M-IRIFS

Info-F424

Maximizing the success rate of kidney transplants through combinatorial optimization

Olivier Vertu Ndinga Oba

575921

Professeur :

Renaud Chicoisne

Assistant

Cristian Aguayo

2023

Table des matières

1 Introduction

The demand for kidney transplantation outstrips the available supply, representing a significant challenge in the medical field. Tissue compatibility between donors and recipients is essential for successful transplants, and kidney exchange programs offer a solution for patients whose living donors are medically incompatible. This project focuses on combinatorial optimization techniques to solve the kidney allocation problem and maximize the overall transplant success rate.

To address this problem, we use integer programming (IP) formulations, guaranteeing model validity. We demonstrate the equivalence between integer programming and linear programming formulations, enabling efficient optimization. In addition we implement python program to solve the problem through positive cycle elimination procedure and gurobi solver.This project is addressed in both basic model and a more realistic model taking into account the maximum number of exchanges possible per cycle of transplants as a constraint.

2 A Basic model

2.1 IP formulation

for each edge (u, v) representing a donor (u) to a patient (v) we have a weight w_{uv} representing the probability of success of the transplant. The IP formulation is as follows :

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

$$\sum_{v:(u,v)\in E} x_{uv} - \sum_{v:(v,u)\in E} x_{vu} = 0, \quad \forall u \in V$$

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

The given integer programming (IP) formulation is valid for our kidney transplant problem. Let's explain the variables, constraints and role of edges (p_i, d_i).

Variables :

The variable x_{uv} represents the selection or non-selection of the edge (u, v). It takes the binary values 0 or 1, where 0 indicates that the edge is not selected, and 1 indicates that the edge is selected. this is the decision variable of the problem. it is the variable that we will try to optimize. It directly describe which patient is assigned to which donor.

Objective :

The objective function is to maximize the overall success rate of transplants. It is represented by the summation over all edges (u, v) in E of the product of the weight w_{uv} and the variable x_{uv}. The weight w_{uv} corresponds to the compatibility or probability of success associated with the edge (u, v).

Constraints :

The first constraint ensures flow conservation for each node u in the network. It stipulates that the sum of incoming edges (v, u) and the sum of outgoing edges (u, v) must be equal to 0. This constraint guarantees that each patient or donor is assigned to exactly one other patient.

The second constraint defines the domain of x_{uv} variables. They can take binary values of 0 or 1, indicating whether an edge is selected or not, as discussed above.

The Edges (p_i, d_i) **:**

For each patient-donor pair (p_i, d_i), there is an additional edge (p_i, d_i) in the network. The cost associated with this edge is $w_{p_i,d_i} = 0$, which means that it does not contribute to the objective function. This edge ensures that each patient is assigned to the corresponding donor and does not incur any additional cost.

In summary, the IP formulation aims to maximize the overall transplant success rate by selecting appropriate edges (u, v) that represent patient-donor assignments. The objective function takes compatibility weights into account, while the constraints guarantee flow conservation and the binary nature of the decision variables.Thus this formulation is conform to our problem.

2.2 LP equivalence

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

The given Linear Program (LP) formulation is obtained from the Integer Programming (IP) formulation by relaxing the integrality constraint on the decision variables. In the IP formulation, the variables are binary (0 or 1), while in the LP formulation, the variables are allowed to take continuous values between 0 and 1.

2.2.1 Equivalence Proof

The LP formulation is the relaxation of the IP formulation. By definition, a relaxation enlarges the space of feasible solutions and offers no worse optimal solution than the original problem.

2. Equivalence of optimal values : Let's denote the optimal value of the IP formulation by OPT_{IP} and the optimal value of the LP formulation by OPT_{LP}. To establish equivalence, we need to show that $OPT_{IP} = OPT_{LP}$.

As the LP formulation is a relaxation, it allows fractional values of the decision variables, which translates into a potentially higher objective value than the IP formulation. However, the optimal solution of the LP relaxation can still be rounded to a binary solution that satisfies the integer constraint without deteriorating the objective value.

