Combinatorial Algorithms
CoContest Semester Project Assignment:
Kidney exchange problem

Industrial Informatics Research Center
https://industrialinformatics.fel.cvut.cz/

March 1, 2023

Abstract

This document introduces the assignment for the CoContest semester project.

1 Motivational Example

For a long time, our colleague Theodor suffered from a chronic kidney disease, which resulted in the failure of his kidneys to function correctly. Since then, he had to visit the hospital every second day, where he was connected to a dialysis machine for many hours. Long hours on dialysis interfered with his life, and he became increasingly unhappy.

After seeing him sad, his wife, Sofia, decided to donate him one of her healthy kidneys. Unfortunately, after performing some medical tests at a hospital, the doctors found out that Theodor and Sofia were not biologically compatible, meaning that Theodor's immune system would attack the donated kidney from Sofia. After some time, the kidney would not work anymore, and Theodor would have to be on dialysis again.

However, the doctors at the hospital offered Theodor and Sofia an alternative: to join an international kidney exchange program. There are many similar pairs in the program as Theodor and Sofia, who are not biologically compatible but might be compatible across pairs. So, Theodor would get a kidney from a compatible donor from a different country, and Sofia would donate her kidney to another compatible recipient.

2 Formal Problem Statement

Let $P = \{p_i \mid i \in \{1, \ldots, n\}\}$ be a set of n donor-recipient pairs. Let $G = (P, E)$ be a directed graph, where edge $(p_i, p_j) \in E, i \neq j$ denotes that donor of pair p_i is *compatible* with the recipient of pair p_j (note that compatibility between pairs is not necessarily symmetrical). Let $\omega(e) \in [0, 1]$ be the preference weight for edge $e = (p_i, p_j) \in E$, denoting how much it is preferable that the recipient of p_j pair receive a kidney from the donor of p_i pair. Let $C(G)$ be a set of all directed cycles in graph G. Let $c_k, c_l \in C(G)$ be two different directed cycles in graph G; we say they are disjoint if they do not share any edge or vertex. The goal of the kidney exchange problem is to find a set of mutually disjoint cycles $C' \subseteq C(G)$ which maximizes the sum of preferences of the performed compatible transplantations, i.e.,

$$\max_{C' \subseteq C(G)} \sum_{c \in C'} \sum_{e \in c} \omega(e) \tag{1}$$

subject to

$$\forall\, c_k, c_l \in C', k \neq l : c_k, c_l \text{ are disjoint} \tag{2}$$

$$\forall\, c \in C' : |c| \leq L \tag{3}$$

where $|c|$ denotes the number of edges in the cycle c and L is a bound on the maximum length of each cycle. The motivation behind limiting the maximal cycle length is that a dropout from a longer cycle would impact more participants (donors may change their minds), which is obviously undesirable (we prefer a certain balance between efficient and stable solutions).

2.1 Example

Consider six donor-recipient pairs P with the mutual compatibility graph G given in Figure 1. The numbers on the edges are preference weights $\omega(e) \in [0, 1]$. Consider $L = 4$. Then, the optimal solution is depicted in Figure 2 with the highlighted edges in blue.

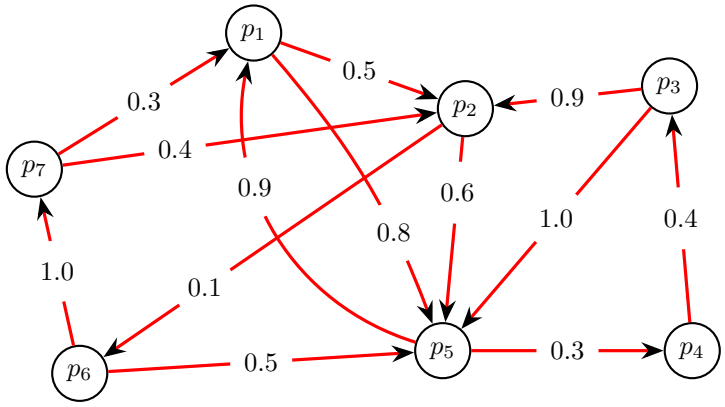

Figure 1: The input compatibility graph on donor-recipient pairs.

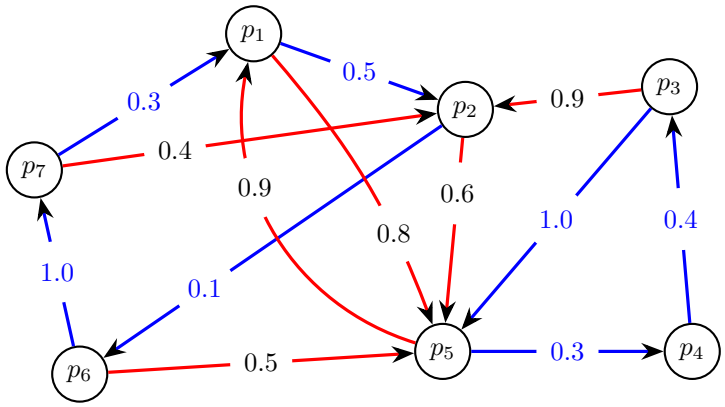

Figure 2: Optimal solution with the objective value of 3.6, $L = 4$.

