1. ⓐ each saving is calculate by this formula

$$w(v_0v_k) + w(v_0v_{k+1}) - w(v_kv_{k+1})$$

and the demand of new route is the sum of customers demand

demand		v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
	v_0	-								
3	v_1	5	-	8	8	2	2	8	10	6
3	v_2	7	4	-	10	0	7	14	8	8
6	v_3	7	4	4	-	0	4	14	12	8
9	v_4	3	6	10	10	-	0	0	6	6
9	v_5	4	7	4	7	7	-	8	2	2
9	v_6	10	7	3	3	13	6	-	12	8
12	v_7	9	4	8	4	6	11	7	-	14
12	v_8	9	8	8	8	6	11	11	4	-

ⓑ

order saving by decreasing order: : feasible tour.

saving	14	14	14	12	12	10	10	
tour	v_2v_6	v_3v_6	v_7v_8	v_3v_7	v_6v_7	v_2v_5	v_1v_7	
saving	8	8	8	8	8	8	8	8
tour	v_1v_4	v_1v_3	v_1v_6	v_4v_7	v_2v_8	v_3v_8	v_5v_8	v_6v_8
saving	7	6	6	6	4	2	2	2
tour	v_2v_5	v_1v_8	v_4v_7	v_4v_8	v_3v_5	v_1v_4	v_1v_5	v_5v_7
saving	2	0	0	0	0			
tour	v_5v_8	v_2v_4	v_3v_4	v_4v_5	v_4v_6			

step 1 : merge $v_2 v_6$ → $(v_0 v_2 v_6 v_0)$. demand : 12 < 16

2 : merge $v_3 v_6$ → $(v_0 v_3 v_6 v_3 v_0)$

demand: 18 > 16 infeasible tour

3 : merge $v_2 v_3$ → $(v_0 v_3 v_2 v_6 v_0)$

demand : 18 > 16 infeasible tour

4. : merge $v_1 v_7$ → $(v_0 v_1 v_7 v_0)$ demand: 15 < 16

5 : merge $v_1 v_2$ → $(v_0 v_6 v_2 v_1 v_7 v_0)$

demand 27 > 16 infeasible tour

6 : merge $v_1 v_3$ → $(v_0 v_3 v_1 v_7 v_0)$

demand 21 > 16 infeasible tour

7 : merge $v_1 v_6$ → $(v_0 v_2 v_6 v_1 v_7 v_0)$

demand 27 > 16 infeasible tour

8 : merge $v_2 v_7$ → $(v_1 v_6 v_2 v_7 v_1 v_0)$

demand 27 > 16 infeasible tour

9 : merge $v_2 v_8$ → $(v_0 v_6 v_2 v_8 v_0)$

demand 24 > 16 infeasible tour

10 : merge $v_2 v_5$ → $(v_0 v_6 v_2 v_5 v_0)$

demand 21 > 16 infeasible tour

11 : merge $v_1 v_8$ → $(v_0 v_8 v_1 v_7 v_0)$

demand 27 > 16 infeasible tour

12 : merge $v_3 v_5$ → $(v_0 v_3 v_5 v_0)$ demand 15 < 16

13 : merge $v_1 v_4$ → $(v_0 v_4 v_1 v_7 v_0)$

demand 24 > 16 infeasible tour

14: merge $v_1 v_5 \rightarrow (v_0\ v_5\ v_1\ v_7\ v_0)$

demand $24 > 16$ infeasible tour

15: merge $v_2 v_4 \rightarrow (v_0\ v_4\ v_2\ v_6\ v_0)$

demand $21 > 16$ infeasible tour

16: merge $v_3 v_4 \rightarrow (v_0\ v_4\ v_3\ v_5\ v_0)$

demand $24 > 16$ infeasible tour

$S = \{ (v_0, v_1, v_7, v_0),\ (v_0, v_2, v_6, v_0),$
$(v_0, v_3, v_5, v_0),\ (v_0, v_4, v_0),$
$(v_0, v_8, v_0) \}.$

(c) we need at least

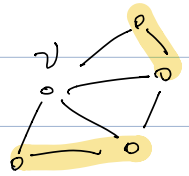
$$\left\lceil \frac{\sum_{v \in S} d_v}{C} \right\rceil = \left\lceil \frac{63}{16} \right\rceil = 4.$$

but according to the answer of (b), we need 5 trucks for 5 routes. And, this is the minimum of needed to serve all customers.

