Balanced Graph Bi-partitioning

Electronic Design Automation (積體電路電路設計自動化)

元智大學資工系

Computer Science and Engineering

Yuan Ze University

Rung-Bin Lin

Date out 12/06/2023 Date due 01/16/2024

1. Purpose

Make students familiar with circuit partitioning problem and integer linear programming modeling.

2. Problem Description

This project has three parts as described below.

Part I:

Develop an integer linear programming model to solve the balanced graph bi-partitioning problem. We assume that the edge weight is 1 and vertex weight is also 1. Solve the model using a GNU mixed integer programming solver called **lp_solve** (https://lpsolve.sourceforge.net/5.5/). The following is the format of the input file that describe a graph (circuit).

```
#-of-edges #-of-nodes
N1 N2
N20 N10
…
…
```

Where #-of-nodes denotes the number of nodes in the graph and #-of-edges denotes the number of edges. Ni, i=1, 2, …,N#-of-nodes, denotes a node index. A line "N1 N2" means that there is an edge between N1 and N2. For an integer linear programming model, your constraints should be linear and your objective should also be linear. You may have encounter some non-linear terms, but you should try to transform the linear terms into non-linear ones. Your program should generate a model file with *lp_file_format*. For example,

```
min: x + y;
c1: x >= 6;
c2: y >= 6;
c3: x + y <= 11;
```

Part II:

Use an open source free software *shmetis* from http://glaros.dtc.umn.edu/gkhome/metis/metis/overview?q=metis/hmetis/overview to do the same job.

Part III:

Write a piece of code to implement *Simulated Annealing* to solve the same problem.

3. Results and Report

have to hand in the followings:

1. A piece of program for generating the 0/1 integer linear programming model for each graph. Name your program **yourStudentIDNumber_LP.cpp** if your program is written in c++. If your program is written in other language, replace .cpp with other extension. You need to provide some comments to your code.
2. A piece of program code for Simulated Annealing. Name your program **yourStudentIDNumber_SA.cpp** if your program is written in c++. If your program is written in other language, replace .cpp with other extension. You need to provide some comments to your code.
3. You will be asked to submit a report for this project. The report should provide the following content.

Integer Linear Programming, hMetis, and Simulated Annealing for Graph Bi-Partitioning

Electronic Design Automation
Computer Science and Engineering Department
Yuan Ze University
First Semester, 2023
Rung-Bin Lin (replaced by your name)

I. Problem Description

Describe the problem you solve in this section.

II. The 0/1 Integer Linear Programming Formulation of Graph Bi-partitioning

Present your formulation that consist of the objective function, problem constraints, and variable constraints. You have to linearize any nonlinear term into a linear formulation.

III. Simulated Annealing Implementation

Provide a piece of pseudo code for the SA you implement. Except the SA code body, your pseudo code should also provide the following information.

a. Initial temperature

b. Temperature annealing coefficient
c. The number of solutions being examined in an inner loop of SA
d. The final temperature
e. Best cost

Your SA program should produce an output like that shown below.

```
Enter graph input file name: s27o
Initial temperature: 10   Annealing coefficient: 0.98
Initial cost = 9
Cost at temperature 10:  12
Cost at temperature 9.8:  12
Cost at temperature 9.604:  12
Cost at temperature 9.41192:  12
Cost at temperature 9.22368:  15
Cost at temperature 9.03921:  10
Cost at temperature 8.85842:  8
Cost at temperature 8.68126:  15
Cost at temperature 8.50763:  9
Cost at temperature 8.33748:  11
```

.

.

.

```
Cost at temperature 0.279908:  3
Cost at temperature 0.27431:  3
Cost at temperature 0.268824:  3
Cost at temperature 0.263447:  3
Cost at temperature 0.258178:  3
Cost at temperature 0.253015:  3
Cost at temperature 0.247954:  3
Cost at temperature 0.242995:  3
Cost at temperature 0.238135:  3
Cost at temperature 0.233373:  3
Best cost: 3   Final temperature: 0.228705
```

Besides the above pseudo code, you should also explain

a. Cost function
b. How you purturb the current solution to obtain a new solution, i.e., the purturbation procedure (function) for SA.
c. How to determine when an SA run reaches a frozen state.

IV. Experimental Results

You should fill in the following table where *# of cnstr* denotes the number of constraints, *# of var* denotes the number of variables, *# of non-zero vars* denotes the number of non-zero variables, and *rTime* denotes runtime of your ILP model, *shmetis*, or SA. If you cannot obtain a solution for an ILP model within two hours, just provide *the best cost* (if available) obtained so far and the *time spent*. *You must perform an analysis of cost and runtime data of each approach.* Besides, you should also provide a clip of output for a graph respectively from *SA*, *lp_solve*, and *shmetis*. For *SA*, if the output has too many lines, just provide the output of the first 10 and the last 10 temperatures as shown above. Similarly, if there are too many output lines from *lp_solve*, you should provide the head part and the tail part (if available) of the output, just like that for *SA*. Compress your program code files and your report file (in PDF) to a file **yourStudentIDNumber.zip** and upload it to the course portal.

```
Model name:  'LPSolver' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size:     1060 constraints,     525 variables,         2643 non-zeros.
Sets:                                   0 GUB,                  0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.


Relaxed solution                  0 after       1040 iter is B&B base.

Feasible solution                66 after       1485 iter,        26 nodes (gap 6600.0%)
Improved solution                65 after       1585 iter,        38 nodes (gap 6500.0%)
Improved solution                63 after       3685 iter,       288 nodes (gap 6300.0%)
Improved solution                61 after       9443 iter,       536 nodes (gap 6100.0%)
Improved solution                56 after      11985 iter,       741 nodes (gap 5600.0%)
Improved solution                55 after      15344 iter,       852 nodes (gap 5500.0%)
Improved solution                52 after      17816 iter,       973 nodes (gap 5200.0%)
Improved solution                51 after      66962 iter,      6840 nodes (gap 5100.0%)

Optimal solution                 51 after      88297 iter,      8038 nodes (gap 5100.0%).
```

Graph			**0/1 ILP model**					**SA**		**shmetis**	
Graph name	**# of nodes**	**# of edges**	**# of cnstr**	**# of vars**	**# of non-zero vars**	**Best cost**	**rTime (sec.)**	**Best cost**	**rTime (sec.)**	**Best cost**	**rTime (sec.)**
s27o											
s298o											
s386o											
s510o											
s953o											
s1448o											
s5378o											
s13207o											
s15850o											
s35932o											
s38417o											
s38584o											