Część trzecia

Ekonometria i badania operacyjne

Marcin ANHOLCER
Katedra Badań Operacyjnych
Akademia Ekonomiczna w Poznaniu

METODA RZUTOWANIA GRADIENTU DLA NIELINIOWYCH ZADAŃ ALOKACJI

Wstęp

Metoda rzutowania gradientu dla zadań programowania wypukłego została wprowadzona przez Rosena w roku 1960. Opiera się na znanym schemacie – w każdej iteracji najpierw wyznaczany jest kierunek poprawy rozwiązania (a więc kierunek wzdłuż którego funkcja celu polepsza swoją wartość), a następnie optymalna wielkość przesunięcia wzdłuż tego kierunku. Optymalną długość kroku wyznacza się – nie tylko w tej metodzie – stosując na przykład metody jednowymiarowego przeszukiwania liniowego. Cechą wyróżniającą algorytm rzutowania gradientu jest sposób wyznaczania kierunku poprawy.

Aby to uczynić, wykonuje się rzut gradientu na brzeg zbioru rozwiązań dopuszczalnych, wykorzystując do tego celu macierz rzutowania P. Jej wyznaczenie wymaga wykonania kilku działań mnożenia i odwrócenia macierzy dość dużych rozmiarów (odpowiednio zdefiniowanych podmacierzy macierzy ograniczeń) i jest najbardziej pracochłonną częścią każdej iteracji.

W przypadku nieliniowych zadań alokacji macierze ograniczeń mają specyficzną postać – składają się wyłącznie z elementów -1, 0 i 1, a ponadto są bardzo rzadkie (niewielki odsetek ich elementów różni się od 0). To prowadzi do dość naturalnego pytania, a mianowicie czy w przypadku tych zadań można procedurę wyznaczania kierunku poprawy uprościć i czy po takim uproszczeniu metoda rzutowania gradientu może konkurować z innymi metodami rozwiązywania nieliniowych zagadnień alokacji. Celem niniejszego artykułu jest udzielenie odpowiedzi na to pytanie.

W pierwszym punkcie przedstawiona została ogólna postać metody z poprawkami (w wersji oryginalnej algorytm nie zawsze był zbieżny, więcej na ten temat można znaleźć na przykład w pracach [5], [7], [8]). W drugim punkcie zaprezentowano zwięźle omawiane zagadnienia alokacji. Trzeci punkt opisuje wyprowadzenie postaci macierzy rzutowania P dla poszczególnych typów nieliniowych zadań alokacji i wynikające z niej uproszczone postacie algorytmu. W czwartym zaprezentowano wyniki eksperymentów obliczeniowych. Pracę zamykają wnioski i sugestie co do dalszych badań.

W artykule zastosowano następujące symbole matematyczne:
∇f – gradient funkcji f,
A^{T} – macierz transponowana do macierzy A,
A^{-1} – macierz odwrotna do macierzy A,
$\|x\|$ – norma wektora x,
$|x|$ – wartość bezwzględna liczby x,
$\mathbf{0}$ – wektor zerowy (wektor o wszystkich współrzędnych równych 0),
I – macierz jednostkowa,
A_i – wiersz i macierzy A,
x_i – współrzędna i wektora x.

1. Metoda rzutowania gradientu

Załóżmy, że dane jest zadanie programowania nieliniowego:

$$\begin{aligned} &\min f(x) \\ &\text{przy} \\ &A_1 x = b_1 \\ &A_2 x \leqslant b_2 \end{aligned} \tag{1}$$

Metodę rzutowania gradientu można zapisać w postaci (por. [5]):

1. Obierz stałą $c>0$. Podstaw $k:=1$ i wyznacz pierwsze rozwiązanie dopuszczalne $x^{(1)}$.
2. Niech $M^{(k)}$ będzie macierzą, której wiersze są wierszami macierzy A_1 i tymi wierszami macierzy A_2, które odpowiadają warunkom wiążącym w aktualnym rozwiązaniu.
3. Jeżeli macierz $M^{(k)}$ jest macierzą pustą i $\nabla f(x^{(k)})=\mathbf{0}$, to STOP. Uzyskane rozwiązanie należy do zbioru rozwiązań zadowalających Ω. Jeżeli

macierz $M^{(k)}$ jest macierzą pustą, ale $\nabla f(x^{(k)}) \neq \mathbf{0}$, wyznacz kierunek poprawy rozwiązania zgodnie ze wzorem

$$d^{(k)} = -\nabla f(x^{(k)}) \tag{2}$$

i przejdź do kroku 8. Jeżeli macierz $M^{(k)}$ nie jest macierzą pustą, wyznacz macierz rzutowania zgodnie ze wzorem:

$$P^{(k)} = I - (M^{(k)})^{\mathrm{T}} (M^{(k)} (M^{(k)})^{\mathrm{T}})^{-1} M^{(k)}. \tag{3}$$

4. Wyznacz wektor $d^{(k)*}$ zgodnie ze wzorem:

$$d^{(k)*} = -P^{(k)} \nabla f(x^{(k)}). \tag{4}$$

5. Wyznacz wektor przybliżonych wartości zmiennych dualnych[1] zgodnie ze wzorem:

$$w^{(k)} = -(M^{(k)} (M^{(k)})^{\mathrm{T}})^{-1} M^{(k)} \nabla f(x^{(k)}). \tag{5}$$

Podziel wektor na dwie części: $(w^{(k)})^{\mathrm{T}} = [(v^{(k)})^{\mathrm{T}} (u^{(k)})^{\mathrm{T}}]$, gdzie wektor $u^{(k)}$ odpowiada warunkom nierównościowym, a $v^{(k)}$ – równaniom. Dalsze postępowanie zależy od tego, który z trzech wykluczających się warunków jest spełniony:

- Jeżeli $u^{(k)} \geqslant \mathbf{0}$ i $d^{(k)*} = \mathbf{0}$, to STOP. Otrzymane rozwiązanie $x^{(k)}$ należy do zbioru rozwiązań zadowalających Ω.
- Jeżeli $u^{(k)} = \mathbf{0}$ i $d^{(k)*} \neq \mathbf{0}$, wyznacz wektor poprawy rozwiązania zgodnie ze wzorem $d^{(k)} = d^{(k)*}$ i przejdź do kroku 8.
- Jeżeli istnieje indeks i, dla którego $u_i^{(k)} < 0$, przejdź do kroku 6.