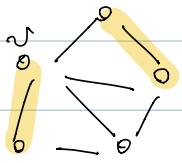
2.

(a) if v has only one neighbor.
the optimal solution of MM must contain v and its neighbor in order to maximize the matching value, so the value will be the same as the optimal solution to MMv.

if v has more than one neighbor,
the optimal solution to MM must be either contain all its neighbor without v itself



or include v and one of its neighbors as a match and not include another neighbor.



b. if the graph is a bipartite graph. (if there is no odd cycle).
the Ford Fulkerson algorithm should start augmenting path that flows through v.

if the graph is not a bipartite graph we can use greedy algorithm to find the maximal solution by starting with vertex v.

Otherwise, we can set an extra constraint in IP formulation of MM the sum of all edges that connect to v should be 1.

$$\sum_i X_{iv} = 1 \quad \forall i.$$

c. Graph $D(V, A)$

$V = \{$ set of patients $\}$.

$A = \{ (u, v)$: donor D_u is compatible with receiver $R_v \}$.

$$\max \sum_{a \in A} x_A$$

$$x(\delta^-(v)) - x(\delta^+(v)) = 0 \quad \forall v \in V$$

$$x(\delta^-(v)) \leq 1$$

$$x_{(v_i, v_j)} + x_{(v_j, v_k)} + x_{(v_k, v_h)} \leq 2$$

$\forall i, j, k, h \in V$ in different cycles.

$$\sum x(v_i) \geq \sum x(v_j)$$

$\forall i, j$ in the same cycle and $i < j$.

$x \in \{0, 1\}$.

3. longest chain problem. (longest distance).

v = the deceased donor

V = the set of all patients.

$x_{ij} = \begin{cases} 1, & \text{the edge between } i \text{ and } j \text{ is selected.} \\ 0, & \text{ow.} \end{cases}$

the longest chain problem is the same as the longest path problem but we have to try all possible combination of starting point (s) and destination point (t).

$$\max \sum_{i \in V \cup \{v\}} \sum_{j \in V \cup \{v\}} x_{ij}$$

$$\sum_i x_{vi} = 1 \qquad \forall i \in V \cup \{v\}$$

$$\sum_i \sum_j x_{ij} - \sum_j \sum_i x_{ji} = 0 \qquad \forall i, j \in V \cap \{t\}$$

$$\sum_i x_{it} = 1 \qquad \forall i \in V \cap \{t\}, \forall t \in V$$

4. a. $G(V, E)$ the two-way exchange graph

$X_{ij} = \begin{cases} 1, & \text{if the edge connects } i \text{ and } j \text{ is selected.} \\ 0, & \text{o.w.} \end{cases}$

$$\max \sum_{(i,j) \in E} X_{ij}$$

$$X(\delta(v)) \leq 1 \quad \forall v \in V$$

$$X \in \{0, 1\}.$$

the optimal solution is 50 according to Gurobi.

There are 50 matches in Two-Way Exchange.

b. V = the set of all patients

$A = \{(u,v)$: donor D_u is compatible with receiver R_v

$$\max \sum_{a \in A} x_a$$

$$x(\delta^-(v)) - x(\delta^+(v)) = 0 \qquad \forall v \in V$$

$$x(\delta^-(v)) \leq 1 \qquad \forall v \in V$$

$$x_{(v_i, v_j)} + x_{(v_j, v_k)} + x_{(v_k, v_r)} \leq 2$$

for all consecutively connected four patients i, j, k, r in V

.

the optimal result is 48.

there are 48 matches in the two and three-ways exchanges.

IEOR 4008: Computational Discrete Optimization 04/06/2020

Assignment 4

Deadline: 04/22/2020, h: 1:10pm

Unless stated otherwise, in solving those exercises you are allowed to assume as known every statement that has been proved or stated in class, as well as basic linear algebra / graph theory facts. If you are uncertain whether you can use some result, please ask. You can also reuse (portion of) the code uploaded on the webpage of the course.

Exercise 1

Consider the following instance of the Vehicle Routing Problem: The depot is denoted by v_0 and customers by $v_1, \ldots, v_8$. Demand of customers are, respectively: $3, 3, 6, 9, 9, 9, 12, 12$. The capacity of the trucks is 16. Distances between nodes are given by the table below.

