

Krzysztof Trajkowski

Przegląd pakietów do optymalizacji liniowej

1. Wprowadzenie

Niniejsze opracowanie dotyczy opisu funkcji `solveLP(linprog)` i `lp(lpSolve)` za pomocą których możemy rozwiązywać zadania programowania liniowego.

Liniowa funkcja celu jest postaci:

$$f(x_1, x_2, \dots, x_n) = c_1x_1 + c_2x_2 + \dots + c_nx_n \rightarrow (\min) \quad \text{lub} \quad (\max) \quad (1.1)$$

gdzie: $c_n \in \mathcal{R}$.

Gradient czyli wektor, który jest prostopadły do liniowej funkcji celu (1.1):

$$\nabla f(x_1, x_2, \dots, x_n) = [c_1, c_2, \dots, c_n] \quad (1.2)$$

Ograniczenia które tworzą zbiór rozwiązań dopuszczalnych są dane następującymi wzorami:

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \leq b \quad (1.3a)$$

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \geq b \quad (1.3b)$$

$$a_1x_1 + a_2x_2 + \dots + a_nx_n = b \quad (1.3c)$$

gdzie: $a_n \in \mathcal{R}$ oraz $b \in \mathcal{R}$.

Ograniczenia brzegowe:

$$x_1 \geq 0, \dots, x_n \geq 0 \quad (1.4)$$

Tak sformułowane ograniczenia (1.3a)-(1.3c) oraz ograniczenia brzegowe (1.4) tworzą zbiór rozwiązań dopuszczalnych.

2. Zadanie programowania liniowego

Funkcja celu:

$$2x_1 + 3x_2 \rightarrow \max \quad (2.1)$$

Gradient:

$$[2, 3] \quad (2.2)$$

Ograniczenia:

$$-3x_1 + 9x_2 \leq 36 \quad (2.3)$$

$$4x_1 - 5x_2 \leq 11 \quad (2.4)$$

$$-3x_1 - 2x_2 \leq -6 \quad (2.5)$$

$$4x_1 + 3x_2 \leq 39 \quad (2.6)$$

Ograniczenia brzegowe: $x_1 \geq 0$ oraz $x_2 \geq 0$.

Aby znaleźć wierzchołki wielokąta wypukłego (zbiór rozwiązań dopuszczalnych - rysunek 2.1) należy rozwiązać kilka układów równań. Na listingu 2.1 został przedstawiony sposób rozwiązania wybranego układu równań (2.7):

$$\begin{cases} -3x + 9y = 36 \\ 4x + 3y = 39 \end{cases} \quad (2.7)$$

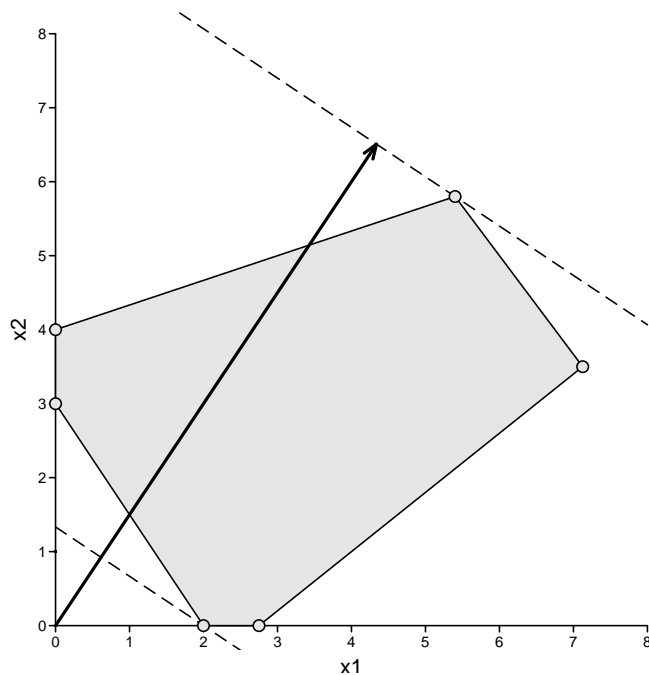
Listing 2.1: Rozwiązanie układu równań (2.7)

```
> A=matrix(c(-3,4,9,3),2,2)
> B=matrix(c(36,39),2,1)
> solve(A)%*%B
      [,1]
[1,]  5.4
[2,]  5.8
```