6. Wybierz indeks i^* taki, że

$$u_{i*} = \min\{u_i : u_i < 0\}. \tag{6}$$

Usuń z macierzy $M^{(k)}$ wiersz związany ze składową u_{i*} wyznacz macierz rzutowania zgodnie ze wzorem:

$$P^{(k)} = I - (M^{(k)})^{\mathrm{T}} (M^{(k)} (M^{(k)})^{\mathrm{T}})^{-1} M^{(k)}. \tag{7}$$

---

[1] Użyte sformułowanie należy rozumieć tak, że ciąg wektorów podanej postaci jest zbieżny do wektora zmiennych dualnych w punkcie spełniającym warunki Karush-Kuhna-Tuckera (patrz np. [5]).

Wyznacz wektor $d^{(k)**}$ zgodnie ze wzorem

$$d^{(k)**} = -P^{(k)} \nabla f(x^{(k)}) \tag{8}$$

i przejdź do kroku 7.

7. Wyznacz kierunek poprawy rozwiązania zgodnie ze wzorem:

$$d^{(k)} = \begin{cases} d^{(k)*}, & \| d^{(k)*} \| > c|u_{i*}|, \\ d^{(k)**}, & \| d^{(k)*} \| \leqslant c|u_{i*}| \end{cases} \tag{9}$$

i przejdź do kroku 8.

8. Wyznacz maksymalną długość kroku ze wzoru:

$$\lambda_{\max}^{(k)} = \begin{cases} \min \left\{ \dfrac{b_i - A_i x^{(k)}}{A_i d^{(k)}} : A_i d^{(k)} > 0 \right\}, & \exists i : A_i d^{(k)} > 0, \\ \infty & A d^{(k)} \leqslant \mathbf{0}, \end{cases} \tag{10}$$

gdzie A_i oznacza wiersz i macierzy warunków ograniczających w postaci nierówności, które pozostają niewiążące w aktualnym rozwiązaniu.

9. Wyznacz optymalną długość kroku jako rozwiązanie następującego zadania przeszukiwania liniowego:

$$\begin{aligned} &\min f(x^{(k)} + \lambda_{\text{opt}}^{(k)} d^{(k)}) \\ &\text{przy} \\ &\lambda_{\text{opt}}^{(k)} \in \langle 0, \lambda_{\max}^{(k)} \rangle \end{aligned} \tag{11}$$

10. Wyznacz kolejne rozwiązanie zgodnie ze wzorem $x_{ij}^{(k+1)} = x_{ij}^{(k)} + \lambda_{\text{opt}}^{(k)} d_{ij}^{(k)}$, podstaw $k := k+1$ i wróć do kroku 2.

W praktyce warunek z kroku 7 sprawdza się bezpośrednio po wyznaczeniu pierwszego kierunku poprawy, dzięki czemu można w niektórych iteracjach uniknąć wykonywania kroku 6.

2. Nieliniowe zagadnienia alokacji

Rozpatrzone będą cztery typy nieliniowych zadań alokacji: zagadnienie transportowo-produkcyjne bez ograniczeń mocy produkcyjnych (bez warunków popytowych), oznaczane dalej jako TP1, zagadnienie transportowo-produkcyjne

z ograniczeniem mocy produkcyjnych, oznaczane dalej jako TP2, zagadnienie pośrednika bez ograniczeń popytowych, oznaczane dalej jako NP1 i zagadnienie pośrednika z ograniczeniami odnośnie do popytu, oznaczane dalej jako NP2.

W każdym z omawianych zadań występuje m dostawców i n odbiorców (w przypadku zadań TP1 i TP2 odbiorcy są utożsamiani z zakładami produkcyjnymi). Przez c_{ij} oznaczono jednostkowy koszt transportu od dostawcy i do odbiorcy j, przez x_{ij} wielkość przewozu od dostawcy i do zakładu j, przez a_i podaż dostawcy i, a przez b_j – popyt odbiorcy j.

Ponadto dane są wypukłe funkcje kosztów produkcji f_j (w przypadku zadań TP1 i TP2) albo wklęsłe funkcje przychodu g_j i wypukłe funkcje kosztów zakupu f_i (w przypadku zadań NP1 i NP2). W każdym wypadku zakładamy, że funkcje te są klasy C^1.

