بسم الله الرحمن الرحیم

برنامه ریزی حرکت قطارها

نرم افزار بهینه سازی LINGO

مدرس: دکتر مسعود یقینی

پائیز ۱۳۸۹

LINGO Software Overview

Optimization Software

- Spreadsheet (e.g, *MS Excel*)
- Solver (Optimizers) (e.g., *LINDO, CPLEX*)
- Modeling Language (e.g., *GAMS*)
- Combination of Modeling Language and Solver (e.g., *LINGO*)

LINGO Software Overview

- LINGO is a software tool designed to efficiently build and solve optimization models
  - Linear programs
  - Nonlinear programs
  - Integer programs

LINGO Software Overview

- It lets you express your problem in a natural manner which is very similar to standard mathematical notation.
- Instead of entering each term of each constraint explicitly, you can express a whole series of similar constraints in a single compact statement .
- In LINGO, you can use set-based models which are useful for problems involving large numbers of similar variables and constraints.

An Optimization model

- An Optimization model consists of 3 parts
  - Objective Function
    - A single formula that describes exactly what the model should optimize
  - Variables
    - Quantities that can be changed to produce the optimal value of the objective function
  - Constraints
    - formulas that define the limits on the values of the variables

Lingo Operators

- Addition: **+**
- Multiplication: ***
- Subtraction: **-**
- Division: **/**
- For exponents: X^n
- Equals: **=**
- Greater than or less than: **>** or **<**
  - Note: Lingo accepts '**<**' as being '**<=**'. It does not support strictly less than or greater than.

A Sample Model

- A cookie store can produce drop cookies and decorated cookies, which sell for $1 and $1.50 apiece, respectively.
- The two bakers each work 8 hours per day and can produce up to 400 drop cookies and 200 decorated cookies.
- It takes 1 minute to produce each drop cookie and 3 minutes to produce each decorated cookie.
- What combination of cookies produced will maximize the baker's profit?

A Sample Model

- Lingo Model:

```
! Cookie Store Model ;

MAX = 1*Drop + 1.5*Deco;

Drop <= 400;
Deco <= 200;

1/60*Drop + 3/60*Deco <=16;
```

Things to notice

- Comments in the model are initiated with an exclamation point (!) and appear in green text
- Enter the objective function by typing:

  MIN= …; or MAX= …;

- Start the constraints immediately after the objective, without:
  - ST
  - SUCH THAT
  - SUBJECT TO

Things to notice

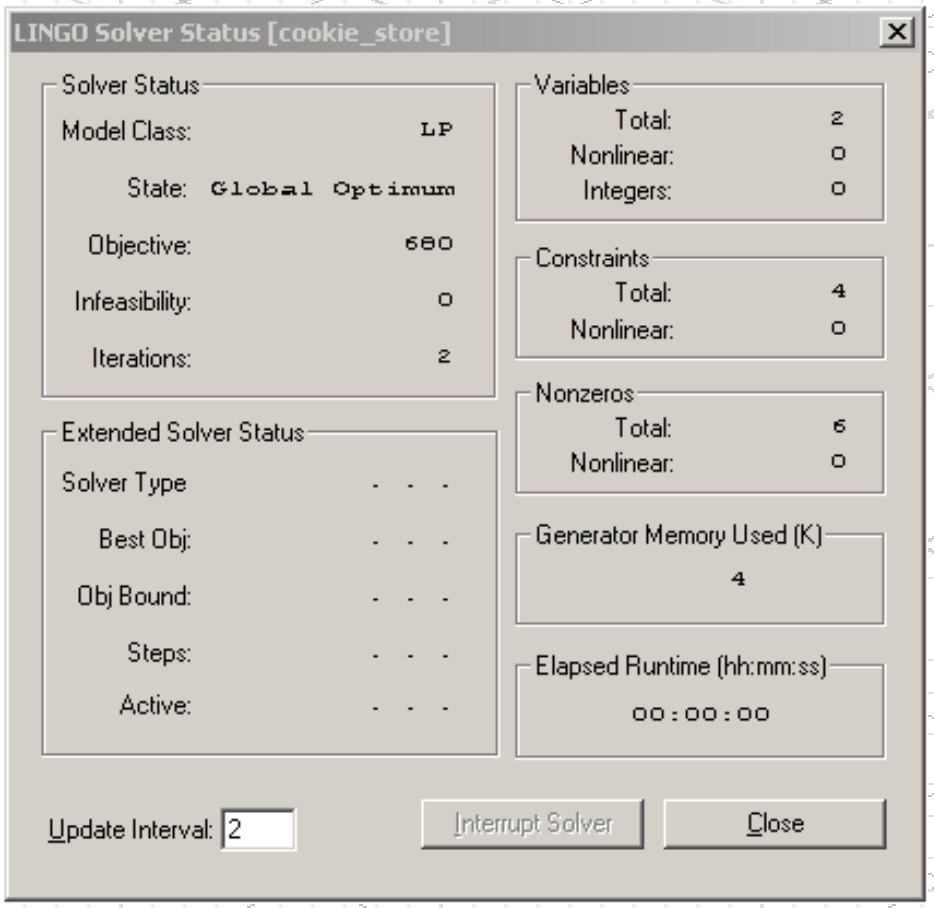
- LINGO specified operators and functions appear in blue text
- All other text is shown in black
- Each LINGO statement must end in a semi-colon (;)
- Variable names are not case-sensitive and must begin with a letter (A-Z)

