# Introduction to AMPL IDE with Demonstration on Combinatorial Optimization Examples

Beichen Lyu

Undergraduate, Computer Science

Lvb@purdue.edu

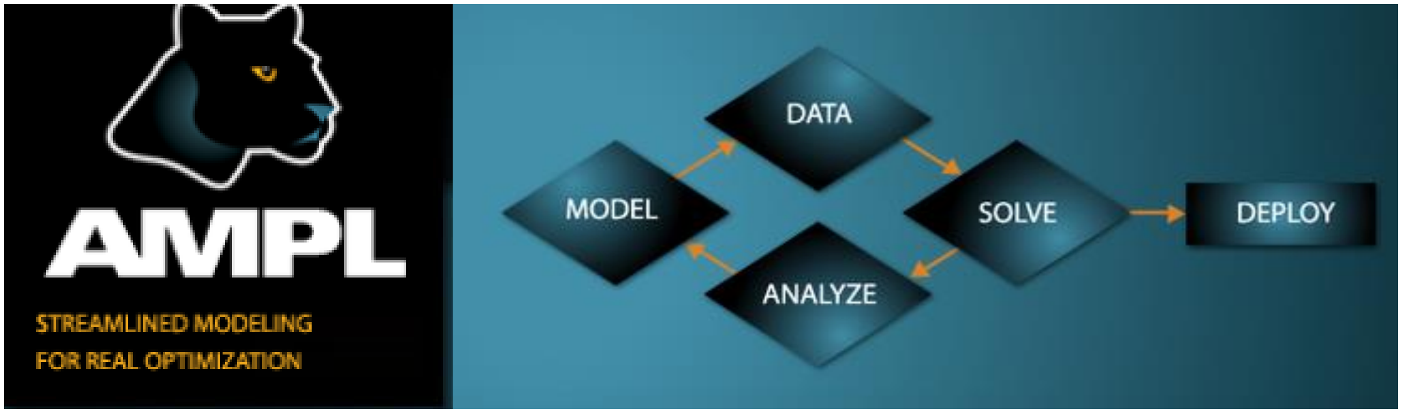
July 6th, 2018

PURDUE
UNIVERSITY

# Overview

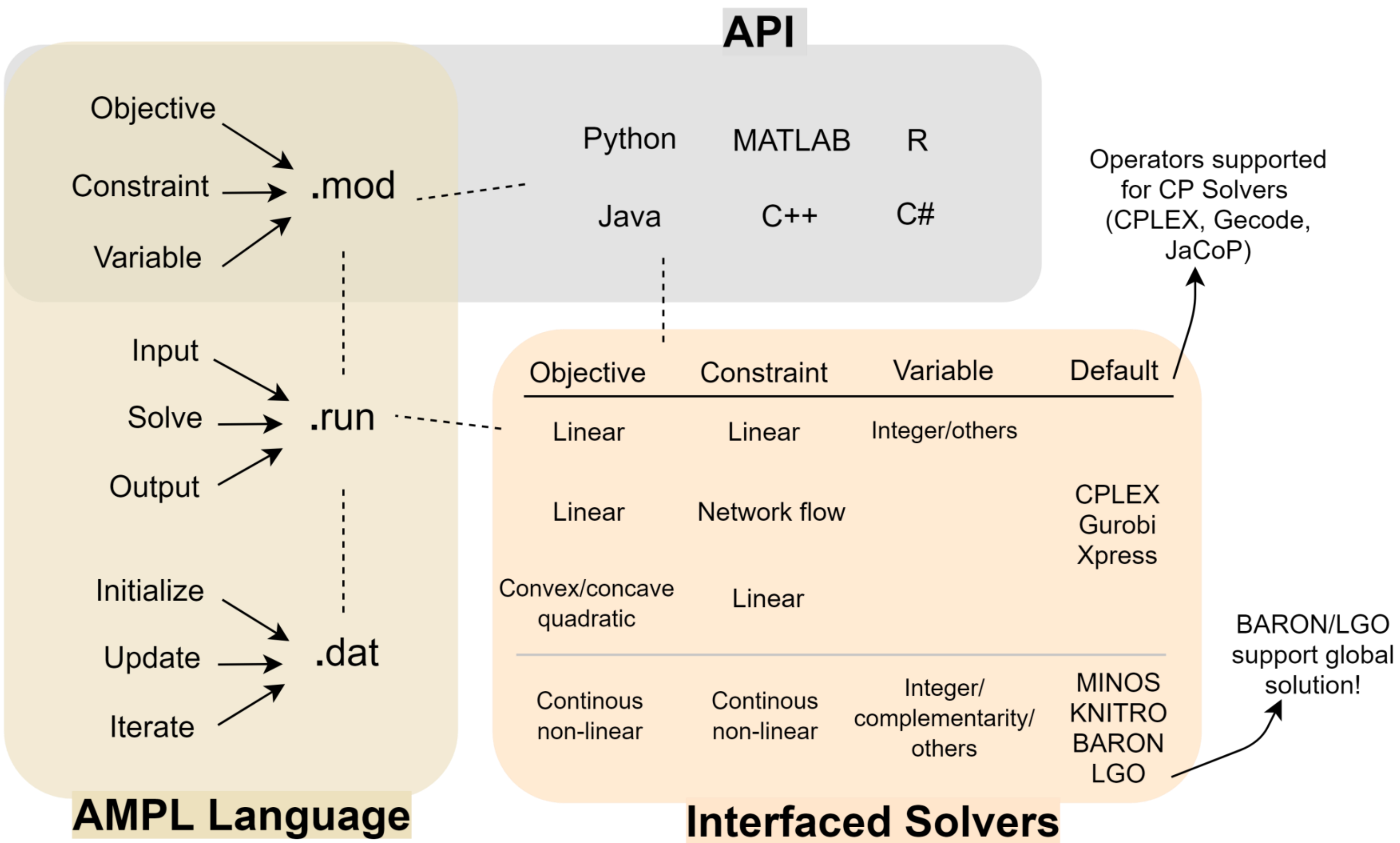
- AMPL IDE Basics
  - Background
  - Workflow
  - Evaluation
- Example Demonstration
  - Non-linear Optimization
  - Mixed Integer Programming
  - Constraint Programming
- Comparison with Peers
  - High-level IDEs: AIMMS, GAMS
  - Low-Level Solver: Google OR-Tools

# AMPL IDE - Background




- AMPL was originally developed as A Mathematical Programming Language for algebraic modelling at Bell Lab in 80s.
- Now it evolves into a comprehensive and state-of-the-art optimization tool.
  - It integrates three processes - data preparation, formulation modelling, and algorithm operation - into one IDE described in AMPL language.
  - It solves a wide range of Combinatorial Optimization problems, in particular (non-) linear/integer optimization and Constraint Programming.

# AMPL IDE - Workflow




*This diagram was created by Beichen Lyu

# AMPL IDE - Evaluation

- Pros
  - High readability with arithmetic, logical, and conditional expressions
  - Quick implementation with high-level algebraic modelling language
  - Broad availability supporting most mainstream solvers on the market
  - Easy customization upon ‘well’-documented programming languages
  - Big community with 2000+ members in AMPL Google Group, and open interface to personal functions/solvers

- Cons
  - Low flexibility on low-level modification, e.g. search, branch and bound
  - Cost for full-function purchase, ~$400 for academia

<A demo version is available, though functions in open-source solvers are unlimited.>

# Example 1
## Three-dimensional Modelling for Non-linear Optimization

- Objective

  *Maximize*

$$z = 3\,(1-x)^2\, e^{-x^2-(y+1)^2} - 10\, e^{-x^2-y^2}\left(-x^3 + \frac{x}{5} - y^5\right) - \frac{1}{3}\, e^{-(x+1)^2-y^2}$$