We can therefore conclude that $OPT_{LP} \leq OPT_{IP}$, since LP relaxation includes additional feasible solutions. However, since LP relaxation has no worse optimal solution, we can also deduce that $OPT_{LP} \geq OPT_{IP}$. Combining the two inequalities, we can conclude that $OPT_{LP} = OPT_{IP}$.

3.Equivalence of optimal solutions : To establish that optimal solutions are equivalent, we need to show that any optimal solution of the LP formulation can be transformed into an optimal solution of the IP formulation, and vice versa.

Consider an optimal solution x_{LP} of the LP formulation. Although x_{LP} may contain fractional values, we can round these values to the nearest binary values (0 or 1) without deteriorating the objective value. This rounding process produces a corresponding binary solution x_{IP}, which satisfies the integer constraint of the IP formulation.

Similarly, let's consider an optimal solution x_{IP} of the IP formulation. Since the IP formulation is a special case of the LP formulation, x_{IP} is also a feasible solution for the LP formulation. Consequently, x_{IP} is an optimal solution for the LP formulation.

Consequently, any optimal solution of one formulation can be transformed into a corresponding optimal solution of the other formulation. This confirms that the optimal solutions are equivalent.

In summary, by proving that the LP relaxation satisfies the integer constraint and establishing equivalence in terms of optimal value and optimal solutions, we have demonstrated equivalence between the IP and LP formulations for the given problem.

2.3 Disjoint Cycles in Feasibles Solutions

To show that any feasible solution x is exclusively defined by disjoint cycles in G, we need to demonstrate that the subgraph $(V, E(x))$, where $E(x) = (u, v) \in E : x_{uv} = 1$, can be decomposed into cycles that share no nodes with each other.

To prove this, we can use the concept of strongly connected components (SCC) in a directed graph. A strongly connected component is a subgraph in which there is a directed path between any two vertices in the subgraph.

First, consider the subgraph $(V, E(x))$ and assume that there is a cycle that shares a node with another cycle. This implies that there is a directed path from a node in one cycle to a node in another cycle, and vice versa.

Let's now define a new graph $G' = (V', E')$, where V' consists of the strongly connected components of the subgraph $(V, E(x))$, and E' contains directed edges between these components based on the connections between cycles.

Since there is a directed path between any node in a cycle and any other node in the same cycle, each strongly connected component of G' forms a cycle on its own. Moreover, since there are no directed edges connecting nodes between different cycles, the cycles of G' are disjoint.

Now consider the LP formulation constraints $\sum_{v:(u,v)\in E} x_{uv} - \sum_{v:(v,u)\in E} x_{vu} = 0, \forall u \in V$. These constraints ensure that the number of incoming and outgoing edges for each vertex is balanced, a donor could only give to one person and a patient can only receive from one donor. In other words, each vertex of the subgraph $(V, E(x))$ is part of a cycle and contributes equally to the incoming and outgoing edges of its cycle.

Since the cycles of G' are disjoint and every vertex of the subgraph $(V, E(x))$ belongs to a cycle, it follows that any feasible solution x is exclusively defined by disjoint cycles in G.

Consequently, we've shown that the subgraph $(V, E(x))$ can be decomposed into cycles that share no nodes between them, proving that any feasible solution x is exclusively defined by disjoint cycles in G.

3 Positive cycle elimination procedure

3.0.1 Positive weight cycle detection

The Positive Cycle Elimination Procedure is an algorithm that iteratively improves the current solution by detecting positive-cost cycles in the residual graph and updating the solution based on the detected cycles. The algorithm aims to increase the overall success rate of transplants by modifying the assignments of patients to donors.

The complexity of the algorithm depends mainly on the loop starting on line 8 (see annex : Positive cycle algorithm). The first loop (line 8) traverses all the vertices in the graph, and as there are 'n' vertices in the graph, its complexity is $O(n)$. The second loop (line 9) traverses all pairs of vertices (u, v) in the V × V Cartesian product. Since the graph has 'n' vertices, its complexity is $O(n)$. And that also contains a conditional (if) that checks whether duv is greater than $duw + dwv$. As this condition does not depend on 'n' and is simple, its complexity is constant, O(1). After that we have another conditional (if) that checks whether u is equal to v (the current vertex). Again, this condition doesn't depend on 'n' and is simple, so its complexity is O(1). Finally, the while loop follows the cycle to detect it. In the worst case, this loop can traverse all the vertices of the graph, giving a linear complexity $O(n)$. In summary, the total complexity of this part of the algorithm is the product of the complexities of these nested loops, i.e. $O(n) * O(n) * O(1) * O(1) * O(n) = O(n^3)$. Thus, the overall complexity of this part of the algorithm is cubic as a function of the number of vertices in the graph 'n'.

