10720 CS 512200 VLSI Design for Manufacturability

Programming Homework (Due: May 14, 2019)

1. Problem statement

Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Each vertex has a weight and each edge also has a weight. A subset $D \subseteq V$ of the vertices is called a *dominating set* if every vertex $v \in V - D$ is adjacent to at least one vertex in D. A subset $I \subseteq V$ of the vertices is called an *independent set* if no two vertices in I are adjacent to each other. A subset $P \subseteq V$ is called a *dominating independent set* if P is both a dominating set and an independent set. We also define $N[v] = \{u \in V \mid (v, u) \in E\}$. For each $v \in V$ $(e \in E)$, we are given an integer weight $w(v) > 0$ $(w(e) > 0)$. Our goal is to find a dominating independent set P in G that minimizes the following cost function:

$$F(P) = \sum_{u \in P} w(u) + \sum_{v \in V-P} \min\{ w(v,u) \mid u \in N[v] \cap P\}$$

As an example, consider the graph in Figure 1. The vertex weights are indicated inside the vertices and the edge weights are provided besides the edges. An input graph is shown in Figure 1a. Figure 1b shows a dominating independent set with cost equals to 7. The optimal solution is shown in Figure 1c with cost equals to 4.

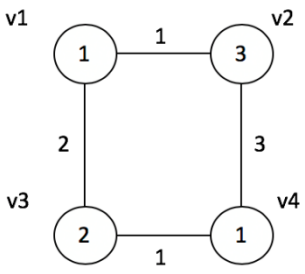

Fig.1a

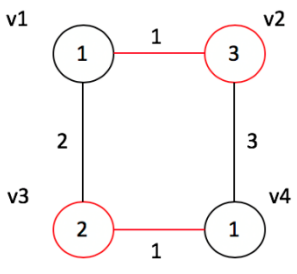

Fig.1b

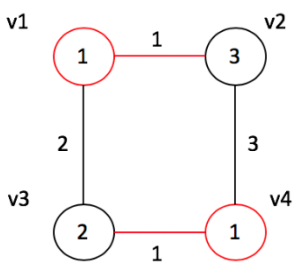

Fig.1c

2. ILP Formulation

We can use integer linear programing (ILP) to compute a dominating independent set that minimizes the cost function F(P). Please fill in constraints 3,4 and 5 and then write a C++ program to solve the ILP by using a tool called Gurobi Optimizer to help you.

min

$$\sum_{v \in V} x_v w(v) + \sum_{e \in E} z_e w(e)$$

Subject to

Constraint 1: $x_v + x_u \leq 1$, for $e = (u, v) \in E$

Constraint 2: $x_v + x_u = y_e$, for $e = (u, v) \in E$

Constraint 3:

Constraint 4:

Constraint 5:

Constraint 6: $x_v \in \{0,1\}$

Constraint 7: $y_e \in \{0,1\}$

Constraint 8: $z_e \in \{0,1\}$

Notations

x_v	1 if v is chosen for the solution
y_e	1 if e can be chosen
z_e	1 if $e \in E$ is selected for connecting a non-chosen node to a chosen one

3. Example and file formats

The input contain 2 files, where *.nv* file gives the names of the vertices and their corresponding weights, and *.as* file gives the edge weight between two vertices. Your program should produce a *.rpt* file which gives the name of each chosen vertex and the cost function F(P).

The format of each file is given below.

.nv

[Vertex name] [Weight]
[Vertex name] [Weight]
…

.as

[Vertex name] [Vertex name] [Weight]
[Vertex name] [Vertex name] [Weight]
…

.rpt

Cost function = F(P)
Dominating Independent Set:
[Vertex name]
[Vertex name]
…

The input files and output file of case1 is shown below.

Case1.nv

v1	1
v2	3
v3	2
v4	1

Case1.as

v1	v2	1
v1	v3	2
v2	v4	3
v3	v4	1

Case1.rpt

Cost function = 4
Dominating Independent Set:
v1
v4

The optimal solution is shown in Figure 1c where the cost function value is 4.

4. Program Command Format

There are some folders in the code folder. **Benchmark/** stores all the testcases, **checker/** stores a checker implemented by the TA using C++ language, **include/** should store the gurobi headers that you downloaded, **lib/** should store the gurobi libraries you downloaded, **result/** should store

the results your program generated, and **src/** should contain your main program using C++ language.

You should download the newest gurobi, get license, set the environment variables, place gurobi headers and libraries into **include/** and **lib/**.

Your program must be called “choose” and invoked in the following manner.

./choose [.nv][.as][.rpt]

For example:

./choose ../benchmark/case1.nv ../benchmark/case1.as ../result/case1.rpt

A checker is provided. Invoke it as follows.

./checker [.nv][.as][.rpt]

For example:

./check ../benchmark/case1.nv ../benchmark/case1.as ../result/case1.rpt

If your result is correct, it should print the messages as follows. Note that the checker only checks if your result is a dominating independent set. It will not check whether your result is optimal or not.

```
Macde-MBP:checker liawrush$ ./check ../benchmark/case1.nv ../benchmark/case1.as ../result/case1.rpt
Read .nv file...
Read .as file...
Read .rpt file...
Check if the number of vertices is matched...
Check if the total value is matched...
Check if the solution is feasible...
Your result satisfy the dominating independent set
```

5. Submission

A. code.tar.gz containing your compliable and runnable source code.

B. A report.pdf describing how your ILP formulation for solving this problem.