Workshop on Optimization Techniques for Data Science in Python and Julia

3. Solving mixed integer programming (MIP) problems with Pyomo

Bogumił Kamiński

Solving sudoku (Hart et al., chap. 14.6.2)

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

Solving optimization problems

1. Mathematical formulation
2. Problem type identification
3. Software implementation
4. Solution

Mathematical formulation

1. Decision variables

2. Objective

3. Constraints

Mathematical formulation

1. Decision variables
   - $y[r, c, v] \in \{0,1\}$, where $r, c, v \in \{1,2,3,4,5,6,7,8,9\}$
     (equals to 1 if cell (r, c) contains value v, otherwise contains 0)
2. Objective

3. Constraints

Mathematical formulation

1. Decision variables
   - $y[r, c, v] \in \{0,1\}$, where $r, c, v \in \{1,2,3,4,5,6,7,8,9\}$
     (equals to 1 if cell (r, c) contains value v, otherwise contains 0)
2. Objective
   - not important (we want to find all feasible solutions)
3. Constraints

Mathematical formulation

1. Decision variables
   - $y[r, c, v] \in \{0,1\}$, where $r, c, v \in \{1,2,3,4,5,6,7,8,9\}$
     (equals to 1 if cell (r, c) contains value v, otherwise contains 0)
2. Objective
   - not important (we want to find all feasible solutions)
3. Constraints
   - each row contains exactly one of the numbers $\{1,2,3,4,5,6,7,8,9\}$
   - each column contains exactly one of the numbers $\{1,2,3,4,5,6,7,8,9\}$
   - each 3x3 cell contains exactly one of the numbers $\{1,2,3,4,5,6,7,8,9\}$
   - each cell contains exactly one number
   - cells that we know contain proper values

How to generate all solutions?

1. Find any solution
2. Update optimization problem by excluding only it from a feasible set
3. Repeat steps 1 and 2 as long as the problem is feasible

How to generate all solutions?

1. Find any solution
2. Update optimization problem by excluding only it from a feasible set
3. Repeat steps 1 and 2 as long as the problem is feasible

Let s$[r, c, v]$ be a feasible solution. We exclude it using the condition:

$$\sum_{r,c,v} f(s[r,c,v], y[r,c,v]) \geq 1, \qquad \text{where } f(s,y) = \begin{cases} y & \text{if} \quad x = 0 \\ 1 - y & \text{if} \quad x = 1 \end{cases}$$

or

$$\sum_{r,c,v} s[r,c,v] y[r,c,v] \leq 80$$

Problem type identification

- Decision variables: boolean
- Objective: linear in variables
- Constraints: linear in variables

Problem type identification

- Decision variables: boolean
- Objective: linear in variables
- Constraints: linear in variables

Type of problem:
mixed integer programming

Software implementation (sudoku.py)

```
from pyomo.environ import *
from pyomo.opt import TerminationCondition

subsq_to_row_col = dict()

subsq_to_row_col[1] = [(i,j) for i in range(1,4) for j in range(1,4)]
subsq_to_row_col[2] = [(i,j) for i in range(1,4) for j in range(4,7)]
subsq_to_row_col[3] = [(i,j) for i in range(1,4) for j in range(7,10)]

subsq_to_row_col[4] = [(i,j) for i in range(4,7) for j in range(1,4)]
subsq_to_row_col[5] = [(i,j) for i in range(4,7) for j in range(4,7)]
subsq_to_row_col[6] = [(i,j) for i in range(4,7) for j in range(7,10)]

subsq_to_row_col[7] = [(i,j) for i in range(7,10) for j in range(1,4)]
subsq_to_row_col[8] = [(i,j) for i in range(7,10) for j in range(4,7)]
subsq_to_row_col[9] = [(i,j) for i in range(7,10) for j in range(7,10)]
```

