Introduction to AMPL

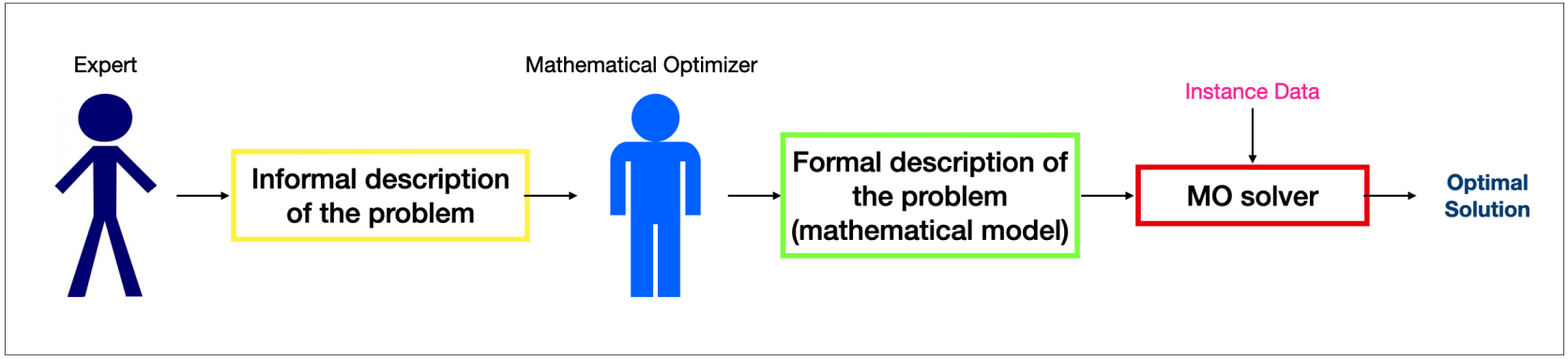
Claudia D'Ambrosio
dambrosio@lix.polytechnique.fr

LIX, CNRS & École Polytechnique
Institut Polytechnique de Paris
France

CIEN E4011 – March 2024

Reminder

Mathematical Optimisation is a knowledge-based approach



Algebraic modeling languages

- AMPL (and others): **algebraic modeling languages**

Algebraic modeling languages

- AMPL (and others): **algebraic modeling languages**
- **Syntax**: very similar to mathematical notation

Algebraic modeling languages

- AMPL (and others): **algebraic modeling languages**
- **Syntax**: very similar to mathematical notation
- **Model** mathematical optimization problems

Algebraic modeling languages

- AMPL (and others): **algebraic modeling languages**
- **Syntax**: very similar to mathematical notation
- **Model** mathematical optimization problems
- Develop **algorithms** based on mathematical optimization

Algebraic modeling languages

- AMPL (and others): **algebraic modeling languages**
- **Syntax**: very similar to mathematical notation
- **Model** mathematical optimization problems
- Develop **algorithms** based on mathematical optimization
- Linked to solvers for LP/MILP/MINLP problems

Algebraic modeling languages

- AMPL (and others): **algebraic modeling languages**
- **Syntax**: very similar to mathematical notation
- **Model** mathematical optimization problems
- Develop **algorithms** based on mathematical optimization
- Linked to solvers for LP/MILP/MINLP problems
- **Download**:
  `https://www.lix.polytechnique.fr/~dambrosio/teaching/`

Algebraic modeling languages

- AMPL (and others): **algebraic modeling languages**
- **Syntax**: very similar to mathematical notation
- **Model** mathematical optimization problems
- Develop **algorithms** based on mathematical optimization
- Linked to solvers for LP/MILP/MINLP problems
- **Download**:
  `https://www.lix.polytechnique.fr/~dambrosio/teaching/`
- **AMPL book**:
  `www.ampl.com/BOOK/download.html`

Algebraic modeling languages

- AMPL (and others): **algebraic modeling languages**
- **Syntax**: very similar to mathematical notation
- **Model** mathematical optimization problems
- Develop **algorithms** based on mathematical optimization
- Linked to solvers for LP/MILP/MINLP problems
- **Download**:
  `https://www.lix.polytechnique.fr/~dambrosio/teaching/`
- **AMPL book**:
  `www.ampl.com/BOOK/download.html`
- **Quick-start guide**:
  `https://www.lix.polytechnique.fr/~dambrosio/teaching/ampl-quick-start-guide_dambrosio.pdf`

AMPL files

- We will use the Python interface, called **amplpy**

AMPL files

- We will use the Python interface, called **amplpy**
  - a **model file** (extension .mod): contains the mathematical formulation of the problem (AMPL syntax).