Biorąc pod uwagę powyższe, model matematyczny zagadnienia typu TP1 będzie miał postać (por. [6], [9], [11]):

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{j=1}^{n} f_j(x_j)$$

przy

$$\begin{aligned} &\sum_{j=1}^{n} x_{ij} = a_i, \quad i = 1, \ldots, m \\ &\sum_{i=1}^{m} x_{ij} = x_j, \quad j = 1, \ldots, n \\ &x_{ij} > 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \end{aligned} \tag{12}$$

zagadnienia typu TP2 (por. [1], [9], [11]):

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{j=1}^{n} f_j(x_j)$$

przy

$$\begin{aligned} &\sum_{j=1}^{n} x_{ij} = a_i, \quad i = 1, \ldots, m \\ &\sum_{i=1}^{m} x_{ij} = x_j, \quad j = 1, \ldots, n \\ &x_j \leqslant b_j, \quad j = 1, \ldots, n \\ &x_{ij} \geqslant 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \end{aligned} \tag{13}$$

zagadnienia typu NP1 (por. [9], [11]):

$$\max \sum_{j=1}^{n} g_j(y_j) - \sum_{i=1}^{m} f_i(x_i) - \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

przy

$$\sum_{j=1}^{n} x_{ij} = x_i, \quad i = 1, \ldots, m \tag{14}$$

$$x_i \leqslant a_i, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} = y_j, \quad j = 1, \ldots, n$$

$$x_{ij} \geqslant 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

zagadnienia typu NP2 (por. [2], [9], [10], [11]):

$$\max \sum_{j=1}^{n} g_j(y_j) - \sum_{i=1}^{m} f_i(x_i) - \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

przy

$$\sum_{j=1}^{n} x_{ij} = x_i, \quad i = 1, \ldots, m \tag{15}$$

$$x_i \leqslant a_i, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} = y_j, \quad j = 1, \ldots, n$$

$$y_j \leqslant b_j, \quad j = 1, \ldots, n$$

$$x_{ij} \geqslant 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

3. Postacie algorytmu dla nieliniowych zadań alokacji

W przypadku zadania typu TP1 macierze A_1 i A_2 można zapisać w postaci (zakładając, że porządkujemy zmienne najpierw według pierwszego, a potem według drugiego indeksu):

$$A_1 = \begin{bmatrix} 1\ 1 \cdots 1\ 0\ 0 \cdots 0 \cdots 0\ 0 \cdots 0 \\ 0\ 0 \cdots 0\ 1\ 1 \cdots 1 \cdots 0\ 0 \cdots 0 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ 0\ 0 \cdots 0\ 0\ 0 \cdots 0 \cdots 1\ 1 \cdots 1 \end{bmatrix} \tag{16}$$

(w każdym wierszu znajduje się dokładanie n jedynek)

$$A_2 = -I_m \tag{17}$$

(I_k oznacza macierz jednostkową rzędu k).

Niech ustalony będzie pierwszy indeks zmiennej i. Przez n_i oznaczono liczbę zmiennych o pierwszym indeksie i równych 0. Macierz M powstanie wówczas z macierzy A_1 oraz tych wierszy macierzy A_2, które odpowiadają zmiennym równym 0. Warto zauważyć, że usunięcie któregokolwiek z wierszy macierzy M nie ma wpływu na jej ogólną postać (a więc również na postacie kolejnych wyznaczanych macierzy), a jedynie na wartości liczb n_i dla poszczególnych i.

Macierz MM^T przyjmie w takiej sytuacji następującą postać blokową:

$$MM^T = \begin{bmatrix} nI_m & B \\ B^T & I_N \end{bmatrix}, \tag{18}$$

gdzie I_k oznacza macierz jednostkową rzędu k,

$$N = \sum_{i=1}^{m} n_i, \tag{19}$$

a B jest macierzą o m wierszach i N kolumnach taką, że w wierszu i występuje dokładnie n_i elementów „-1" postaci:

$$B = \begin{bmatrix} -1\ -1 \cdots -1\ 0\ 0 \cdots 0 \cdots 0\ 0 \cdots 0 \\ 0\ 0 \cdots 0\ -1\ -1 \cdots -1 \cdots 0\ 0 \cdots 0 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ 0\ 0 \cdots 0\ 0\ 0 \cdots 0 \cdots -1\ -1 \cdots -1 \end{bmatrix}. \tag{20}$$

Macierz odwrotna ma postać blokową:

$$(MM^{\mathrm{T}})^{-1} = \begin{bmatrix} B_0 & C_1 & C_2 & .... & C_m \\ C_1^{\mathrm{T}} & B_1 & \mathbf{0} & ... & \mathbf{0} \\ C_2^{\mathrm{T}} & \mathbf{0} & B_2 & ... & \mathbf{0} \\ . & . & . & . & . \\ C_m^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & ... & B_m \end{bmatrix}, \tag{21}$$

gdzie macierz B_0 jest macierzą diagonalną postaci:

$$B_0 = \begin{bmatrix} \frac{1}{n-n_1} & 0 & ... & 0 \\ 0 & \frac{1}{n-n_2} & ... & 0 \\ . & . & . & . \\ 0 & 0 & ... & \frac{1}{n-n_m} \end{bmatrix}, \tag{22}$$

0 oznacza macierz zerową odpowiednich rozmiarów, B_i jest macierzą kwadratową o n_i wierszach i kolumnach postaci:

$$B_i = \begin{bmatrix} \frac{n-n_i+1}{n-n_i} & \frac{1}{n-n_i} & ... & \frac{1}{n-n_i} \\ \frac{1}{n-n_i} & \frac{n-n_i+1}{n-n_i} & ... & \frac{1}{n-n_i} \\ . & . & . & . \\ \frac{1}{n-n_i} & \frac{1}{n-n_i} & ... & \frac{n-n_i+1}{n-n_i} \end{bmatrix}, \tag{23}$$

a C_i oznacza macierz o m wierszach i n_i kolumnach, której wiersz i składa się wyłącznie z elementów $\frac{1}{n-n_i}$, pozostałe elementy są zerami.

Biorąc pod uwagę powyższe, postać blokową macierzy $(MM^{\mathrm{T}})^{-1}M$ można zapisać następująco:

$$(MM^{\mathrm{T}})^{-1}M=\begin{bmatrix} D_1 & D_2 & \dots & D_m \\ F_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & F_2 & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \dots & F_m \end{bmatrix}, \tag{24}$$

D_i jest macierzą o m wierszach i n kolumnach taką, że wszystkie jej wiersze poza wierszem i są wypełnione zerami, a w wierszu i na wszystkich miejscach odpowiadających usuniętym wierszom macierzy A_2 stoją elementy $\frac{1}{n-n_i}$, na pozostałych zaś 0; F_i jest macierzą o n_i wierszach i n kolumnach, w której kolumny odpowiadające usuniętym wierszom macierzy A_2 wypełnione są elementami $\frac{1}{n-n_i}$, a pozostałe kolumny (przy zachowaniu ich kolejności) tworzą macierz $-I_{n_i}$; $\mathbf{0}$ jest macierzą zerową odpowiednich rozmiarów.

Ostatecznie macierz rzutowania ma postać blokowo-diagonalną:

$$P=\begin{bmatrix} G_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & G_2 & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \dots & G_m \end{bmatrix}, \tag{25}$$

przy czym $\mathbf{0}$ oznacza macierz zerową odpowiednich rozmiarów, a G_i jest macierzą kwadratową, w której wszystkie kolumny i wszystkie wiersze odpowiadające nie usuniętym wierszom macierzy A_2 są wypełnione zerami, a po ich usunięciu macierz przyjęłaby postać:

$$G_i=\begin{bmatrix} \frac{n-n_i+1}{n-n_i} & \frac{-1}{n-n_i} & \dots & \frac{-1}{n-n_i} \\ \frac{-1}{n-n_i} & \frac{n-n_i+1}{n-n_i} & \dots & \frac{-1}{n-n_i} \\ \dots & \dots & \dots & \dots \\ \frac{-1}{n-n_i} & \frac{-1}{n-n_i} & \dots & \frac{n-n_i+1}{n-n_i} \end{bmatrix}, \tag{26}$$

Biorąc pod uwagę powyższe ustalenia oraz mając na uwadze fakt, że macierz M nigdy nie będzie macierzą pustą[2] algorytm rzutowania gradientu dla zadań typu TP1 można zapisać w postaci:

1. Obierz stałą $c>0$. Podstaw $k:=1$ i wyznacz pierwsze rozwiązanie dopuszczalne $x^{(1)}$.
2. Niech $M^{(k)}$ będzie macierzą, której wiersze tworzą warunki podażowe i te warunki nieujemności, które odpowiadają zmiennym równym w aktualnym rozwiązaniu 0.
3. Wyznacz wartości n_i, $i=1,\ldots,m$ i przejdź do kroku 4.
4. Wyznacz współrzędne wektora $d^{(k)*}$ zgodnie ze wzorem:

$$d_{ij}^{(k)*}=\begin{cases}0, & i=1,\ldots,m,\quad j\in J_i,\\ \dfrac{1}{n-n_i}\displaystyle\sum_{\substack{h\notin J_i\\ h\neq j}}\frac{\partial f}{\partial x_{ih}}(x^{(k)})-\frac{n-n_i-1}{n-n_i}\cdot\frac{\partial f}{\partial x_{ij}}(x^{(k)}), & i=1,\ldots,m,\quad j\notin J_i,\end{cases}\tag{27}$$

   gdzie J_i oznacza zbiór drugich indeksów tych zmiennych o pierwszym indeksie i, których warunki nieujemności nie zostały usunięte z macierzy $M^{(k)}$.
5. Wyznacz przybliżone wartości zmiennych dualnych, odpowiadających warunkom nieujemności zmiennych uwzględnionym w macierzy $M^{(k)}$ zgodnie ze wzorem:

$$u_{ij}^{(k)}=-\frac{1}{n-n_i}\sum_{h\notin J_i}\frac{\partial f}{\partial x_{ih}}(x^{(k)})+\frac{\partial f}{\partial x_{ij}}(x^{(k)}),\qquad i=1,\ldots m,\quad j\in J_i.\tag{28}$$

   Dalsze postępowanie zależy od tego, który z trzech wykluczających się warunków jest spełniony:
   - Jeżeli $u^{(k)}\geqslant\mathbf{0}$ i $d^{(k)*}=\mathbf{0}$, to STOP. Otrzymane rozwiązanie $x^{(k)}$ należy do zbioru rozwiązań zadowalających Ω.
   - Jeżeli $u^{(k)}\geqslant\mathbf{0}$ i $d^{(k)*}\neq\mathbf{0}$, wyznacz wektor poprawy rozwiązania zgodnie ze wzorem $d^{(k)}=d^{(k)*}$ i przejdź do kroku 8.
   - Jeżeli istnieje para indeksów i, j, dla których $u_{ij}^{(k)}<0$, przejdź do kroku 6.
6. Wybierz parę indeksów i^*, j^* taką, że

$$u_{i^*j^*}^{(k)}=\min\{u_{ij}^{(k)}: u_{ij}^{(k)}<0\}.\tag{29}$$

[2] Wynika to z faktu, że na pewno będą do niej należeć wiersze macierzy ograniczeń podażowych A_1.

Usuń z macierzy $M^{(k)}$ wiersz związany ze składową $u_{i^*j^*}$ i wyznacz wektor $d^{(k)**}$ w taki sam sposób, jak wektor $d^{(k)*}$ w kroku 4. Przejdź do kroku 7.