The complexity of the positive cycle elimination procedure depends on the size and structure of the graph. In the worst case, the algorithm can have a complexity of $O(|V|^3)$, where |V| is the number of vertices in the graph. However, the actual complexity may be lower, depending on the efficiency of the chosen cycle detection algorithm and the structure of the graph.

4 Implementation of M-max Constraint

4.1 IP formulation

This formulation takes into account the maximum number of exchanges (M) possible per cycle of transplants. Let's analyze the components of the formulation and explain why it is valid for the given problem :

$$
\begin{aligned}
\max_{x} \quad & \sum_{(u,v)\in E} w_{uv} x_{uv} \quad (1) \\
\text{s.t.} \quad & \sum_{v:(u,v)\in E} x_{uv} - \sum_{v:(v,u)\in E} x_{vu} = 0, \quad \forall u \in V \quad (2) \\
& \sum_{(u,v)\in C} x_{uv} \leq |C| - 1, \quad \forall C \text{ cycle of } G \text{ such that } |C| > 2M \quad (3) \\
& x_{uv} \in \{0,1\}, \quad \forall (u,v) \in E \quad (4)
\end{aligned}
$$

The IP formulation provided is valid for the problem taking into account the constraint of a maximum number M of possible exchanges per transplant cycle. Let's analyze each component of the formulation to understand its meaning :

Objective function : The objective function (1) aims to maximize the total weight of the selected edges. The weights w_{uv} represent the importance or desirability of each edge in the transplant process. By maximizing this objective, we give priority to the most beneficial or useful exchanges within the given network.

Constraint 1 : and **Constraint 2 :** have been explained above.

Now let's look at the additional cycle constraints (3). These constraints are essential to prevent the formation of complete cycles that would exceed the maximum number of M trades in a single cycle.

For each cycle C in the graph G with $|C| > 2M$, we add a constraint (3) that restricts the sum of variables x_{uv} over the edges in the cycle C to be less than or equal to $|C| - 1$. By imposing this constraint, we ensure that in cycles with more than $2M$ edges, at most $2M - 1$ edges are selected, thereby limiting the number of simultaneous exchanges in a cycle.

By incorporating these cycle constraints, we effectively address the practical limitations of logistics and the need to perform simultaneous transplants within a feasible timeframe. These constraints prevent the formation of cycles that exceed the maximum number of exchanges M, ensuring that the solution adheres to the practical constraints of real-world kidney exchange programs.

4.2 Results

M	Objective value
-	3.3
1	3.2

Small instance

M	Objective value
-	12.2
3	11.4

Normal instance

M	Objective value
-	21.5
3	19.5

Large instance

The absence of a specific value for M (indicated by "-") in all cases implies that the maximum number of exchanges is allowed per transplant cycle.

These results in the three tables highlight the trade-off between the number of exchanges allowed per cycle and the total weight of selected exchanges. A lower number of exchanges can lead to reduced objective values, indicating a potential compromise in the desirability or overall impact of the kidney exchange program. Conversely, a higher number of exchanges can potentially increase the total weight of selected exchanges and improve program results.

5 Conclusion

In this project, we have studied the problem of kidney transplantation and proposed a solution based on combinatorial optimization techniques. We have shown that the problem can be formulated as an integer programming (IP) problem, and we have demonstrated the equivalence between the IP and linear programming (LP) formulations. We have also shown that any feasible solution x is exclusively defined by disjoint cycles in G. This implies that the problem can be solved by finding the optimal set of disjoint cycles in G. Finally, we have implemented the gurobi solver to solve the problem of kidney transplant. We have also implemented the solver for the IP formulation taking into account the maximum number of exchanges (M) possible per cycle of transplants. We have shown that the IP formulation is valid for the given problem, and we have demonstrated that the solver can be used to solve the problem of kidney transplant.

