Combinatorial optimization Cocontest semester project assignment: Bureau Optimization Problem

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

February 27, 2020

Abstract

This document introduces the assignment for the Cocontest semester project.

1 Motivational example

In the year 2042, most of the useful work has been substituted by robots. To maintain the unemployment rate reasonably low, the number of people employed in offices in Czechia 4.0 has grown prominently. To give people a sense of usefulness, the level of bureaucracy was artificially increased in order to generate more paperwork to be processed by bureau workers.

At first, the plan worked perfectly, and soon it brought additional unforeseen benefits, such as newly generated job positions for gender studies graduates, more social interactions for awkward technical Ph.D. students, and skyrocketing profits of companies manufacturing the round stamps. However, at some point, the system has become unsustainable as it turned out that the bureau tends to generate more bureau, which led to a positive feedback loop. That is why the Ministry of Bureau Affairs was established to optimize the bureau processes and prevent the system from a complete breakdown. They appealed to Industrial Informatics Research Center (known for their love for bureaucracy) and asked them to help with this situation. Hence, the task is not to remove bureaucracy but make it more sustainable.

The government obtains each day a set of requests from the civilians, and each request consists of a set of fulfillments that must be passed in the given order. Such a fulfillment means that a civilian needs to visit a specific bureau worker with the application form to be filled in and given a stamp, which takes some time. Consider, for example, the situation in Figure 1. There, we have 9 bureaucrats and three citizens. The red citizen wishes to build a new cesspool, which requires a total of six specific stamps, such as from the department of the environment, lavatory chief, and many others. The navy blue citizen wishes to deposit additional funds to his university cafeteria card requiring visiting three specific bureaucrats. The sky blue one tries to obtain approval for enrolling for KO semestral test. However, each bureaucrat may serve one civilian at a time. What is somehow unusual is that giving stamps by a single bureaucrat on documents for two different citizens can take different amounts of time (which is perhaps a sign of corruption), and figuring out a good processing order becomes complicated. Hence, it appears to the Ministry of Bureau Affairs that optimizing the order of requests to be performed by bureaucrats might reduce overtimes, thus optimizing human costs. The goal is to assign the fulfillments in time such that the time for which the authorities are operating is minimized.

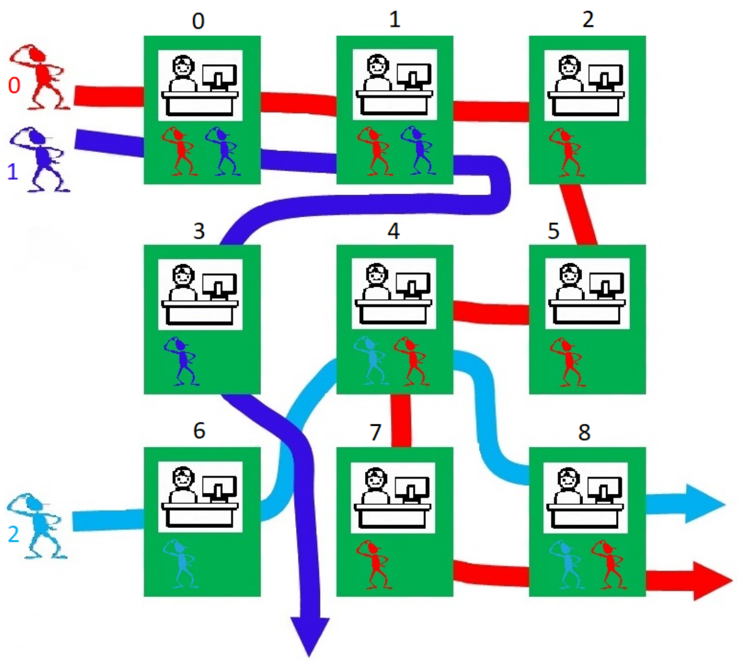

Figure 1: Example how the Ministry of Bureau Affairs completes agenda of the citizens.

2 Formal problem statement

You are given a set of bureaucrats $B = \{0, 1, \ldots, |B|-1\}$ and a set of citizens $C = \{0, 1, \ldots, |C|-1\}$. Each citizen $i \in C$ has its paperwork described by a sequence of bureaucrats $b_{i,1} \to b_{i,2} \to \ldots \to b_{i,k} \to \ldots \to b_{i,n_i}$, $b_{i,k} \in B$ with the expected durations for obtaining the proper stamps $d_{i,1} \to d_{i,2} \to \ldots \to d_{i,k} \to \ldots \to d_{i,n_i}$, $d_{i,k} \in \mathbb{N}$ by the corresponding bureaucrats. Once the work is started by a bureaucrat, he/she will finish the action without interruption. The paperwork of each citizen $i \in C$ has to be completed in the required order of bureaucrats while a bureaucrat can start stamping for citizen i after the preceding bureaucrat has finished stamping for i. The goal is to determine the order for each bureaucrat $k \in B$ in which to perform the required actions of citizens such that the finish time of the last bureaucrat is minimized.

Hence, the goal is

$$
\begin{aligned}
&\text{minimize}_{\pi_0,\ldots,\pi_{|B|-1}} \max\{F_0, \ldots, F_{|B|-1}\}\\
&\qquad \text{subject to:}\\
&\qquad\qquad (i)\ \pi_k \text{ is a permutation of citizens requiring stamps by } k \in B\\
&\qquad\qquad (ii)\ \text{bureaucrat } k \in B \text{ stamps documents in the order defined by } \pi_k\\
&\qquad\qquad (iii)\ \text{the agenda of each citizen } i \in C \text{ is completed in the given order}\\
&\qquad\qquad (iv)\ F_k \text{ is the finish time of bureaucrat } k \in B
\end{aligned}
$$

3 Rules

If you decide to choose the contest as your semestral project, then you are expected to implement a correct solver for Bureau Optimization Problem. The implementation will be submitted to BRUTE `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 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 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 some fraction of the point if the cost in your solution is not worse than our threshold (5 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 the single-purpose problem-specific solvers is prohibited (i.e., a MILP solver is allowed, but somebody's else code for solving Bureau Optimization 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

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 homeworks, these parameters represent the path to the input and output files, respectively (see below for 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 the time limit is reached and you will be awarded with 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):

$$\begin{array}{ccccc}
|C| & |B| & & & \\
b_{0,1} & d_{0,1} & \cdots & b_{0,n_0} & d_{0,n_0} \\
b_{1,1} & d_{1,1} & \cdots & b_{1,n_1} & d_{1,n_1} \\
b_{2,1} & d_{2,1} & \cdots & b_{2,n_2} & d_{2,n_2} \\
\vdots & \vdots & & \vdots & \vdots \\
b_{|C|-1,1} & d_{|C|-1,1} & \cdots & b_{|C|-1,n_{|C|-1}} & d_{|C|-1,n_{|C|-1}}
\end{array}$$

We ensure that each bureaucrat is required by at least one citizen.

The output file has the following format

obj

π_0

π_1

$\vdots$

π_k

$\vdots$

$\pi_{|B|-1}$

where obj is the finish time of the last bureaucrat and π_k is a permutation of citizens $C_k \subseteq C$ requiring the stamp from bureaucrat $k \in B$.

Example 1

This example corresponds to the motivation example.

Input:

```
3 9
0 10 1 4 2 3 5 18 4 9 7 3 8 6
0 24 1 15 3 18
6 26 4 12 8 17
```

Output:

```
67
0 1
0 1
0
1
2 0
0
2
0
2 0
```