7. Wyznacz kierunek poprawy rozwiązania zgodnie ze wzorem:

$$d^{(k)} = \begin{cases} d^{(k)*}, & \|d^{(k)*}\| > c|u_{i^*}|, \\ d^{(k)**}, & \|d^{(k)*}\| \leqslant c|u_{i^*}| \end{cases} \tag{30}$$

i przejdź do kroku 8.

8. Wyznacz maksymalną długość kroku ze wzoru:

$$\lambda_{\max}^{(k)} = \min_{i=1,\ldots,m,\ j=1,\ldots,n_i} \left\{ \frac{x_{ij}^{(k)}}{d_{ij}^{(k)}},\ d_{ij}^{(k)} < 0 \right\}. \tag{31}$$

9. Wyznacz optymalną długość kroku jako rozwiązanie następującego zadania przeszukiwania liniowego:

$$\begin{aligned} &\min f(x^{(k)} + \lambda_{\text{opt}}^{(k)} d^{(k)}) \\ &\text{przy} \\ &\lambda_{\text{opt}}^{(k)} \in \langle 0, \lambda_{\max}^{(k)} \rangle \end{aligned} \tag{32}$$

10. Wyznacz kolejne rozwiązanie zgodnie ze wzorem $x_{ij}^{(k+1)} = x_{ij}^{(k)} + \lambda_{\text{opt}}^{(k)} d_{ij}^{(k)}$, podstaw $k := k+1$ i wróć do kroku 2.

W przypadku zadań typu NP1, jeżeli wszystkie warunki podażowe są wiążące, postacie poszczególnych macierzy są identyczne z zadaniami typu TP1. W sytuacji gdy któryś z warunków jest niewiążący, należy nieznacznie zmodyfikować poszczególne macierze. W macierzy $(MM^{\mathrm{T}})^{-1}$ należy usunąć wszystkie wiersze odpowiadające niewiążącym warunkom (ta modyfikacja dotyczy podmacierzy D_i), natomiast w odpowiednich macierzach F_i elementy $\frac{1}{n-n_i}$ należy zastąpić zerami. W macierzy rzutowania natomiast po usunięciu z odpowiednich podmacierzy G_i zerowych kolumn i wierszy powinny powstać macierze jednostkowe rzędu $n-n_i$.

Biorąc pod uwagę powyższe, algorytm rzutowania gradientu dla zadań typu NP1W można zapisać w postaci:

1. Obierz stałą $c>0$. Podstaw $k := 1$ i wyznacz pierwsze rozwiązanie dopuszczalne $x^{(1)}$.

2. Niech $M^{(k)}$ będzie macierzą, której wiersze tworzą warunki podażowe spełnione w danej iteracji z równością i te warunki nieujemności, które odpowiadają zmiennym równym w aktualnym rozwiązaniu 0.
3. Jeżeli macierz $M^{(k)}$ jest macierzą pustą i $\nabla f(x^{(k)}) = \mathbf{0}$, to STOP. Uzyskane rozwiązanie należy do zbioru rozwiązań zadowalających Ω. Jeżeli macierz $M^{(k)}$ jest macierzą pustą, ale $\nabla f(x^{(k)}) \neq \mathbf{0}$, wyznacz kierunek poprawy rozwiązania zgodnie ze wzorem

$$d^{(k)} = -\nabla f(x^{(k)}) \tag{33}$$

i przejdź do kroku 8. Jeżeli macierz $M^{(k)}$ nie jest macierzą pustą, wyznacz wartości n_i, $i = 1, \ldots, m$ i przejdź do kroku 4.
4. Wyznacz współrzędne wektora $d^{(k)*}$ zgodnie ze wzorem:

$$d_{ij}^{(k)*} = \begin{cases} 0, & i = 1, \ldots, m, \quad j \in J_i, \\ -\dfrac{1}{n - n_i} \displaystyle\sum_{\substack{h \notin J_i \\ h \neq j}} \frac{\partial f}{\partial x_{ih}}(x^{(k)}) + \frac{n - n_i - 1}{n - n_i} \cdot \frac{\partial f}{\partial x_{ij}}(x^{(k)}), & i \in I^*, \ j \notin J_i, \\ \dfrac{\partial f}{\partial x_{ij}}(x^{(k)}), & i \notin I^*, \quad j \notin J_i, \end{cases} \tag{34}$$

gdzie J_i oznacza zbiór drugich indeksów tych zmiennych o pierwszym indeksie i, których warunki nieujemności nie zostały usunięte z macierzy $M^{(k)}$, a I^* zbiór indeksów warunków podażowych wiążących w danym rozwiązaniu.
5. Wyznacz przybliżone wartości zmiennych dualnych odpowiadających warunkom nieujemności zmiennych uwzględnionym w macierzy $M^{(k)}$ zgodnie ze wzorem:

$$u_{ij}^{(k)} = \begin{cases} -\dfrac{1}{n - n_i} \displaystyle\sum_{h \notin J_i} \frac{\partial f}{\partial x_{ih}}(x^{(k)}) + \frac{\partial f}{\partial x_{ij}}(x^{(k)}), & i \in I^*, \quad j \in J_i, \\ \dfrac{\partial f}{\partial x_{ij}}(x^{(k)}), & i \notin I^*, \quad j \in J_i \end{cases} \tag{35}$$

oraz przybliżone wartości zmiennych dualnych odpowiadających wiążącym warunkom podażowym zgodnie ze wzorem:

$$u_i^{(k)} = -\frac{1}{n - n_i} \sum_{j \notin J_i} \frac{\partial f}{\partial x_{ij}}(x^{(k)}), \quad i \in I^*. \tag{36}$$

Dalsze postępowanie zależy od tego, który z trzech wykluczających się warunków jest spełniony:

- Jeżeli $u^{(k)} \leqslant \mathbf{0}$ i $d^{(k)*} = \mathbf{0}$, to STOP. Otrzymane rozwiązanie $x^{(k)}$ należy do zbioru rozwiązań zadowalających Ω.
- Jeżeli $u^{(k))} \leqslant \mathbf{0}$ i $d^{(k)*} \neq \mathbf{0}$, wyznacz wektor poprawy rozwiązania zgodnie ze wzorem $d^{(k)} = d^{(k)*}$ i przejdź do kroku 8.
- Jeżeli istnieje para indeksów i, j, dla których $u_{ij}^{(k)} > 0$, albo indeks i, dla którego $u_i^{(k)} > 0$, przejdź do kroku 6.

6. Wybierz parę indeksów i^*, j^* taką, że $u_{i^*j^*}^{(k)} = \max\{u_{ij}^{(k)} : u_{ij}^{(k)} > 0\}$. Wybierz indeks h^* taki, że $u_{h^*}^{(k)} = \max\{u_i^{(k)} : u_i^{(k)} > 0\}$. Jeżeli $u_{i^*j^*}^{(k)} > u_{h^*}^{(k)}$, usuń z macierzy $M^{(k)}$ wiersz związany ze składową $u_{i^*j^*}$ i wyznacz wektor $d^{(k)**}$ w taki sam sposób, jak wektor $d^{(k)*}$ w kroku 4. W przeciwnym razie usuń z macierzy $M^{(k)}$ wiersz związany ze składową u_{h^*} i wyznacz wektor $d^{(k)**}$ w taki sam sposób, jak wektor $d^{(k)*}$ w kroku 4. Przejdź do kroku 7.
7. Wyznacz kierunek poprawy rozwiązania zgodnie ze wzorem:

$$d^{(k)} = \begin{cases} d^{(k)*}, & \|d^{(k)*}\| > c|u_{i*}|, \\ d^{(k)**}, & \|d^{(k)*}\| \leqslant c|u_{i*}| \end{cases} \tag{37}$$

i przejdź do kroku 8.

8. Wyznacz maksymalną długość kroku ze wzoru:

$$\lambda_{\max}^{(k)} = \min\{\lambda_{\max}^{(k,1)}, \lambda_{\max}^{(k,2)}\}, \tag{38}$$

gdzie

$$\lambda_{\max}^{(k,1)} = \min_{i=1,\ldots,m,\ j=1,\ldots,n_i} \left\{ \frac{x_{ij}^{(k)}}{d_{ij}^{(k)}},\ d_{ij}^{(k)} < 0 \right\} \tag{39}$$

oraz

$$\lambda_{\max}^{(k,2)} = \min_{i=1,\ldots,m} \left\{ \frac{a_i - \sum_{j=1}^{n} x_{ij}^{(k)}}{\sum_{j=1}^{n} d_{ij}^{(k)}},\ \sum_{j=1}^{n} d_{ij}^{(k)} < 0 \right\} \tag{40}$$

9. Wyznacz optymalną długość kroku jako rozwiązanie następującego zadania przeszukiwania liniowego:

$$\begin{aligned} &\max f(x^{(k)}+\lambda_{\text{opt}}^{(k)} d^{(k)}) \\ &\text{przy} \\ &\lambda_{\text{opt}}^{(k)} \in \langle 0, \lambda_{\max}^{(k)} \rangle \end{aligned} \tag{41}$$

10. Wyznacz kolejne rozwiązanie zgodnie ze wzorem $x_{ij}^{(k+1)}=x_{ij}^{(k)}+\lambda_{\text{opt}}^{(k)} d_{ij}^{(k)}$, podstaw $k:=k+1$ i wróć do kroku 2.

W zadaniach typu TP2 i NP2, ze względu na występowanie warunków popytowych, macierze rzutowania mają trudną do przewidzenia postać. W związku z tym do ich rozwiązywania trzeba używać ogólnej postaci algorytmu. Ze względu jednak na fakt, że odpowiednie macierze są symetryczne, do operacji odwracania, będącej najbardziej czasochłonnym elementem algorytmu, użyto algorytmu Frobeniusa-Schura dla macierzy symetrycznych.

Nieco uproszczone zostały także operacje mnożenia odpowiednich macierzy, biorąc pod uwagę to, że w obu przypadkach macierz A jest macierzą rzadką, a jej jedyne niezerowe elementy mają wartość 1 albo -1.

4. Wyniki eksperymentów

Działanie przedstawionego algorytmu zostało porównane z działaniem metody wyrównań uogólnionej na zadania z dowolnymi wypukłymi funkcjami celu[3] (patrz np. [3], [9], [11]) oraz metody kierunków dopuszczalnych dla nieliniowych zadań alokacji przedstawionej w pracy [4]. Testy przeprowadzono dla zadań z kwadratowymi, wypukłymi funkcjami kosztów i kwadratowymi, wklęsłymi funkcjami przychodów (bez stałej, jedynie z częścią liniową i kwadratową). Przetestowano po 100 zadań każdego typu, z czego po 50 z 10 dostawcami i 10 odbiorcami oraz po 50 z 5 dostawcami i 20 odbiorcami. Parametry każdego z 400 zadań zostały wygenerowane niezależnie jako liczby pseudolosowe o rozkładzie jednostajnym odpowiednio z przedziałów:

– podaż dostawców: przedział (0, 100);

[3] Tylko w przypadku zadań postaci TP1 i NP1, gdyż tylko dla tych typów zadań zdefiniowano metodę wyrównań.

