^1, \ldots, \lambda^k\} = V(P)$, to ZP$(U, \alpha)$ nie ma rozwiązań. Stop.

W przeciwnym przypadku wyznacz wierzchołek λ^{k+1}, $\lambda^{k+1} \in V(P) \setminus \{\lambda^1, \ldots, \lambda^k\}$ taki, że

$$(\forall\ \lambda \in V(P) \setminus \{\lambda^1, \ldots, \lambda^k\}) \sum_{i \in I} \lambda_i \leqslant \sum_{i \in I} \lambda_i^{k+1}.$$

Zwiększ k o 1; idź do 4°.

Jedynym elementem powyższej procedury wymagającym odwołania się do metody [3] jest wyznaczenie punktu λ^{k+1} w 6°.

6. Własności zadania pomocniczego. Ustalmy liniowo niezależny zbiór $U = \{u^1, \ldots, u^n\} \subset R^n$ i liczbę α taką, że $f(u^i) \geqslant \alpha$ dla $i = 1, \ldots, n$; $f(0) \geqslant \alpha$. W sposób opisany w § 5.1 wyznaczamy macierz B i zbiór indeksów I.

LEMAT 4. *Dla każdego $x \in K(B) \setminus P(B)$ zachodzi $f(x) \geqslant \alpha$.*

D o w ó d. Macierz B określiliśmy tak, że $f(B_i) \geqslant \alpha$, $i = 1, \ldots, n$, a ponadto $f(\Theta B_i) \geqslant \alpha$ dla $\Theta \geqslant 1$, $i \in N \setminus I$. Zbiór $K(B) \setminus P(B)$ jest albo wielościanem o wierzchołkach $B_1, \ldots, B_n$, 0 (jeśli $I = N$), albo nieograniczonym zbiorem wielościennym o wierzchołkach B_i dla $i \in I$, 0 i promieniach ekstremalnych B_j dla $j \in N \setminus I$. W pierwszym przypadku tezę otrzymujemy natychmiast z lematu 1, w drugim zaś z wniosku podanego po lemacie 3. ■

Poniższe spostrzeżenie wyjaśnia rolę zadań pomocniczych ZP(U, α) w algorytmie problemu (PW).

WNIOSEK. *Jeśli* ZP(U, α) *nie ma rozwiązań, to w stożku* cone U *nie istnieje wierzchołek wielościanu* D, *któremu odpowiada mniejsza niż* α *wartość funkcji* f.

Dowód. Przypomnijmy, że cone $U = K(B)$. Z założenia wynika, że jeśli $x \in V(D) \cap \text{cone } U$, to $x \in K(B) \setminus P(B)$. Z lematu 4 natychmiast otrzymujemy $f(x) \geqslant \alpha$. ■

7. Algorytm. Przed podaniem schematu algorytmu omówimy sposób modyfikowania rodziny Z oraz wyznaczania ograniczeń $\alpha(U)$ dla $U \in Z$.

7.1. Jak stwierdziliśmy już w § 4.1, rozwiązując zadanie pomocnicze dla wybranego elementu $U \in Z$, szukamy odpowiedzi na pytanie, czy w zbiorze cone U istnieją wierzchołki wielościanu D o mniejszej wartości funkcji f niż $\hat{\alpha}$. Jeśli odpowiedź jest negatywna, czyli ZP(U, $\hat{\alpha}$) nie ma rozwiązań, to nowa rodzina Z' ma postać $Z' = Z \setminus \{U\}$. W przeciwnym przypadku nie możemy wykluczyć zbioru U (dokładniej: cone U) z dalszych rozważań i postępujemy wówczas następująco:

Przede wszystkim, jeśli w punkcie x^* będącym rozwiązaniem zadania pomocniczego zachodzi $f(x^*) < \hat{\alpha} = f(\hat{x})$, gdzie $\hat{x}$ jest najlepszym znalezionym dotychczas wierzchołkiem, to przyjmujemy nowe $\hat{x} = x^*$ oraz $\hat{\alpha} = f(x^*)$. Zbiór U usuwamy z rodziny Z wprowadzając do niej zamiast niego zbiory

$$U_j = \{B_1, \ldots, B_{j-1}, x^*, B_{j+1}, \ldots, B_n\}$$

dla

$$j \in N(x^*) = \{i \in N \mid \lambda_i^* = (B^{-1}x^*)_i > 0\},$$

gdzie B_i są kolumnami macierzy B znalezionej dla U (zob. § 5.1). Zauważmy, że U_j dla $j \in N(x^*)$ są liniowo niezależne oraz $\bigcup_{j \in N(x^*)} \text{cone } U_j = \text{cone } U$. Tak więc nowa rodzina Z' ma postać

$$Z' = (Z \setminus \{U\}) \cup \{U_j \mid j \in N(x^*)\}.$$

7.2. Uzupełnieniem procedury modyfikującej wartość $\hat{\alpha}$, wektor $\hat{x}$ oraz rodzinę Z w każdej iteracji jest szacowanie dolnych ograniczeń wartości funkcji f w zbiorach $V(D) \cap \text{cone } U_j$ dla $j \in N(x^*)$. Mianowicie określmy liczby

$$\alpha(U_j) = \min\left(\{f(\eta B_i) \mid i \in I, i \neq j\} \cup \{f(0), f(x^*)\}\right)$$

dla $j \in N(x^*)$, przy czym $\eta = \sum_{i \in I} \lambda_i^*$ jest optymalną wartością funkcji celu w zadaniu (ZP).

LEMAT 5. *Jeśli* $x \in V(D) \cap \text{cone } U_j$, *to* $f(x) \geqslant \alpha(U_j)$.

Dowód. Ustalmy $x \in V(D) \cap \text{cone } U_j$. Istnieją liczby $\gamma_1, \ldots, \gamma_n \geqslant 0$ takie, że $x = \sum_{i \neq} \gamma_i B_i + \gamma_j x^*$. Ale $x^* = B\lambda^* = \sum_{i=1}^{n} \lambda_i^* B_i$. Z tego otrzymujemy następujące

przedstawienie punktu x:

$$x = \sum_{i \neq j} \gamma_i B_i + \gamma_j \Big(\sum_{i=1} \lambda_i^* B_i \Big) = \sum_{i \neq j} (\gamma_i + \gamma_j \lambda_i^*) B_i + \gamma_j \lambda_j^* B_j = \sum_{i=1}^{n} \lambda_i B_i,$$

gdzie

$$\lambda_i = \begin{cases} \gamma_i + \gamma_j \lambda_i^* & \text{dla} \quad i \neq j, \\ \gamma_j \lambda_j^* & \text{dla} \quad i = j. \end{cases} \tag{1}$$

Ponieważ x^* jest optymalnym rozwiązaniem zadania pomocniczego, a $x \in \operatorname{cone} U = K(B)$, więc

$$\sum_{i \in I} \lambda_i \leqslant \sum_{i \in I} \lambda_i^* = \eta. \tag{2}$$

Po prostych rachunkach otrzymujemy z (1) i (2)

$$\sum_{\substack{i \in I \\ i \neq j}} \gamma_i + \eta \gamma_j \leqslant \eta. \tag{3}$$

Wykonujemy następujące trywialne przekształcenie:

$$x = \sum_{i \neq j} \gamma_i B_i + \gamma_j x^* = \sum_{\substack{i \in I \\ i \neq j}} \gamma_i B_i + \sum_{\substack{i \notin I \\ i \neq j}} \gamma_i B_i + \gamma_j x^* =$$

$$= \sum_{\substack{i \in I \\ i \neq j}} \frac{\gamma_i}{\eta} (\eta B_i) + \gamma_j x^* + \sum_{\substack{i \notin I \\ i \neq j}} \gamma_i B_i.$$

Mamy przy tym $\gamma_i/\eta \geqslant 0$ dla $i \in I$, $i \neq j$, $\gamma_j \geqslant 0$ oraz $\gamma_i \geqslant 0$ dla $i \notin I$, a ponadto z (3)

$$\sum_{\substack{i \in I \\ i \neq j}} \gamma_i/\eta + \gamma_j \leqslant 1.$$

Wynika z tego, że $x \in \operatorname{conv}(\{0, x^*\} \cup \{\eta B_i |\ i \in I, i \neq j\}) + \operatorname{cone}\{B_i |\ i \notin I, i \neq j\}$. Ponieważ dla $i \notin I$ zachodzi $f(\Theta B_i) \geqslant \alpha$ dla każdego $\Theta \geqslant 1$, to na podstawie wniosku z lematu 3 otrzymujemy

$$f(x) \geqslant \min\big(\{f(\eta B_i) |\ i \in I, i \neq j\} \cup \{f(0), f(x^*)\}\big). \quad \blacksquare$$

