AMMM Final Project

Roger Garcia Nevada, Mohana Fathollahi

December 13, 2021

1 Problem statement

The goal of this project is finding a subgraph in Image that has same structure with shape. Therefore, we need to map vertices in shape to vertices in image. We may have several subgraphs, we should find a subgraph that difference between edges in subgraph of image and shape has minimum value. We will model this problem through Integer linear programming and heuristic models.

2 Integer Linear Model

2.1 Input data

Input data is the number of nodes in Image and Shape and adjacency matrix for each graph.
n Number of nodes in Image.
m Number of nodes in Shape.
G Adjacency matrix for Image.
H Adjacency matrix for Shape.

2.2 Decision variables

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

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

2.3 Auxiliary variables

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

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

2.4 Objective function

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

2.5 Constraints

- **Constraint1:**

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

- **Constraint2:**

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

- **Constraint3:**

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

- **Constraint4:**

$$cost_{klbv} = zX_{klbv} * |H_{kb} - G_{lv}|$$

- **Constraint5:**

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

In constraint1, we will guaranty that all of vertices in shape will be mapped to Image, additionally we do not need to use all of vertices in Image. Our goal is finding a sub graph in Image to be matched with shape therefore constraint 2 has been defined.

Constraint3: If vertex k mapped to vertex l and vertex b mapped to vertex v, therefore we must have an edge from k to b in shape and an edge from l to v in Image like equation (1), based on mathematics formula, $p \implies q$, is equal to $\neg p \vee q$ and we can conclude equation(2). Other equations ate trivial and we skip to describe them. At the end we get equation(3).
In mathematic format something like this:

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

$$\neg(a_{kl} = 1 \wedge a_{bv} = 1) \vee H_{kb} = G_{lv} \quad (2)$$

$$(\neg a_{kl} = 1) \vee (\neg a_{bv} = 1) \vee H_{kb} = G_{lv} \quad (3)$$

$$(1 - a_{kl}) + (1 - a_{bv}) + (H_{kb} = G_{lv}) \geq 1 \quad (4)$$

Constraint 4 and 5: To calculate cost of mapping, we should subtract weight of edges in both graph if their vertices are assigned, vertex k mapped to vertex l and vertex b mapped to vertex v. To make the problem simpler, a boolean variable assigned to situation that both mapping happen, $zX_{klbv} = a_{kl} \wedge a_{bv}$. Therefore, zX_{klbv} means that k is mapped to l and b is mapped to v. We need to write mathematical formula and based on equation (5) we can conclude constraint 4:

$$y = x_1 \wedge x_2 \wedge \ldots \wedge x_n \quad \implies 0 \leq x_1 + x_2 + \ldots + x_n - ny \leq n - 1 \quad (5)$$

$$zX_{klbv} = a_{kl} \wedge a_{bv} \quad \implies 0 \leq a_{kl} + a_{bv} - 2zX_{klbv} \leq n - 1 \quad (Constrain4)$$

3 Heuristics

In Integer Linear programming we can find optimal solution but when the size of problem is medium or large, this method will be time consuming and because of that we used heuristic methods. In this project three heuristics methods have been applied, greedy, local search and GRASP.

3.1 Greedy

To write greedy algorithm, we considered cycles that may exist in Image and Shape. First we tried to find cycles and match them based on their degree. To assign vertices in shape to image, first we decide based on number of neighbors that they have and then we assigned based on minimum difference weight that we have. After assigning cycles, we should assign remaining vertices in shape base on similarity between number of neighbors in shape to number of neighbors in image and difference between weights.

```
#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 1: Figure: Greedy algorithm pseudocode

4 Cplex Performance

Graphs have been generated based on number of vertices and number of edges.
to dooooo Cplex could not find a feasible solution for some of combinations in 10 or 30 minutes. For example when load is equal to 0.7 and density in shape is smaller than density in Image (Experimen1). We fixed value of density in image to find where we can find a solution: based on Experiment2, the problem is infeasible when load decreases to 0.6 and the density in shape increases.
Therefore, we need to decrease load and density in shape, with decreasing load to 0.5 cplex can solve problem. In Experiment 3 and 4, we can see that when we have lower density in shape, time that we need to solve the problem will be decreased.
In experiment 6 we can see that with load of 0.6 feasible solution can exist if density in shape and image are equal.

Another question can be raised is that, do the number of vertices and edges affect on solving problem? In Experiment 7, we duplicated number of vertices in Image and increase density in image too. To avoid spending alot of time for running, we started with low load, 0.2. If we increase load to 0.5 like Experiment 9, cplex could not solve the problem with even smaller density in shape,

compare to Experiment 8. Therefore, we decreased load to 0.4 and density in shape to 0.36, to find a solution. Then size of problem has direct effect on solving time and finding feasible solution.

- V.I: Number of vertices in Image
- E.I: Number of edges in Image
- W.S: Number of vertices in Shape
- F.S: Number of edges in Shape

N	V.I	E.I	W.S	F.S	load	Density.I	Density.S	Time	Result	V* C
1	10	20	7	6	0.7	0.44	0.29	30 min	Infeasible	
2	10	20	6	13	0.6	0.44	0.87	10 min	Infeasible	3660*7216
3	10	20	5	7	0.5	0.44	0.7	80 milsecond	1.85	2550*5015
4	10	20	4	4	0.4	0.44	0.67	70 milsecond	0.45	1640*3214
5	10	27	6	9	0.6	0.6	0.6	1.3 second	2.58	3660*7216
6	12	30	4	5	0.33	0.45	0.83	1.15 second	0.68	2352*4624
7	20	99	4	3	0.2	0.52	0.5	2.40 second	0.08	6480*12824
8	20	99	6	8	0.3	0.53	0.52	19 second	0.9	14520*28826
9	20	99	10	19	0.5	0.52	0.42	11 min	Infeasible	40200*80010
10	20	99	8	10	0.4	0.52	0.36	41 second	3.28	25760*51228

Table 1: Compare different experiments

load	Size	Time(second)
0.5	12,788,250	0.08
0.5	3,216,402,000	600
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

Table 2: Compare time based on size of problem

5 Compare ILP and Huristic

In ILP for small graphs with less than 0.7 load, we could find global minimum in reasonable time. But when the size of problem increase we have large density or load we could not find a global solution. In heuristic we could solve problems faster than ILP, but not all kind of problems, just specific kinds. Quality of solutions in greedy algorithm is not as well as ILP because it gives us local solution.