Graficzna prezentacja rozwiązania ZPL jest przedstawiona na rysunku 2.1. Kod jest przedstawiony na listingu 2.2:

Listing 2.2: Rozwiązanie ZPL - metoda graficzna

```
> plot(0,1,col="black",pch=".",xlim=c(0,8),ylim=c(0,8),axes=F,xlab="",ylab="↔
")
> mtext(side=1, line=0, paste("x1"),cex=0.8);mtext(side=2, line=0, paste("↔
x2"),cex=0.8)
> axis(1,c(0:8),pos=0,las=1,mgp=c(3, -0.1 ,0),tcl=-0.2,cex.axis=.6)
> axis(2,c(0:8),pos=0,las=1,mgp=c(3, 0.3 ,0),tcl=-0.2,cex.axis=.6)
> curve(add=T,lty=5,(-2/3)*x+47/5)
> curve(add=T,lty=5,(-2/3)*x+4/3)
> p=data.frame(x1=c(2,0,0,5.4,7.125,2.75),x2=c(0,3,4,5.8,3.5,0)) # ↔
      wierzchołki wielokąta wypukłego
> polygon(p,col="gray90")
> points(p,pch=21,bg="gray90")
> arrows(0,0,282/65,423/65,length=0.1,angle=20,lwd=2) # gradient funkcji ↔
celu
```



Rysunek 2.1: Rozwiązanie graficzne.

Listing 2.3: Rozwiązanie ZPL

```
> fc=function(x){ # funkcja celu
+   x1=x[1]
+   x2=x[2]
+   2*x1+3*x2
+ }
```

```

> p=data.frame(x1=c(2,0,0,5.4,7.125,2.75),x2=c(0,3,4,5.8,3.5,0)) # ←
      wierzchołki wielokąta wypukłego
> min(fc(p)) # minimalna wartość funkcji celu
[1] 4
> max(fc(p)) # maksymalna wartość funkcji celu
[1] 28.2
> range(fc(p)) # wartości ekstremalne funkcji celu
[1] 4.0 28.2

```

Na listingu 2.4 zostało przedstawione rozwiązanie ZPL za pomocą funkcji solveLP(linprog).

Listing 2.4: Rozwiązanie ZPL - funkcja solveLP

```

> library(linprog) # załadowanie pakietu linprog
> f=c(2,3) # współczynniki funkcji celu
> b=c(36,11,-6,39)
> m=matrix(c(-3, 4, -3, 4, 9, -5, -2,3),4,2)
> solveLP(f,b,m,T)

```

Results of Linear Programming / Linear Optimization

Objective function (Maximum): 28.2

Iterations in phase 1: 1

Iterations in phase 2: 3

Solution

opt

1 5.4

2 5.8

Basic Variables

opt

1 5.4

2 5.8

S 2 18.4

S 3 21.8

Constraints

	actual	dir	bvec	free	dual	dual.reg
1	36.0	<=	36	0.0	0.133333	25.875
2	-7.4	<=	11	18.4	0.000000	18.400
3	-27.8	<=	-6	21.8	0.000000	21.800
4	39.0	<=	39	0.0	0.600000	27.000

All Variables (including slack variables)

	opt	cvec	min.c	max.c	marg	marg.reg
1	5.4	2	-1.000000	4.000000	NA	NA
2	5.8	3	1.500000	Inf	NA	NA
S 1	0.0	0	-Inf	0.133333	-0.133333	25.875
S 2	18.4	0	-0.187500	1.285714	0.000000	NA
S 3	21.8	0	-0.818182	6.000000	0.000000	NA
S 4	0.0	0	-Inf	0.600000	-0.600000	27.000

Oprócz rozwiązania optymalnego które wynosi $f_{max} = 28,2$ dla $x_1 = 5,4$ i $x_2 = 5,8$ otrzymaliśmy także kilka dodatkowych informacji:

- zasoby s_1 i s_4 zostały wykorzystane w całości. Z kolei zasobu s_2 i s_3 pozostało odpowiednio 18,4 i 21,8.
- rozwiązanie dualne wynosi: $y_1 = 0,13$, $y_2 = 0$, $y_3 = 0$, $y_4 = 0,6$.
- zmiana wybranego współczynnika funkcji celu (2.1) dla $c_1 = 2$ na przedziale $c_1 \in \langle -1; 4 \rangle$ lub dla $c_2 = 3$ na przedziale $c_2 \in \langle 1, 5; \infty \rangle$ nie spowoduje zmiany wartości x_1 oraz x_2 .

