Politechnika Warszawska

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH

Referat

Laboratorium 3, Grupa B

Krzysztof Milde 277229

Programowanie Matematyczne

7 listopada 2022

Spis treści

1 Problem Optymalizacyjny

Tematem zadnia jest rozwiązanie problemu maksymalizacyjnego w następującej postaci.

$$\max_{x \in \Omega} c^T x$$

$$\Omega: \begin{cases} [A|I]\,x \leq b, \quad b \geq 0 \\ [x_1, \cdots, x_m] \leq 0, [x_{m+1}, \cdots, x_n] \geq 0 \end{cases}$$

$$c, x \in R^n \quad b \in R^m \quad [A|I] \in R^{mxn} \quad n = 2 * m$$

Zadanie zostanie rozwiązane metodą symplex. Aby zadanie było rozwiązywalne algorytmem sympleksowym, należy sprowadzić je do postaci standarowej (bazowej), która wygląda w przedstawiony poniżej sposób.

$$\max[c|zeros([m,1])]^T x$$
$$[x_1, ..., x_m] = -[x_1, ..., x_m]$$
$$x = [x|x^d]$$
$$\begin{cases} [A|I|I]x = b \\ x >= 0 \end{cases}$$

Ponieważ zawsze musimy dodać zmienną dopełniającą, aby zniwelować nierówność, nie ma potrzeby wprowadzania zmiennych sztucznych. Jak widać na powyższych wzorach, wszystkie zmienne w wektorze x muszą być nieujemne, w związku z czym na czas wykonywania algorytmu zmienne $x_1, ..., x_m$ zostaną potraktowane jako dodatnie, a przy wyniku połowa wektora wynikowego X zostanie z powrotem przemnożona przez -1. Ta postać zadania jest również kanoniczna, ponieważ jesteśmy w stanie wybrać takie bazowe zmienne, aby w macierzy $[A|I|I]$ przyporządkować im odpowiednio ulokowana macierz jednostkową. Jak zostało powiedziane na wykładzie, dla zadania w takiej formie jako **punkt startowy** wybieramy zmienne dopełniające a początkowe rozwiązanie bazowe wynosi 0.

2 Opis Działania Algorytmu w Matlab

Możliwe rezultaty algorytmu to jedynie rozwiązanie optymalne oraz nieograniczone. Ponieważ punkt startowy z definicji spełnia ograniczenia nie istnieje możliwość na zistnienie sprzeczności.

2.1 Rozwiązanie Optymalne

W pierwszej kolejności przeanalizuję krok po kroku, w jaki sposób algorytm rozwiązuje problem, gdy istnieje rozwiązanie optymalne.

$$c = \begin{pmatrix} 1 \\ -2 \\ 5 \\ -1 \\ 2 \\ 2 \end{pmatrix}$$

$$A = \begin{pmatrix} 5 & 1 & 3 \\ -1 & 4 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 8 \\ 4 \\ 6 \end{pmatrix}$$

```
iteration 1
c:
     1    -2     5    -1     2     2     0     0     0

base indeces:
     0
     0
     0

Simplex Matrix:
     5     1     3     1     0     0     1     0     0
    -1     4     0     0     1     0     0     1     0
     1     2     1     0     0     1     0     0     1

b
     8
     4
     6

Current solution:
     0

z:
     0     0     0     0     0     0     0     0     0

z - c:
    -1     2    -5     1    -2    -2     0     0     0

Minimum value found in z - c vector: -5
Solution is not optimal
Smallest value found for x3, which will be new base variable substituting x7
```

