PROGRAMOWANIE LINIOWE – SZKIC ROZWIĄZANIA ZADANIA

Tu proszę podać imiona i nazwiska autorów

8 czerwca 2017

```r
# Określenie parametrów zadania prymalnego
A <- matrix(c(3, 3, 1, 2, 4, 5), ncol = 2)
b <- matrix(c(7, 8, 8))
c <- c(3, 5)
```

$$\max_{(x_1,x_2)\in D} Z = 3x_1 + 5x_2$$

pod warunkiem

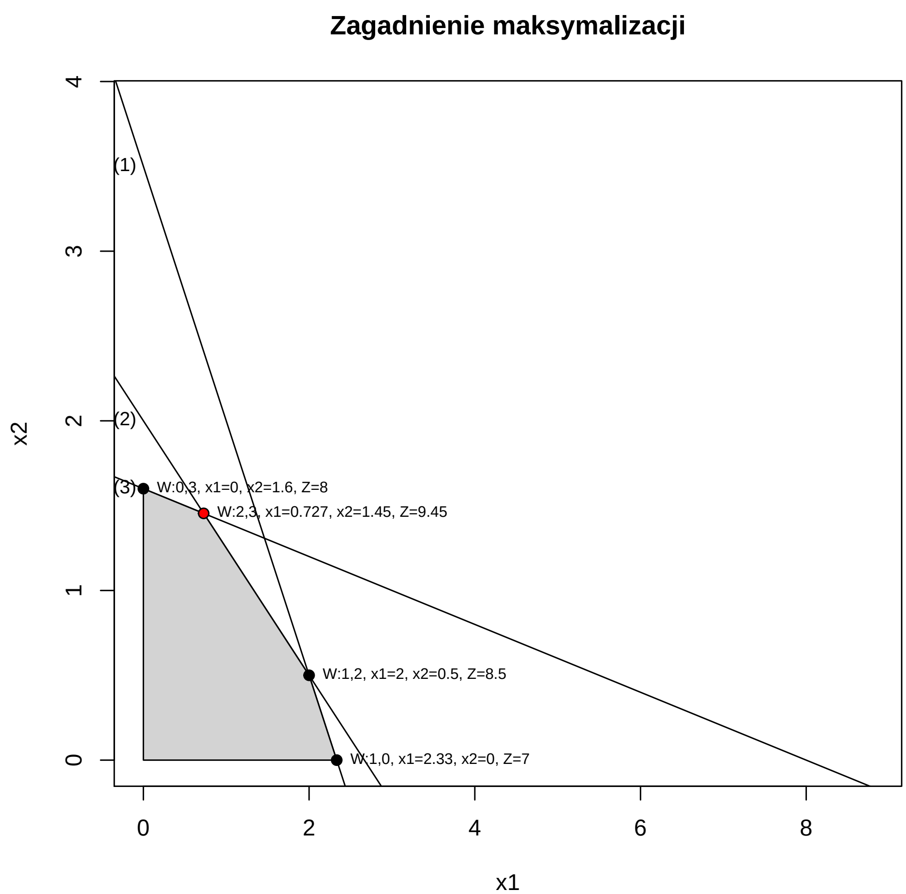
$$\begin{aligned} D = \{(x_1, x_2) \in \mathbf{R}^2 : & & \\ 3x_1 + 2x_2 & \leqslant & 7 \\ 3x_1 + 4x_2 & \leqslant & 8 \\ 1x_1 + 5x_2 & \leqslant & 8 \\ x_1 \geqslant 0, x_2 \geqslant 0 & & \} \end{aligned}$$

Rozwiązanie metodą geometryczną

```r
# Górna granica liczby iteracji
n <- dim(A)[2]
m <- dim(A)[1]
it <- choose(m + n, n)
```

Górna granica liczby iteracji wynosi 10.

```r
# Metoda geometryczna - rozwiązanie zadania prymalnego
optg <- maxgeom(A, b, c)
```



Rozwiązanie optymalne x_1^*, x_2^*, Z^* zadania prymalnego (maksymalizacji) przedstawia się następująco:

```
       x1        x2         Z
0.7272727 1.4545455 9.4545455
```

Zadanie dualne - minimalizacja:

$$\min_{(y_1,y_2,y_3)\in D'} K = 7y_1 + 8y_2 + 8y_3$$

pod warunkiem

$$
\begin{aligned}
D' = \{(y_1, y_2, y_3) \in \mathbf{R}^3 : & \\
3y_1 + 3y_2 + 1y_3 &\geqslant 3 \\
2y_1 + 4y_2 + 5y_3 &\geqslant 5 \\
y_1 \geqslant 0, y_2 \geqslant 0, y_3 \geqslant 0 & \quad \}
\end{aligned}
$$

Wykorzystanie optymalnego rozwiązania zadania prymalnego metodą geometryczną oraz twierdzeń o dualności w celu rozwiązania zadania dualnego:

Funkcyjne warunki ograniczające 2 i 3 dla optymalnego rozwiązania przybierają postać równań, natomiast dla tegoż rozwiązania warunek 1 przyjmuje formę ostrej nierówności. Tak więc w oparciu o twierdzenia o dualności mamy: Dla zadania dualnego $y_1^* = 0$, $y_2^* > 0$ $y_3^* > 0$.