Rozwiązanie tego samego zadania za pomocą funkcji `lp(lpSolve)` jest przedstawione na listingu 2.5.

Listing 2.5: Rozwiązanie ZPL - funkcja `lp`

```
> library(lpSolve) # załadowanie pakietu lpSolve
> f=c(2,3) # współczynniki funkcji celu
> m=matrix ( c(-3,4,-3,4, 9,-5,-2,3), 4,2)
> o=c("<=", "<=", "<=", "<=")
> b=c(36,11,-6,39)
> lp("max",f,m,o,b) # maksymalna wartość funkcji celu
Success: the objective function is 28.2
> lp("max",f,m,o,b)$solution # wartości zmiennych decyzyjnych x1 oraz x2
[1] 5.4 5.8
> lp("max",f,m,o,b,compute.sens=T)$duals # rozwiązanie dualne
[1] 0.1333333 0.0000000 0.0000000 0.6000000 0.0000000 0.0000000
```

Warto w tym miejscu zaznaczyć, że ograniczenia brzegowe $x_1 \geq 0$ oraz $x_2 \geq 0$ są ustawieniami domyślnymi. A więc nie musimy ich uwzględniać w kodzie.

Jeżeli chcemy, aby wszystkie zmienne decyzyjne należały do zbioru liczb całkowitych dodatnich ($x_1 \in \mathcal{Z}_+$ oraz $x_2 \in \mathcal{Z}_+$) należy użyć opcji `int.vec=TRUE`.

Listing 2.6: Rozwiązanie ZPL - funkcja `lp`

```
> lp("max",f,m,o,b,int.vec=T)
Success: the objective function is 27
> lp("max",f,m,o,b,int.vec=T)$solution
[1] 6 5
```

Z kolei dla założenia $x_1 \in \{0,1\}$ oraz $x_2 \in \mathcal{Z}_+$ należy użyć opcję `int.vec=2` (dla zmiennej x_2) oraz `binary.vec=1` (dla zmiennej x_1).

Listing 2.7: Rozwiązanie ZPL - funkcja `lp`

```
> lp("max",f,m,o,b,int.vec=2,binary.vec=1)
Success: the objective function is 14
> lp("max",f,m,o,b,int.vec=2,binary.vec=1)$solution
[1] 1 4
```

3. Zadanie transportowe

Zadanie transportowe możemy rozwiązać z wykorzystaniem funkcji `lp(lpSolve)`. Jednak w środowisku R mamy do dyspozycji funkcję `lp.transport(lpSolve)` za pomocą której mamy możliwość rozwiązać takie zadanie. Przykładowe dane zostały przedstawione w tabeli 3.1.

Tabela 3.1. ZT.

	Odb_1	Odb_2	Odb_3	
$Dost_1$	20	16	18	30
$Dost_2$	15	15	20	25
$Dost_3$	14	11	12	22
$Dost_4$	11	10	14	20
	16	27	22	

Ponieważ $97 > 65$ (tzn. $30 + 25 + 22 + 20 > 16 + 27 + 22$) to mamy do czynienia z niebilansowanym zadaniem transportowym. Występuje przewaga podaży (dostawcy) nad popytem (odbiorcy).

