GLPK: Referência rápida

1 Saída

Cabeça

```
Problem:
Rows:       3
Columns:    2
Non-zeros:  4
Status:     OPTIMAL
Objective:  obj = 3200 (MAXimum)
```

Problem Nome do problema

Rows Número de linhas (restrições)

Columns Número de colunas (variáveis)

Non-zeros Número de coeficientes diferente de 0.

Status Resultado da otimização. Pode ser ótimo (OPTIMAL), não definido (UNDEFINED), ilimitado (UNBOUNDED), invíavel (INFEASIBLE).

Objective Valor da função objetivo.

Descrição das linhas

```
   No.   Row name   St   Activity
------ ------------ -- -------------
  Lower bound   Upper bound    Marginal
 ------------- ------------- -------------
```

No. Número da linha.

Row name Nome da linha, caso possui rótulo, `r.N` senão.

St Status da variável de folga associada com a linha. B = básica, N = nula.

Activity Valor das expressões variáveis da linha.

Lower bound Limite inferior, no caso de uma restrição $a_i x \geq b_i$ ou "=" no caso de $a_i x = b_i$.

Upper bound Limite superior, no casa de uma restrição $a_x x \leq b_i$ ou "=" no caso de $a_i x = b_i$.

Marginal Lucro (ou custo) marginal: Crescimento da função objetivo se o lado direito for aumentado uma unidade: $\delta z / \delta b_i$.

Descrição das colunas

```
   No. Column name  St   Activity
------ ------------ -- -------------
 Lower bound   Upper bound    Marginal
------------- ------------- -------------
```

No. Número da coluna.

Column name Nome da coluna (variável).

St Status da variável. B = básica, N = nula.

Activity Valor da variável.

Lower bound Limite inferior, ou nada caso não possui.

Upper bound Limite superior, ou nada caso não possui.

Marginal Lucro (ou custo) marginal da variável.

2 Formato CPLEX lp

⟨*specification*⟩ ::= ⟨*objective*⟩
⟨*restrictions*⟩?
⟨*bounds*⟩
⟨*general*⟩?
⟨*binary*⟩?
'`End`'

⟨*objective*⟩ ::= ⟨*goal*⟩ ⟨*name*⟩? ⟨*linear expression*⟩

⟨*goal*⟩ ::= '`MINIMIZE`' | '`MAXIMIZE`' | '`MIN`' | '`MAX`'

⟨*restrictions*⟩ ::= ('`SUBJECT TO`' | '`ST`') ⟨*restriction*⟩+

⟨*restriction*⟩ ::= ⟨*name*⟩? ⟨*linear expression*⟩ ⟨*cmp*⟩ ⟨*number*⟩

⟨*cmp*⟩ ::= '`<`' | '`<=`' | '`=`' | '`>`' | '`>=`'

⟨*linear expression*⟩ ::= ⟨*number*⟩ ⟨*variable*⟩ ('+' | '-') ⟨*number*⟩ ⟨*variable*⟩)*

⟨*bounds*⟩ ::= '`BOUNDS`' ⟨*bound*⟩+

⟨*bound*⟩ ::= ⟨*name*⟩? (⟨*limit*⟩ '`<=`' ⟨*variable*⟩ '`<=`' ⟨*limit*⟩
| ⟨*limit*⟩ '`<=`' ⟨*variable*⟩
| ⟨*variable*⟩ '`<=`' ⟨*limit*⟩
| ⟨*variable*⟩ '`=`' ⟨*number*⟩
| ⟨*variable*⟩ '`free`')

⟨*limit*⟩ ::= '`infinity`' | '`-infinity`' | ⟨*number*⟩

⟨*general*⟩ ::= '`GENERAL`' ⟨*variable*⟩+

⟨*binary*⟩ ::= '`BINARY`' ⟨*variable*⟩+

- Nenhuma variável pode comecar com "e".
- No lado da esquerda: somente variáveis. No lado da direita: somente constantes.

3 Formato MathProg

- Um modelo em AMPL consiste em
  - parâmetros,
  - variáveis,
  - restrições, e
  - objetivos
- AMPL usa *conjuntos* (ou arrays de múltiplas dimensões)

$$A : I \to D$$

que mapeiam um conjunto de índices $I = I_1 \times \cdots \times I_n$ para valores D.

3 Formato

- Parte do modelo

```
s1
...
sn
end;
```

com s_i sendo um comando ou uma declaração.

- Parte de dados

```
data
d1
...
dn
end;
```

com d_i sendo uma especificação de dados.