AMPL files

- We will use the Python interface, called **amplpy**
  - a **model file** (extension .mod): contains the mathematical formulation of the problem (AMPL syntax).
  - **input and output data**: matches Python lists, sets, dictionaries, **pandas, and numpy** objects.

AMPL files

- We will use the Python interface, called **amplpy**
  - a **model file** (extension .mod): contains the mathematical formulation of the problem (AMPL syntax).
  - **input and output data**: matches Python lists, sets, dictionaries, **pandas, and numpy** objects.
  - `https://amplpy.ampl.com/`

File .mod

- parameters, lines starting with the keyword `param`

File .mod

- parameters, lines starting with the keyword `param`
- sets, lines starting with the keyword `set`

File .mod

- parameters, lines starting with the keyword `param`
- sets, lines starting with the keyword `set`
- decision variables, lines starting with the keyword `var`

File .mod

- parameters, lines starting with the keyword `param`
- sets, lines starting with the keyword `set`
- decision variables, lines starting with the keyword `var`
- objective function(s), lines starting with the keyword `minimize` or `maximize`

File .mod

- parameters, lines starting with the keyword `param`
- sets, lines starting with the keyword `set`
- decision variables, lines starting with the keyword `var`
- objective function(s), lines starting with the keyword `minimize` or `maximize`
- constraints, lines starting with the keyword `subject to`

File .mod

```
param n > 0;
```

File .mod

```
param n > 0;
```

```
param w{1..n} > 0;
```

File .mod

```
param n > 0;
```

```
param w{1..n} > 0;
```

```
param n > 0;
param m > 0;
param a{1..n, 1..m};
```

File .mod

```
set N := 1..n;
```

File .mod

```
set N := 1..n;
```

```
param w{N} > 0;
```

File .mod

```
set N := 1..n;
```

```
param w{N} > 0;
```

```
param w{N} > 0;
param p{j in N} <= 10*w[j];
```

File .mod

Decision variables:

```
var x{j in 1..n} >= 0, <= 1, binary;
```

File .mod

Decision variables:

```
var x{j in 1..n} >= 0, <= 1, binary;
```

Objective function:

```
maximize total_profit:
  sum{j in N} p[j]*x[j];
```

File .mod

```
subject to capacity_constraint:
  sum{j in N} w[j]*x[j] <= c;
```

File .mod

```
subject to capacity_constraint:
  sum{j in N} w[j]*x[j] <= c;
```

```
subject to random_constraint{j in 2..n}:
  w[j]*x[j] - w[j-1]*x[j-1]<= 1;
```

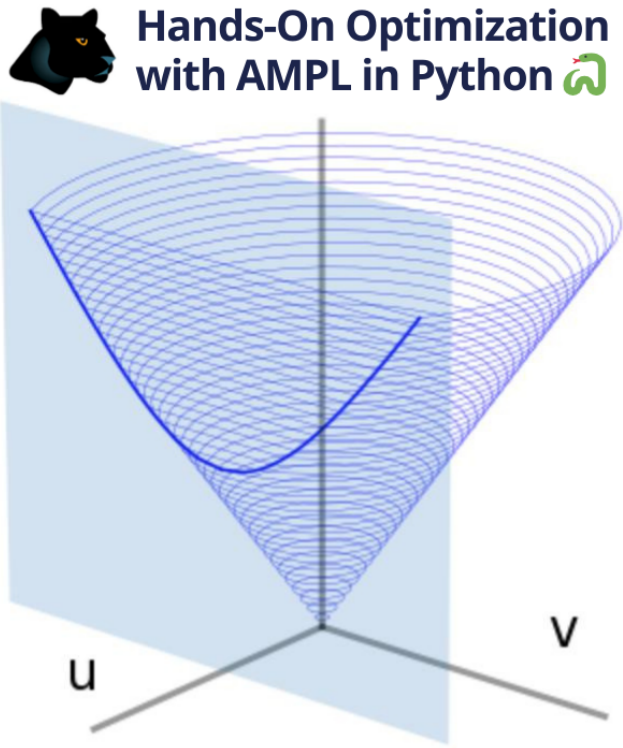
The liquid transportation problem: File .mod

```
set L ordered;
param U {j in L} > 0; # max avail per liquid
param p {j in L} > 0; # unit profit per liquid
param w {j in L} > 0; # unit weight per liquid
param W > 0;  # truck max capacity

# decision variables
var x {j in L} >= 0, <= U[j];

# objective function
maximize Total_Profit:
  sum {j in L} p[j]*x[j];

# constraints
subject to max_weight_constraint:
  sum{j in L} w[j]*x[j] <= W;
```

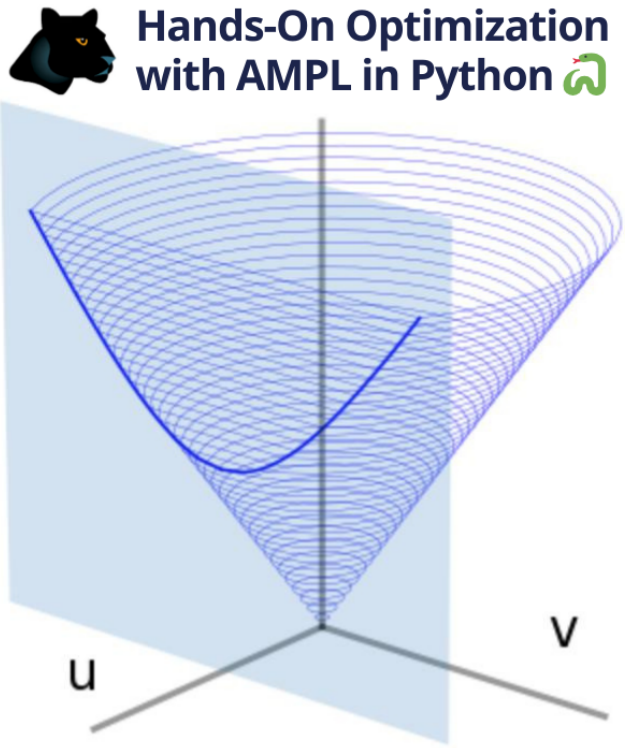
References

- Manual: `www.ampl.com/BOOK/download.html`
- How to install AMPL: jupyter notebook AMPLinstallation.ipynb
- Modeling languages like **ampl**: `ampl.com` or **gams**: `www.gams.com` or **jump** `https://jump.dev/JuMP.jl/` or **pyomo** `https://www.pyomo.org/`
- Open source solvers like **scip**: `scip.zib.de`
- **NEOS Server**, State-of-the-Art Solvers for Numerical Optimization: `www.neos-server.org/neos/`