- Variables

$$x, y$$

- Constraints

$$-4 \leq x \leq 4$$
$$-4 \leq y \leq 4$$

# Example 1 (cont'd)
## Three-dimensional Modelling for Non-linear Optimization

### .mod

```
#parameter initialization
param lowerBound;
param upperBound;

#define input variables
var x >= lowerBound;
var y >= lowerBound;

#non-linear objective
maximize z: 3*(1-x)^2*exp(-(x^2) -
(y+1)^2) - 10*(x/5 - x^3 - y^5)*exp(-x^2-
y^2)- 1/3*exp(-(x+1)^2 - y^2);

#upper bound of variables
subject to xUpperBound: x <= upperBound;
subject to yUpperBound: y <= upperBound;
```

### .run

```
#reset environment
reset;

#load model
model peaks_test.mod;

#load data
data peaks_test.dat;

#decide solver
option solver lgo;

#solve
solve;

#display results
display _varname, _var;
display _objname, _obj;
display _conname, _con;
display _total_solve_time;
```

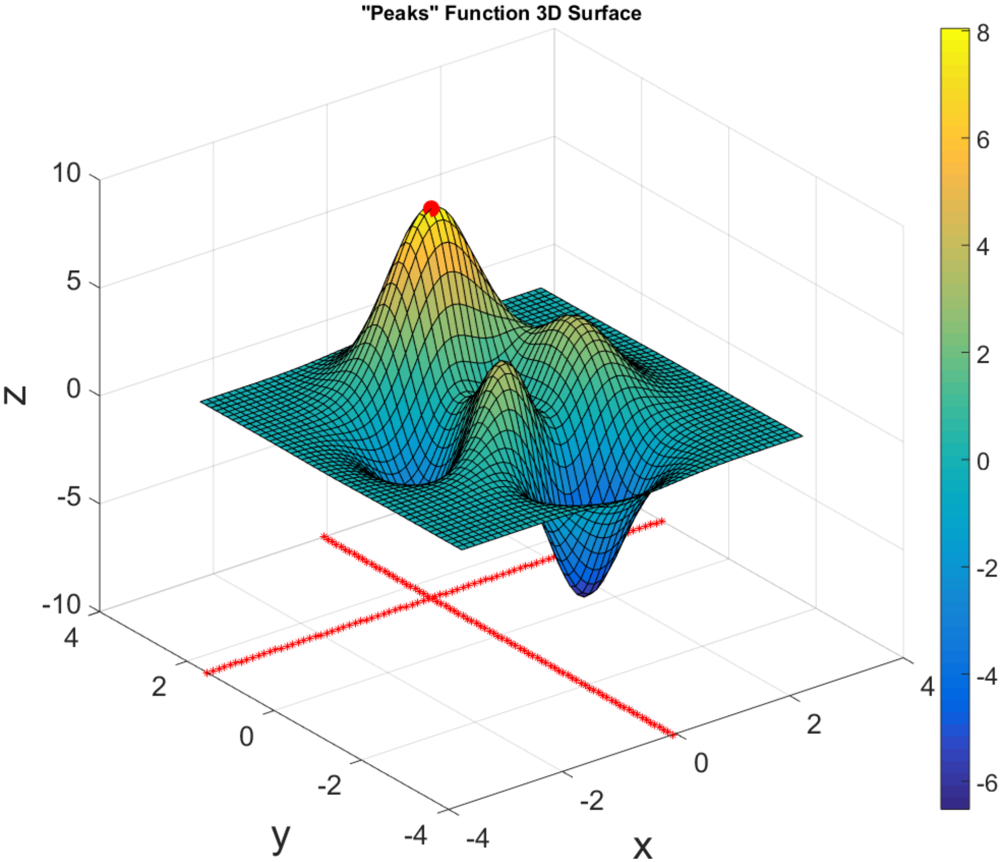
### .dat

```
#assigned by user
param lowerBound := -4;
param upperBound := 4;
```