Funkcja celu:

$$20x_{11} + 16x_{12} + 18x_{13} + 15x_{21} + 15x_{22} + 20x_{23} + 14x_{31} + 11x_{32} + 12x_{33} + 11x_{41} + 10x_{42} + 14x_{43} \rightarrow \min \quad (3.1)$$

Ograniczenia dla dostawców:

$$\begin{aligned} x_{11} + x_{12} + x_{13} &\leq 30 \\ x_{21} + x_{22} + x_{23} &\leq 25 \\ x_{31} + x_{32} + x_{33} &\leq 22 \\ x_{41} + x_{42} + x_{43} &\leq 20 \end{aligned}$$

Ograniczenia dla odbiorców:

$$\begin{aligned} x_{11} + x_{21} + x_{31} + x_{41} &= 16 \\ x_{12} + x_{22} + x_{32} + x_{42} &= 27 \\ x_{13} + x_{23} + x_{33} + x_{43} &= 22 \end{aligned}$$

Listing 3.1: Rozwiązanie ZT - funkcja lp.transport

```
> m= matrix(c(20,15,14,11,16,15,11,10,18,20,12,14),4,3)
> rs= rep("<=",4)
> rm= c(30,25,22,20)
> cs= rep("=",3)
> cm= c(16,27,22)
> lp.transport(m, "min", rs, rm, cs, cm)
Success: the objective function is 809
> lp.transport(m, "min", rs, rm, cs, cm)$solution
  [,1] [,2] [,3]
[1,]  0   0   0
[2,] 16   7   0
[3,]  0   0  22
[4,]  0  20   0
```

4. Zadanie przydziału

Zadanie przydziału polega na przydzieleniu pracownika do stanowiska pracy. Przykładowe dane do tego typu zadania zostały przedstawione w tabeli 4.1 - liczba godzin jaką potrzebuje każdy z pracowników na wykonanie określonego zadania.

Tabela 4.1. ZP.

	Zad ₁	Zad ₂	Zad ₃	Zad ₄	
Prac ₁	8	8	7	4	1
Prac ₂	5	5	8	7	1
Prac ₃	6	7	5	9	1
Prac ₄	4	4	8	5	1
	1	1	1	1	

Funkcja celu:

$$8x_{11} + 8x_{12} + 7x_{13} + 4x_{14} + 5x_{21} + 5x_{22} + 8x_{23} + 7x_{24} + 6x_{31} + 7x_{32} + 5x_{33} + 9x_{34} + 4x_{41} + 4x_{42} + 8x_{43} + 5x_{44} \rightarrow \min \quad (4.1)$$

Ograniczenia dla pracowników:

$$\begin{aligned}x_{11} + x_{12} + x_{13} + x_{14} &= 1 \\x_{21} + x_{22} + x_{23} + x_{24} &= 1 \\x_{31} + x_{32} + x_{33} + x_{34} &= 1 \\x_{41} + x_{42} + x_{43} + x_{44} &= 1\end{aligned}$$

Ograniczenia dla zadań:

$$\begin{aligned}x_{11} + x_{21} + x_{31} + x_{41} &= 1 \\x_{12} + x_{22} + x_{32} + x_{42} &= 1 \\x_{13} + x_{23} + x_{33} + x_{43} &= 1 \\x_{14} + x_{24} + x_{34} + x_{44} &= 1\end{aligned}$$

Rozwiązanie zadania przydziału z wykorzystaniem funkcji `lp.transport(lpSolve)`:

Listing 4.1: Rozwiązanie ZP - funkcja `lp.transport`

```
> m= matrix(c(8,5,6,4,8,5,7,4,7,8,5,8,4,7,9,5),4,4)
> rs= rep("=",4)
> rm= c(1,1,1,1)
> cs= rep("=",4)
> cm= c(1,1,1,1)
> lp.transport(m, "min", rs, rm, cs, cm)
Success: the objective function is 18
> lp.transport(m, "min", rs, rm, cs, cm)$solution
      [,1] [,2] [,3] [,4]
[1,]    0    0    0    1
[2,]    1    0    0    0
[3,]    0    0    1    0
[4,]    0    1    0    0
```

W przypadku rozwiązywania zbilansowanego zadania przydziału (liczba pracowników jest równa liczbie zadań - tabela 4.1) wygodniej jest wykorzystać funkcję `lp.assign(lpSolve)`:

Listing 4.2: Rozwiązanie ZP - funkcja `lp.assign`

```
> m= matrix(c(8,5,6,4,8,5,7,4,7,8,5,8,4,7,9,5),4,4)
> lp.assign(m, "min")
Success: the objective function is 18
> lp.assign(m, "min")$solution
      [,1] [,2] [,3] [,4]
[1,]    0    0    0    1
[2,]    1    0    0    0
[3,]    0    0    1    0
[4,]    0    1    0    0
```