Jak widać na załączonym obrazku, W pierwszej kolejności skrypt dodaje **m** zer odpowiadających zmiennym dopełniającym do wektora c oraz dwie macierze jednostkowe o wymiarach **m x m** do macierzy A tworząc w ten sposób macierz **simp_matrix**. Indeksy bazowe w pierwszej iteracji wynoszą zero, ponieważ odpowiadają zmiennym dopełniającym właśnie. Z uwagi na to, iż jest to dopiero początek algorytmu, zarówno Macierz A, jak i b są niezmienione w porównaniu do parametrów wejściowych. Mnożąc macież simp_matrix przez wetkor stworzony z indeksów bazowych osiągany jest nowy wektor - **z**. Nas najbardziej interesuje różnica **z-c**, gdyż na jej podstawie określamy, czy rozwiązanie jest optymalne. Jeśli w wektorze **z-c** znajduje się wartość ujemna, rozwiązanie nie jest optymalne i należy kontynuować obliczenia, w przeciwnym przypadku zmienne bazowe tworzą rozwiązanie optymalne. Jeżli rozwiązanie nie jest optymalne, szukamy największej w sensie bezwzględnym ujemnej wartośći, następnie, w macierzy **simp_matrix** wyznaczamy odpowiadającą tej wartości kolumne **column**. Liczymy stosunek **b/column**, z zastrzeżeniem, że w mianowniku mogą być jedynie liczby dodatnie. Mając wyznaczony stosunek, szukamy indeksu liczby, która podzielona przez odpowiadającą jej wartość w wektorze **b** jest najniższa (ale wciąz nieujemna - z dziedziny). W ten sposób zmienna, z wyznaczonej kolumny, zmieni w bazie zmienną wyznaczoną przez znaleziony wiersz. Następnie, należy znormalizować cały wiersz oraz korespondującą wartość w wektorze b, tak aby na przecięciu zmiennej w bazie oraz odpowiadającej jej kolumnie została wartość 1. Na koniec odejmujemy, bądź dodajemy do pozostałych wierszy znormalizowany wiersz, tak aby w danej kolumnie wartość 1 pojawiła się dokładnie raz (dokładnie te same operacje wykonujemy na wektorze **b**). Po tym kroku przechodzimy do kolejnej iteracji.

```
iteration 2
c:
     1    -2     5    -1     2     2     0     0     0

base indeces:
     5
     0
     0

Simplex Matrix:
    1.6667    0.3333    1.0000    0.3333         0         0    0.3333         0         0
   -1.0000    4.0000         0         0    1.0000         0         0    1.0000         0
   -0.6667    1.6667         0   -0.3333         0    1.0000   -0.3333         0    1.0000

b
    2.6667
    4.0000
    3.3333

Current solution:
   13.3333

z:
    8.3333    1.6667    5.0000    1.6667         0         0    1.6667         0         0

z - c:
    7.3333    3.6667         0    2.6667   -2.0000   -2.0000    1.6667         0         0

Minimum value found in z - c vector: -2
Solution is not optimal
Smallest value found for x5, which will be new base variable substituting x8
```

```
iteration 4
c:
     1    -2     5    -1     2     2     0     0     0

base indeces:
     5
     2
     2

Simplex Matrix:
    1.6667    0.3333    1.0000    0.3333         0         0    0.3333         0         0
   -1.0000    4.0000         0         0    1.0000         0         0    1.0000         0
   -0.6667    1.6667         0   -0.3333         0    1.0000   -0.3333         0    1.0000

b
    2.6667
    4.0000
    3.3333

Current solution:
    28

z:
    5.0000   13.0000    5.0000    1.0000    2.0000    2.0000    1.0000    2.0000    2.0000

z - c:
    4.0000   15.0000         0    2.0000         0         0    1.0000    2.0000    2.0000

Minimum value found in z - c vector: 0
Solution is optimal
```

Powtarzamy, powyższe instrukcje do momentu, gdy w wektorze **z-c** pozostaną jedynie wartości nieujemne. W ten sposób wyznaczamy **RO**. Wartość rozwiązania optymalnego obliczamy za pomocą produktu wektora z indeksami bazowymi oraz wektora **b** w ostatniej iteracji. Przy zwracaniu wyniku należy pamiętać, aby wartości **od 1 do m** wektora wynikowego **X** pomnożyć razy **-1**.

$$X = \begin{pmatrix} 0 \\ 0 \\ -2.6667 \\ 0 \\ 4 \\ 3.3333 \end{pmatrix}$$

$$FVAL = 28$$

$$EXITFLAG = 1$$

2.2 Brak Rozwiązania Optymalnego

Niestety niektóre problemy nie posiadają optymalnego rozwiązania. W takich przypadkach skrypt zwraca flagę 3 (unbounded).

$$c = \begin{pmatrix} 0 \\ 2 \\ 1 \\ 0 \\ -5 \\ -4 \\ -2 \\ 5 \end{pmatrix}$$

$$A = \begin{pmatrix} -2 & -4 & -5 & 0 \\ 1 & 0 & 5 & -1 \\ 2 & -1 & -1 & -2 \\ 4 & -3 & 5 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 2 \\ 4 \\ 2 \\ 5 \end{pmatrix}$$

W każdej iteracji, która nie jest optymalna, sprawdzamy, czy w każdej kolumnie, dla której **z-c** jest ujemne istnieje kandydat niesprzeczny z dziedziną. Jeżeli istnieje chociaż jedna taka kolumna, nie można wyznaczyć rozwiązania optymalnego. Na załączonym poniżej obrazku widać, że w 2 kolumnie, pomimo że wartość w wektorze **z-c** jest ujemna, nie bylibyśmy w stanie włączyć tej zmiennej do bazy.

```
iteration 1
c:
     0     2     1     0    -5    -4    -2     5     0     0     0     0

base indeces:
     0
     0
     0
     0

Simplex Matrix:
    -2    -4    -5     0     1     0     0     0     1     0     0     0
     1     0     5    -1     0     1     0     0     0     1     0     0
     2    -1    -1    -2     0     0     1     0     0     0     1     0
     4    -3     5     1     0     0     0     1     0     0     0     1

b
     2
     4
     2
     5

Current solution:
     0

z:
     0     0     0     0     0     0     0     0     0     0     0     0

z - c:
     0    -2    -1     0     5     4     2    -5     0     0     0     0

Solution is unbounded
```