# Example 1 (cont'd)
## Three-dimensional Modelling for Non-linear Optimization

**output**

```
ampl: include peaks_test.run

LGO 2015-01-17: Feasible solution from
global search;
function evaluation limit reached
(affected by g_maxfct = 800).
Objective 8.106213589
1602 function evaluations.
Runtime = 0 seconds
: _varname      _var        :=
1   x          -0.00931758
2   y           1.58137
;

: _objname    _obj      :=
1   z          8.10621
;

:    _conname   _con    :=
1   xUpperBound   0
2   yUpperBound   0
;

_total_solve_time = 0.015625 <seconds>
```



# Example 2

## Cost Minimization for Mixed Integer Programming

- Objective

*Minimize*

$$totalCost = \sum_{\text{worker},task} cost_{woker,task} * matches_{worker,task}$$

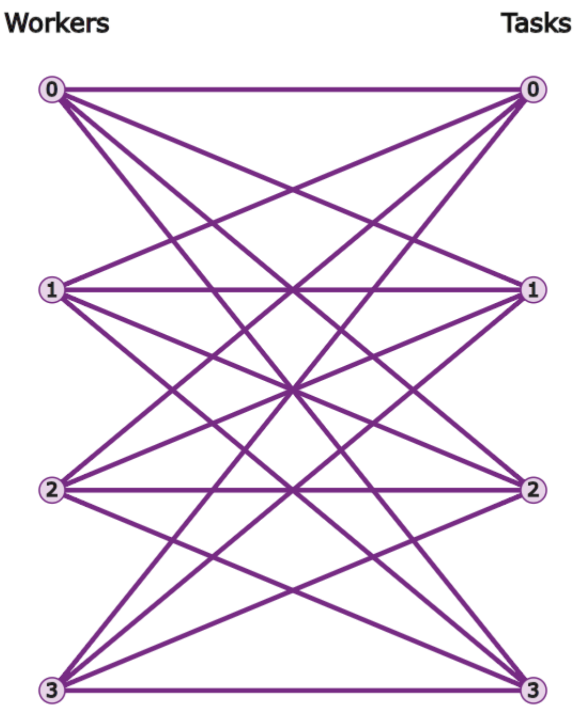
- Variables

$$matches_{worker,task}$$

- Constraints

$$\sum_{\text{worker}} matches_{worker,task} = 1$$

$$\sum_{\text{task}} matches_{worker,task} = 1$$



# Example 2 (cont'd)
## Cost Minimization for Mixed Integer Programming

**.mod**

```
#sets of indices
set WORKER;
set TASK;

#parameter initialization
param cost {WORKER, TASK};

#define lower bound of input variables
var matches {WORKER, TASK} binary;

#minimize total cost
minimize totalCost:
sum {i in WORKER, j in TASK} cost[i, j] * matches[i, j];

#upper bound of variables
subject to workAssign {i in WORKER}:
sum {j in TASK} matches[i, j] = 1;

subject to taskAssign {j in TASK}:
sum {i in WORKER} matches[i, j] = 1;
```