	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_0	-								
v_1	5	-							
v_2	7	4	-						
v_3	7	4	4	-					
v_4	3	6	10	10	-				
v_5	4	7	4	7	7	-			
v_6	10	7	3	3	13	6	-		
v_7	9	4	8	4	6	11	7	-	
v_8	9	8	8	8	6	11	11	4	-

a) Compute the savings as in the Clarke-Wright algorithm.

b) Solve the instance above with the Clarke-Wright algorithm.

c) What is the minimum number of trucks needed to serve all customers? Motivate your answer.

Exercise 2

Recall that in class we saw that the Maximum Matching problem (MM) and the Maximum Weighted Matching problem (MWM) can be solved with efficient algorithms. Consider the following:

The Maximum Matching problem with a specific vertex (MMv)

Input: A graph $G(V, E)$ and a vertex $v \in V$.
Output: A matching M of G of maximum cardinality among those that contain v.

a) Show that, assuming v has at least a neighbor, the value of the optimal solution to (MM) is equal to the value of the optimal solution to (MMv) on the same input graph.

b) Give an algorithm that solves (MMv) (Hint: Reduce it to a problem you know how to solve)

c) In class we saw that we can formulate the problem of finding the maximum number of 2- and 3-way exchanges in a set of patients/donors as an Integer Program[1]. Now suppose that patients are ordered according to the time they spent waiting for a donation. This gives them priorities: patient v_1 has priority over v_2, which has priority over v_3, etc. We say that a solution S to the 2- and 3-way exchange problem *respects the priorities (RP)* if the following happens:

[1] See the formulation from page 7 of the lecture notes from the 16th and 17th lectures.

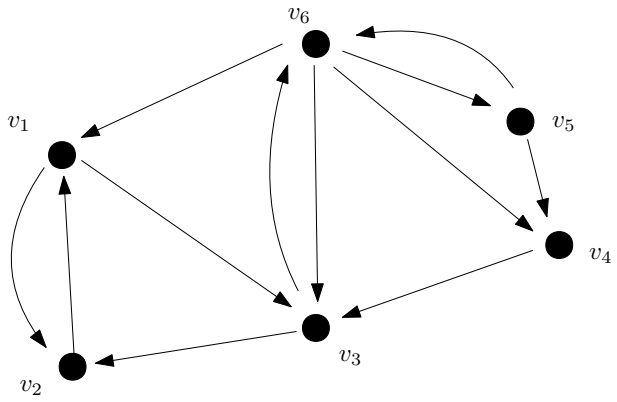


Figure 1: An instance of the Kidney exchange problem. Consider the following three solutions: A: The 3-way exchange v_1, v_2, v_3 plus the 2-way exchange v_5, v_6; B: The 3-way exchange v_3, v_6, v_4 plus the 2-way exchange v_1, v_2; B: The 3-way exchange v_1, v_3, v_6. Solution C does not achieve the maximum number of transplants, hence it is not RP. Both solutions A and B realize transplants for v_1, v_2, v_3, but A does not realize a transplant for v_4, while B does. Hence, A is not RP. One can check that indeed B is an RP solution.

- it achieves the maximum number of possible transplants with 2- and 3-way exchanges.
- take any other solution S' to the problem that achieves the same number of transplants with 2- and 3-way exchanges, and let v_i be the first (according to the priorities given above) vertex that receives a transplants only in one of S and S'. Then v_i receives a transplant in S and not in S'.

See Figure 1 for an example. Write an IP that finds a RP solution.

Exercise 3

Recall that, in the Kidney Exchange problem, deceased donors are voluntarily donating a kidney, without asking for one back. When a deceased donor v enters the market, it can sparkle a sequence of donations: v donates to v_1, whose donor donates to v_2, whose donor donates to v_3, etc. Finding the longest sequence of possible donation is called the **longest chain problem** (cfr. the notes and slides). Write an IP formulation for the longest chain problem.

Exercise 4

Consider the file kidney.py. It contains an instance of the Kidney exchange problem with 100 nodes stored as a list (called "data") of directed edges in the associated directed graph. Nodes with no incoming edges are deceased donors; all the others are patients, each with his/her own living donor.

a) Write an IP for the maximum possible number of 2-way exchanges in kidney.py, and solve it using Gurobi.

b) Write an IP for the maximum possible number 2- and 3-way exchanges in kidney.py, and solve it using Gurobi.

c) [**Optional, bonus points**] Now take into account the existence of deceased donors. Can you find a solution containing chains from deceased donors, 2- and 3-way exchanges that lead to more transplants than those found in part b)?