Sprawozdanie 1

Problem transportowy LINGO

Spis treści

Opis ogólny

Podczas przygotowywania sprawozdania korzystałem z oprogramowania LINGO 18.0. Zadanie polegało na modyfikacji przedstawionego na zajęciach przykładowego pliku *TRAN.lg4* wbudowanego w używany program.

Zadanie 1

Pierwszym krokiem było niejako spersonalizowanie programu modyfikując dane wejściowe na takie, które byłyby powiązane z numerem mojego indeksu. Czynność tę wykonałem w następujący sposób:

```
DATA:
   CAPACITY =     25, 42, 79 ; ! Dane tworzą mój numer indeksu;
   DEMAND =       20, 24, 97, 5;
   COST =         8,  8,  4,  8,
                  6,  8,  9,  7,
                  4,  5,  8,  9,
                  8,  8,  4,  8;
ENDDATA
END
```

Zadanie 2

Następną czynnością było skorygowanie parametrów wejściowych w taki sposób, by zadanie było problemem zbalansowanym. O takiej sytuacji mówimy, kiedy popyt odpowiada podaży, czyli ilość rzeczy przechowywanych w magazynie odpowiada zapotrzebowaniu na nie. Ograniczenia powinny być wtedy równościowe, co również zostało przeze mnie uwzględnione.

$$Popyt = podaż \quad \rightarrow \quad 25 + 42 + 79 = 20 + 24 + 97 + 5$$

```
! The demand constraints;
 @FOR( CUSTOMER( J): [DEM]
  @SUM( WAREHOUSE( I): VOLUME( I, J)) = ! Ograniczenie równościowe;
   DEMAND( J));

! The supply constraints;
 @FOR( WAREHOUSE( I): [SUP]
  @SUM( CUSTOMER( J): VOLUME( I, J)) = ! Ograniczenie równościowe;
   CAPACITY( I));


! Here are the parameters;
DATA:
   CAPACITY =     25, 42, 79 ; ! Dane tworzą mój numer indeksu;
   DEMAND =       20, 24, 97, 5;
   COST =         6,  2,  6,  7,
                  4,  9,  5,  3,
                  8,  8,  1,  5;
ENDDATA
END
```

Zadanie 3

Zmiana rodzaju zmiennej decyzyjnej na całkowitoliczbową, czyli taką, która w naszym przypadku jedynie ma sens (koszt wykorzystania pojazdu niezależnie od tego w jakim stopniu jest zapełniony jest taki sam).

```
! Zmienna decyzyjna całkowitoiczbowa;
@FOR(ROUTES:@GIN(VOLUME));
```

Zadanie 4

Funkcję kosztu zmieniam na funkcję zysku, dodając parametry kosztu transportu oraz zysku ze sprzedaży towarów. Funkcja teraz zamiast minimalizować, będzie maksymalizować.

```
DATA:
m=3;
n=4;
o=8;
ENDDATA
SETS:
   WAREHOUSE / 1..m/    : CAPACITY;
   CUSTOMER    / 1..n/  : DEMAND;
   TRUCKS /1..o/        : TRUCK_COST;      ! Dodanie pojazdów;
   ROUTES( WAREHOUSE, CUSTOMER) : STONKS, VOLUME;
ENDSETS

! Zaktualizowana funkcja celu;
 [OBJ] MAX = @SUM( ROUTES: (STONKS - TRUCK_COST) * VOLUME);
```

```
DATA:
   CAPACITY =     25, 42, 79 ; ! Dane tworzą mój numer indeksu;
   DEMAND =       20, 24, 97, 5;
   STONKS =       46,  12,  46,  97,   ! Ceny towarów;
                  64,  29,  65,  73,
                  38,  38,  61,  65;
   TRUCK_COST =   12, 13, 11, 8,       ! Koszty transportu;
                  25, 42, 79, 28;
ENDDATA
END
```