3 Rules

If you choose the contest as your semestral project, you are expected to implement a correct solver for the Kidney Exchange Problem. BRUTE https://cw.felk.cvut.cz/brute/ will be used to evaluate it automatically. The number of submissions is not limited. The grading combines the ability to find optimal solutions for testing instances and the achieved rank relative to other students (w.r.t. the objective function) on competition instances. Therefore, you can acquire some points even if your solver is not very efficient relative to other students.

In BRUTE, you will find 3 tasks related to the contest. Each task has specific instances, rules, and grading. The contest is split into different tasks so that we avoid re-evaluation of the instances (which is time-consuming) and so that you can implement a specific solver for each task.

1. `SP_CC_O`: You have to implement an exact MILP solver for the problem. If your solver solves all the instances optimally in this task, you will get 5 points. If the solver returns a suboptimal solution for **any** instance in this task, then the evaluation of your solver is stopped, and you will get 0 points in this task.

2. `SP_CC_T`: the goal is to find the best possible feasible solution within the specified time limit, i.e., the optimal solutions are not required, and you are encouraged to implement clever heuristics to solve these instances. For each instance in this task, you will obtain some fraction of the point if the cost of the sum of preferences in your solution is not worse than our threshold (6 points at max).

3. `SP_CC_R`: The goal is to find the best possible feasible solution within the specified time limit, i.e., optimal solutions are not required, and you are encouraged to implement clever heuristics to solve these instances. Similarly, as in `SP_CC_T`, in this task, we are also interested in finding the best possible feasible solution within the specified time limit. The evaluation of your solver will depend on how good your solver is relative to other students' solvers, i.e., the number of points obtained will depend on your rank (4 points at max).

Some general contest rules also apply:

1. Usage of single-purpose problem-specific solvers is prohibited (i.e., a MILP solver is allowed, but somebody's else code for solving the Kidney Exchange Problem is not).

2. Every participant is required to write their own code. However, sharing ideas and discussion about the problem is encouraged.

4 Input and Output Format

In `SP_CC_O`, your solver will be called as

```
$ ./your-solver PATH_INPUT_FILE PATH_OUTPUT_FILE
```

whereas in `SP_CC_T` and `SP_CC_R` we include a time-limit

```
$ ./your-solver PATH_INPUT_FILE PATH_OUTPUT_FILE TIME_LIMIT
```

- `PATH_INPUT_FILE` and `PATH_OUTPUT_FILE`: Similarly, as in homework, these parameters represent the path to the input and output files, respectively (see below for a description of the file formats).

- `TIME_LIMIT`: A float representing the time limit in seconds given to your solver. Your solver will be killed after reaching the time limit, and you will be awarded 0 points. Hence, the output of your solver is considered only if your program exits with status code 0 before it timeouts.

The input file has the following form (we use one space as a separator between values on one line)

$$\begin{array}{lll} n & m & L \\ i_1 & j_1 & \omega_1 \\ i_2 & j_2 & \omega_2 \\ i_3 & j_3 & \omega_3 \\ \vdots & \vdots & \vdots \\ i_m & j_m & \omega_m \end{array}$$

where $n = |V|$, $m = |E|$, $e_k = (i_k, j_k) \in E$, and $\omega(e_k) = \omega_k$. ω_k is a float with up to 2 decimal place. Finally, L is a maximum length of the cycle allowed in the solution.

The output file has the following format

$$\begin{array}{ll} obj & \\ i'_1 & j'_1 \\ i'_2 & j'_2 \\ \vdots & \vdots \\ i'_k & j'_k \end{array}$$

where obj is optimal objective value (rounded to 2 decimal places) and $(i'_k, j'_k) \in E$ is an edge of graph G corresponding to a transplant that should be performed based on your solution.

Example 1

Input and output correspond to the example problem in Figs. 1, 2.

Input:

```
7 13 4
0 1 0.5
0 4 0.8
1 4 0.6
1 5 0.1
2 1 0.9
2 4 1.0
3 2 0.4
4 3 0.3
4 0 0.9
5 4 0.5
5 6 1.0
6 0 0.3
6 1 0.4
```

Output:

```
3.6
0 1
1 5
2 4
3 2
4 3
5 6
6 0
```

Example 2

Unfortunately, it may happen that the graph does not contain any cycle satisfying the constraints of our problem. However, your solution still needs to report this - even an empty set of edges is a valid solution.

Input:

```
8 15 3
2 7 0.3
2 3 0.06
2 6 0.16
2 5 0.78
7 3 0.56
7 6 0.63
7 4 0.2
7 1 0.57
7 5 0.41
7 0 0.77
6 4 0.43
4 1 0.28
4 0 0.26
1 5 0.5
5 0 0.66
```

Output:

```
0.0
```