# Example 2 (cont'd)

## Cost Minimization for Mixed Integer Programming

.java

```
importcom.ampl.AMPL;
import com.ampl.DataFrame;
import java.io.IOException

"public class WorkerTaskDemo {
public static void main(String[] args) throws IOException {
// initialize object
AMPL ampl = new AMPL();
ampl.setOption("solver", "gurobi");

try {
// read .mod file
ampl.read("<parent directory>"+"WorkerTaskDemo.mod");

// initialize .dat object for workers
DataFrame df = new DataFrame(1, "WORKER");
String[] workers = "worker1 worker2 worker3
worker4".split("\\s+");

df.setColumn("WORKER", workers);
ampl.setData(df, "WORKER");
```

```
// refresh .dat object for tasks
df = new DataFrame(1, "TASK");
String[] tasks = "task1 task2 task3 task4".split("\\s+");
df.setColumn("TASK", tasks);
ampl.setData(df, "TASK");

// refresh .dat object for costs
df = new DataFrame(2, "WORKER", "TASK", "cost");
double[][] cost = {{90, 76, 75, 70},{35, 85, 55, 65},{125,
95, 90, 105},{45, 110, 95, 115}};
df.setMatrix(cost, workers, tasks);
ampl.setData(df);

//solve
ampl.solve();

//display
System.out.println("total cost:\n" +
ampl.getObjective("totalCost").value());
System.out.println("matches:\n" +
ampl.getVariable("matches").getValues());
} finally {
ampl.close();}}}
```

# Example 2 (cont'd)
## Cost Minimization for Mixed Integer Programming

**AMPL output**

```
Gurobi 8.0.0: optimal solution; objective 265
6 simplex iterations
total cost:
265.0
matches:
i1       i2     |  val
worker1  task4  |  1.0
worker2  task3  |  1.0
worker3  task2  |  1.0
worker4  task1  |  1.0
```

**AMPL runs for 15.6 milliseconds;**

**Google OR-Tools .py script uses 1.6 milliseconds**

**with the same match and cost.**

**OR-Tools .py**

```
from ortools.graph import pywrapgraph


def main():
cost = create_data_array()
rows = len(cost)
cols = len(cost[0])
assignment = pywrapgraph.LinearSumAssignment()
for worker in range(rows):
 for task in range(cols):
  if cost[worker][task]:
   assignment.AddArcWithCost(worker, task, cost[worker][task])
solve_status = assignment.Solve()


if solve_status == assignment.OPTIMAL:
 <7 lines of print scripts>


def create_data_array():
cost = [[90, 76, 75, 70], [35, 85, 55, 65],
         [125, 95, 90, 105], [45, 110, 95, 115]]
return cost
```

