Tutorial AMPL

Part I

Eduardo Camponogara

Department of Automation and Systems Engineering
Federal University of Santa Catarina

October 2016

Summary

Algebraic Modeling Languages

Definition:

- They are high-level programming languages with which one can specify and solve optimization problems.

Properties:

- These languages do not solve the problems directly, but rather invoke algorithms (*solvers*) to obtain a solution.
- Some languages have the advantage of having a syntax similar to the mathematical notation used to describe optimization problems.

Examples:

- GAMS, AMPL, Pyomo, CMPL, MPL, PuLP, . . .

Algebraic Modeling Languages

Definition:

- They are high-level programming languages with which one can specify and solve optimization problems.

Properties:

- These languages do not solve the problems directly, but rather invoke algorithms (*solvers*) to obtain a solution.
- Some languages have the advantage of having a syntax similar to the mathematical notation used to describe optimization problems.

Examples:

- GAMS, AMPL, Pyomo, CMPL, MPL, PuLP, . . .

Algebraic Modeling Languages

Definition:

- They are high-level programming languages with which one can specify and solve optimization problems.

Properties:

- These languages do not solve the problems directly, but rather invoke algorithms (*solvers*) to obtain a solution.
- Some languages have the advantage of having a syntax similar to the mathematical notation used to describe optimization problems.

Examples:

- GAMS, AMPL, Pyomo, CMPL, MPL, PuLP, ...

Algebraic Modeling Languages

Definition:

- They are high-level programming languages with which one can specify and solve optimization problems.

Properties:

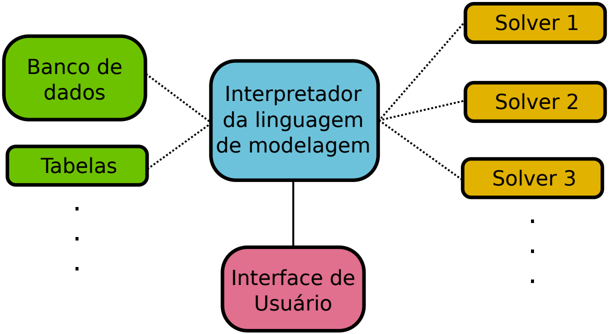
- These languages do not solve the problems directly, but rather invoke algorithms (*solvers*) to obtain a solution.
- Some languages have the advantage of having a syntax similar to the mathematical notation used to describe optimization problems.

Examples:

- GAMS, AMPL, Pyomo, CMPL, MPL, PuLP, ...

Algebraic Modeling Languages

Basic Structure:



AMPL: A Mathematical Programming Language

- The user interface is a terminal for input of command lines. It is reached by running the command **ampl.exe**.
- The files contain the model, data, configurations, and other programming structures that can be edited by a regular editor.
- There exists a developing interface, AMPL IDE, which facilitates the editing and execution of AMPL commands.

AMPL: A Mathematical Programming Language

- The user interface is a terminal for input of command lines. It is reached by running the command **ampl.exe**.
- The files contain the model, data, configurations, and other programming structures that can be edited by a regular editor.
- There exists a developing interface, AMPL IDE, which facilitates the editing and execution of AMPL commands.

AMPL: A Mathematical Programming Language

- The user interface is a terminal for input of command lines. It is reached by running the command **ampl.exe**.
- The files contain the model, data, configurations, and other programming structures that can be edited by a regular editor.
- There exists a developing interface, AMPL IDE, which facilitates the editing and execution of AMPL commands.

AMPL

The basic files are:

.mod - used to declare the elements of the models: variables, objective, constraints and data (sets and parameters).

.dat - used to define the data for the model.

.run - where variable configurations are defined, "scripting constructs," such as reading tables or data bases.

AMPL

The basic files are:

.mod - used to declare the elements of the models: variables, objective, constraints and data (sets and parameters).

.dat - used to define the data for the model.

.run - where variable configurations are defined, "scripting constructs," such as reading tables or data bases.

AMPL

The basic files are:

.mod - used to declare the elements of the models: variables, objective, constraints and data (sets and parameters).

.dat - used to define the data for the model.

.run - where variable configurations are defined, "scripting constructs," such as reading tables or data bases.

AMPL

Sintaxe:

- Variable: **var** VariableName;
- Objective: **minimize** or **maximize** ObjectiveName: . . . ;
- Constraint: **subject to** RestrictionName: . . . ;

Remarks:

- Every line instruction must be terminated with "**;**".
- Line comments are preceded by the symbol "#".
- Block commands are enclosed by the symbols "//*. . . *//".
- AMPL is "case-sensitive".
- Variable names must be unique.