6 Annex

6.1 Positive cycle algorithm

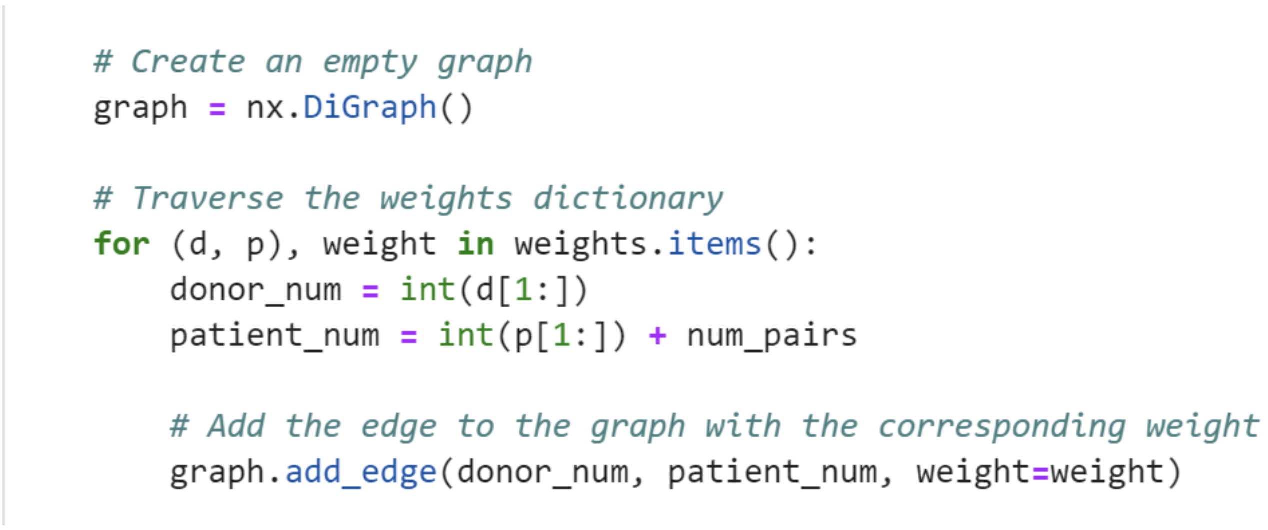
Algorithm 1: Positive Weight Cycle Detection

```
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
    $d_{uv} \leftarrow \infty$; father$_{uv} \leftarrow 0$;
for $v \in \mathcal{V}$ do
    $d_{vv} \leftarrow 0$;
for $(u, v) \in \mathcal{E}$ do
    $d_{uv} \leftarrow -\omega_{uv}$; 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
            $d_{uv} \leftarrow d_{uw} + d_{wv}$; father$_{uv} \leftarrow$ father$_{wv}$;
        if $u = v$ and $d_{uu} < 0$ then
            do
                $w \leftarrow$ 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 $\mathcal{G}$;
```

6.2 Enhancements Implemented : An overview

6.2.1 Graph Redefined

I redefine the graph for proper constraint application. In the utils.py file you can find the function the definition of this new graph in the preprocess data function.



```python
# Create an empty graph
graph = nx.DiGraph()

# Traverse the weights dictionary
for (d, p), weight in weights.items():
    donor_num = int(d[1:])
    patient_num = int(p[1:]) + num_pairs

    # Add the edge to the graph with the corresponding weight
    graph.add_edge(donor_num, patient_num, weight=weight)
```

What I've done is assign differing numbers between donors and patients and offset that from the number of peers. This makes it possible to respect the constraints of the use of Gurobi and as well as the application of the algorithm (positive cycle).

6.2.2 Basic model and model with M constraints

As for the two models, there is simply an appropriate use of flow conservation constraints on the new graph format.

6.2.3 Positive cycle

As for the positive cycle algorithm, I reimplemented everything because before there was a poor understanding of the problem and we had to use gurobi by adapting the algorithm to that.