7.3. Znamy już wszystkie elementy opisywanego algorytmu. Poniżej przedstawiamy jego pełny schemat.

Krok 0. (Iteracja początkowa) Zgodnie z rozważaniami w § 4.2 wyznacz $x^0 \in V(D)$, $\alpha_0 := f(x^0)$ (zakładamy dalej, że $x^0 = 0$). $Z_0 := \{V_s(x^0)\}$; $\alpha(V_s(x^0)) := d$, gdzie d jest dolnym ograniczeniem f na D (dostatecznie mała liczba). $k := 0$.

Krok 1. Jeśli $Z_k = \emptyset$, to x^k jest minimum globalnym. Stop.

W przeciwnym razie oblicz $\bar{\alpha} := \min\{\alpha(U) |\ U \in Z_k\}$. Idź do kroku 2.

Krok 2. Jeśli $\bar{\alpha} \geqslant \alpha_k$, to x^k jest minimum globalnym. Stop.

W przeciwnym razie wybierz $U \in Z_k$ taki, że $\alpha(U) = \bar{\alpha}$. Połóż $Z'_k := \{U \in Z_k | \alpha(U) < \alpha_k\}$. Idź do kroku 3.

Krok 3. Rozwiąż zadanie ZP(U, α_k). Jeśli nie ma rozwiązania, to idź do kroku 4. W przeciwnym przypadku idź do kroku 5.

Krok 4. Połóż $Z_{k+1} := Z'_k \setminus \{U\}$, $x^{k+1} := x^k$, $\alpha_{k+1} := \alpha_k$. Zwiększ k o 1. Idź do kroku 1.

Krok 5. Niech $x^* = B\lambda^*$ będzie rozwiązaniem ZP(U, α_k). Przyjmij $x^{k+1} := x^*$, jeśli $f(x^*) < \alpha_k$ albo $x^{k+1} := x^k$, jeśli $f(x^*) \geqslant \alpha_k$; $\alpha_{k+1} := f(x^{k+1})$.

Krok 6. Niech $N(x^*) = \{i | \lambda_i^* > 0\}$. $U_j := \{B_1, \ldots, B_{j-1}, x^*, B_{j+1}, \ldots, B_n\}$ dla $j \in N(x^*)$. Oblicz $\alpha(U_j)$ dla $j \in N(x^*)$ zgodnie z § 7.2. Połóż $Z_{k+1} := (Z'_k \setminus \{U\}) \cup \cup \{U_j | \ j \in N(x^*)\}$. Zwiększ k o 1. Idź do kroku 1.

8. Dowód zbieżności algorytmu.

8.1. Pokażemy najpierw, że algorytm jest skończony, tzn. istnieje k takie, że $Z_k = \emptyset$ lub $\bar{\alpha} \geqslant \alpha_k$. Zauważmy, że z konstrukcji rodziny Z_k dla $k = 0, 1, \ldots$ oraz definicji zadania pomocniczego wynika, że każde rozwiązywane w toku działania algorytmu zadanie pomocnicze jest elementem przynajmniej jednego ciągu postaci

$$\text{ZP}(U^0, \alpha_{k_0}), \ldots, \text{ZP}(U^s, \alpha_{k_s}),$$

gdzie $k_0 = 0$, $U^0 \in Z_0$, $U^{j+1} = (\tilde{B}^j \setminus \{B^j_{i_j}\}) \cup \{x^*_j\}$ dla pewnego $i_j \in N(x^*_j)$, $j = 0, \ldots, s-1$, przy czym $\tilde{B}^j$ jest zbiorem kolumn macierzy B dla zadania ZP(U^j, α_{k_j}), $B^j_{i_j}$ jest kolumną o numerze i_j, x^*_j zaś jest rozwiązaniem zadania ZP(U^j, α_{k_j}). Ponadto $k_j < k_{j+1}$ dla $j = 0, 1, \ldots, s-1$.

