Combinatorial optimization
CoContest semester project assignment: debt settlement problem

Industrial Informatics Research Center

March 24, 2017

Abstract

This document introduces the assignment for the CoContest semester project.

1 Motivational example

The members of Industrial Informatics Research Center group decided to throw a grilling party. After a short discussion, few "volunteers" are selected for visiting different shops and buying the stuff. The group agreed that once the volunteers come back from the shops and bring the bills, it will be decided how much each participant must pay to whom so that in the end, everybody will pay the same amount (it is a privately-funded party, the university will not cover even a single penny). To solve this problem, the smartest researcher of the group decides to formulate it as an combinatorial optimization problem with objective function that minimizes the number of money transactions between the participants.

As an example, consider that the group consists of four persons Attila, Bedřich, Nadezhda and Jaroslav. Bedřich bought some whiskey for 590, soft-drinks for 110 were bought by Nadezhda, Jaroslav went to buy some sausages for 300 while lazy Attila bought nothing. In total, the party costs $590 + 110 + 300 = 1000$ and every person has in the end pay $\frac{1000}{4} = 250$. To settle-up the debts in least number of transactions, Attila will give 200 to Bedřich, 50 to Jaroslav and Nadezhda will give 140 to Bedřich.

As you can see, this is a very practical problem, there are even applications for solving it, e.g. see Dlužníček app for Android[1].

2 Debt Settlement problem - formal statement

Let $P = \{1, \ldots, n\}$ be a set of persons and $B = \{1, \ldots, m\}$ be the set of bills. The cost of each bill $b \in B$ is denoted as $c_b \in \mathbb{N}_{>0}$ and function $\Omega : B \to P$ denotes the person who paid for the corresponding bill.

Let $s = \frac{1}{n} \sum_{b \in B} c_b$ be the settlement value, i.e. the amount of money that all persons must pay. The goal of the Debt Settlement problem is to find tuple (T, a) where $T \subseteq P \times P$ is the set of transactions and $a : T \to \mathbb{Q}_{>0}$ is a function representing the amount of money transferred in each transaction such that

$$\min_{T,a} \quad \sum_{p \in P} \sum_{q \in P} \mathbf{1}_T(p, q) \tag{1}$$

$$\text{s.t.} \quad \sum_{(p,q) \in T} a(p, q) + \sum_{b \in B\colon \Omega(b) = p} c_b - \sum_{(q,p) \in T} a(q, p) = s, \quad \forall p \in P \tag{2}$$

[1] https://play.google.com/store/apps/details?id=cz.destil.settleup

where $\mathbf{1}_X(x)$ is the indicator function

$$\mathbf{1}_X(x) = \begin{cases} 1 & x \in X \\ 0 & x \notin X \end{cases} \tag{3}$$

3 Rules

If you decide to choose the contest as your semestral project, then you are expected to implement a correct solver for the debt settlement problem. The implementation will be submitted to UploadSystem https://cw.felk.cvut.cz/brute/ where it will be automatically evaluated (number of submissions is not limited). The grading is combination of ability of finding good solutions and the achieved rank relative to other students (w.r.t. the objective function). Therefore, you can acquire some minimum number of points even if your solver is not very efficient relative to other students.

In UploadSystem, 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 specific solver for each task.

1. `SP_CC_O`: you have to implement an exact, MILP solver for the problem. If your solver solves optimally all the instances in this task, then you will get 3 points for this task. If the solver returns 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 solving these instances. For each instance in this task, you will obtain one point if the number of transactions in your solution is not worse than our threshold (4 points at max).
3. `SP_CC_R`: 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. However, 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 (3 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 Debt Settlement Problem is not).
2. every participant is required to write its own code. However, sharing ideas and other discussion about the problem is encouraged

4 Input and Output Format

Your solver will be called with the following three parameters, e.g.

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

where

- `PATH_INPUT_FILE` and `PATH_OUTPUT_FILE`: similarly as in homeworks, these parameters represent the path to the input and output files, respectively (see below for description of the file formats).
- `TIME_LIMIT`: an integer representing the time-limit in seconds given to your solver. Your solver will be killed after the time-limit is reached and the solution written in the output file (if any) will be taken as the result of your program.

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

$$\begin{array}{cccc} n & m & & \\ c_1 & c_2 & \dots & c_m \\ \Omega(1) & \Omega(2) & \dots & \Omega(m) \end{array}$$

The output file starts with the following line (again, one space as a separator between values on one line)

$|T|$

Then, for each transaction $(p, q) \in T$ there is one line in the following form (the order of the transactions is not important)

$p \quad q \quad a(p, q)$

Example 1

This example corresponds to the motivational example.

Input:

```
4 3
590 110 300
1 3 4
```

Output:

```
3
2 1 200
2 4 50
3 1 140
```