Programowanie matematyczne - zadanie 3

Paulina Przybyłek

Grupa laboratoryjna C
14 listopada 2022

1 Treść zadania

Zadanie dotyczy poszukiwania przecięcia (tzw. kolizji) dwóch trójkątów w $\mathbb{R}^2$, których współrzędne wierzchołków są całkowite. Każda współrzędna wierzchołka $w = (w_1, w_2)$ trójkąta jest losowana z przedziału $[0, 10]$, czyli $w_i \in \{0, 1, 2, \ldots, 10\}$ dla $i = 1,\ 2$. Dla utworzonych w ten sposób dwóch trójkątów należy znaleźć rozwiązanie zadania programowania liniowego określające czy jest kolizja między trójkątami czy też są rozłączne. W przypadku kolizji należy podać ten punkt.

Do rozwiązania zadania należy wykorzystać własną implementację wybranego algorytmu oraz wbudowaną metodę z Matlaba - funkcję linprog wykorzystującą metodę sympleks.

1.1 Zadanie programowania liniowego

Przyjmijmy więc, że $P = [p_1, p_2, p_3]$ oraz $Q = [q_1, q_2, q_3]$ to macierze określające wylosowane trójkąty. Punkty p_i, q_i posiadają dwie współrzędne. Trójkąty te możemy opisać następująco:

$$x_P = x_1 * p_1 + x_2 * p_2 + x_3 * p_3$$

$$x_1 + x_2 + x_3 = 1$$

$$x_Q = x_4 * q_1 + x_5 * q_2 + x_6 * q_3$$

$$x_4 + x_5 + x_6 = 1$$

Oczywiście wszystkie $x_i \geqslant 0$ dla $i = 1, \ldots, 6$. Poszukujemy kolizji, więc chcemy, aby $x_P = x_Q$. Zadanie programowania liniowego można przedstawić jako minimalizację dowolnej funkcji (na przykład: $min \sum_{i=1}^{6} 0 * x_i$), gdyż chcemy znaleźć jedynie punkt dopuszczalny. Zdefiniujmy więc ZPL odpowiednio zgodnie z definicją.

$$\max_x -(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T x) \tag{1}$$

$$\Omega : \begin{cases} \begin{bmatrix} p_{11} & p_{21} & p_{31} & -q_{11} & -q_{21} & -q_{31} \\ p_{12} & p_{22} & p_{32} & -q_{12} & -q_{22} & -q_{32} \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} x = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \\ x \geqslant 0 \ (warunki\ brzegowe) \end{cases} \tag{2}$$

gdzie $x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T$ a $p_{ij}, q_{ij} \in \mathbb{Z}$ ($p_i = \begin{bmatrix} p_{i1} & p_{i2} \end{bmatrix}$). Tak zdefiniowane zadanie nazywane jest **postacią standardową** zadania ZPL. Zadanie to nie ma jednak **postaci kanonicznej**, gdyż w macierzy A nie ma podmacierzy odpowiadającej macierzy jednostkowej stopnia m, gdzie m odpowiada liczbie liniowych ograniczeń oraz rzędowi macierzy A. Nie możemy więc ustalić bazy początkowej dla algorytmu sympleks. W tym celu należy dołożyć zmienne techniczne, zwane **zmiennymi sztucznymi**. Sprowadzi to zadanie do postaci kanonicznej oraz pozwoli zastosować algorytm sympleks z metodą dwufazową. Zmodyfikowany ZPL dla Ω (2) wygląda następująco:

$$\Omega' : \begin{cases} \begin{bmatrix} p_{11} & p_{21} & p_{31} & -q_{11} & -q_{21} & -q_{31} & 1 & 0 & 0 & 0 \\ p_{12} & p_{22} & p_{32} & -q_{12} & -q_{22} & -q_{32} & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \\ x \geqslant 0,\ s \geqslant 0 \ (warunki\ brzegowe) \end{cases} \tag{3}$$

gdzie $s = \begin{bmatrix} x_7 & x_8 & x_9 & x_{10} \end{bmatrix}^T$. Mając postać standardową kanoniczną można przystąpić do rozwiązania zadania ZPL. W tym celu zastosowana zostanie metoda dwufazowa, która wykorzystuje zastosowanie owych sztucznych zmiennych.