Amplpy

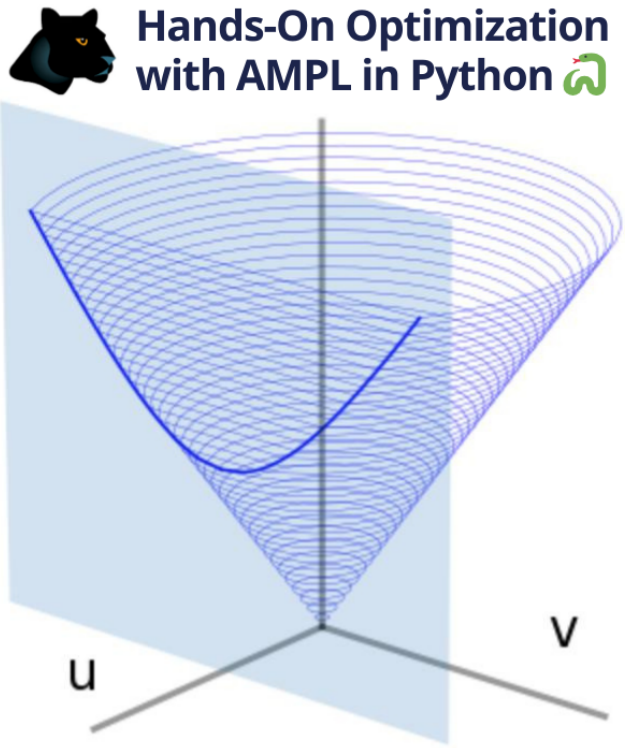


Amplpy



https://amplpy.ampl.com/en/latest/quick-start.html

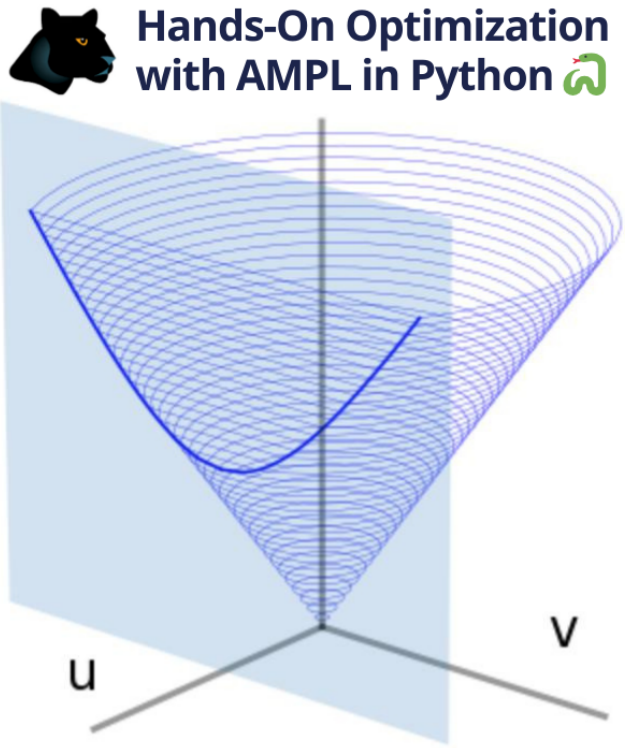
Amplpy



`https://amplpy.ampl.com/en/latest/quick-start.html`

AMPL modules are installed (see AMPLinstallation.ipynb).

Amplpy



`https://amplpy.ampl.com/en/latest/quick-start.html`

AMPL modules are installed (see AMPLinstallation.ipynb).

Any installation issue?

How to load a model

Load model using **seval**

```
ampl.eval(r"""
        set L ordered;
        param U {j in L} > 0; # max avail per liquid
        param p {j in L} > 0; # unit profit per liquid
        param w {j in L} > 0; # unit weight per liquid
        param W > 0;  # truck max capacity

        # decision variables
        var x {j in L} >= 0, <= U[j];

        # objective function
        maximize Total_Profit:
                sum {j in L} p[j]*x[j];

        # constraints
        subject to max_weight_constraint:
                sum{j in L} w[j]*x[j] <= W;
""")
```

How to load a model

Load model using **seval**

```
ampl.eval(r"""
        set L ordered;
        param U {j in L} > 0; # max avail per liquid
        param p {j in L} > 0; # unit profit per liquid
        param w {j in L} > 0; # unit weight per liquid
        param W > 0;  # truck max capacity

        # decision variables
        var x {j in L} >= 0, <= U[j];

        # objective function
        maximize Total_Profit:
                sum {j in L} p[j]*x[j];

        # constraints
        subject to max_weight_constraint:
                sum{j in L} w[j]*x[j] <= W;
""")
```

Load a model **reading the .mod file** .

```
ampl.read("diet.mod")
```

Load the data using Pandas objects

```
amplpy.AMPL.set_data()
```

is used to load data from the pandas.DataFrame objects

Load the data using Pandas objects

```
amplpy.AMPL.set_data()
```

is used to load data from the pandas.DataFrame objects

```
amplpy.Parameter.set_values()
```

is used to load data into an AMPL parameter

Load the data using Pandas objects

```
amplpy.AMPL.set_data()
```

is used to load data from the pandas.DataFrame objects

```
amplpy.Parameter.set_values()
```

is used to load data into an AMPL parameter

Examples:

- Send the data from "nutr_df" to AMPL and initialize the indexing set "NUTR"

```
ampl.set_data(nutr_df, "NUTR")
```

- Set the values for the parameter "amt" using "amt_df"

```
ampl.get_parameter("amt").set_values(amt_df)
```

Solve a problem

Once the model and the data are loaded:

Solve a problem

Once the model and the data are loaded:

Specify the **solver** to use (e.g., HiGHS)

```
ampl.option["solver"] = "highs"
```

Solve a problem

Once the model and the data are loaded:

Specify the **solver** to use (e.g., HiGHS)

```
ampl.option["solver"] = "highs"
```

Solve the problem

```
ampl.solve()
```

Solve a problem

Once the model and the data are loaded:

Specify the **solver** to use (e.g., HiGHS)

```
ampl.option["solver"] = "highs"
```

Solve the problem

```
ampl.solve()
```

Stop if the model was not solved

```
assert ampl.solve_result == "solved"
```

Get an AMPL entity in the programming environment

Relevant functions :

Get an AMPL entity in the programming environment

Relevant functions :

- amplpy.AMPL.get_variable()

Get an AMPL entity in the programming environment

Relevant functions :

- amplpy.AMPL.get_variable()
- amplpy.AMPL.get_constraint()

Get an AMPL entity in the programming environment

Relevant functions :

- amplpy.AMPL.get_variable()
- amplpy.AMPL.get_constraint()
- amplpy.AMPL.get_objective()

Get an AMPL entity in the programming environment

Relevant functions :

- amplpy.AMPL.get_variable()
- amplpy.AMPL.get_constraint()
- amplpy.AMPL.get_objective()
- amplpy.AMPL.get_parameter()

Get an AMPL entity in the programming environment

Relevant functions :

- amplpy.AMPL.get_variable()
- amplpy.AMPL.get_constraint()
- amplpy.AMPL.get_objective()
- amplpy.AMPL.get_parameter()
- amplpy.AMPL.get_set().