- popyt odbiorców: w przypadku zadań z 10 dostawcami i 10 odbiorcami – przedział (0, 150); w przypadku pozostałych zadań – przedział (0, 50);
- jednostkowe koszty transportu: przedział (0, 10);
- współczynniki funkcji kosztów dla zadań typu TP1 i TP2 – część liniowa: przedział (0, 10); część kwadratowa: przedział (0, 0,1); dla zadań typu NP1 i NP2 – część liniowa: przedział (0,10); część kwadratowa: przedział (0, 0,05);
- współczynniki funkcji przychodu dla zadań typu NP1 i NP2 – część liniowa: przedział (10, 30); część kwadratowa: przedział (−0,1, 0).

Algorytm zatrzymywał się, gdy pierwiastek kwadratowy iloczynu skalarnego kierunku poprawy i gradientu funkcji celu osiągał wartość mniejszą od $\varepsilon = 0{,}01$.

Zmierzone zostały czasy (w sekundach), po jakich wybrane metody znajdowały rozwiązanie zadowalające, zdefiniowane jak wyżej. Wszystkie metody zostały zaimplementowane w języku VBA.

Porównaniu poddane zostały: średni czas działania algorytmu w każdym przypadku, będący miernikiem jego efektywności, oraz miary zmienności tego czasu na badanej populacji zadań: rozstęp (czyli różnicę między maksymalnym i minimalnym czasem rozwiązywania) i odchylenie standardowe z próby, przy czym obie te wielkości zostały podzielone przez średnie wartości czasu rozwiązywania (aby uzyskać łatwiejsze do porównania współczynniki zmienności). Te ostatnie wskaźniki pozwalają ocenić, w jakim stopniu zmiany parametrów wpływają na czas pracy danego algorytmu, czyli innymi słowy, jaka jest jego wrażliwość na zmiany parametrów.

Wartości najkrótszych (t_{min}), najdłuższych (t_{max}) i średnich (t_{av}) czasów pracy poszczególnych metod dla wszystkich typów zadań oraz obu współczynników zmienności (odpowiednio v_s dla odchylenia standardowego i v_d dla rozstępu) zostały zamieszczone w poniższej tabeli.

Jak widać, w przypadku zadań postaci TP1 i NP1 metoda rzutowania gradientu cechuje się krótszymi średnimi czasami poszukiwania optimum niż metoda kierunków dopuszczalnych, a w porównaniu z metodą wyrównań wypada nieco gorzej (w dwóch przypadkach średnie czasy są nieco lepsze, w dwóch pozostałych gorsze niż dla metody wyrównań).

Należy jednak zwrócić uwagę na fakt, że we wszystkich czterech przypadkach algorytm rzutowania gradientu cechuje się mniejszą wrażliwością na zmiany parametrów niż metoda wyrównań (choć w dwóch przypadkach jest pod tym względem gorszy od metody kierunków dopuszczalnych).

Tabela 1. Wyniki eksperymentów obliczeniowych

Zadanie	Metoda wyrównań				
	t_{av}	t_{max}	t_{min}	v_s	v_d
TP1 10 × 10	0,512	3,165	0,090	1,739	6,007
TP1 5 × 20	0,354	0,821	0,100	0,632	2,038
NP1 10 × 10	0,420	1,402	0,050	0,838	3,221
NP1 5 × 20	0,397	1,022	0,020	0,772	2,527
TP2 10 × 10	–	–	–	–	–
TP2 5 × 20	–	–	–	–	–
NP2 10 × 10	–	–	–	–	–
NP2 5 × 20	–	–	–	–	–
	Metoda kierunków dopuszczalnych				
TP1 10 × 10	0,770	2,543	0,080	1,015	3,198
TP1 5 × 20	1,201	2,293	0,500	0,481	1,493
NP1 10 × 10	0,937	1,822	0,221	0,633	1,708
NP1 5 × 20	0,447	0,872	0,241	0,404	1,412
TP2 10 × 10	0,727	3,966	0,090	1,503	5,330
TP2 5 × 20	0,669	2,133	0,070	0,901	3,084
NP2 10 × 10	1,370	5,258	0,240	0,999	3,663
NP2 5 × 20	0,656	1,873	0,070	0,907	2,747
	Metoda rzutowania gradientu				
TP1 10 × 10	0,419	0,841	0,170	0,524	1,600
TP1 5 × 20	0,789	1,632	0,241	0,529	1,421
NP1 10 × 10	0,736	1,162	0,030	0,485	1,538
NP1 5 × 20	0,379	1,001	0,090	0,654	2,402
TP2 10 × 10	24,750	47,468	2,293	0,519	1,825
TP2 5 × 20	40,649	98,581	6,770	0,661	2,259
NP2 10 × 10	295,655	365,655	216,471	0,179	0,505
NP2 5 × 20	255,958	437,238	147,072	0,332	1,134

Podsumowując powyższe, metoda rzutowania gradientu dla nieliniowych zadań alokacji może konkurować z innymi znanymi metodami, o ile ograniczamy się do zadań bez warunków popytowych.

Dużo wolniej metoda rzutowania gradientu działa w przypadku zadań TP2 i NP2. W tej sytuacji czas działania jest kilkudziesięciokrotnie lub kilkusetkrotnie większy, niż w przypadku metody kierunków dopuszczalnych. Oznacza to wprawdzie dyskwalifikację metody, jednakże z kilkoma zastrzeżeniami.