1.2 Metoda dwufazowa sympleks

Metoda dwufazowa wykorzystuje zmodyfikowaną wersję ZPL w fazie 1, a oryginalną w fazie 2. Mając zdefiniowane ograniczenia Ω' (3) rozwiązywane jest następujące zadanie pomocnicze dla tych ograniczeń:

$$\min_{x,s} \sum_{i=1}^{4} s_i$$

Początkowe bazowe rozwiązanie dopuszczalne to $s = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}^T$ oraz $x = 0$. Zadanie to jest rozwiązywane w pierwszej fazie i polega na usunięciu zmiennych sztucznych z bazy przy pomocy algorytmu sympleks.

Jeśli uda się zakończyć algorytm z bazą bez sztucznych zmiennych to możliwe jest przejście do fazy drugiej, gdzie rozwiązywane jest przy pomocy algorytmu sympleks zadanie właściwe (1). Wówczas uzyskane wartości z fazy pierwszej stają się bazowym rozwiązaniem dopuszczalnym fazy drugiej. Mianowicie końcowa baza jest bazą początkową, to samo się tyczy końcowych wyników wartości A oraz b. Trzeba pamiętać, że w przypadku macierzy A, interesuje nas tylko pierwsze 6 kolumn (bez wprowadzonych sztucznych zmiennych).

1.2.1 Metoda sympleks

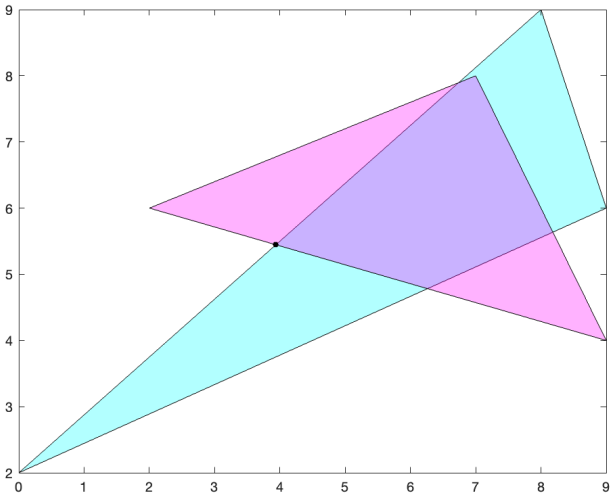
Mając bazowe rozwiązanie dopuszczalne (BRD) dla danej bazy opartej na macierzy jednostkowej będącej podmacierzą A, wyliczane są współczynniki optymalności odpowiadające aktualnemu BRD. Jeśli wszystkie współczynniki są większe bądź równe zero to algorytm się zatrzymuje, w przeciwnym przypadku poszukiwany jest nowy element x, który zastąpi stary w bazie. Dzieje się to na podstawie kryterium wejścia i wyjścia z bazy, a biorą w nim udział te elementy co mają ujemne wartości współczynników optymalności. Po wybraniu nowego elementu zastępuje on stary i wyliczane są nowe wartości macierzy A i wektora b n a podstawie odpowiednich wzorów. Wówczas mamy nowe BRD i wracamy do punktu sprawdzania współczynników. Dzieje się tak aż do zatrzymania algorytmu gdy znajdziemy rozwiązanie bądź okaże się, że zadanie jest sprzeczne.

2 Rozwiązanie Matlab

Rozwiązanie zadania zawiera się trzech plikach .m. Plik *symplex_2_phase_method.m* zawiera funkcje implementujące metodę dwufazową sympleks. Komentarze informują w krótki sposób co się tam dzieje. W pliku *script.m* jest przykładowe wywołanie dla jednej pary trójkątów wraz z możliwością narysowania wyniku. Natomiast plik *test.m* wykorzystany został do realizacji części testowej opisanej w kolejnej sekcji.

2.1 Przykład z kolizją

Wylosowano punkty dla obu trójkątów i zdefiniowano ZPL zgodnie z jego właściwą formą (wzory 1, 2), jednak z tą zmianą, że funkcja celu jest wspomnianym wcześniej minimum a nie maksimum, ponieważ takie wartości były potrzebne dla funkcji linprog. Wbudowany algorytm 'dual-simplex' w Matlabie znalazł kolizję, a na Rysunku 1 przedstawiono oba trójkąty i punkt, który znalazł ten algorytm.



Rysunek 1: Wylosowane trójkąty do ZPL. Punkt znaleziony przez algorytm dual-simplex w funkcji linprog oznaczono kolorem czarnym.

W przypadku własnej implementacji zostano przy maksymalizacji funkcji celu, a do algorytmu podano A, b, funkcję celu oraz ograniczenie na to, aby x były nieujemne. Już algorytm sam dokłada zmienne sztuczne i zmienia postać ZPL na zmodyfikowaną. Oryginalne dane były następujące:

$$A = \begin{bmatrix} 8 & 9 & 0 & -9 & -7 & -2 \\ 9 & 6 & 2 & -4 & -8 & -6 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}^T$$

Wektor b jest kolumnowy, macierz A w pierwszych dwóch wierszach ma wierzchołki trójkątów, co można porównać z rysunkiem, aby się przekonać. Algorytm przekształca to do postaci przekształconej, więc nowa macierz A' będzie wyglądać następująco:

$$A' = \begin{bmatrix} 8 & 9 & 0 & -9 & -7 & -2 & 1 & 0 & 0 & 0 \\ 9 & 6 & 2 & -4 & -8 & -6 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Natomiast mamy również nowy wektor x':

$$x' = \begin{bmatrix} x & -1 & -1 & -1 & -1 \end{bmatrix}^T$$

gdzie wartości x użyte do jego stworzenia są równe 0. Dla tak zdefiniowanych powstaje tabelka sympleks, gdzie zmienne sztuczne (4 ostatnie) są w bazie. Faza pierwsza zakończyła się po 5 iteracjach. Wyniki ostatnich tabel przedstawia Rysunek 2. Kolorem czerwonym zaznaczono kolejny punkt, który wybrano do bazy, czyli x_3 ma zastąpić x_9. W drugiej tabelce widzimy już w bazie brak zmiennych sztucznych oraz wektor $z - c$ jest nieujemny, stąd znaleziono rozwiązanie będące BRD dla fazy drugiej. Po wpisaniu danych do tabelki sympleks dla fazy drugiej (Rysunek 3) okazało się, że jest to rozwiązanie optymalne również tam. Czyli do RO należą x_1, x_3, x_4, x_6.

```
ITERACJA 4

             0     0     0     0     0     0     -1    -1    -1    -1
c_B   x_B    x1    x2    x3    x4    x5    x6    x7    x8    x9    x10   b
-------------------------------------------------------------------------------
0     x1     1     0.76  0.18  0     -0.30 0     0.03  0.09  0     0.58  0.58
0     x4     0     -0.42 0.20  1     0.37  0     -0.11 0.10  0     0.38  0.38
-1    x9     0     0.24  0.82  -0    0.30  0     -0.03 -0.09 1     -0.58 0.42
0     x6     0     0.42  -0.20 0     0.63  1     0.11  -0.10 0     0.62  0.62
-------------------------------------------------------------------------------
      z      0     -0.24 -0.82 0     -0.30 0     0.03  0.09  -1    0.58  -0.42
      z-c    0     -0.24 -0.82 0     -0.30 0     1.03  1.09  0     1.58


ITERACJA 5

             0     0     0     0     0     0     -1    -1    -1    -1
c_B   x_B    x1    x2    x3    x4    x5    x6    x7    x8    x9    x10   b
-------------------------------------------------------------------------------
0     x1     1     0.71  0     0     -0.37 0     0.03  0.11  -0.22 0.71  0.49
0     x4     0     -0.48 0     1     0.29  0     -0.11 0.12  -0.25 0.52  0.28
0     x3     0     0.29  1     -0    0.37  0     -0.03 -0.11 1.22  -0.71 0.51
0     x6     0     0.48  0     0     0.71  1     0.11  -0.12 0.25  0.48  0.72
-------------------------------------------------------------------------------
      z      0     0     0     0     0     0     0     0     0     0     0
      z-c    0     0     0     0     0     0     1     1     1     1     >=0


Znaleziono rozwiązanie
```