Get an AMPL entity in the programming environment

Example:

Get an AMPL entity in the programming environment

Example:

totalcost = ampl.get_objective("Total_Cost")

Get an AMPL entity in the programming environment

Example:

```
totalcost = ampl.get_objective("Total_Cost")
print("Objective is:", totalcost.get().value())
```

Get an AMPL entity in the programming environment

Example:

totalcost = ampl.get_objective("Total_Cost")

print("Objective is:", totalcost.get().value())

totalcost.value() is **equivalent** to totalcost.get().value()

Get numeric values from variables

To access all the numeric values contained in a Variable or any other entity:

Get numeric values from variables

To access all the numeric values contained in a Variable or any other entity:

- use a amplpy.DataFrame object

Get numeric values from variables

To access all the numeric values contained in a Variable or any other entity:

- use a amplpy.DataFrame object
- can be converted in other objects such as pandas.DataFrame

Get numeric values from variables

To access all the numeric values contained in a Variable or any other entity:

- use a amplpy.DataFrame object
- can be converted in other objects such as pandas.DataFrame

Example:

Get numeric values from variables

To access all the numeric values contained in a Variable or any other entity:

- use a amplpy.DataFrame object
- can be converted in other objects such as pandas.DataFrame

Example:

buy = ampl.get_variable("Buy")

Get numeric values from variables

To access all the numeric values contained in a Variable or any other entity:

- use a amplpy.DataFrame object
- can be converted in other objects such as pandas.DataFrame

Example:

```
buy = ampl.get_variable("Buy")
df = buy.get_values().to_pandas()
```

Get numeric values from variables

To access all the numeric values contained in a Variable or any other entity:

- use a amplpy.DataFrame object
- can be converted in other objects such as pandas.DataFrame

Example:

```
buy = ampl.get_variable("Buy")
df = buy.get_values().to_pandas()
print(df)
```

Alternatives to load data...

Load the data using **lists and dictionaries** .

Alternatives to load data...

Load the data using **lists and dictionaries** .

Load data **from files** , e.g.
`https://plugins.ampl.com/amplcsv.html`.

Alternatives to load data...

Load the data using **lists and dictionaries** .

Load data **from files** , e.g.
`https://plugins.ampl.com/amplcsv.html`.

Read a **.dat file**. Example:

```
ampl.read_data("models/diet.dat")
```

Let's code!

1. Implement the **liquid transportation example** together. Find the optimal solution by running a solver.

Let's code!

1. Implement the **liquid transportation example** together. Find the optimal solution by running a solver.

2. Change the model of point 1 by **adding the constraint** about the minimum liquid quantity to transport. Find the optimal solution by running a solver (or check it is infeasible!).

Let's code!

1. Implement the **liquid transportation example** together. Find the optimal solution by running a solver.

2. Change the model of point 1 by **adding the constraint** about the minimum liquid quantity to transport. Find the optimal solution by running a solver (or check it is infeasible!).

3. Change the model of point 1 by **mofifying the objective function** as in the slides about the defenerate case. Find the optimal solution by running a solver.

Let's code!

1. Implement the **liquid transportation example** together. Find the optimal solution by running a solver.

2. Change the model of point 1 by **adding the constraint** about the minimum liquid quantity to transport. Find the optimal solution by running a solver (or check it is infeasible!).

3. Change the model of point 1 by **mofifying the objective function** as in the slides about the defenerate case. Find the optimal solution by running a solver.

4. Code the example of **unboundedness** and run a solver to check it is really infeasible.

Let's code!

1. Implement the **liquid transportation example** together. Find the optimal solution by running a solver.

2. Change the model of point 1 by **adding the constraint** about the minimum liquid quantity to transport. Find the optimal solution by running a solver (or check it is infeasible!).

3. Change the model of point 1 by **mofifying the objective function** as in the slides about the defenerate case. Find the optimal solution by running a solver.

4. Code the example of **unboundedness** and run a solver to check it is really infeasible.

5. If we modify the objective of the model at point 4, by **minimizing instead of maximizing**, is the problem still unbounded?