AMPL

Sintaxe:

- Variable: **var** VariableName;
- Objective: **minimize** or **maximize** ObjectiveName: . . . ;
- Constraint: **subject to** RestrictionName: . . . ;

Remarks:

- Every line instruction must be terminated with "**;**".
- Line comments are preceded by the symbol "**#**".
- Block commands are enclosed by the symbols "**//*. . . *//**".
- AMPL is "case-sensitive".
- Variable names must be unique.

AMPL – Example 1

Problem Description:

- Paint Deals produces two colors of paint, **blue** and **black**.
- Blue paint is sold for US$10 per liter, while black paint is sold for US$15 per liter.
- The company owns a process plant which can produce one color paint at a time.
- However, blue paint is produced at a rate of 40 liters per hour, while the production rate for black paint is 30 liters per hour.
- Besides, the marketing department estimates that at most 860 liters of black paint and 1000 liters of blue paint can be sold in the market.
- During a week, the plant can operate for 40 hours and the paint can be stored for the following week.
- Determine how many liters of each paint should be produced to maximize week revenue.

AMPL – Example 1

Problem Description:

- Paint Deals produces two colors of paint, **blue** and **black**.
- Blue paint is sold for US$10 per liter, while black paint is sold for US$15 per liter.
- The company owns a process plant which can produce one color paint at a time.
- However, blue paint is produced at a rate of 40 liters per hour, while the production rate for black paint is 30 liters per hour.
- Besides, the marketing department estimates that at most 860 liters of black paint and 1000 liters of blue paint can be sold in the market.
- During a week, the plant can operate for 40 hours and the paint can be stored for the following week.
- Determine how many liters of each paint should be produced to maximize week revenue.

AMPL – Example 1

Problem Description:

- Paint Deals produces two colors of paint, **blue** and **black**.
- Blue paint is sold for US$10 per liter, while black paint is sold for US$15 per liter.
- The company owns a process plant which can produce one color paint at a time.
- However, blue paint is produced at a rate of 40 liters per hour, while the production rate for black paint is 30 liters per hour.
- Besides, the marketing department estimates that at most 860 liters of black paint and 1000 liters of blue paint can be sold in the market.
- During a week, the plant can operate for 40 hours and the paint can be stored for the following week.
- Determine how many liters of each paint should be produced to maximize week revenue.

AMPL – Example 1

Problem Description:

- Paint Deals produces two colors of paint, **blue** and **black**.
- Blue paint is sold for US$10 per liter, while black paint is sold for US$15 per liter.
- The company owns a process plant which can produce one color paint at a time.
- However, blue paint is produced at a rate of 40 liters per hour, while the production rate for black paint is 30 liters per hour.
- Besides, the marketing department estimates that at most 860 liters of black paint and 1000 liters of blue paint can be sold in the market.
- During a week, the plant can operate for 40 hours and the paint can be stored for the following week.
- Determine how many liters of each paint should be produced to maximize week revenue.

AMPL – Example 1

Mathematical Programming Model:

$$\max \quad 10 \cdot BluePaint + 15 \cdot BlackPaint \tag{1}$$

$$\text{s.t.}: (\frac{1}{40}) \cdot BluePaint + (\frac{1}{30}) \cdot BlackPaint \leq 40 \tag{2}$$

$$0 \leq BluePaint \leq 1000 \tag{3}$$

$$0 \leq BlackPaint \leq 860 \tag{4}$$

AMPL – Example 1

Basic Structure of ".mod" file:

```
# Part 1: Variable Declaration (var, set, param, etc)
var BluePaint;
var BlackPaint;
# Part 2: Objective Function
maximize Revenue: 10*BluePaint + 15*BlackPaint;
# Part 3: Constraints
subject to Time: (1/40)*BluePaint + (1/30)*BlackPaint
<= 40;
subject to BlueLimit: 0 <= BluePaint <= 1000;
subject to BlackLimit: 0 <= BlackPaint <= 860;
```

AMPL – Example 1

Basic Structure of ".mod" file:

```
# Part 1: Variable Declaration (var, set, param, etc)
var BluePaint;
var BlackPaint;
# Part 2: Objective Function
maximize Revenue: 10*BluePaint + 15*BlackPaint;
# Part 3: Constraints
subject to Time: (1/40)*BluePaint + (1/30)*BlackPaint
<= 40;
subject to BlueLimit: 0 <= BluePaint <= 1000;
subject to BlackLimit: 0 <= BlackPaint <= 860;
```

AMPL – Example 1

Basic Structure of ".mod" file:

```
# Part 1: Variable Declaration (var, set, param, etc)
var BluePaint;
var BlackPaint;
# Part 2: Objective Function
maximize Revenue: 10*BluePaint + 15*BlackPaint;
# Part 3: Constraints
subject to Time: (1/40)*BluePaint + (1/30)*BlackPaint
<= 40;
subject to BlueLimit: 0 <= BluePaint <= 1000;
subject to BlackLimit: 0 <= BlackPaint <= 860;
```