Solving a LINGO Model

- Once the model has been entered into the *Model Window*, it can be solved by:
  - clicking the *Solve* button
  - Selecting *Solve* from the LINGO menu
  - Using the ctrl+s keyboard shortcut
- Errors (if any) will be reported

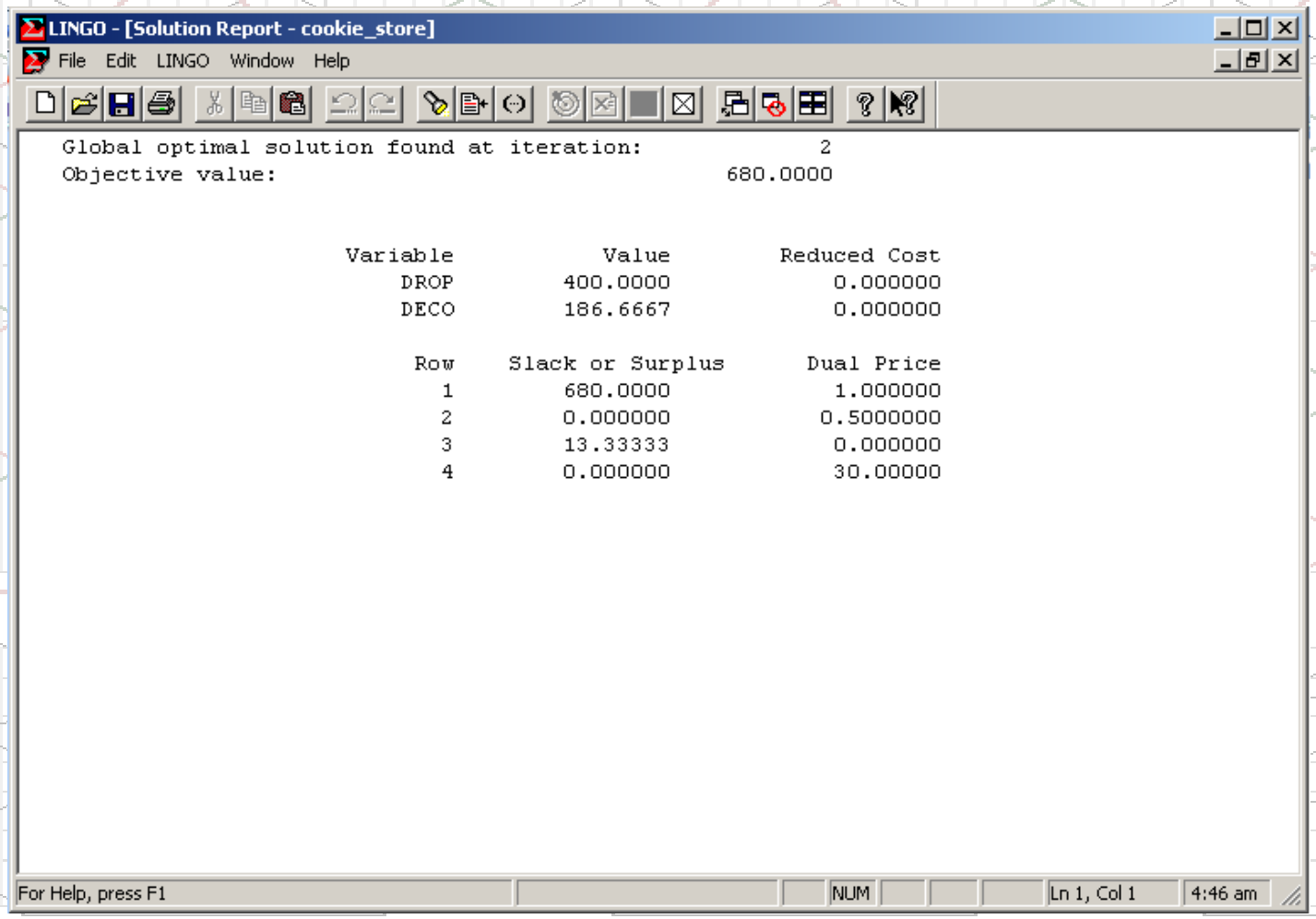
LINGO Solver Status Window

- If no errors are found, the LINGO Solver Status window appears



LINGO Solution Report Window

- Close the Solver Status window to see the Solution Report window



LINGO Solution Report Window

- Slack or Surplus
  - **Zero**: if a constraint is completely satisfied as an equality
  - **Positive**: shows how many more units of the variable could be added to the optimal solution before the constraint becomes an equality

