Combinatorial Optimization - Project

Renaud Chicoisne
renaud.chicoisne@ulb.be

Cristian Aguayo
cristian.aguayo.quintana@ulb.be

The preferred treatment for kidney failure is transplantation. However, the demand is in general far greater than the supply. Successful transplantation of a kidney relies, among other factors, on tissue-type compatibility between the donor organ and patient. Compatibility is determined through a tissue-type crossmatch between a potential donor and patient's bloodtypes; if the two types differ substantially, the patient's body will reject the donor organ.

Typically, the donor organs come from deceased patients or from patients' close relatives. Complementing deceased donation is kidney exchange, which allows patients with a willing but medically incompatible living donor to swap their donor with other patients. If the success rate of each transplant is high, both patients are able to receive a viable transplant via the other patient's donor. Notice that even if both pairs were compatible, it can be sometimes possible to get more efficient transplants by exchanging donors. The process is - of course - generalizable to more than two pairs, forming *cycles* of compatible transplants.

In this project, we solve the *kidney assignment problem* via Integer Programming techniques, in the aim of maximizing the overall success rate of the transplantations.

1 A basic model

Consider n patient-donor pairs (p_i, d_i), and define c_{ij} as the compatibility of donor d_i with patient p_j. For any $i \in \{1, ..., n\}$, let $p(i)$ be the set of indices $j \in \{1, ..., n\}$ such that p_j is compatible with d_i. Let us consider the oriented network $G = (V, E)$ where

- Each node is either a patient or a donor.
- For each donor d_i, there is an edge (d_i, p_j), for each $j \in p(i)$, i.e. from a donor and all its compatible patients. The cost associated to this edge is $w_{d_i,p_j} = c_{ij}$.
- For each patient-donor pair (p_i, d_i), there is an edge (p_i, d_i). The cost associated to this edge is $w_{p_i,d_i} = 0$.

Consider for example $n = 6$ pairs with the compatibilities in Table 1. We obtain the network depicted in

	p_1	p_2	p_3	p_4	p_5	p_6
d_1	0.5	0.6				
d_2	0.8	0.4				
d_3				0.4		
d_4			0.5		0.5	
d_5			0.7			
d_6					0.9	0.8

Table 1: Compatibilities c_{ij}

Figure 1 where an example of kidney exchange assignment is drawn in blue: d_1 gives to p_2, d_2 gives to p_1, d_5 gives to p_3, d_3 gives to p_4, d_4 gives to p_5 and d_6 gives to p_6.

Seeing the compatibilities c_{ij} as *probabilities of successful surgery*, our problem is to find an assignment maximizing the number of transplants in *expected value*.

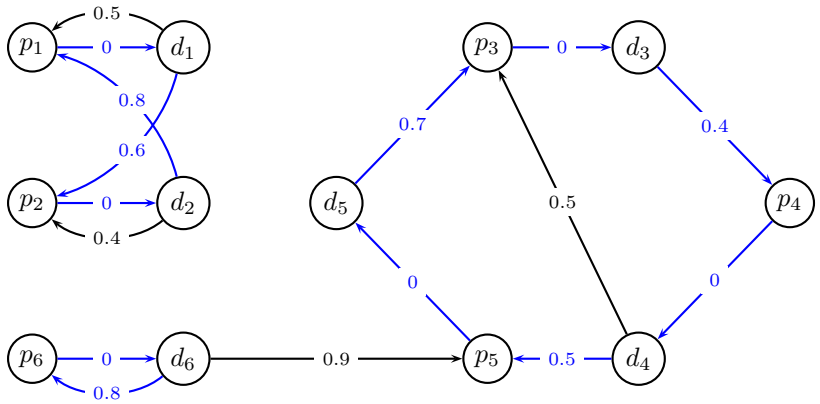

Figure 1: Instance's network and possible exchange (edge (u, v) labels: w_{uv})

1. Explain why the following Integer Programming (IP) formulation is valid for our problem: (variables, contraints, why the edges (p_i, d_i), etc)

$$\begin{aligned}
&\max_x \sum_{(u,v)\in E} w_{uv}x_{uv} \\
&\text{s.t.} \sum_{v:(u,v)\in E} x_{uv} - \sum_{v:(v,u)\in E} x_{vu} = 0, && \forall u \in V \\
&\quad x_{uv} \in \{0,1\}, && \forall (u,v) \in E
\end{aligned}$$

2. Explain why the IP is equivalent to the following Linear Program (LP) (i.e. they share exactly the same optimal value *and* optimal solutions)

$$\begin{aligned}
&\max_x \sum_{(u,v)\in E} w_{uv}x_{uv} \\
&\text{s.t.} \sum_{v:(u,v)\in E} x_{uv} - \sum_{v:(v,u)\in E} x_{vu} = 0, && \forall u \in V \\
&\quad x_{uv} \in [0,1], && \forall (u,v) \in E
\end{aligned}$$

3. Show that any feasible solution x is exclusively defined by *disjoint* cycles in G. I.e. the subgraph $(V, E(x))$ where $E(x) := \{(u, v) \in E : x_{uv} = 1\}$ can be decomposed in cycles that do not share any node with each other.

2 Positive cycle elimination procedure

We now see an algorithm to solve our problem that does not need any solver. Given a feasible x, let us define the residual network $G(x) := (V, F(x) \cup B(x)))$ as follows:

- *Forward edges:* $F(x) := \{(u, v) \in E : x_{uv} = 0\}$: those are edges that already exist in E, their cost stay the same $\bar{w}_{uv} = w_{uv}$
- *Backward edges:* $B(x) := \{(v, u) : x_{uv} = 1\}$: those are edges that are merely reversed edges from E, their cost is also reversed $\bar{w}_{vu} = -w_{uv}$.

It is possible to show that there is a cycle C in the residual network $G(x)$ having a strictly positive cost

$$\bar{w}(C) = \sum_{e\in C\cap(F(x)\cup B(x))} \bar{w}_e,$$

if and only if x is not optimal. More precisely, for any positive cost (w.r.t. $\bar{w}$) cycle C in $G(x)$, and defining:

$$x_{uv}(C) := \begin{cases} 0 & \text{If } (u,v) \notin C \text{ and } (v,u) \notin C \\ 1 & \text{If } (u,v) \in C \\ -1 & \text{If } (v,u) \in C \end{cases}$$

It is possible to prove that $x + x(C)$ is still feasible (within bounds $[0, 1]$ and satisfying the flow conservation) and has a strictly better objective value than x. This means that we can iteratively find positive cost cycles in the residual graph and update $x \to x + x(C)$ until no positive cost cycle can be found in the residual graph. We show the pseudocode of a positive weight cycle detection routine in an oriented graph in Algorithm 1.

Algorithm 1: Positive Weight Cycle Detection

Data: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edge weights $(\omega_e)_{e \in \mathcal{E}}$
Result: A positive weight cycle $\mathcal{C}$ if any.

$\mathcal{C} \leftarrow \emptyset$;
for $(u, v) \in \mathcal{V} \times \mathcal{V}$ **do**
  $d_{uv} \leftarrow \infty$; $\text{father}_{uv} \leftarrow 0$;
for $v \in \mathcal{V}$ **do**
  $d_{vv} \leftarrow 0$;
for $(u, v) \in \mathcal{E}$ **do**
  $d_{uv} \leftarrow -\omega_{uv}$; $\text{father}_{uv} \leftarrow u$;
for $w \in \mathcal{V}$ **do**
  for $(u, v) \in \mathcal{V} \times \mathcal{V}$ **do**
    if $d_{uv} > d_{uw} + d_{wv}$ **then**
      $d_{uv} \leftarrow d_{uw} + d_{wv}$; $\text{father}_{uv} \leftarrow \text{father}_{wv}$;
    if $u = v$ **and** $d_{uu} < 0$ **then**
      do
        $w \leftarrow \text{father}_{uv}$;
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{(w, v)\}$;
        $v \leftarrow w$;
      while $v \neq u$;
      return $\mathcal{C}$;
return There is no positive weight cycle in $\mathcal{G}$;

Implement the algorithm and check that the values match with the results obtained in the previous section. What is the worst case complexity of Algorithm 1?