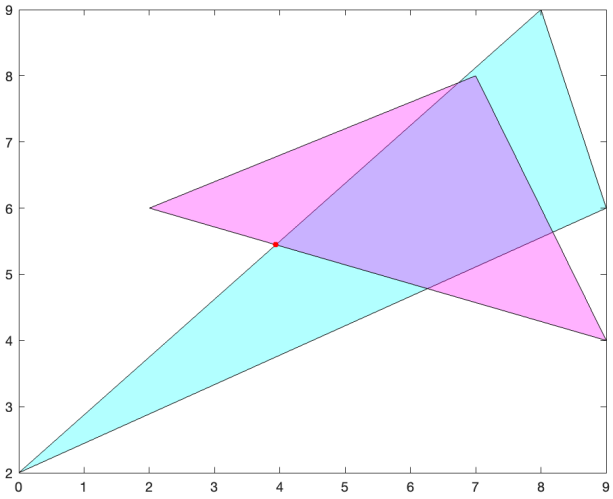
Rysunek 2: Ostatnie dwie tabelki sympleks fazy 1 w metodzie dwufazowej.

```
ITERACJA 1

             0     0     0     0     0     0
c_B   x_B    x1    x2    x3    x4    x5    x6    b
-----------------------------------------------------
0     x1     1     0.71  0     0     -0.37 0     0.49
0     x4     0     -0.48 0     1     0.29  0     0.28
0     x3     0     0.29  1     -0    0.37  0     0.51
0     x6     0     0.48  0     0     0.71  1     0.72
-----------------------------------------------------
      z      0     0     0     0     0     0     0
      z-c    0     0     0     0     0     0     >=0
```

Rysunek 3: Tabelka sympleks fazy 2 w metodzie dwufazowej.

Na Rysunku 4 wynik analogiczny do wywołania Matlaba. ZŁożyło się nawet, że znalezione zostały te same punkty. Oczywiście nie zawsze tak jest.



Rysunek 4: Wylosowane trójkąty do ZPL. Punkt znaleziony przez algorytm sympleks we własnej implementacji oznaczono kolorem czerwonym.

