Lab 4c,
Programowanie matematyczne

Piotr Onyszczuk, gr. C

1 XII 2021

1 Treść zadania

Celem zadania jest rozwiązanie za pomocą funkcji **linprog** oraz za pomocą własnej implementacji algorytmu **sympleks** zagadnienia dualnego do następującego zagadnienia.

$$\max_{x \in \Omega} c^T x \tag{1}$$

$$\Omega : \begin{cases} Ax \leq b & \text{, b>0} \\ |x| \leq g & \text{, g>0} \end{cases} \tag{2}$$

$$c, x \in R^n; b \in R^m; g \in R^n; A \in R^{mxn}; m = n \tag{3}$$

1.1 Dane testowe

Do testów należy użyć losowe wektory (n=5, m=5).

- c i A zawierają wartości całkowite z przedziału [-2,2]
- b i g zawierają wartości całkowite z przedziału [1,5]

2 Algorytm

Algorytm opisany zostanie na jednym z wylosowanych przykładów (Wycinek 1)

2.1 Przekształcenie układu

Pierwszym etapem jest przekształcenie układu. Etap ten polega rozbiciu równań z wartością bezwzględną na dwa równania.

$$x \leq |g| => \begin{cases} -x \leq g \\ x \leq g \end{cases} \tag{4}$$

```
A =

     0    -2     0     1     2
     1    -2     1     0     2
    -2    -1    -1     0    -1
    -1    -1     2    -2     1
    -2     0    -2    -2     2


b =

     5     1     1     1     5


c =

    -2     0     2     0     1


g =

     2     4     5     1     4
```

Wycinek 1: Wylosowany przykład

W praktyce jest to dodanie do macierzy A pięciu wierszy zanegowanej macierzy identyczności, a następnie kolejnych pięciu wierszy macierzy identyczności z dodatnim znakiem. Wektor b roszerzany jest dwukrotnie o wektor g, otrzymując na koniec następujące zagadnienie (Wycinek 2).

2.2 Zadanie dualne

Na podstawie przekształconego zagadnienia tworzone jest zadanie dualne. Przekształcenie to jest w zasadzie natychmiastowe. Wektory b i c zamieniają się rolami, a macierz A należy transponować.

Aby móc wykorzystać podstawową wersję algorytmu **sympleks**, nowy wektor b powinien zawierć nieujemne wartości. W tym celu negowane są wiersze odpowiadające wartościom ujemnym, zarówno w macierzy A, jak i w wektorze b.

Potrzebna jest jeszcze baza. Do bazy trafiają elementy odpowiadające kolumnom powstałym po rozbiciu wartości bezwzględnej. Będą to jednak kolumny zawierające wartość 1, a nie -1. Ten wybór jest o tyle trudniejszy, że niektóre wiersze mogą być zanegowane.

Tak otrzymywane jest ostateczne zadanie dla algorytmu **sympleks** (Wycinek 3).

```
c =

    -2     0     2     0     1


A =

     0    -2     0     1     2
     1    -2     1     0     2
    -2    -1    -1     0    -1
    -1    -1     2    -2     1
    -2     0    -2    -2     2
    -1     0     0     0     0
     0    -1     0     0     0
     0     0    -1     0     0
     0     0     0    -1     0
     0     0     0     0    -1
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1


b =

     5     1     1     1     5     2     4     5     1     4     2     4     5     1     4
```

Wycinek 2: Nowe zagadnienie

```
A =

     0    -1     2     1     2     1     0     0     0     0    -1     0     0     0     0
    -2    -2    -1    -1     0     0    -1     0     0     0     0     1     0     0     0
     0     1    -1     2    -2     0     0    -1     0     0     0     0     1     0     0
     1     0     0    -2    -2     0     0     0    -1     0     0     0     0     1     0
     2     2    -1     1     2     0     0     0     0    -1     0     0     0     0     1

b =

     2     0     2     0     1

c =

    -5    -1    -1    -1    -5    -2    -4    -5    -1    -4    -2    -4    -5    -1    -4

base =

     6    12    13    14    15
```

Wycinek 3: Ostateczne zagadnienie i baza

```
x_primary =

    2.0000   -4.0000   -1.1667   -1.0000   -2.6667
```

Wycinek 4: Rozwiązanie zadania pierwotnego

2.3 sympleks

Implementacja tego algorytmu nie będzie tu omówiona, ponieważ jest dokładnie taka sama jak w Lab3c.

2.4 Odzyskanie rozwiązania dla zadania pierwotnego

Oba zadania (pierwotne i dualne) mają tę samą wartość optymalną, z dokładnością do negacji.

Rozwiązaniem optymalnym dla zadania dualnego jest rozwiązanie otrzymane z algorytmu **sympleks**.

Aby uzyskać rozwiązanie pierwotne należy przemnożyć wektor wynikowy dla zadania dualnego przez odwróconą macierz bazy. Trzeba jednak najpierw zanegować te kolumny macierzy bazy, dla których odpowiadające wiersze były negowane przed rozpoczęciem **sympleksu** (Wycinek 4). Negowane są kolumny, ponieważ macierz jest odwróconą macierzą bazy.

3 Wyniki i wnioski

Testując na 1000 przypadków, oba algorytmy zawsze zwracają wynik, zawsze ten sam wynik (z dokładnością co do znaku). Stąd też potwierdzenie wniosku z Lab3c, że zagadnienie zawsze posiada rozwiązanie. Niemożliwe jest w takim razie podanie przykładu bez rozwiązania.

Na 1000 przypadków (dla zadania dualnego):

- Własna implementacja zwraca wynik szybciej (mniej iteracji) niż linprog 807 razy
- Wyniki nie różnią się nigdy. Sprawdzana dokładność to $1e-6$
- Linprog wykonuje średnio prawie dwa razy więcej iteracji niż własna implementacja ($2,26$ vs $4,38$).
- Własna implementacja wykonuje maksymalnie 9 iteracji, podczas gdy linprog maksymalnie 8
- Różne rozwiązania (wektory) uzyskiwane są w 415 przypadkach

3.1 Mnożniki Lagrange'a

Mnożniki Lagrange'a można otrzymać w ostatnim parametrze wyjściowym funkcji **linprog**. Analizując mnożniki dla kilku przypadków można dojść do wniosku że mnożniki Lagrange'a dla zadania pierwotnego są w zasadzie rozwiązaniem zadania dualnego. Przypuszczalnie wartości te mogą się różnić, gdy istnieje więcej niż jedno rozwiązanie optymalne

4 Pliki

- Losowanie.m - skrypt losujący dane do zadania
- Rozwiazanie.m - główny skrypt, losuje zadania w pętli i uruchamia algorytmy
- get_dual.m - zawiera funkcję przekształcającą zagadnienie oraz funkcję zwracającą zadanie dualne
- compute_dual.m - zawiera funkcję modyfikującą zadanie dualne, znajdującą bazę oraz zwracającą rozwiązania zadania dualnego i pierwotnego
- my_simplex.m - zawiera implementację algorytmu sympleks wraz z wypisywaniem wyniku
- sympleks.m - zawiera funkcję zgodną ze specyfikacją podaną w poleceniu.