Row	Slack or Surplus
1	680.0000
2	0.000000
3	13.33333
4	0.000000

LINGO Solution Report Window

- Reduced Cost
  - How much the objective function would degrade if one unit of a variable (not included in the current solution) were to be included

Variable	Value	Reduced Cost
DROP	400.0000	0.000000
DECO	186.6667	0.000000

LINGO Solution Report Window

- Dual Price
  - How much the objective function would improve if the constraining value is increased by one unit

Row	Slack or Surplus	Dual Price
1	680.0000	1.000000
2	0.000000	0.5000000
3	13.33333	0.000000
4	0.000000	30.00000

LINGO Modeling Language

Using Sets in LINGO

- LINGO allows you to group many instances of the same variable into sets
  - Example: If a model involved 27 delivery trucks, then these 27 trucks could be described more simply as a single set
- Sets may also include attributes for each member, such as the hauling capacity for each delivery truck

Using Sets

- SETS section must be defined before any of the set members are used in the model's constraints
- Primitive set example:

```
SETS:
   Trucks/TR1..TR27/:Capacity;
ENDSETS
```

Using Sets

- Derived set example:

```
SETS:
   Product/X Y/;
   Machine/L M/;
   Make(Product Machine)/X L, X M,Y M/;
ENDSETS
```

Set Looping Statement Examples

```
@FOR(Trucks(T): Capacity(T)<=3000);
```

- This @FOR statement sets the hauling capacity for all 27 delivery trucks in the Trucks set to at most 3000 pounds

```
TOTAL_HAUL=@SUM(Trucks(J): Capacity(J));
```

- This @SUM statement calculates the total hauling capacity from the individual trucks

LINGO Data Example

```
SETS:
    SET1 /A, B, C/: X, Y;
ENDSETS

DATA:
    X = 1, 2, 3;
    Y = 4, 5, 6;
ENDDATA
```

Variable Types in LINGO

- All variables in a LINGO model are considered to be non-negative and continuous unless otherwise specified
- LINGO's four variable domain functions can be used to override the default domain for given variables

Variable Types in LINGO

- @GIN – any positive integer value
- @BIN – a binary value (ie. 0 or 1)
- @FREE – any positive or negative real value
- @BND – any value within the specified bounds

Mathematical Functions

- @ABS(X) – returns the absolute value of X
- @SIGN(X) – returns -1 if X is negative and +1 if X is positive
- @EXP(X) –calculates e^X
- @LOG(X) – calculates the natural log of X
- @SIN(X) – returns the sine of X, where X is the angle in radians
- @COS(X) – returns the cosine of X
- @TAN(X) – returns the tangent of returns the tangent of X

Other Functions in LINGO

- LINGO also contains a lot of financial, probability, and import/export functions
- These are commonly used in more advanced models

Example I: Knapsack Problem

Knapsack Problem

```
SETS:
   ITEMS / ANT_REPEL, BEER, BLANKET,
      BRATWURST, BROWNIES, FRISBEE, SALAD,
WATERMELON/:
      INCLUDE, WEIGHT, RATING;
ENDSETS

DATA:
   WEIGHT, RATING =
      1, 2
      3, 9
      4, 3
      3, 8
      3, 10
      1, 6
      5, 4
      10, 10;

      KNAPSACK_CAPACITY = 15;
ENDDATA
```

Knapsack Problem

```
MAX = @SUM( ITEMS: RATING * INCLUDE);

@SUM( ITEMS: WEIGHT * INCLUDE) <=
KNAPSACK_CAPACITY;

@FOR( ITEMS: @BIN( INCLUDE));
```

