Optimization Algorithms & Tools for AI

Neng-Fa Zhou

CUNY Graduate Center

2024

Syllabus

- http://www.sci.brooklyn.cuny.edu/~zhou/teaching/gc24/

Constraint Satisfaction and Optimization Problems

- CSP $= (X, D, C)$
  - $X = X_1, X_2, \ldots, X_n$
  - $D = D_1, D_2, \ldots, D_n$
  - $C = C_1, C_2, \ldots, C_m$
  - Find a valuation of X that satisfies all the constraints.
- COP $= (X, D, C, O)$
  - $O = \min(E)$ or $\max(E)$
  - Find a valuation of X that satisfies all the constraints and optimizes O.

Examples of CSP and COP

- Basic NP-complete and NP-hard problems
  - Satisfiability, graph coloring, maximum clique, minimum cover, Hamiltonian cycle, TSP, ...
- Puzzles
  - N-queens, sudoku, knight's tour, magic square, cryptarithmetic, ...
- Real-world problems
  - Resource allocation and scheduling (cloud computing, operations management, supply chain optimization, ...)
  - Network design (utility networks, communication networks, AI networks)

Solving Methods

- The Symplex method for linear programming problems
- Mixed integer programming (branch-and-bound and cutting-plane algorithms)
- Constraint programming (backtracking search and constraint propagation)
- SAT solving (backtracking search, unit propagation, and conflict-driven clause learning)

Modeling and Solving Tools (Picat)

```
import sat.

main =>
    Board = {{6,_,2,_,5,_,_,_,_,_}, ..., {_,_,_,_,6,_,_,_,_,_}},
    sudoku(Board),
    foreach(Row in Board) writeln(Row) end.

sudoku(Board) =>
   N = len(Board),
   Board :: 1..N,
   foreach (Row in Board) all_different(Row) end,
   foreach (J in 1..N) all_different([Board[I,J] : I in 1..N]) end,
   M = round(sqrt(N)),
   foreach (I in 1..M..N-M, J in 1..M..N-M)
      all_different([Board[I+K,J+L] : K in 0..M-1, L in 0..M-1])
   end,
   solve(Board).
```

Modeling and Solving Tools (Julia)

```
using JuMP, HiGHS
grid = [6 0 2 0 5 0 0 0 0; ...; 0 0 0 6 0 0 0 0 0]

csp = Model(HiGHS.Optimizer)
@variable(csp, 1 <= x[1:9,1:9] <= 9, Int)
for r=1:9, c=1:9
    if grid[r,c] != 0
        @constraint(csp, x[r,c] == grid[r,c])
    end
end
for rc = 1:9
    all_different(csp, x[rc,:])
    all_different(csp, x[:,rc])
end
for br=0:2
    for bc=0:2
        all_different(csp, vec(x[br*3+1:(br+1)*3, bc*3+1:(bc+1)*3]))
    end
end
optimize!(csp)
grid = convert.(Int, JuMP.value.(x))
display(grid)
```