gurobi_code

November 2, 2022

[30]:
```
import gurobi as grb # can be installed via Gurobi's Anaconda channel; fairly␣
 ↪easy to setup.
# see, e.g., https://gitlab.ibr.cs.tu-bs.de/alg/student-guides/-/blob/main/
 ↪gurobi.md
import itertools
```

1 Modeling Language

https://cocoto.github.io/glpk-online/#G4QwTgBAHgjBB8BeCAGA3AWAFCklATAsutrtAMxGqY7jQAsVJWAz
AC8eCGC5ADYlODUADkcXIyVtCABOPVMRABtgHhoAE38WAAdskABPaBgAOlBsgBoKOpBG8Jbsmh5JArC

2 Simple example from the board

[31]:
```
# create an empty LP model
model = grb.Model()
# create a list of four variables (x_1 to x_4)
x = [model.addVar(name=f"x_{xi}") for xi in range(1,5)]
# set objective to max 6x_1 + 8x_2 + 5x_3 + 9x_4
model.setObjective(6 * x[0] + 8 * x[1] + 5 * x[2] + 9 * x[3], grb.GRB.MAXIMIZE)
# add constraints
model.addConstr(2*x[0] + x[1] + x[2] + 3 * x[3] <= 5)
model.addConstr(x[0] + 3*x[1] + x[2] + 2*x[3] <= 3)
# solve!
model.optimize()
# after solving, model.objVal accesses the optimal objective value
print("Objective value:", model.objVal)
for i, x_i in enumerate(x):
    # after solving, variable.x accesses the value in the optimal solution
    print(f"x_{i+1} = {x_i.x}")
```

```
Gurobi Optimizer version 9.1.1 build v9.1.1rc0 (mac64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 2 rows, 4 columns and 8 nonzeros
Model fingerprint: 0xd18e4328
Coefficient statistics:
  Matrix range     [1e+00, 3e+00]
```

```
  Objective range  [5e+00, 9e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e+00, 5e+00]
Presolve time: 0.01s
Presolved: 2 rows, 4 columns, 8 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    4.9000000e+31   5.750000e+30   4.900000e+01      0s
       3    1.7000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.02 seconds
Optimal objective  1.700000000e+01
Objective value: 17.0
x_1 = 2.0
x_2 = 0.0
x_3 = 1.0
x_4 = 0.0
```

3 Flow computation

[32]:
```
class SolveMaxFlow:
    def __add_variables(self):
        self.variables = {(v,w): self.model.addVar(name=f"x_{v,w}", lb=0.0,␣
↪ub=capacities[v][w])
                          for v,w in itertools.product(range(self.n_vertices),␣
↪repeat=2)
                          if self.capacities[v][w] != 0}

    def __add_vertex_constraints(self):
        for v in range(self.n_vertices):
            if v in (self.source, self.sink): continue
            incoming = 0
            outgoing = 0
            for w in range(self.n_vertices):
                if (v,w) in self.variables:
                    outgoing += self.variables[v,w]
                if (w,v) in self.variables:
                    incoming += self.variables[w,v]
            self.model.addConstr(outgoing == incoming)

    def __add_objective(self):
        sink_in = (self.variables[w,self.sink] for w in range(self.n_vertices)
                   if (w,self.sink) in self.variables)
        self.model.setObjective(sum(sink_in), sense=grb.GRB.MAXIMIZE)

    def __init__(self, capacities, source, sink):
```

```
        self.n_vertices = len(capacities)
        self.capacities = capacities
        self.model = grb.Model()
        self.source = source
        self.sink = sink
        self.__add_variables()
        self.__add_vertex_constraints()
        self.__add_objective()

    def solve(self):
        self.model.optimize()
        return {e: v.x for e,v in self.variables.items()
                if v.x >= 1e-6}
```

[33]:
```
capacities = [
    [0, 1000, 1000,    0], # v0 = s
    [0,    0,    1, 1000], # v1
    [0,    0,    0, 1000], # v2
    [0,    0,    0,    0]  # v3 = t
]
s = SolveMaxFlow(capacities, 0, 3)
```

[34]:
```
s.solve()
```

```
Gurobi Optimizer version 9.1.1 build v9.1.1rc0 (mac64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 2 rows, 5 columns and 6 nonzeros
Model fingerprint: 0x1e521ef6
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+03]
  RHS range        [0e+00, 0e+00]
Presolve removed 2 rows and 5 columns
Presolve time: 0.01s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.0000000e+03   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds
Optimal objective  2.000000000e+03
```

[34]:
```
{(0, 1): 1000.0, (0, 2): 1000.0, (1, 3): 1000.0, (2, 3): 1000.0}
```

[ ]: