Management Science

Group Assignment

Due date: November 30, 23:59

Instructions

1. Each group should submit a pdf file and a notebook. The pdf file should have all names of the group.

2. Your model code (but not pre-processing / post-processing code) **should also be included in the pdf using screenshots**. You should not assume we will read the notebook— but you should still submit one.

3. Handwritten equations are ok, as long as they are clear. Consider using Latexit (MACOS) or KLatexFormula (Windows) to typeset equations.

4. Points awarded are not necessarily proportional to the effort needed to answer a question.

5. Any questions on the assignment should be posted on the corresponding discussion forum in Canvas and not be sent via email.

6. Collaboration across groups is not allowed.

7. There is a limit on the size of problems you can solve with the 3-month license provided with Gurobi's installation. At least one member of each group will have to install a license (after

connecting to the university via VPN) following the instructions in the introductory slides of week 1.

Introduction

Nearly 0.3% of the population suffers from kidney failure. Patients with kidney failure that cannot get a transplant, must endure dialysis to clean their blood multiple times per week, which results in a severe reduction in their quality of life. Getting a transplant however is not easy, and patients often have to stay in dialysis for many years. Often a patient has a relative that is willing to donate a kidney, however it is not always the case that the potential donor and the recipient are compatible.

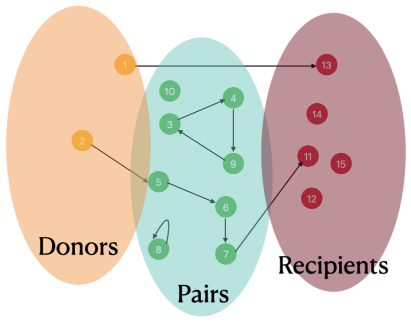
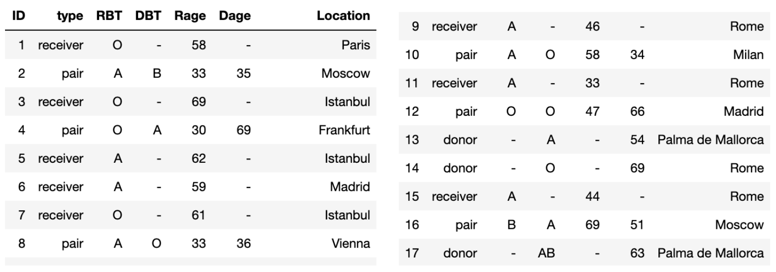


Figure 1: *Example of a donor-recipient assignment. The donor-recipient pair 8 has a direct transplant. The pairs 3,4 and 9 form a cycle. Donor 2 initiates a chain that terminates at recipient 11 and donor 1 donates directly to recipient 13. The pair 10 and recipients 12,14,15 did not manage to get a transplant.*

To increase the number of people that can get transplants, a strategy that emerged in recent years is to construct chains of donor-recipient pairs. In this assignment you will use your optimization skills to help towards this goal. Download the datasets , *"pairs.csv","distances.csv"*. The dataset "pairs.csv" contains information about donor-recipient pairs. It also includes (a few) un-paired



ID	type	RBT	DBT	Rage	Dage	Location
1	receiver	O	-	58	-	Paris
2	pair	A	B	33	35	Moscow
3	receiver	O	-	69	-	Istanbul
4	pair	O	A	30	69	Frankfurt
5	receiver	A	-	62	-	Istanbul
6	receiver	A	-	59	-	Madrid
7	receiver	O	-	61	-	Istanbul
8	pair	A	O	33	36	Vienna
9	receiver	A	-	46	-	Rome
10	pair	A	O	58	34	Milan
11	receiver	A	-	33	-	Rome
12	pair	O	O	47	66	Madrid
13	donor	-	A	-	54	Palma de Mallorca
14	donor	-	O	-	69	Rome
15	receiver	A	-	44	-	Rome
16	pair	B	A	69	51	Moscow
17	donor	-	AB	-	63	Palma de Mallorca

Figure 2: *First rows of* pairs.csv

donors that are willing to donate and (more) recipients that are hoping to get a transplant but do not have a relative willing to donate a kidney. Donor-recipient pairs only donate a kidney if they receive one.

In reality many factors contribute to the compatibility of a donor and a patient, such as the human leukocyte antigens. For the purpose of this assignment, however, you can assume a donor is compatible with a recipient if their blood types are compatible and their age difference is not above 10 years. In addition you will assume that donors can be matched only to recipients that are geographically close, in particular within a distance of 300 miles. The compatibility matrix of the donor blood type(DBT) and recipient blood type (RBT) is given in Table 1. The file *"distances.csv"* contains the distances of the cities in the file *"pairs.csv"*.

	RBT			
DBT	A	B	AB	O
A	x		x	
B		x	x	
AB			x	
O	x	x	x	x

Table 1: *Blood type compatibility Matrix. Donors with O are universal donors and patients with AB are universal receivers.*

1 Unconstrained chain length (85 points)

Use integer programming to find a matching that maximizes the number of patients that receive a kidney. Remember to express the fact that all donors and recipients can donate/receive at most one kidney and that donor-receiver pairs donate a kidney only if they also receive one.

1. **Mathematical modelling** Using mathematical notation, write down an integer program expressing the problem. Clearly explain the meaning of every group of constraints.

   Hint: Use binary variable x_{ij} to indicate that donor with ID i gives a kidney to a receiver with ID j. Use $\mathcal{D}$ as the set of un-paired donors, $\mathcal{P}$ for the set of donor-recipient pairs (that do not/cannot necessarily have a direct transplant) and $\mathcal{R}$ for the set of un-paired recipients. You can assume that the set $\mathcal{C}$ contains all potential matchings (i, j) that are compatible in terms of blood type, age and distance, including self-matchings (i, i), when blood types and ages permit it.

2. **Data Preparation** Read in the csv files using python. Prepare data in a convenient way to ease the implementation of your model in GUROBI.

   Hint: One, but not the only way, is to create a list *Arcs* that contains all (potential) matchings (i, j) that are compatible. The list should

   - include (8,8) as the donor and the recipient of the pair are compatible
   - include (10,11), (10,15) and (12,6), note that all conditions hold for those.
   - not include (4,4) as the donor and receiver are both blood and age incompatible.
   - not include (12,6) due to the age difference.
   - not include (13,5) due to the large distance.
   - $\cdots$

3. **Implement and solve in GUROBI** Implement your model in GUROBI and solve it.

   Hint: For every node (receiver/donor/pair), it might be useful to first compute the subsets of the (suggested from the previous hint) list *Arcs* that are inflows/outflows to the node.

4. **Analyze Solution** For the optimal solution

   (a) How many transplantations will take place according to your solution?

(b) What percentage of paired patients and what percentage of un-paired patients will receive a kidney?

(c) What percentage of un-paired donors donate a kidney?

(d) What percentage of paired patients receive a kidney for the different blood type combinations (RBT/DBT) of the pair? Visualize with a bar chart.

(e) What is the percentage of paired patients that receive a kidney for each city? Visualize with a bar chart.

5. **Analyze Solution 2** Re-calculate the same metrics if the acceptable distance is increased to 500 miles.

2 Avoiding long chains (15 points)

The operations required for a long series of transplants, either a cycle or a chain that initiates with a donor, must be asynchronous. This implies that donor receiver pairs, receive a kidney before they donate one. For example, in Figure 3, R3 received a kidney on July 24 2008 and the son of R3 donated a kidney a few months later on October 2 of 2008. Even though most of the time people keep their word and donate a kidney as promised, it is good practice to plan for a solution with relatively small chains that are more robust to cancellations, for whatever reason.

Your job is to modify your solution to ensure that it does not include any chains with more than 6 people. You will do that by detecting the long paths and repeatedly adding constraints to exclude them.

1. **Calculate longest path for current solution (300 miles)** Extract the solution from GUROBI in the form of a list of all active connections. What is the longest path in the current solution? You can use the following code to extract all paths.

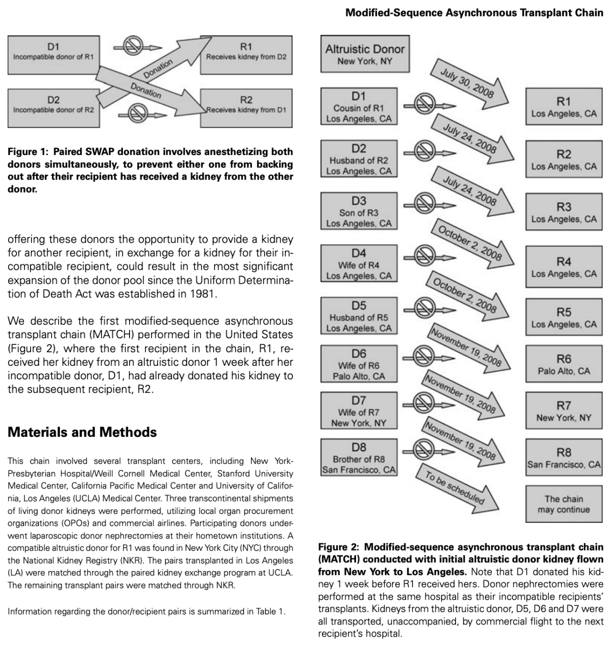


Figure 1: Paired SWAP donation involves anesthetizing both donors simultaneously, to prevent either one from backing out after their recipient has received a kidney from the other donor.

offering these donors the opportunity to provide a kidney for another recipient, in exchange for a kidney for their incompatible recipient, could result in the most significant expansion of the donor pool since the Uniform Determination of Death Act was established in 1981.

We describe the first modified-sequence asynchronous transplant chain (MATCH) performed in the United States (Figure 2), where the first recipient in the chain, R1, received her kidney from an altruistic donor 1 week after her incompatible donor, D1, had already donated his kidney to the subsequent recipient, R2.

Materials and Methods

This chain involved several transplant centers, including New York-Presbyterian Hospital/Weill Cornell Medical Center, Stanford University Medical Center, California Pacific Medical Center and University of California, Los Angeles (UCLA) Medical Center. Three transcontinental shipments of living donor kidneys were performed, utilizing local organ procurement organizations (OPOs) and commercial airlines. Participating donors underwent laparoscopic donor nephrectomies at their hometown institutions. A compatible altruistic donor for R1 was found in New York City (NYC) through the National Kidney Registry (NKR). The pairs transplanted in Los Angeles (LA) were matched through the paired kidney exchange program at UCLA. The remaining transplant pairs were matched through NKR.

Information regarding the donor/recipient pairs is summarized in Table 1.

Figure 2: Modified-sequence asynchronous transplant chain (MATCH) conducted with initial altruistic donor kidney flown from New York to Los Angeles. Note that D1 donated his kidney 1 week before R1 received hers. Donor nephrectomies were performed at the same hospital as their incompatible recipients' transplants. Kidneys from the altruistic donor, D5, D6 and D7 were all transported, unaccompanied, by commercial flight to the next recipient's hospital.

Figure 3: *Figures from article* ***Asynchronous, Out-of-Sequence, Transcontinental Chain Kidney Transplantation: A Novel Concept****, Butt et al. 2009. The operations of a long chain are asynchronous.*

```python
def computePaths(arcs):
    paths=[[arc] for arc in arcs]
    join=True
    while(join):
        join=False
        for i,p1 in enumerate(paths):
            if join==True:
                break
            for j,p2 in enumerate(paths):
                if i<j:
                    if p1[-1][1]==p2[0][0]:
                        p2=paths.pop(j)
                        paths[i]=paths[i]+p2
                        join=True
                    elif p1[0][0]==p2[-1][1]:
                        p2=paths.pop(j)
                        paths[i]=p2+paths[i]
                        join=True
    return paths

computePaths(activeArcs)
```

2. **Construct an iterative procedure to obtain a solution without long paths** Create a loop performing multiple iterations of:

   (a) Computing all paths that are longer than 3.

   (b) Add constraints to exclude them.

   (c) Resolving the problem

until no path longer than 3 is in the solution.

Hint: For example, if the solution contains the path

$$x_{01} \to x_{12} \to x_{23} \to x_{34} \to x_{45} \to x_{56}$$

you must write 3 constraints

- one to make sure that x_{01},x_{12}, x_{23}, x_{34} are not concurrently in the solution
- one to make sure that x_{12}, x_{23}, x_{34}, x_{45} are not concurrently in the solution
- one to make sure that x_{23}, x_{34}, x_{45}, x_{56} are not concurrently in the solution

3. **Analyze Solution** For the new optimal solution (300 miles)

   (a) How many transplantations will take place according to your solution?

   (b) How many constraints did your add? You can use the Gurobi attribute *NumConstrs* to query the number of constraints in a model.