Software implementation (sudoku.py)

```
def create_sudoku_model(board):
    model = ConcreteModel()

    model.board = board

    model.ROWS = RangeSet(1,9)
    model.COLS = RangeSet(1,9)
    model.SUBSQUARES = RangeSet(1,9)
    model.VALUES = RangeSet(1,9)

    model.y = Var(model.ROWS, model.COLS, model.VALUES, within=Binary)

    for (r,c,v) in board:
        model.y[r,c,v].fix(1)

    model.obj = Objective(expr= 1.0)



# (function continued on the next slide ...)
```

Software implementation (sudoku.py)

```
# (function continued from the previous slide ...)

    def _RowCon(model, r, v):
        return sum(model.y[r,c,v] for c in model.COLS) == 1
    model.RowCon = Constraint(model.ROWS, model.VALUES, rule=_RowCon)

    def _ColCon(model, c, v):
        return sum(model.y[r,c,v] for r in model.ROWS) == 1
    model.ColCon = Constraint(model.COLS, model.VALUES, rule=_ColCon)

    def _SqCon(model, s, v):
        return sum(model.y[r,c,v] for (r,c) in subsq_to_row_col[s]) == 1
    model.SqCon = Constraint(model.SUBSQUARES, model.VALUES, rule=_SqCon)

    def _ValueCon(model, r, c):
        return sum(model.y[r,c,v] for v in model.VALUES) == 1
    model.ValueCon = Constraint(model.ROWS, model.COLS, rule=_ValueCon)

    return model
```

Software implementation (sudoku.py)

```
def add_integer_cut(model):
    if not hasattr(model, "IntegerCuts"):
        model.IntegerCuts = ConstraintList()

    cut_expr = 0.0
    for r in model.ROWS:
        for c in model.COLS:
            for v in model.VALUES:
                if not model.y[r,c,v].fixed:
                    if value(model.y[r,c,v]) >= 0.5:
                        cut_expr += (1.0 - model.y[r,c,v])
                    else:
                        cut_expr += model.y[r,c,v]
    model.IntegerCuts.add(cut_expr >= 1)

def print_solution(model):
    for r in model.ROWS:
        print(" ".join(str(v) for c in model.COLS
                              for v in model.VALUES
                              if value(model.y[r,c,v]) >= 0.5))
```

Solution (sudoku.py)

```
board = [(1,1,5),(1,2,3),(1,5,7),(2,1,6),(2,4,1),(2,5,9),(2,6,5),
         (3,2,9),(3,3,8),(3,8,6),(4,1,8),(4,5,6),(4,9,3),(5,1,4),
         (5,4,8),(5,6,3),(5,9,1),(6,1,7),(6,5,2),(6,9,6),(7,2,6),
         (7,7,2),(7,8,8),(8,4,4),(8,5,1),(8,6,9)]

model = create_sudoku_model(board)

solution_count = 0
while 1:
    with SolverFactory("glpk") as opt:
        results = opt.solve(model)
        if results.solver.termination_condition != TerminationCondition.optimal:
            print("All board solutions have been found")
            break
        solution_count += 1
        add_integer_cut(model)
        print("Solution #%d" % (solution_count))
        print_solution(model)
```

Solution (sudoku.py)

```
$ python sudoku.py
WARNING: Constant objective detected, replacing with a placeholder to prevent
    solver failure.
Solution #1
5 3 4 6 7 8 1 9 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 7 6 5
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 9 7 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 5 4 1 9 6 3 7
3 4 7 2 8 6 5 1 9
WARNING: Constant objective detected, replacing with a placeholder to prevent
    solver failure.
Solution #2
5 3 4 6 7 8 1 9 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 9 7 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 7 1 9
WARNING: Constant objective detected, replacing with a placeholder to prevent
    solver failure.
Solution #3
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
WARNING: Constant objective detected, replacing with a placeholder to prevent
    solver failure.
All board solutions have been found
```

Self-check task

Change the exclusion constraint to the second form.