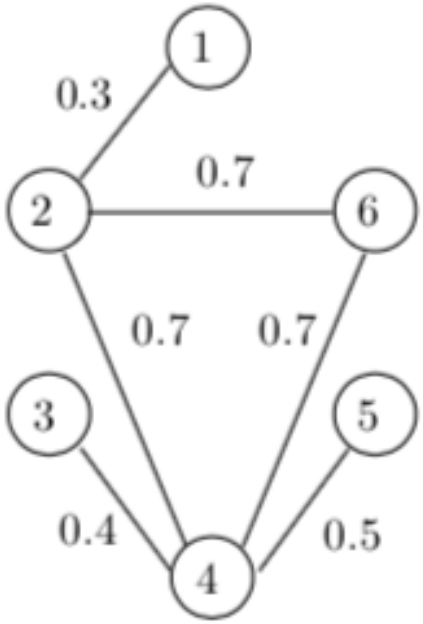
Algorithmic Methods for Mathematical Models

Roger Garcia Nevado, Mohana Fathollahi

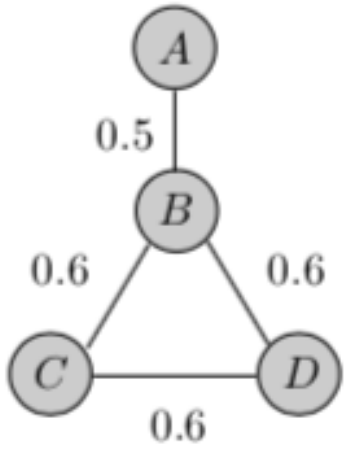
December 2021

Problem Description

- Finding a subgraph in Image
- Has same structure with shape
- Assign vertex of shape to vertex in image
- Minimize total of absolute value of difference between weight in shape and image.



Image



Shape

Integer Linear Programming

Decision variables

$$a_{kl} = \begin{cases} 1 & \text{if vertex } k \text{ assigned to vertex } l \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$zX_{kblv} = \begin{cases} 1 & \text{vertex } k \text{ assigned to } l \text{ and } b \text{ assigned to } v \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Auxiliary variables

$$gX_{lv} = \begin{cases} 1 & if \quad G_{lv} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$hX_{kb} = \begin{cases} 1 & if \quad H_{kb} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Integer Linear Programming

Objective function

$$minimize \sum_{k=1}^{m} \sum_{l=1}^{n} \sum_{b=1}^{m} \sum_{v=1}^{n} cost_{klbv}$$

Integer Linear Programming

Constraints

- All of vertices in shape will be mapped to Image

$$\sum_{l=1}^{n} a_{kl} = 1 \qquad \forall k \in m \qquad (1)$$

- we do not need to use all of vertices in Image

$$\sum_{k=1}^{m} a_{kl} <= 1 \quad \forall l \in n \qquad (2)$$

-

$$a_{kl} = 1 \wedge a_{bv} = 1 \implies H_{kb} = G_{lv}$$

$$(1 - a_{kl}) + (1 - a_{bv}) + (H_{kb} = G_{lv}) \geq 0 \qquad (3)$$

Integer Linear Programming

Constraints

- Cost of each mapping depend on

$$cost_{klbv} = zX_{klbv} * |H_{kb} - G_{lv}| \quad (4)$$

- $$zX_{klbv} = a_{kl} \wedge a_{bv}$$

$$0 \leq a_{kl} + a_{bv} - 2 * zX_{klbv} \leq 1, \forall k \in m, \forall l \in n, \forall b \in m, \forall v \in n (5)$$

Analyze the result of ILP

- Effect of load in feasibility and time
- Effect of density in feasibility and time

V.I	E.I	W.S	F.S	load	Density.I	Density.S	Time
10	20	7	6	0.7	0.44	0.29	30 min
10	20	6	13	0.6	0.44	0.87	10 min
10	27	6	9	0.6	0.6	0.6	1.3 second

- Size of problem

load	Size	Time(second)
0.4	5,270,960	0.07
0.4	1,319,633,280	41
0.3	10,875,648	1.15
0.3	418,553,529	19

Greedy algorithm pseudocode

```
#Algorithm 1 - Greedy
imageCycles <- getCycles(0, n, G)
shapeCycles <- getCycles(0, m, H)
usedShapes <- []
assignments <- {}
while (assignments.size() != m)
    if ((shapeCycles.size() > 0 && imageCycles.size() > 0) && (max(shapeCycles.size()) > max(imageCycles.size())))
        then return INFEASIBLE
    else
        if (assignments.size() == 0 && shapeCycles.size() != 0)
            currentShapeCycle <- shapeCycles[max(shapeCycles.size())][0]
            bestCycleAssignment <- getBestFeasableCycle(currentShapeCycle, imageCycles)
            if (bestCycleAssignment)
                assignments[bestCycleAssignment[shape]] = bestCycleAssignment[image]
                usedShapes.add(bestCycleAssignment[shape])
            else
                then return INFEASIBLE
        else
            if (assignments.size() == 0)
                result = getFirstAssignment()
                if (result)
                    assignments[result[shape]] = result[image]
                    usedShapes.add(result[shape])
                else
                    then return INFEASIBLE
            else
                remainingShapes <- assignments[shapes] - usedShapes
                result <- getFeasibleAssignment(remainingShapes, usedShapes, assignments)
                if (result)
                    assignments[result[shape]] = result[image]
                    usedShapes.add(result[shape])
                else
                    then return INFEASIBLE
return assignments

objectiveValue = getFinalObjectiveValue(assignments)
```

Figure: Figure: Greedy algorithm pseudocode