# Example 3

## N Queens Problem for Constraint Programming

- Variables

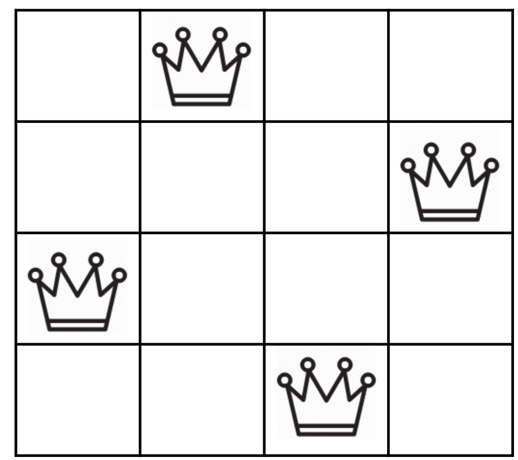
$$grid_{row,column} = \{0, 1\}$$

$$\text{row}, column \in 1..size$$

- Constraints

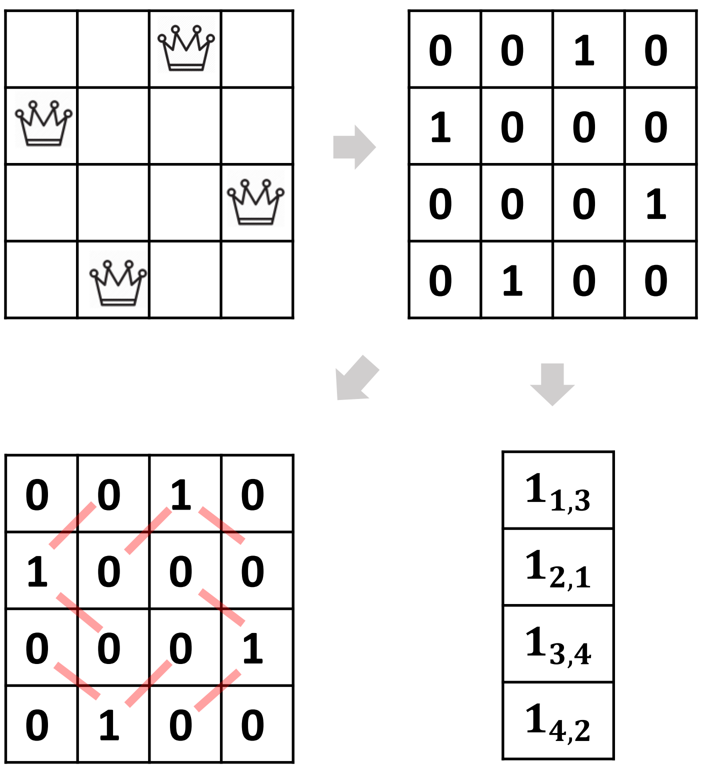
$$grid_{i,k} \neq grid_{j,k}$$
$$grid_{i,k} \neq grid_{i,j}$$
$$grid_{i,k} \neq grid_{i\pm1,k\pm1}$$

$$i, j, k \in 1..size$$



# Example 3 (cont'd)

## N Queens Problem for Constraint Programming




**.mod**

```
# chess size
param size := 300;

# 2D matrix to 1D vector
# constraint on row repetition
var row {1..size} integer >= 1 <= size;

# constraint on column repetition
s.t. consRow:
alldiff ({i in 1..size} row[i]);

# constraint on diagonal repetition
s.t. consDia:
alldiff ({i in 1..size} row[i]+i)
and
alldiff ({i in 1..size} row[i]-i);
```

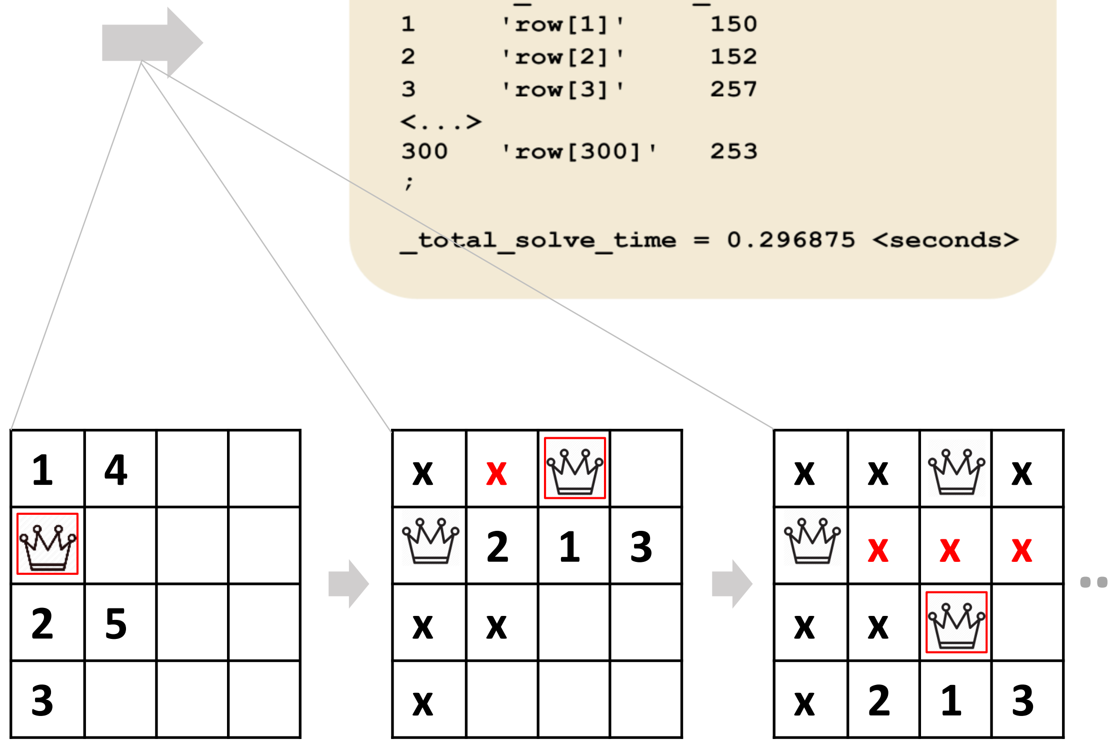
# Example 3 (cont'd)
## N Queens Problem for Constraint Programming

**.run**

```
# reset environment
reset;

# load model
model NQueensDemo.mod;

# CP solver with options
option solver gecode;
option gecode_options
'icl=bnd solutionlimit=1000
timelimit=10
val_branching=med
var_branching=size_min';

# solve
solve;

# display results
display _varname, _var;
display _total_solve_time;
```

**output**

```
gecode 4.4.0: feasible solution
3780 nodes, 745 fails
:     _varname    _var    :=
1     'row[1]'     150
2     'row[2]'     152
3     'row[3]'     257
<...>
300   'row[300]'   253
;

_total_solve_time = 0.296875 <seconds>
```



# Comparison with High-level IDEs

| | Programming Language | Low-level Customization | Cloud Computing | Support | Academic Price |
|---|---|---|---|---|---|
| **AMPL** | AMPL, algebraic modelling language | Directives for branch | On NEOS Server (free), AWS, Gurobi Instant Cloud, etc. | ~2600 members in Google Groups; complete API documentation | $400 base module; free demo version available |
| **AIMMS** | AIMMS, graphic development environment | Directives for branch, cut, and bound | On its own platform through its Apps; commercial | ~1500 members in Google Groups | $100 base module; free academic license available |
| **GAMS** | GAMS, algebraic modelling language | Directives for branch, cut, and bound | On NEOS Server (free), AWS, Gurobi Instant Cloud, etc | ~600 members on its forum; complete API documentation | $640 base module |
| **MiniZinc** | FlatZinc, algebraic modelling language | Directives for branch, cut, and bound; interfaced to open source CP solvers like OR-Tools | N/A | ~160 members in Google Groups; hosts CP annual competition over 10 years | Open source |

# Comparison with Low-level Solver

| | Interfaced API | Interfaced Solvers' Algorithm | Low-level Customization | Support |
|---|---|---|---|---|
| **AMPL** | Java, C++, C#, Python, R, MATLAB | Linear Programming, Mixed Integer Programming, non-linear optimization; | Directives for branch; commercial | ~2600 members in Google Groups; complete API documentation |
| **Google OR-Tools** | Java, C++, C#, Python | Linear Programming, Mixed Integer Programming | In-depth configuration for branch, bound, and cut; open source on GitHub | ~1200 members in Group Groups; only API for C++ in maintenance (because the CP solver is written in C++); used internally at Google |

# Comments

- AMPL runs with two options
  - (.mod + (.dat) + .run) and (.mod + API)
- It solves non-linear and linear optimization problems by interfacing mainstream solvers.
- Use it for high-level modelling and quick computation; use OR-Tools for low-level customization.
- You can download its demo version for free, full-solver access but limited constraint/variable size (~300), at https://ampl.com/try-ampl/download-a-free-demo/

# References


- AIMMS B. V. (n. d.). Solvers. Retrieved July 5th, 2018 from web
- AMPL Optimization Inc. (n. d.). AMPL Products. Retrieved May 24, 2018 from web
- GAMS Software GmbH. (n. d.). An Introduction to GAMS. Retrieved July 5th, 2018 from web
- Google Inc. (May 29, 2018). Google Optimization Tools. Retrieved June 6th, 2018 from web
- MiniZinc.org. (n. d.). Getting Started. Retrieved July 5th, 2018 from web