TWIERDZENIE 1. *Każdy ciąg* ZP(U^0, α_{k_0}), ..., ZP(U^s, α_{k_s}) *opisany powyżej ma nie więcej niż* $\overline{\overline{V(D)}}+1$ *elementów.*

Dowód. Pokażemy, że $x^*_j \neq x^*_p$ dla $j < p$; $j, p = 0, \ldots, s-1$. Ustalmy $p \leqslant s-1$. Dla $j = 0, 1, \ldots, p$ oznaczmy $K_j = \text{cone } U^j$ (przypomnijmy, że cone $U^j = K(B^j)$) oraz $P_j = P(B^j)$, gdzie B^j jest macierzą B dla zadania ZP(U^j, α_{k_j}). Dla $j = 0, 1, \ldots, p-1$ zachodzi inkluzja $U^{j+1} \subset K_j$. Z tego otrzymujemy

$$K_0 \supset K_1 \supset K_2 \supset \ldots \supset K_p. \tag{1}$$

Można teraz pokazać (łatwy dowód rachunkowy wykorzystujący wprost definicję zbiorów P_j, P_{j+1}, K_{j+1} oraz związki (c) z § 5.3 pomijamy), że:

$$K_{j+1} \setminus P_{j+1} \supset K_{j+1} \setminus P_j, \quad j = 0, 1, \ldots, p-1. \tag{2}$$

Z (1) i (2) wynika następująca zależność:

$$K_p \setminus P_{j+1} \supset K_p \setminus P_j, \quad j = 0, 1, \ldots, p-1. \tag{3}$$

Przypuśćmy, że dla pewnego $j < p$ mamy $x^*_j = x^*_p$. Wówczas z konstrukcji zadania pomocniczego otrzymujemy

$$x^*_j \in K_p \cap P_p. \tag{4}$$

Zauważmy jednak, że z definicji zbiorów K_{j+1} i P_{j+1} oraz z konstrukcji macierzy B^{j+1} wynika

$$x_j^* \in K_{j+1} \setminus P_{j+1}. \tag{5}$$

Otrzymujemy więc $x_j^* \in K_p \setminus P_{j+1}$, a ponieważ na podstawie (3) $K_p \setminus P_{j+1} \subset K_p \setminus P_p$, to ostatecznie wynika z tego

$$x_j^* \notin P_p, \tag{6}$$

co jest oczywiście sprzeczne z (4). Dlatego musi być $x_j^* \neq x_p^*$.

Pokazaliśmy w ten sposób, że dla każdego $p = 1, \ldots, s-1$ i każdego $j = 0, 1, \ldots, p-1$ zachodzi $x_j^* \neq x_p^*$. Ponieważ $x_j^* \in V(D)$, $j = 0, 1, \ldots, s-1$, a zbiór $V(D)$ jest skończony, to otrzymujemy z tego $s \leqslant \overline{\overline{V(D)}}$. ∎

8.2. Wniosek. *Algorytm jest skończony, tzn. istnieje k takie, że* $\bar{\alpha} \geqslant \alpha_k$ *lub* $Z_k = \emptyset$.

Dowód. Każde zadanie pomocnicze rozwiązywane podczas stosowania algorytmu należy do pewnego ciągu opisanego w § 8.1. Liczba takich ciągów jest skończona, każdy bowiem zbiór postaci U^{j+1} możemy otrzymać z U^j dla danego α_{k_j} i wyznaczonego x_j^* na co najwyżej n sposobów. Wynika z tego i z twierdzenia 1, że wszystkich ciągów jest nie więcej niż n^M, gdzie $M = \overline{\overline{V(D)}}$, a więc skończona liczba. Konsekwencją powyższego jest to, że algorytm zawsze prowadzi do rozwiązywania skończonej liczby zadań pomocniczych. Jeżeli więc dla każdego k otrzymamy $\bar{\alpha} < \alpha_k$, to istnieje k takie, że $Z_k = \emptyset$. ∎

8.3. Pozostaje jedynie pokazać, że ostatni wierzchołek x^k wyznaczony przez algorytm jest rozwiązaniem optymalnym zadania (PW). Rozważymy dwa możliwe przypadki zakończenia algorytmu opisane w krokach 1 i 2. Wprowadźmy w tym celu zbiory T_k', T_k'' i T_k określone następująco:

$$T_k' = \bigcup_{j=0}^{k-1} (Z_j - Z_j'),$$

przy czym Z_j' jest zbiorem zdefiniowanym w kroku 2 algorytmu;

T_k'' = zbiór takich układów U, że zadanie $\mathrm{ZP}(U, \alpha_j)$ było rozwiązywane dla pewnego $j < k$, ale nie miało rozwiązań;

$$T_k = T_k' \cup T_k''.$$

Zauważmy, że dla każdego $k = 1, 2, \ldots$ zachodzi

$$\bigcup_{U \in T_k \cup Z_k} \operatorname{cone} U \supset D.$$

Twierdzenie 2. *Ostatni wyznaczony przez algorytm punkt* x^k *jest rozwiązaniem optymalnym zadania* (PW).

Dowód. Z wniosku w § 8.2 wiemy, że algorytm jest skończony. Niech k będzie numerem ostatniej iteracji. Sprowadza się ona do sprawdzenia, że $Z_k = \emptyset$ lub $\bar{\alpha} \geqslant \alpha_k$. Musimy pokazać, że x^k jest rozwiązaniem zadania (PW).

Niech $x \in V(D)$ będzie dowolnym wierzchołkiem. Istnieje $U \in T_k \cup Z_k$ takie, że $x \in \operatorname{cone} U$. Jeśli $U \in T_k = T_k' \cup T_k''$, to zachodzi jeden z następujących przypadków:

(a) $U \in Z_j \setminus Z'_j$ dla pewnego $j < k$; oznacza to, że $\alpha(U) \geqslant \alpha_j$, a ponieważ $\alpha_j \geqslant \alpha_k$ oraz $f(x) \geqslant \alpha(U)$, więc $f(x) \geqslant \alpha_k = f(x^k)$;

(b) $\mathrm{ZP}(U, \alpha_j)$ było rozwiązywane dla pewnego $j < k$, przy czym nie miało rozwiązania; wynika z tego, że $f(x) \geqslant \alpha_j$ (zob. wniosek w § 6.2), ponieważ jednak $\alpha_j \geqslant \alpha_k$, to znowu otrzymujemy $f(x) \geqslant \alpha_k = f(x^k)$.

Jeśli natomiast $U \in Z_k$ (czyli $Z_k \neq \emptyset$), to mamy do czynienia z przypadkiem $\bar{\alpha} \geqslant \alpha_k$ w iteracji k-tej; wiemy jednak, że $\alpha(U) \geqslant \bar{\alpha}$ i $f(x) \geqslant \alpha(U)$. Z tego wynika, że $f(x) \geqslant \bar{\alpha} \geqslant \alpha_k = f(x^k)$.

W ten sposób otrzymaliśmy następujące stwierdzenie:

$$(\forall x \in V(D))\ f(x) \geqslant f(x^k).$$

Z lematu 1 wynika teraz bezpośrednio, że x^k jest rozwiązaniem optymalnym zadania (PW). ■

Prace cytowane

[1] S. Kryński, *Algorytm programowania wklęsłego*, Prace IOK PAN 31, Warszawa 1976.

[2] —, *Minimization of a concave functions under linear constrains (Modification of Tuy's method)*, w: Survey of Mathematical Programming, Proceedings (ed. A. Prékopa), Budapest 1976.

[3] M. A. Pollatschek, B. Avi-Itzhak, *Sorting feasible basic solutions of a linear program*, w: Development in Operations Research (ed. B. Avi-Itzhak), Gordon&Breach, 1970.

[4] R. T. Rockafellar, *Convex analysis*, Princeton 1970.

[5] [H. Tuy], X. Туй, *Вогнутое программирование при линейных ограничениях*, Докл. АН СССР 159, 1 (1964).

[6] P. B. Zwart, *Nonlinear programming: Counterexamples to two global optimization algorithms*, Operations Res. 21 (1973).