Mixed Integer Non Linear Optimization: Methods and Applications
–
Introduction to AMPL

Claudia D'Ambrosio
dambrosio@lix.polytechnique.fr

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**:
  `https://ampl.com/learn/ampl-book/`

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**:
  `https://ampl.com/learn/ampl-book/`
- **Quick-start guide**:
  `https://www.lix.polytechnique.fr/~dambrosio/teaching/ampl-quick-start-guide_dambrosio.pdf`

AMPL files

- Each problem instance is coded in AMPL using three files:

AMPL files

- Each problem instance is coded in AMPL using three files:
  - a **model file** (extension .mod): contains the mathematical formulation of the problem.

AMPL files

- Each problem instance is coded in AMPL using three files:
  - a **model file** (extension .mod): contains the mathematical formulation of the problem.
  - a **data file** (extension .dat): contains the numerical values of the problem parameters.

AMPL files

- Each problem instance is coded in AMPL using three files:
  - a **model file** (extension .mod): contains the mathematical formulation of the problem.
  - a **data file** (extension .dat): contains the numerical values of the problem parameters.
  - a **run file** (extension .run): specifies the solution algorithm (external and/or coded by the user in the AMPL language itself).

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;
```

Continuous knapsack problem: File .mod

```
param N > 0;          # number of objects
set VARS ordered := {1..N};
param U {j in VARS} > 0;
param c {j in VARS} > 0;
param w {j in VARS} > 0;
param C > 0;  # knapsack capacity


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

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

subject to KP_constraint: # constraint
  sum{j in VARS} w[j]*x[j] <= C;
```

Continuous knapsack problem: File .dat

```
# Author: Claudia D'AMBROSIO
# Date: 20200106
# kp.dat

param N := 10;
param C := 546.000000;

param: w :=
1       78.770199
2       77.468892
3       93.324757
4       96.180080
5       55.137398
6       40.101851
7       36.007819
8       5.317250
9       9.964929
10      60.265707
;
```

```
param: c :=
1       3.062328
2       43.280130
3       52.983122
4       62.101010
5       58.531125
6       47.574366
7       53.101406
8       6.902601
9       16.985577
10      62.576610
;

param: U :=
1       100.000000
2       100.000000
3       100.000000
4       100.000000
5       100.000000
6       100.000000
7       100.000000
8       100.000000
9       100.000000
10      100.000000
;
```

Continuous knapsack problem: File .run

```
#   Author: Claudia D'Ambrosio
#   Date: 20190121
#   kp.run

reset;
reset data;

model kp.mod;

data "../dat/kp.dat";

option solver "gurobi";
option gurobi_options "outlev 1";

solve > kp.out;
```

Other commands

```
reset;
reset data;
```

Other commands

```
reset;
reset data;
```

```
option solver gurobi;
option gurobi_options "outlev 1";
solve;
```

Other commands

```
model myMILPmodel.mod;
data myInstance.dat;
option solver cplex;
option relax_integrality 1; # relaxing the
    integrality requirements on all the decision
    variables
solve;
option relax_integrality 0; # restoring the
    integrality requirements on all the decision
    variables
solve;
```

Other commands

```
display n, c;
display N;
display w, p;
```

Other commands

```
display n, c;
display N;
display w, p;
```

```
n = 7
c = 19

set N := 1 2 3 4 5 6 7;

:    w     p       :=
1    11    10
2     6     3
3     6     4
4     5     5
5     5     6
6     4     7
7     1     2
;
```

Other commands

```
display x;
display cost;
```

Other commands

```
display x;
display cost;
```

```
x [*] :=
1   0.363636
2   0
3   0
4   1
5   1
6   1
7   1
;

cost = 23.6364
```

Other commands

```
display capacity_constraint;
```

Other commands

```
display capacity_constraint;
```

```
capacity_constraint = 0.909091
```

Other commands

```
expand capacity_constraint;
```

Other commands

```
expand capacity_constraint;
```

```
subject to capacity_constraint:
        11*x[1] + 6*x[2] + 6*x[3] + 5*x[4] + 5*x[5] + 4*x[6] + x[7] <= 19;
```

Other commands

```
printf "param n := %d;\n", n;
printf "\n";

# param c
printf "param c :";
for {j in N} {
  printf "\t%d", j;
}
printf "\t:=\n";
for {i in N} {
  printf "\t%d", i;
  for {j in N} {
    printf "\t%d", c[i,j];
  }
  printf "\n";
}
printf ";\n\n";
```

Other commands

```
for {j in 1..n} {
...
}
```

Other commands

```
for {j in 1..n} {
...
}
```

```
repeat {
  . . .
   }
until x[n] > 0;
```

How to call AMPL from the command line:

```
Claudias-MacBook-Pro-8: ampl myrunfile.run
```

How to call AMPL from the command line:

```
Claudias-MacBook-Pro-8: ampl myrunfile.run
```

or

```
Claudias-MacBook-Pro-8: ampl myrunfile.run > myoutputfile.out
```

How to call AMPL from the AMPL environment/IDE:

```
ampl: include "../myfolder/myrunfile.run";
```

Continuous knapsack problem: File kp.out obtained

```
Gurobi 9.5.2: Set parameter Username
outlev 1
logfreq 1
Set parameter OutputFlag to value 1
Set parameter InfUnbdInfo to value 1
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (mac64[rosetta2])
Thread count: 10 physical cores, 10 logical processors, using up to 10 threads
Optimize a model with 1 rows, 10 columns and 10 nonzeros
Model fingerprint: 0x93cf5499
Coefficient statistics:
  Matrix range     [5e+00, 1e+02]
  Objective range  [3e+00, 6e+01]
  Bounds range     [1e+02, 1e+02]
  RHS range        [5e+02, 5e+02]
Presolve removed 1 rows and 10 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    9.3067648e+02   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds (0.00 work units)
Optimal objective  9.306764797e+02
Gurobi 9.5.2: optimal solution; objective 930.6764797
```

How to use AMPL

AMPL

- Either use the IDE or console

How to use AMPL

AMPL

- Either use the IDE or console
- To open the console: the executable *ampl* is already in the path, thus callable from any directory

How to use AMPL

AMPL

- Either use the IDE or console
- To open the console: the executable *ampl* is already in the path, thus callable from any directory
- Available solvers: baron, conopt, cplex, gurobi, knitro, minos, snopt, xpress

References

- Manual: `https://ampl.com/learn/ampl-book/`
- Modeling languages like **ampl**: `ampl.com` or **gams**: `www.gams.com` or **jump** `https://jump.dev/JuMP.jl/` or **pyomo** `http://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/`