Po pierwsze, we wszystkich czterech przypadkach czasy wykonania cechują się mniejszą wrażliwością na zmiany parametrów. Po drugie, algorytmy

testowano tylko dla jednego typu funkcji kosztów i przychodów, a nie można mieć całkowitej pewności, czy zmiana typu funkcji nie powoduje, przynajmniej niewielkiej, zmiany wyników na korzyść metody rzutowania gradientu. Po trzecie, dotychczasowe nieudane próby wyznaczenia uproszczonej postaci macierzy rzutowania dla zadań z ograniczeniami popytowymi nie oznaczają, że jest to niemożliwe. Gdyby takie uproszczenie doszło do skutku, mogłoby znacznie skrócić czas pracy algorytmu.

Uwagi końcowe

Jak widać, prezentowana metoda pozwala efektywnie rozwiązywać nieliniowe zadania alokacji bez warunków popytowych, a więc zadania typu TP1 i typu NP1, nie jest zaś raczej przydatna w przypadku zadań z warunkami popytowymi, czyli typu TP2 i NP2 (choć nawet w tym ostatnim przypadku cechuje się niewielką wrażliwością na zmiany parametrów zadania). Porównanie wyników dla zadań poszczególnych typów pozwala stwierdzić, że uproszczenie procedur wyznaczania macierzy rzutowania, kierunku poprawy rozwiązania i wektora zmiennych dualnych znacznie skraca czas pracy algorytmu.

Kontynuacja badań nad przedstawioną metodą powinna, zdaniem autora, objąć przede wszystkim dwa kierunki.

Po pierwsze – w miarę możliwości należy porównać działanie metody rzutowania gradientu z innymi metodami, nie tylko wymienionymi w tekście. Można tu jednak napotkać trudność w postaci zakresu działań tych metod – opisane w literaturze algorytmy są przeważnie dostosowane do rozwiązywania zadań jednego, maksymalnie dwóch, spośród czterech omawianych typów. Należy również przeprowadzić testy dla różnych typów funkcji (wielomianów wyższych rzędów, funkcji wykładniczej, funkcji logarytmicznej itp.).

Po drugie, należy próbować wyznaczyć uproszczone metody generowania kierunku poprawy i wektora zmiennych dualnych dla zadań postaci TP2 i NP2. Mimo dotychczasowych niepowodzeń, zdaniem autora, uproszczenia takie są możliwe, gdyż macierz ograniczeń aktywnych także w przypadku tych typów zadań ma specyficzną postać – jest bardzo rzadką macierzą, której wszystkie niezerowe elementy mają wartość 1 albo -1.

Literatura

[1] Anholcer, M., *Modyfikacja metody quasi-baz dla zagadnienia transportowo-produkcyjnego*, w: D. Kopańska-Bródka (red.), *Metody i zastosowania badań operacyjnych 2000*, Akademia Ekonomiczna w Katowicach, Katowice 2001.

[2] Anholcer, M., *Modyfikacja metody quasi-baz dla nieliniowego zagadnienia pośrednika z kwadratową, wklęsłą funkcją celu*, w: E. Ignasiak (red), *Optymalizacja decyzji, symulacja i prognozowanie procesów gospodarczych*, Wydawnictwo Akademii Ekonomicznej w Poznaniu, Poznań 2002.
[3] Anholcer, M., *Zbieżność metody wyrównań dla nieliniowych zadań alokacji*, w: K. Piasecki, W. Sikora (red.), *Z prac Katedry Badań Operacyjnych*, Wydawnictwo Akademii Ekonomicznej w Poznaniu, Poznań 2005.
[4] Anholcer, M., *Metoda kierunków dopuszczalnych dla nieliniowego zagadnienia transportowo-produkcyjnego*, w: *Metody i Zastosowania Badań Operacyjnych 2003*, Prace Naukowe Politechniki Łódzkiej (w druku).
[5] Bazaraa, M.S., Sherali, H.D., Shetty, C.M., *Nonlinear Programing. Theory and Algorithms*, John Wiley & Sons Inc., New York – Chichester – Brisbane – Toronto – Singapore 1993.
[6] Borucki, W., Ignasiak, E., Marcinkowski, J., Sikora, W., *Badania operacyjne*, PWE, Warszawa 1996.
[7] Du, D.-Z., Zhang, X.-S., *A Convergence Theorem of Rosen's Gradient Projection Method*, Mathematical Programming 36, 1986.
[8] Du, D.-Z., Zhang, X.-S., *Global Convergence of Rosen's GRaient Projection Method*, Mathematical Programming 44, 1989.
[9] Sikora, W., *Modele i metody optymalnej dystrybucji dóbr*, Wydawnictwo Akademii Ekonomicznej w Poznaniu, Poznań 1993.
[10] Sikora, W., *Sterowanie oprocentowaniem wkładów i kredytów bankowych*, w: E. Ignasiak (red.), *Badania operacyjne w zarządzaniu firmami i przedsięwzięciami*, Wydawnictwo Akademii Ekonomicznej w Poznaniu, Poznań 1998.
[11] Sikora, W., Runka, H., Pyrzyński, D., *Optymalizacja przepływów w sferze dystrybucji dóbr jednorodnych*, Projekt badawczy H\12\209\90-2. Akademia Ekonomiczna w Poznaniu, Poznań 1991.

GRADIENT PROJECTION METHOD FOR NONLINEAR ALLOCATION PROBLEMS

Summary

In the paper author presents the adaptation of gradient projection method of Rosen to several classes of nonlinear transportation problems. The special character of constraints matrix is exploited to obtain much simpler form of the projection matrix and dual variables vector. Comparison between presented method and feasible directions and equalization algorithms shows that it may be successfully used to solve some of concerned problems.