Otimização de Sistemas

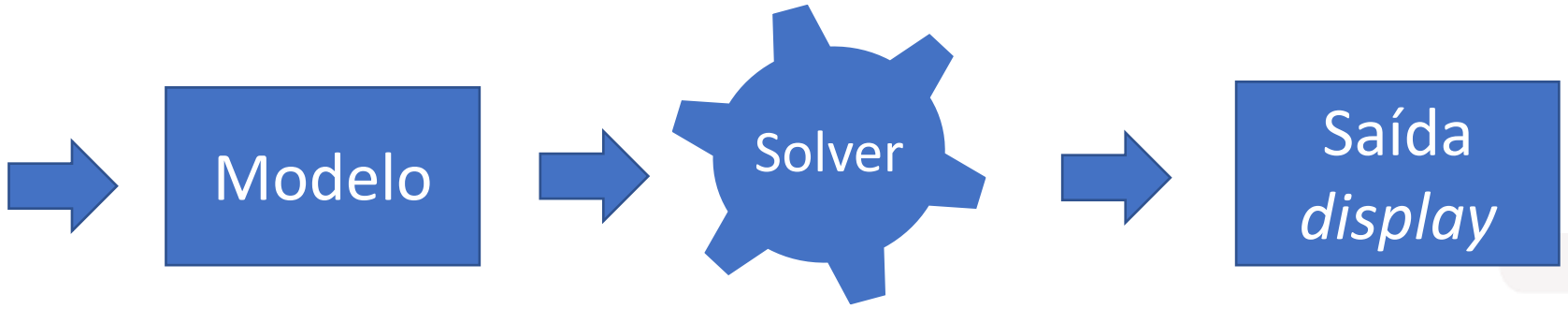
Prof. Sandro Jerônimo de Almeida, PhD.

Solvers e AMPL

Como resolver problemas complexos

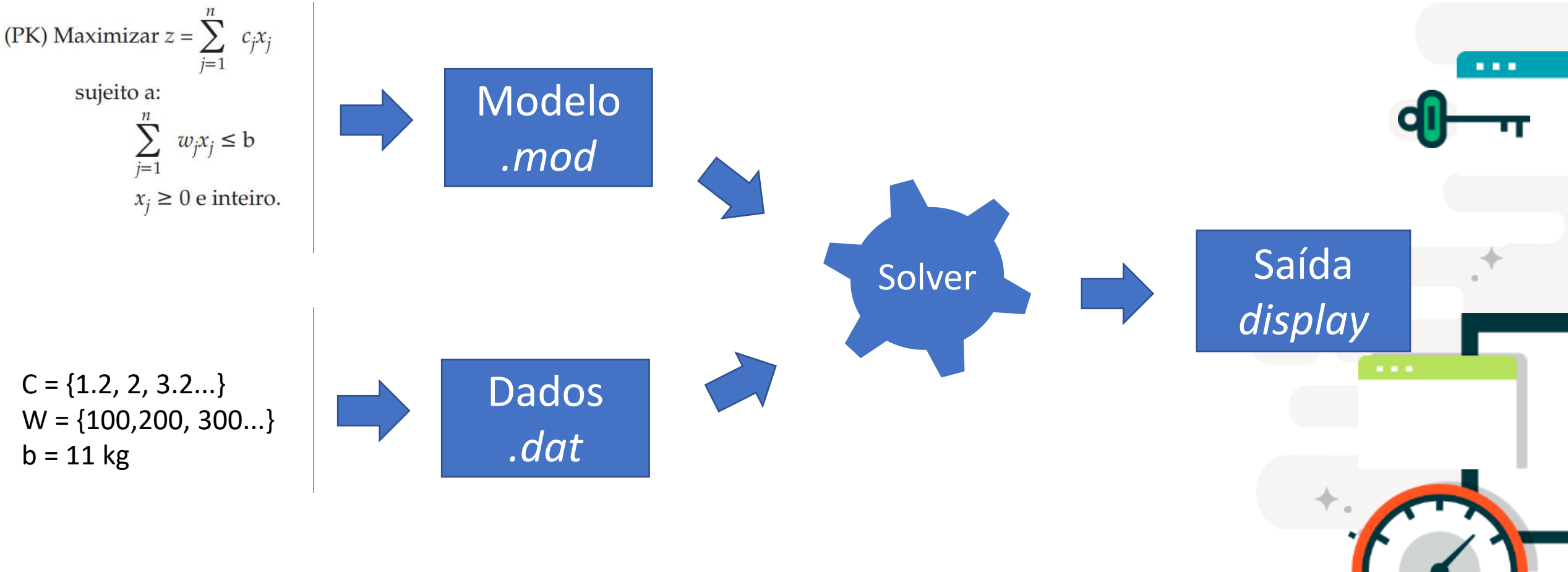
- Opção 1 – Modelo com Dados + *Solver*

$$\max \; F = 4x_1 - 5x_2$$
$$\text{suj. a:}$$
$$2x_1 + 3x_2 \leq 8$$
$$5x_1 + 2x_2 \leq 11$$
$$x_1, x_2 \geq 0$$



Como resolver problemas complexos

- Opção 2 – Modelo separado dos Dados + *Solver*



Linguagem de Modelagem

- AMPL (*A Mathematical Programming Language*) é uma linguagem de modelagem para descrever e reolver problemas complexos de otimizaçao
- Criado por Robert Fourer, David Gay, and Brian Kernighan at Bell Laboratories (1985).

Linguagem de Modelagem

- Livro: AMPL: A Modeling Language for Mathematical Programming by Robert Fourer, David M. Gay, and Brian W. Kernighan
- Disponível para download
- https://ampl.com/resources/the-ampl-book/

Solver

- Software que resolve problemas matemáticos
- Formatos: programa *stand-alone*, biblioteca, Serviço WEB, API
- Cada Solver possui uma forma de operação e formato de entrada e saída, que podem estar sujeitas a uma linguagem de modelagem específica

Solver

Alguns exemplos

- *General Problem Solver (GPS)* – software criado em 1957 por Herbert Simon, J. C. Shaw e Allen Newell
- Software *stand-alone*: Solver for Excel
- Bibliotecas: GLPK (*GNU Linear Programming Kit*)
- Serviço WEB: https://neos-server.org/

Solver

Alguns exemplos

- Serviço *(pago)*: IBM ILOG CPLEX Optimizer

  Possui ferramentas (CPLEX Otimization Studio)

  Possível integração com aplicações em C/C++, Java, C#, Python

Como utilizar o AMPL

- Baixe o software para seu computador
- https://ampl.com/try-ampl/download-a-free-demo/
- Versão gratuita (com limitações)
- Windows, Linux, macOS
- Pode-se usar *IDE (amplitude.exe)* ou linha de comando (sw.exe)

Exemplo 1

$$\max \; F = 4x_1 - 5x_2$$

suj. a:

$$2x_1 + 3x_2 \leq 8$$

$$5x_1 + 2x_2 \leq 11$$

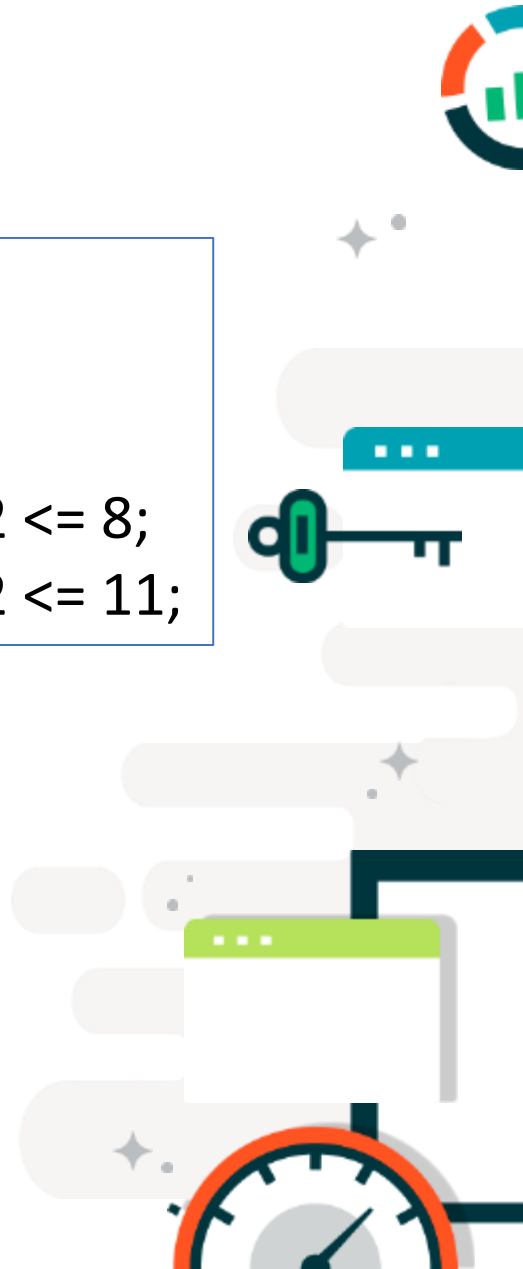
$$x_1, x_2 \geq 0$$

```
var X1 >=0;
var X2 >=0;
maximize F: 4 * X1 - 5 * X2;
subject to Restricao_1: 2 * X1 + 3 * X2 <= 8;
subject to Restricao_2: 5 * X1 + 2 * X2 <= 11;
```

exemplo1.mod

Comandos

ampl: model exemplo1.mod
ampl:solve;
ampl: display X1, X2;



Exemplo 2 – Problema da Mochila

Dados

Item	Valor ($)	Peso (Kg)
1	100	1,2
2	200	2
3	300	3,1
4	400	4,2
5	500	4,8
6	600	5,9
7	700	6,9

Capacidade da Mochila
K = 11 kg

Modelo

$$\text{(PK) Maximizar } z = \sum_{j=1}^{n} c_j x_j$$

sujeito a:

$$\sum_{j=1}^{n} w_j x_j \leq b$$

$$x_j \geq 0 \text{ e inteiro.}$$

Exemplo 2 – Problema da Mochila

Dados

```
data;
set N := Item1 Item2 Item3 Item4
         Item5 Item6 Item7;

param:    c    w :=
 Item1  100   1.2
 Item2  200   2
 Item3  300   3.1
 Item4  400   4.2
 Item5  500   4.8
 Item6  600   5.9
 Item7  700   6.9 ;
param b := 11; # (capacidade máxima)
```

mochila_exemplo.dat

Comandos

1) ampl: model mochila.mod
2) ampl: data mochila_exemplo.dat
3) ampl: option solver **cplex**;

Modelo

```
set N;

param b; # Capacidade Máxima da Mochila
param c {j in N}; # vetor de valor
param w {j in N}; # vetor de pesos

var X {j in N} integer >= 0; # variáveis de decisao (vetor X)

maximize Lucro: sum {j in N} c[j] * X[j];

subject to Capacidade: sum {j in N} w[j] * X[j] <= b;

# Lembre-se de usar um SOLVER que aceite Programação Linear Inteira
# ampl: option solver cplex;
```

mochila.mod

Comandos

4) ampl: solve
5) ampl: display X;

Problema da Mochila Múltipla Binária

Exercício 1: modele e resolva usando AMPL

- O problema da mochila múltipla (PMM) consiste em colocar *n* itens em *m* mochilas. Cada mochila *i* possui uma capacidade de peso b*i* que não pode ser ultrapassada. Cada item *j* possuem um peso w_j e um valor c_j . O objetivo é colocar (escolher) itens nas mochilas de forma a maximizar o valor a ser carregado na mochila. Considere o seguinte cenário ilustrativo:

Mochilas disponíveis:

Mochila	Capacidade (Kg)
A	12
B	10
C	7
D	3
E	9

Itens disponíveis em estoque:

Itens	Valor (R$)	Peso (Kg)
1	70	6
2	30	3
3	28	3
4	40	4
5	13	2
6	12	2
7	100	12

Problema da Mochila Múltipla Binária

- O problema da mochila múltipla (PMM) consiste em colocar *n* itens em *m* mochilas. Cada mochila *i* possui uma capacidade de peso b*i* que não pode ser ultrapassada. Cada item *j* possuem um peso w_j e um valor c_j . O objetivo é colocar (escolher) itens nas mochilas de forma a maximizar o valor a ser carregado na mochila. Considere o seguinte cenário ilustrativo:

Modelo de Programação Matemática

$$\text{(PKM) Maximizar } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_j x_{ij}$$

sujeito a:

$$\sum_{j=1}^{n} w_j x_{ij} \leq b_i \qquad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \qquad j = 1, \ldots, n$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \ldots, m; j = 1, \ldots, n.$$

Problema da Mochila Múltipla Binária

```
set M;
set N;

param b {i in M}; # Capacidade Máxima das Mochilas
param c {j in N}; # vetor de valor
param w {j in N}; # vetor de pesos

#variável binária
var X {i in M, j in N} binary; # variáveis de decisao (vetor X)

maximize Lucro: sum {i in M, j in N} c[j] * X[i,j];

subject to Limites_Por_Mochila {i in M}:
sum {j in N} w[j] * X[i,j] <= b[i];

#específico para a mochila binária: itens são únicos
subject to Item_Unico {j in N}:
sum {i in M} X[i,j] <= 1;

# Lembre-se de usar um SOLVER que aceite Prog. Linear Inteira
# ampl: option solver cplex;
```

mochila_multipla.mod

```
data;

set M := MochilaA MochilaB MochilaC MochilaD MochilaE;
set N := Item1 Item2 Item3 Item4 Item5 Item6 Item7;

# (capacidade máxima das mochilas)
param:     b  :=
  MochilaA  12
  MochilaB  10
  MochilaC  7
  MochilaD  3
  MochilaE  9;

param:     c     w :=
  Item1   70    6
  Item2   30    3
  Item3   28    3
  Item4   40    4
  Item5   13    2
  Item6   12    2
  Item7   100   12;
```

mochila_multipla.dat

Problema da Mochila Múltipla Binária

Resultados via Command line

```
ampl: model mochila_multipla.mod
ampl: data mochila_multipla.dat
ampl: option solver cplex;
ampl: solve;
CPLEX 12.9.0.0: optimal integer solution; objective 293
9 MIP simplex iterations
0 branch-and-bound nodes
ampl: display X;
X [*,*] (tr)
:     MochilaA MochilaB MochilaC MochilaD MochilaE    :=
Item1     0        1        0        0        0
Item2     0        0        1        0        0
Item3     0        1        0        0        0
Item4     0        0        0        0        1
Item5     0        0        1        0        0
Item6     0        0        1        0        0
Item7     1        0        0        0        0
;
```

```
data;

set M := MochilaA MochilaB MochilaC MochilaD MochilaE;
set N := Item1 Item2 Item3 Item4 Item5 Item6 Item7;

# (capacidade máxima das mochilas)
param:     b  :=
  MochilaA  12
  MochilaB  10
  MochilaC  7
  MochilaD  3
  MochilaE  9;

param:     c     w :=
  Item1  70   6
  Item2  30   3
  Item3  28   3
  Item4  40   4
  Item5  13   2
  Item6  12   2
  Item7  100  12;
```

mochila_multipla.dat

Problema da Mochila Múltipla Com múltiplos itens

Exercício 2: modele e resolva usando AMPL

- O problema da mochila múltipla (PMM) consiste em colocar *n* itens em *m* mochilas. Cada mochila *i* possui uma capacidade de peso b*i* que não pode ser ultrapassada. Cada item *j* possuem um peso w_j , um valor c_j e a quantidade em estoque l_j. O objetivo é colocar (escolher) itens nas mochilas de forma a maximizar o valor a ser carregado na mochila. Considere o seguinte cenário ilustrativo:

Mochilas disponíveis:

Mochila	Capacidade (Kg)
A	12
B	10
C	7
D	3
E	9

Itens disponíveis em estoque:

Itens	Valor (R$)	Peso (Kg)	Quantidade disponível
1	70	6	4
2	30	3	3
3	28	3	6
4	40	4	7
5	13	2	2
6	12	2	5
7	100	12	1