

Using AMPL inside C

Johnathan Rhyne

CU Denver

March 13, 2023

Table of Contents

- 1 Brief AMPL Overview
- 2 Problem Overview
- 3 Embedding in C
 - Files Created Externally
 - All Inside C
- 4 Extensions

What is AMPL?

AMPL Stands for **A Mathematical Programming Language**.
In short, it is a relatively human readable way of asking a computer to solve optimization problems for us.

What is Sudoku?

8				2	6			
						7		4
			7					5
			1				3	6
	1			8			4	
9	8				3			
3					1			
7		5						
			2	5				8

Figure: Source: Integer Programming - Michele conforti Gerard Cornuejols Giacomo Zambelli

Sudoku as an Integer Program

$$\sum_{i=1}^9 x_{ijk} = 1, \quad 1 \leq jk \leq 9 \quad (\text{each number } k \text{ appears in each column})$$

$$\sum_{j=1}^9 x_{ijk} = 1, \quad 1 \leq ik \leq 9 \quad (\text{each number } k \text{ appears in each row})$$

$$\sum_{q,r=0}^2 x_{i+q,j+r,k} = 1, \quad i,j = 1,4,7, 1 \leq k \leq 9 \quad (\text{each } k \text{ appears once in each box})$$

$$\sum_{k=1}^9 x_{ijk} = 1, \quad 1 \leq ij \leq 9 \quad (\text{each cell contains exactly one number})$$

$$x_{ijk} \in \{0, 1\}, \quad 1 \leq ijk \leq 9$$

$$x_{ijk} = 1, \quad \text{When the initial board has number } k \text{ in cell } (i,j).$$

Use Case For External File Creation

- Already tried running it locally, so you have the files
- Running on a cluster with no GUI or very high latency
- Only part of what you are trying to solve

Requirements

- AMPL installation accessible to your user
- A list of AMPL commands inside a text file

Example AMPL Files

```
set x := {1, 2, 3, 4, 5, 6, 7, 8, 9}; # x-coordinate inside our grid
set y := {1, 2, 3, 4, 5, 6, 7, 8, 9}; # y-coordinate inside our grid
set value := {1, 2, 3, 4, 5, 6, 7, 8, 9}; # value that is taken at coordinate (i,j)

param givenInfo {x,y,value} binary;

# 3D variable where the value at grid[i,j,k] = 1
# if and only if the number k is present at the
# coordinate (i,j) inside our 9x9 grid
var grid {x,y,value} binary;
# Sudoku doesn't require an objective function
# So, we just add a function to let
# AMPL run. This can be anything
minimize dummyFunc: 0;

# Constraint that ensures we only have 1 number
# per column
subject to onePerColumn {(j,k) in {y,value}}:
    sum {i in x} grid[i,j,k] = 1;

# Constraint that ensures we only have 1 number
# per row
subject to onePerRow {(i,k) in {x,value}}:
    sum {j in y} grid[i,j,k] = 1;

# Set that helps with our onePerBox constraint
set offset = {1, 4, 7};

# Set that helps with our onePerBox constraint
set boundaries = {0,1,2};

# Constraint that ensures we only have 1 number
# per 3x3 box
subject to onePerBox {(i,j,k) in {offset, offset, value}}:
    sum {(a,b) in {boundaries,boundaries}} grid[i + a, j + b, k] = 1;

# Constraint that ensures we only have 1 number
# per cell in our grid
subject to onePerCell {(i,j) in {x, y}}:
    sum {k in value} grid[i,j,k] = 1;

# Constraint that ensures we don't violate the given data
subject to startData {(i,j,k) in {x,y,value}}:
    grid[i,j,k] >= givenInfo[i,j,k];

data;
# These following lines
# will be the given state of a sudoku grid
for{(i,j,k) in {x,y,value}} {let givenInfo[i,j,k] := 0}

let givenInfo[1,1,8] := 1;
let givenInfo[1,5,2] := 1;
let givenInfo[1,6,6] := 1;
let givenInfo[2,7,7] := 1;
let givenInfo[2,9,4] := 1;
let givenInfo[3,4,7] := 1;
let givenInfo[3,9,5] := 1;
let givenInfo[4,4,1] := 1;
let givenInfo[4,8,3] := 1;
let givenInfo[4,9,6] := 1;
let givenInfo[5,2,1] := 1;
let givenInfo[5,5,8] := 1;
let givenInfo[5,8,4] := 1;
let givenInfo[6,1,9] := 1;
let givenInfo[6,2,8] := 1;
let givenInfo[6,6,3] := 1;
let givenInfo[7,1,3] := 1;
let givenInfo[7,6,1] := 1;
let givenInfo[8,1,7] := 1;
let givenInfo[8,3,5] := 1;
let givenInfo[9,4,2] := 1;
let givenInfo[9,5,5] := 1;
let givenInfo[9,9,8] := 1;

end;
```


Command File

```
model sudoku.mod;  
data  sudoku.dat;  
option solver "./ampl_linux-intel64/cplex";  
solve;  
display {k in value}: {i in x, j in y} grid[i,j,k];
```

Calling From C!

```
system(amplPath < commandFilePath > outputFilePath)
```

Nice Output

8	4	7	5	2	6	1	9	3
6	5	1	8	3	9	7	2	4
2	3	9	7	1	4	6	8	5
4	7	2	1	9	5	8	3	6
5	1	3	6	8	2	9	4	7
9	8	6	4	7	3	2	5	1
3	6	8	9	4	1	5	7	2
7	2	5	3	6	8	4	1	9
1	9	4	2	5	7	3	6	8

All Inside C: Key Things to Consider

- No official API like Java, Python, C++
- However, can do most of it through basic File I/O and system calls!

Example C File

```
writeModelFile("sudoku.mod");
int grid[9][9];
for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
        grid[i][j] = 0;
    }
}
// Lazy way of initially making the grid
// You could instead read in a bunch of files
// containing real sudoku grids
// This will create a random "valid" starting state
srand(time(NULL));
grid[0][0] = rand()%10;
grid[1][3] = rand()%10;
grid[2][6] = rand()%10;
grid[3][1] = rand()%10;
grid[4][4] = rand()%10;
grid[5][7] = rand()%10;
grid[6][2] = rand()%10;
grid[7][5] = rand()%10;
grid[8][8] = rand()%10;

writeDataFile("sudoku.dat", grid);
writeCommandFile("sudoku.dat", "sudoku.mod", "./ampl_linux-intel64/cplex",
"sudoku.command");

runAMPL("./ampl_linux-intel64/ampl", "sudoku.command", "out.txt");
int *finalGrid = readOutFile("out.txt");
for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
        printf("%d ", finalGrid[i + j * 9]);
    }
    printf("\n");
}
free(finalGrid);
```

Example Output

6	7	9	4	5	8	1	2	3
2	8	5	3	6	1	9	4	7
1	4	3	9	7	2	5	6	8
7	6	1	2	3	5	8	9	4
4	5	2	8	9	7	6	3	1
3	9	8	6	1	4	7	5	2
8	3	7	5	2	9	4	1	6
9	2	4	1	8	6	3	7	5
5	1	6	7	4	3	2	8	9

Variant Sudoku

- Arrow Sum Lines
- Killer Boxes