Podsumowanie

Implementacja całego programu wygląda w następujący sposób:

```
MODEL:
! A 3 Warehouse, 4 Customer
   Transportation Problem;

DATA:
m=3;
n=4;
o=8;
ENDDATA
SETS:
   WAREHOUSE / 1..m/    : CAPACITY;
   CUSTOMER   / 1..n/   : DEMAND;
   TRUCKS /1..o/        : TRUCK_COST;
   ROUTES( WAREHOUSE, CUSTOMER) : STONKS, VOLUME;
ENDSETS

! The objective;
 [OBJ] MAX = @SUM( ROUTES: (STONKS - TRUCK_COST) * VOLUME);

! The demand constraints;
 @FOR( CUSTOMER( J): [DEM]
  @SUM( WAREHOUSE( I): VOLUME( I, J)) = ! Ograniczenie równościowe;
   DEMAND( J));

! The supply constraints;
 @FOR( WAREHOUSE( I): [SUP]
  @SUM( CUSTOMER( J): VOLUME( I, J)) = ! Ograniczenie równościowe;
   CAPACITY( I));

! Zmienna decyzyjna całkowitoiczbowa;
@FOR(ROUTES:@GIN(VOLUME));

! Here are the parameters;
DATA:
   CAPACITY =     25, 42, 79 ; ! Dane tworzą mój numer indeksu;
   DEMAND =       20, 24, 97, 5;
   STONKS =       46,  12,  46,  97,
                  64,  29,  65,  73,
                  38,  38,  61,  65;
   TRUCK_COST =   12, 13, 11, 8,
                  25, 42, 79, 28;
ENDDATA
END
```

Otrzymany finalny wynik przedstawiam poniżej:

```
Global optimal solution found.
Objective value:                              6743.000
Objective bound:                              6743.000
Infeasibilities:                              0.000000
Extended solver steps:                               0
Total solver iterations:                             6
Elapsed runtime seconds:                          0.03

Model Class:                                      PILP

Total variables:                     12
Nonlinear variables:                  0
Integer variables:                   12

Total constraints:                    8
Nonlinear constraints:                0

Total nonzeros:                      36
Nonlinear nonzeros:                   0



                                Variable           Value        Reduced Cost
                                       M        3.000000            0.000000
                                       N        4.000000            0.000000
                                       O        8.000000            0.000000
                            CAPACITY( 1)        25.00000            0.000000
                            CAPACITY( 2)        42.00000            0.000000
                            CAPACITY( 3)        79.00000            0.000000
                              DEMAND( 1)        20.00000            0.000000
                              DEMAND( 2)        24.00000            0.000000
                              DEMAND( 3)        97.00000            0.000000
                              DEMAND( 4)        5.000000            0.000000
                          TRUCK_COST( 1)        12.00000            0.000000
                          TRUCK_COST( 2)        13.00000            0.000000
                          TRUCK_COST( 3)        11.00000            0.000000
                          TRUCK_COST( 4)        8.000000            0.000000
                          TRUCK_COST( 5)        25.00000            0.000000
                          TRUCK_COST( 6)        42.00000            0.000000
                          TRUCK_COST( 7)        79.00000            0.000000
                          TRUCK_COST( 8)        28.00000            0.000000
                           STONKS( 1, 1)        46.00000            0.000000
                           STONKS( 1, 2)        12.00000            0.000000
                           STONKS( 1, 3)        46.00000            0.000000
                           STONKS( 1, 4)        97.00000            0.000000
                           STONKS( 2, 1)        64.00000            0.000000
                           STONKS( 2, 2)        29.00000            0.000000
                           STONKS( 2, 3)        65.00000            0.000000
                           STONKS( 2, 4)        73.00000            0.000000
                           STONKS( 3, 1)        38.00000            0.000000
                           STONKS( 3, 2)        38.00000            0.000000
                           STONKS( 3, 3)        61.00000            0.000000
                           STONKS( 3, 4)        65.00000            0.000000
                           VOLUME( 1, 1)        20.00000           -34.00000
                           VOLUME( 1, 2)        0.000000            1.000000
                           VOLUME( 1, 3)        0.000000           -35.00000
                           VOLUME( 1, 4)        5.000000           -89.00000
                           VOLUME( 2, 1)        0.000000           -52.00000
                           VOLUME( 2, 2)        0.000000           -16.00000
                           VOLUME( 2, 3)        42.00000           -54.00000
                           VOLUME( 2, 4)        0.000000           -65.00000
                           VOLUME( 3, 1)        0.000000           -26.00000
                           VOLUME( 3, 2)        24.00000           -25.00000
                           VOLUME( 3, 3)        55.00000           -50.00000
                           VOLUME( 3, 4)        0.000000           -57.00000

                                     Row    Slack or Surplus      Dual Price
                                     OBJ        6743.000            1.000000
                                 DEM( 1)        0.000000            0.000000
                                 DEM( 2)        0.000000            0.000000
                                 DEM( 3)        0.000000            0.000000
                                 DEM( 4)        0.000000            0.000000
                                 SUP( 1)        0.000000            0.000000
                                 SUP( 2)        0.000000            0.000000
                                 SUP( 3)        0.000000            0.000000
```