3 Implementability

The last model lacks from its simplicity. In practice, it is impossible to perform a cycle of more than a couple dozens simultaneous transplants because of the logistics implied: in fact, all the surgeries must be performed close to each other because live organs have a very limited lifespan. Furthermore, by law, donors are not forced to commit to donate organs, meaning that if there is significant delay between surgeries and some donor finds a way to know if his/her partner patient has already been transplanted, there is no way to impeach him/her to retract, breaking the entire chain he was part of.

1. Given a maximum number M of exchanges possible *per cycle of transplants*, explain why the following IP formulation is valid for our problem:

$$\max_x \sum_{(u,v) \in E} w_{uv} x_{uv} \tag{1}$$

$$\text{s.t.} \sum_{v:(u,v) \in E} x_{uv} - \sum_{v:(v,u) \in E} x_{vu} = 0, \qquad \forall u \in V \tag{2}$$

$$\sum_{(u,v) \in C} x_{uv} \leqslant |C| - 1 \qquad , \forall C \text{ cycle of } G \text{ such that } |C| > 2M \tag{3}$$

$$x_{uv} \in \{0, 1\}, \qquad \forall (u, v) \in E \tag{4}$$

2. Given that the number of cycles in a graph is too large to include all of them from the start in the last formulation, we will omit the cycle constraints (3) and add them as we need them. Implement the following algorithm:

   (a) Start with a set of cycles $\mathcal{C} = \emptyset$

   (b) Solve the following relaxation

$$\max_{x} \sum_{(u,v)\in E} w_{uv} x_{uv} \tag{5}$$

$$\text{s.t.} \sum_{v:(u,v)\in E} x_{uv} - \sum_{v:(v,u)\in E} x_{vu} = 0, \qquad \forall u \in V \tag{6}$$

$$\sum_{(u,v)\in C} x_{uv} \leqslant |C| - 1 \qquad ,\forall C \in \mathcal{C} \tag{7}$$

$$x_{uv} \in \{0,1\}, \qquad \forall (u,v) \in E \tag{8}$$

   i. If the current solution contains a cycle C' performing strictly more than M exchanges, add C' to the pool $\mathcal{C}$, i.e. $\mathcal{C} \to \mathcal{C} \cup \{C'\}$ and go back to step (b)

   ii. Otherwise, return the solution at hand, which is optimal.

Guidelines and Evaluation

Guidelines

- The project must be carried out by groups of no more than three people.
- Besides this statement, you received 3 instances: `small.csv`, `normal.csv` and `large.csv`, as well as a `README.txt` file describing the data structure.
- The required models and algorithms must be implemented in `python` or `julia` along with the solver `Gurobi` when necessary. To use `Gurobi`, you have to use the `gurobipy` package for `python` and the `JuMP` package for `julia`. You can use other packages to read and manage files.
- The implementations must be done in **one and only one** of the mentioned programming languages.
- You are asked to prepare a report answering all theoretical questions presented in this project. Briefly discuss your implementation and your findings on the joint instances.

Evaluation

- You must submit the report in `.pdf` format, as well as the `.py` or `.jl` files used to solve the problem along with their respective instructions for use. The report must be written in english and it must contain the names and last names of the members of the group, and their respective ULB ID numbers.
- Verify that your codes work correctly before submitting them.
- The deadline is on June 1st, 2023 at 23:59.
- Any disrespect of project and/or submission guidelines will lead to penalization.