2.2 Przykład rozłącznych trójkątów

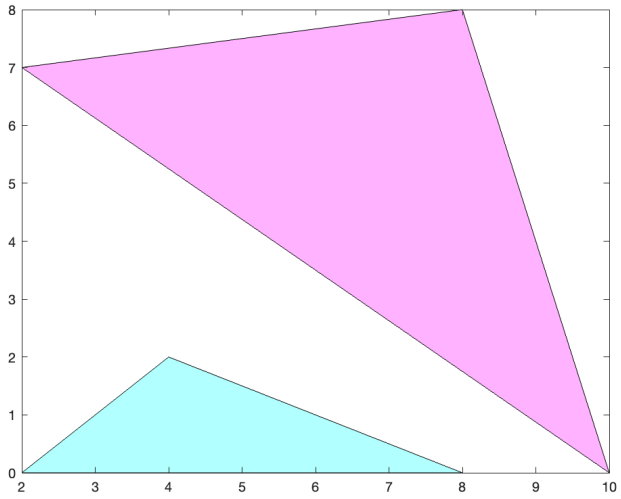
W przypadku wylosowanie rozłącznych trójkątów zwracane jest rozwiązanie sprzeczne. Na Rysunku 5 przedstawiono tabelkę sympleks, gdzie nie ma rozwiązania (jest sprzeczne). Wynika to z tego, że $z - c$ jest już nieujemne, więc kroki się skończyły, ale w bazie nadal znajdują się zmienne sztuczne (x_{10}). Oznacza to, że trójkąty nie mają przecięcia. Na Rysunku 6 przedstawiono te trójkąty.

```
ITERACJA 7

            0     0     0     0     0     0     -1    -1    -1    -1
c_B   x_B   x1    x2    x3    x4    x5    x6    x7    x8    x9    x10   b
-------------------------------------------------------------------------------
0     x2    1     1     1     0     0     0     0     0     1     0     1
0     x5    0.60  0     0.46  0     1     0.57  -0.10 0.03  0.80  0     0.80
0     x4    0     0     -0.29 1     0     1.14  0     -0.14 0     0     0
-1    x10   -0.60 0     -0.17 0     0     -0.71 0.10  0.11  -0.80 1     0.20
-------------------------------------------------------------------------------
      z     0.60  0     0.17  0     0     0.71  -0.10 -0.11 0.80  -1    -0.20
      z-c   0.60  0     0.17  0     0     0.71  0.90  0.89  1.80  0     >=0


Rozwiązanie sprzeczne
```

Rysunek 5: Ostatnia tabelka sympleks fazy 1, gdzie uzyskano sprzeczne rozwiązanie, czyli nie można przejść do fazy 2, algorytm kończy swoje działanie.



Rysunek 6: Wylosowane trójkąty do ZPL, które nie mają kolizji.

3 Testy

Przeprowadzono 100 razy zadanie programowania liniowego w celu porównania wyników między implementacją własną a wynikami wbudowanych funkcji w Matlabie. Na Rysunku 7 przedstawiono zagregowane wyniki tych wywołań.

Liczba znajdowania rozwiązań optymalnych wynosi 61 dla Matlaba i 57 dla własnej implementacji. Różnica dotyczy 4 przypadków, czyli jest to jakieś 6.5% pomyłki. Co daje całkiem dobrą skuteczność. Oczywiście jeśli chodzi o wyniki sprzecznych przypadków to musimy pamiętać, że mogło się tak akurat wylosować, że dane trójkąty nie mają kolizji, stąd taki jest to podział "procentowy".

Jeśli chodzi o średnią liczbę iteracji to oczywiście własna implementacja jest wolniejsza, wynosi prawie 6 iteracji, przy czym Matlab potrzebuje średnio 3.5 iteracji. Nie jest to zaskoczeniem, ale trzeba zauważyć, że 6 iteracji to nie jest taki zły wynik w porównaniu z Matlabem. To wciąż mniej niż dwukrotność jego liczby.

Niestety w testowanych przypadkach nie trafiły się trójkąty, które miałyby nieograniczoną liczbę rozwiązań, stąd nie możemy o nich nic powiedzieć.

```
MATLAB                                  WŁASNA IMPLEMENTACJA SYMPLEKS

Liczba skończonych rozwiązań: 61        Liczba skończonych rozwiązań: 57
Liczba sprzecznych rozwiązań: 39        Liczba sprzecznych rozwiązań: 43
Liczba nieograniczonych rozwiązań: 0    Liczba nieograniczonych rozwiązań: 0
Średnia liczba iteracji: 3.32           Średnia liczba iteracji: 5.95
```

Rysunek 7: Liczba skończonych, sprzecznych i nieograniczonych rozwiązań oraz średnia liczba iteracji dla Matlaba i własnej implementacji algorytmu sympleks w metodzie dwufazowej przy rozwiązywaniu ZPL.