3 Tipo de dados

- Números: `2.0,-4`
- Strings: `'Comida'`
- Conjuntos: `{2,3,4}`

3 Expressões numéricas

- Operações básicas: +,−,∗,/,div,mod,less,∗∗
  Exemplo: x less y (= $\max\{x-y,0\}$)
- Funções: abs, ceil , floor ,exp
  Exemplo: abs(−3)
- Condicional: **if** x>y **then** x **else** y

3 Expressões sobre strings

- AMPL converte números automaticamente em strings
- Concatenação de strings: &
  Exemplo: x & ' unidades'

3 Expressões para conjuntos de índices

- Uma dimensão
  - t **in** S: variável "dummy" t, conjunto S
  - (t1 ,... tn) **in** S: para conjuntos de tuplos
  - S: sem nomear a variável
- Multiplas dimensões
  - {e1 ,..., en} com e_i uma dimensão (acima).
- Variáveis "dummy" servem para referenciar e modificar.
  Exemplo: (i−1) **in** S

3 Conjuntos

- Conjunto básico: {v1 ,..., vn}
- Valores: Tratados como conjuntos com conjunto de índices de dimensão 0
- Índices: [i1 ,..., **in**]
- Sequências: n1 ... n2 by d ou n1 ... n2
- Construção: setof I e: $\{e(i_1,\ldots,i_n) \mid (i_1,\ldots,i_n) \in I\}$
  Exemplo: `setof {j in A} abs(j)`

3 Operações de conjuntos

- X union Y: União $X \cup Y$
- X diff Y: Diferença $X \setminus Y$
- X symdiff Y: Diferença simétrica $(X \setminus Y) \cup (Y \setminus X)$
- X inter Y: Intersecção $X \cap Y$
- X cross Y: Produto cartesiano $X \times Y$

3 Expressões lógicas

- Interpretação de números: n vale "v", sse $n \neq 0$.
- Comparações simples: <,<=,= ou ==,>=,>,<> ou !=
- Pertinência: x **in** Y, x not **in** Y, x !**in** Y
- Subconjunto: X within Y, X !within Y, X not within Y
- Operadores lógicos: && ou and, || ou or, ! ou not
- Quantificação: com índices I, expressão booleana b

  forall I b: $\bigwedge_{(i_1,\ldots,i_n)\in I} b(i_1,\ldots,i_n)$

  exists I b: $\bigvee_{(i_1,\ldots,i_n)\in I} b(i_1,\ldots,i_n)$

3 Declarações: Conjuntos

set N I [dimen n] [within S] [default e1] [:= e2]

param N I [**in** S] [<=,>=,!=,... n] [default e1] [:= e2]

- Nome N
- Conjunto de índices I (opcional)
- Conjunto de valores S
- Valor default e_1
- Valor inicial e_2

3 Declarações: Restrições e objetivos

subject to N I : e1 = e2 | e1 <= e2, e1 >= e2

minimize [I] : e

maximize [I] : e

3 Comandos

- solve: Resolve o sistema.
- check [I] : b: Valida expressão booleana b, erro caso falso.
- display [I] : e1 ,... en: Imprime expressões $e_1,\ldots,e_n$.
- printf [I] : fmt,e1 ,...., en: Imprime expressões $e-1,\ldots,e_n$ usando formato fmt.
- for I : c, `for I : {c1 ... cn}`: Laços.

3 Dados: Conjuntos

set N r1 ,... rn

Com nome N e records $r_1,\ldots,r_n$, cada record

- um tuplo: $v_1,\ldots,v_n$
  Exemplo: 1 2, 1 3, 2 2, 2 7
- a definição de uma fatia $(v_1|*, v_2|*, \ldots, v_n|*)$: depois basta de listar os elementos com ∗.
  Exemplo: (1 *) 2 3, (2 *) 2 7
- uma matriz

```
  : c1 c2  ... cn :=
r1 a11 a12 ... a1n
r2 a21 a22 ... a2n
       ...
rm am1 am2 ... amn
```

  com aij "+"/"-" para inclusão/exclusão do par "ri cj" do conjunto.

3 Dados: Parâmetros

param N r1,...rn

Com nome N e records $r_1,\ldots,r_n$, cada record

- um valor $i_1,\ldots,i_n,v$
- a definição de uma fatia $[i_1|*, i_2|*, \ldots, i_n|*)$: depois basta definir índices com ∗.
- uma matriz

```
  : c1 c2  ... cn :=
r1 a11 a12 ... a1n
r2 a21 a22 ... a2n
       ...
rm am1 am2 ... amn
```

  com aij o valor do par "ri cj".
- uma tabela

```
param default v : s : p1 p2 ... pk :=
t11 t12 ... t1n  a11 a12 ... a1k
t21 t22 ... t2n  a21 a22 ... a2k
        ...
tm1 tm2     tmn  am1 am2 ... amk
```

  para definir simultaneamente o conjunto

```
set s := (t11 t12 ... t1n), ... , (tm1 tm2 ... tmn);
```

  e os parâmetros

```
param p1 default v := [t11 t12 ... t1n] a11, ..., [tm1 tm2 ... tmn] am1;
param p2 default v := [t11 t12 ... t1n] a12, ..., [tm1 tm2 ... tmn] am2;
...
param pk default v := [t11 t12 ... t1n] a1k, ..., [tm1 tm2 ... tmn] amk;
```