Knapsack Problem

Global optimal solution found at iteration: 0
Objective value: 38.00000

Variable	Value	Reduced Cost
KNAPSACK_CAPACITY	15.00000	0.000000
INCLUDE(ANT_REPEL)	1.000000	-2.000000
INCLUDE(BEER)	1.000000	-9.000000
INCLUDE(BLANKET)	1.000000	-3.000000
INCLUDE(BRATWURST)	1.000000	-8.000000
INCLUDE(BROWNIES)	1.000000	-10.00000
INCLUDE(FRISBEE)	1.000000	-6.000000
INCLUDE(SALAD)	0.000000	-4.000000
INCLUDE(WATERMELON)	0.000000	-10.00000
WEIGHT(ANT_REPEL)	1.000000	0.000000
WEIGHT(BEER)	3.000000	0.000000
WEIGHT(BLANKET)	4.000000	0.000000
WEIGHT(BRATWURST)	3.000000	0.000000
WEIGHT(BROWNIES)	3.000000	0.000000
WEIGHT(FRISBEE)	1.000000	0.000000
WEIGHT(SALAD)	5.000000	0.000000
WEIGHT(WATERMELON)	10.00000	0.000000
RATING(ANT_REPEL)	2.000000	0.000000
RATING(BEER)	9.000000	0.000000
RATING(BLANKET)	3.000000	0.000000
RATING(BRATWURST)	8.000000	0.000000
RATING(BROWNIES)	10.00000	0.000000
RATING(FRISBEE)	6.000000	0.000000
RATING(SALAD)	4.000000	0.000000
RATING(WATERMELON)	10.00000	0.000000

Row	Slack or Surplus	Dual Price
1	38.00000	1.000000
2	0.000000	0.000000

Example II: Bisco Problem

Bisco Problem

- Bisco's new sugar-free, fat-free chocolate squares are so popular that the company cannot keep up with demand. Regional demands shown in the following table total 2000 cases per week, but Bisco can produce only 60% (1200 cases) of that number.

	NE	SE	MW	W
Demand	620	490	510	380
Profit	1.6	1.4	1.9	1.2

- The table also shows the different profit levels per case experienced in the regions due to competition and consumer tastes. Bisco wants to find a maximum profit plan that fulfils between 50% and 70% of each region's demand.

Problem Formulation

$$\max \sum_{i=1}^{4} p_i x_i$$

$$\sum_{i=1}^{4} x_i \leq 1200$$

$$l_i \leq x_i \leq u_i, i = 1,2,3,4$$

Problem Formulation

max 1.60 x1 + 1.40 x2 + 1.90 x3 + 1.20 x4

x1 + x2 + x3 + x4 <=1200

x1 >= 310

x1 <= 434

x2 >= 245

x2 <= 343

x3 >= 255

x3 <= 357

x4 >= 190

x4 <= 266

LINGO Model

```
MAX= 1.60 *x1 + 1.40* x2 + 1.90 *x3 + 1.20 *x4;

   x1 + x2 + x3 + x4 <=1200;

   x1 >= 310;
   x1 <= 434;
   x2 >= 245;
   x2 <= 343;
   x3 >= 255;
   x3 <= 357;
   x4 >= 190;
   x4 <= 266;
```

LINGO Solution

Objective value: 1902.100

Variable	Value	Reduced Cost
X1	408.0000	0.000000
X2	245.0000	0.000000
X3	357.0000	0.000000
X4	190.0000	0.000000

Row	Slack or Surplus	Dual Price
1	1902.100	1.000000
2	0.000000	1.600000
3	98.00000	0.000000
4	26.00000	0.000000
5	0.000000	-0.2000000
6	98.00000	0.000000
7	102.0000	0.000000
8	0.000000	0.3000000
9	0.000000	-0.4000000
10	76.00000	0.000000

LINGO Model 2

```
SETS:
    REGIONS / NE SE MW W/: LBOUND, UBOUND, PROFIT, CASES;
ENDSETS

DATA:
    LBOUND = 310 245 255 190;
    UBOUND = 434 343 357 266;
    PROFIT = 1.6 1.4 1.9 1.2;
ENDDATA

MAX = @SUM(REGIONS(I): PROFIT(I)*CASES(I));

@SUM(REGIONS(I): CASES(I)) <=1200;

@FOR(REGIONS(I): CASES(I) <= UBOUND(I));

@FOR(REGIONS(I): CASES(I) >= LBOUND(I));
```

LINGO Solution

Objective value: 1902.100

Variable	Value	Reduced Cost
LBOUND(NE)	310.0000	0.000000
LBOUND(SE)	245.0000	0.000000
LBOUND(MW)	255.0000	0.000000
LBOUND(W)	190.0000	0.000000
UBOUND(NE)	434.0000	0.000000
UBOUND(SE)	343.0000	0.000000
UBOUND(MW)	357.0000	0.000000
UBOUND(W)	266.0000	0.000000
PROFIT(NE)	1.600000	0.000000
PROFIT(SE)	1.400000	0.000000
PROFIT(MW)	1.900000	0.000000
PROFIT(W)	1.200000	0.000000
CASES(NE)	408.0000	0.000000
CASES(SE)	245.0000	0.000000
CASES(MW)	357.0000	0.000000
CASES(W)	190.0000	0.000000

Row	Slack or Surplus	Dual Price
1	1902.100	1.000000
2	0.000000	1.600000
3	26.00000	0.000000
4	98.00000	0.000000
5	0.000000	0.3000000
6	76.00000	0.000000
7	98.00000	0.000000
8	0.000000	-0.2000000
9	102.0000	0.000000
10	0.000000	-0.4000000

پایان