3 Ciekawy Przykład

W rzadkich przypadkach zdarza się, że algorytm nie znajduje rozwiązania w maksymalnej dopuszczalnej liczbie iteracji $(40 * m)$, a wbudowany **linprog** radzi sobie z nimi dość szybko. Może to być spowodowane wielością możliwości wyboru punktu startowego oraz dalszych konsekwencji tego wyboru.

$$c = \begin{pmatrix} 5 \\ 1 \\ 4 \\ -3 \\ 4 \\ 2 \\ 0 \\ 1 \\ 3 \\ 5 \end{pmatrix}$$

$$A = \begin{pmatrix} -5 & 2 & 1 & -3 & 5 \\ -5 & 2 & 3 & 4 & 2 \\ 4 & 2 & -1 & -4 & 4 \\ -1 & 0 & 1 & -4 & -1 \\ 4 & 0 & -4 & 2 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 2 \\ 8 \\ 8 \\ 8 \\ 8 \end{pmatrix}$$

```
iteration 200
c:
     5     1     4    -3     4     2     0     1     3     5     0     0     0     0     0

base indeces:
     0
     0
     5
     0
    -3

Simplex Matrix:
         0    1.8333   -4.2500         0    6.0000    1.0000         0   -0.0833         0    1.3333    1.0000         0   -0.0833         0    1.3333
         0    4.1667    1.2500         0    6.5000         0    1.0000    1.0833         0    0.1667         0    1.0000    1.0833         0    0.1667
    1.0000    0.1667   -0.7500         0    0.5000         0         0    0.0833         0    0.1667         0         0    0.0833         0    0.1667
         0   -1.1667   -1.7500         0   -2.5000         0         0   -0.5833    1.0000    0.8333         0         0   -0.5833    1.0000    0.8333
         0   -0.3333   -0.5000    1.0000   -0.5000         0         0   -0.1667         0    0.1667         0         0   -0.1667         0    0.1667

b
    12
    18
     2
    10
     0

Current solution:
    10

z:
    5.0000    1.8333   -2.2500   -3.0000    4.0000         0         0    0.9167         0    0.3333         0         0    0.9167         0    0.3333

z - c:
         0    0.8333   -6.2500         0         0   -2.0000         0   -0.0833   -3.0000   -4.6667         0         0    0.9167         0    0.3333

Minimum value found in z - c vector: -6.250000e+00
Solution is not optimal
Smallest value found for x3, which will be new base variable substituting x4
Solution is unbounded
```

4 Testy

Wykonano 2 testy:

1. Dla losowych danych N=100 razy i porównując z linprog zbadaj procentową skuteczność swojej implementacji (liczba iteracji, zadania posiadające RO, zadania sprzeczne, zadania nieograniczone).
2. Dla losowych danych N=100 razy dla zadań, które posiadają RO i porównując z linprog zbadaj procentową skuteczność swojej implementacji.

Z powodu problemu z porównaniem liczb zmiennoprzecinkowych spowodowanej naturą zapisu liczb zmiennoprzecinkowych w pamięci ustanowiłem czynnik tolerancji na poziomie 10^{-6}. Jeżeli wyniki różnią się od siebie o nie więcej niż jedna milionowa, uznaję że są identyczne.

W pierwszym eksperymencie osiągnąłem skuteczność na poziomie **94%**. Całkowita liczba iteracji, które przeprowadził mój skrypt wynosiła 884, podczas gdy linprog zrealizował jedynie 266. Jest to spowodowane długim szukaniem rozwiązania opisanym w sekcji **Ciekawy Przykład**.

Drugi eksperyment, polegał na porównaniu jedynie przypadków, dla których wbudowany linprog zwrócił rozwiązanie optymalne zakończył się z wynikiem **95%**. Kiedy linprog wskazywał rozwiązanie optymalne, wówczas mój skrypt wskazał identyczne rozwiązanie w 95 przypadkach na 100.

Oba eksperymenty można powtórzyć. Ziarna, dla których przeprowadzałem testy, to odpowiednio 100 dla pierwszego eksperymentu oraz 1000 dla drugiego. W celu stworzenia zróżnicowanych przypadków testowych zaimplementowano następujące reguły:

> Do testów wygeneruj **losowe** wektory i macierze o wartościach całkowitoliczbowych (**randi**):
> $\boldsymbol{m} = \mathbf{3} \div \mathbf{5}$
> dla $\boldsymbol{c}$ oraz $\boldsymbol{A}$ wartości całkowite z przedziału $[\mathbf{-5},\ \mathbf{5}]$
> dla $\boldsymbol{b}$ wartości całkowite z przedziału $[\mathbf{1},\ \mathbf{8}]$