Ponadto jako, że obie zmienne decyzjne dla optymalnego rozwiązania zadania prymalnego przyjmują wartości dodatnie $x_1^* = 0.7272727, x_2^* = 1.4545455$, na podstawie twierdzeń można stwierdzić, że w zadaniu dualnym dla rozwiązania optymalnego oba funkcyjne warunki ograniczające przybierają postać równości.

```
# Zadanie dualne - rozwiązanie z wykorzystaniem twierdzeń o dualności
# rozwiązanie optymalne zadania dualnego
dual <- rbind(0, solve(t(A)[, -1], matrix(c)))
optmin <- t(b) %*% dual
rozwdual <- c(dual, optmin)
names(rozwdual) <- c("y1", "y2", "y3", "K")
```

Zgodnie z powyższymi ustaleniami wartości zmiennych dualnej dla rozwiązania optymalnego są następujące: y_1^* wynosi 0, a wartości zmiennych y_2^*, y_3^* otrzymujemy jako rozwiązanie układu równań:

$$
\begin{aligned}
3y_2^* + 1y_3^* &= 3 \\
4y_2^* + 5y_3^* &= 5
\end{aligned}
$$

Optymalne rozwiązanie dualne dane jest następująco:

```
       y1        y2        y3         K 
0.0000000 0.9090909 0.2727273 9.4545455 
```

Rozwiązaniem zadania dualnego $y_1^* = 0, y_2^* = 0.9090909, y_3^* = 0.2727273$ są ceny dualne surowców.

Interpretacja cen dualnych: ...
Jak widać wartości funkcji celu dla rozwiązań optymalnych są równe dla zadania prymalnego i zadania dualnego: $Z^* = K^* = 9.4545455$

Rozwiązanie metodą simpleks

```
# Metoda simpleks - rozwiązanie zadania prymalnego
if (!require(lpSolve)) install.packages("lpSolve")

Loading required package: lpSolve

library(lpSolve)

choose(m + n, m)

[1] 10

zn <- rep("<=", 3)
optsim <- lp("max", c, A, zn, t(b), compute.sens = 1)
optsim$solution

[1] 0.7272727 1.4545455

optsim$objval

[1] 9.454545

optsimdual <- optsim$duals
# zerowy wiersz tablicy simpleks dla rozwiązania #optymalnego
names(optsimdual) <- c("s1", "s2", "s3", "x1", "x2")
optsimdual

       s1        s2        s3        x1        x2 
0.0000000 0.9090909 0.2727273 0.0000000 0.0000000 
```

Zerowy wiersz w tablicy simpleks dla rozwiązania optymalnego zadania prymalnego, zawiera w komórkach odpowiadających zmiennym swobodnym tzn. s_1^*, s_2^*, s_3^* rozwiązanie zadania dualnego:$s_1^* = y_1^* = 0, s_2^* = y_2^* = 0.9090909, s_3^* = y_3^* = 0.2727273$, które określa ceny dualne poszczególnych surowców.

Funkcja *lp* z pakietu *lpSolve* umożliwiająca rozwiązania zadania PL metodą simpleks zwraca następujące elementy:

```
names(optsim)

 [1] "direction"        "x.count"          "objective"
 [4] "const.count"      "constraints"      "int.count"
 [7] "int.vec"          "bin.count"        "binary.vec"
[10] "num.bin.solns"    "objval"           "solution"
[13] "presolve"         "compute.sens"     "sens.coef.from"
[16] "sens.coef.to"     "duals"            "duals.from"
[19] "duals.to"         "scale"            "use.dense"
[22] "dense.col"        "dense.val"        "dense.const.nrow"
[25] "dense.ctr"        "use.rw"           "tmp"
[28] "status"
```

Rozważmy zwracane elementy pod kątem uzyskania wyników analizy wrażliwości - analizy przedziałowej ...

DLA CHĘTNYCH – szkic rozwiązania zadania PL metodą simpleks w PYTHONIE

Można skorzystać z funkcji *linprog* w pakiecie *scipy.optimize*. W procedurze domyślnie funkcja celu podlega minimlaizacji, w celu wykonania maksymalizacji wystarczy przemnożyć wektor parametrów funkcji celu c przez -1. Stąd też wartość funkcji celu dla rozwiązania optymalnego podawana jest z przeciwnym znakiem.

```
A = [[3,2], [3,4], [1,5]]
b = [7, 8, 8]
c = [-3,-5]

# Warunki na nieujemność zmiennych decyzyjnych x1, x2 >= 0
x1_bounds = (0, None)
```

```
x2_bounds = (0, None)

# Importowanie funkcji linprog z pakietu scipy.optimize
from scipy.optimize import linprog

res = linprog(c, A_ub=A, b_ub=b, bounds=(x1_bounds, x2_bounds),
              options={"disp": True})
print(res)
```

```
## Optimization terminated successfully.
##          Current function value: -9.454545
##          Iterations: 2
##      fun: -9.4545454545454533
##  message: 'Optimization terminated successfully.'
##      nit: 2
##    slack: array([ 1.90909091,  0.        ,  0.        ])
##   status: 0
##  success: True
##        x: array([ 0.72727273,  1.45454545])
```