AMPL – Example 1

Basic Structure of ".run" file:

```
# Reset Memory
reset ;
# Load Model
model example1.mod;
# Change Configuration (optional)
option solver cplex;
# Solve Problem
solve;
# Show Results
display BluePaint, BlackPaint;
display Revenue;
expand Time;
```

AMPL – Example 1

Basic Structure of ".run" file:

```
# Reset Memory
reset ;
# Load Model
model example1.mod;
# Change Configuration (optional)
option solver cplex;
# Solve Problem
solve;
# Show Results
display BluePaint, BlackPaint;
display Revenue;
expand Time;
```

AMPL – Example 1

Basic Structure of ".run" file:

```
# Reset Memory
reset ;
# Load Model
model example1.mod;
# Change Configuration (optional)
option solver cplex;
# Solve Problem
solve;
# Show Results
display BluePaint, BlackPaint;
display Revenue;
expand Time;
```

AMPL – Example 1

Running file example1.run:

```
ampl: include example1.run;
CPLEX 12.2.0.0: No LP presolve or aggregator reductions.
optimal solution; objective 17433.33333
1 dual simplex iterations (0 in phase I)
BluePaint = 453.333
BlackPaint = 860

Revenue = 17433.3

subject to Time:
	0.025*BluePaint + 0.0333333*BlackPaint <= 40;
ampl:
```

AMPL – Example 2

Problem Description - MPC:

- AMPL is used to model a model-based predictive control problem (MPC).
- We consider a linear discrete-time system with state x_k and an input variable u_k.
- The prediction horizon is $N = 4$, with system dynamics given by $a = 0.8104$ and $b = 0.2076$, $x_{init} = 0.4884$.
- The cost imposes a quadratic penalty on state and control signals, leading to an optimization problem:

$$\min \sum_{k=1}^{N} (x_{k+1})^2 + \sum_{k=1}^{N} (u_k)^2$$

$$\text{subject to: } x_{k+1} = ax_k + bu_k, \quad k = 1, \ldots, N$$

$$x_1 = x_{init}$$

AMPL – Example 2

Problem Description - MPC:

- AMPL is used to model a model-based predictive control problem (MPC).
- We consider a linear discrete-time system with state x_k and an input variable u_k.
- The prediction horizon is $N = 4$, with system dynamics given by $a = 0.8104$ and $b = 0.2076$, $x_{init} = 0.4884$.
- The cost imposes a quadratic penalty on state and control signals, leading to an optimization problem:

$$\min \sum_{k=1}^{N} (x_{k+1})^2 + \sum_{k=1}^{N} (u_k)^2$$

$$\text{subject to: } x_{k+1} = ax_k + bu_k, \quad k = 1, \ldots, N$$

$$x_1 = x_{init}$$

AMPL – Example 2

Problem Description - MPC:

- AMPL is used to model a model-based predictive control problem (MPC).
- We consider a linear discrete-time system with state x_k and an input variable u_k.
- The prediction horizon is $N = 4$, with system dynamics given by $a = 0.8104$ and $b = 0.2076$, $x_{init} = 0.4884$.
- The cost imposes a quadratic penalty on state and control signals, leading to an optimization problem:

$$\min \sum_{k=1}^{N} (x_{k+1})^2 + \sum_{k=1}^{N} (u_k)^2$$

$$\text{subject to: } x_{k+1} = a x_k + b u_k, \quad k = 1, \ldots, N$$

$$x_1 = x_{init}$$

AMPL – Example 2

- Complete the AMPL model in **example2.mod**.

```
# Parte 1: Variable Declaration (var, set, param, etc)
param a = 0.8104;
param b = 0.2076;
param N = 4;
param Xinit = 0.4884;
var x{k in 1..N+1};
var u{k in 1..N};
# Part 2: Objective Function
minimize Cost: sum{k in 1..N}(x[k+1]^2) + ...;
```

- Create a file **example2.run** and obtain the values u_k

AMPL – Example 2

example2.mod:

```
# Parte 1: Variable Declaration (var, set, param, etc)
param a = 0.8104;
param b = 0.2076;
param N = 4;
param Xinit = 0.4884;
var x{k in 1..N+1};
var u{k in 1..N};
# Part 2: Objective Function
minimize Cost: sum{k in 1..N}(x[k+1]^2) + sum{k in
1..N}(u[k]^2);
# Part 3: Constraints
subject to sysmodel{k in 1..N}: x[k+1]=a*x[k]+b*u[k];
subject to initial_condition: x[1]= Xinit;
```

Fundamentals

- Thank you for attending this lecture!!!