Matching Mechanism for Kidney Transplantations

Yutong Meng[a], Hongyu Zhou[b]

February 7, 2021

1 Direct Donation Algorithm and Greedy Algorithm

1.1 Q1

Algorithm 1.1: Direct Donation Algorithm

Input: Donor-Patient pair $\{k_i, t_i\}$, compatible kidney set $\{K_i\}$,a waiting list W

Output: An assignment of the patients to either a kidney or the waiting list

```
W is empty, M is empty.; // W is a priority list, M is a redimensional table to
 store matching pairs
for t_i in patient set do
    if k_i in K_i then
        Assign k_i to t_i
    end
    else
        Add t_i to W
    end
end
```

1.2 Q2

Greedy algorithm

Input:A priority list P of patients, preference P_i for each patient t_i,compatible kidney set $\{K_i\}$

Output:An assignment of the patients to either a kidney or the waiting list `/* patient is a class containing 4 domains:patient` t `and corresponding donor` k`,compatible set` K`,preference list Pr */`

M is empty; `// M is a redimensional table to store matching pairs` [1]
for *patient t in P* **do**
  if *t.k in t.K* **then**
    Add (t.t,t.k) to M; Remove t from P;
  end
end
while *P is not empty* **do**
  Remove the patient with the highest priority from P, denoted by t **while** *t.Pr is not empty* **do**
    Remove the option with the highest preference from t.Pr, denoted by op
    if *op=t or op=w* **then**
      Add t.t to the waiting list W
    end
    else if *op in P and t.k in the compatible set op.K* **then**
      Add (t.t,op.k) and (op.t,t.k) to M
    end
  end
end

2 Efficient strategy-proof exchange mechanism

2.1 Q3

Lemma 2.1. *Consider a graph, in which vertices are kidneys, patients and the waiting list w. Every patient points to either a kidney or to w. Every kidney points to its paired patient. Then either there exists a cycle, or each pair is the tail of some w-chain.*

Proof. For each kidney-patient pair (k, t), let c be the longest directed path starting from k(if it is not unique, we just randomly pick one).Let v be the end vertex of the path, then v is either w or another patient t'. If v is w, we conclude that the path is a w-chain and (k, t) is the tail. Otherwise,v is t'. Noting that w has no outedges, w

is not in the path.Since the path is the longest, t' cannot point to w. Suppose that t' points to a kidney k'', then k'' is in the path. No kidney can be assigned to more than one person, hence k''=k, which creates a circle where there is (k,t). □

2.2 Q4

Algorithm 2.1: Cycle and Chain Matching Algorithm Rule A

Input: Adjacent list D whose vertices are patients $V = \{t_i\}$ and a waiting list w without arcs

Output: An assignment of the patients to either a kidney or the waiting list

```
/* patient is a class containing 3 domains:patient t, corresponding donor k
   and corresponding preference list Pr,and an assignment op            */
```

W is empty; `// W is a waiting priority list`

$V' = V$; `// patient to be assigned`

while V' *is not empty* **do**

  for t *in* V' **do**

    t.op=the most prefered option in t.Pr

  end

  if *there exists a cycle denoted by* c **then**

    Remove the vertices in the cycle from V'; Remove the vertices in the cycle from the preference lists of the vertices in V';

  end

  else

    Find the longest w-chain starting from V' in a sense of priority; Remove the vertices in the w-chain from V'; Remove the vertices(except for the tail) in the w-chain from the preference lists of the vertices in V';

  end

end

Algorithm2.2: Cycle and Chain Matching Algorithm Rule B

Input:Adjacent list D whose vertices are patients $V = \{t_i\}$ and a waiting list w without arcs

Output:An assignment of the patients to either a kidney or the waiting list

```
/* patient is a class containing 3 domains:patient t, corresponding donor k and
   corresponding preference list Pr,and an assignment op                */
```

[1]

W is empty; `// W is a waiting priority list`

$V' = V$; `// patient to be assigned`

while V' *is not empty* **do**
  for t *in* V' **do**
    t.op=the most prefered option in t.Pr
  end
  if *there exists a cycle denoted by* c **then**
    Remove the vertices in the cycle from V' Remove the vertices in the cycle from the preference lists of the vertices in V' **else**
      p =the w-chain starting from the vertex in V' with the highest priority Remove the vertices in p from V' Remove the vertices(except for the tail) in p from the preference lists of the vertices in V'
    end
  end
end

3 Examples

3.1 Q5

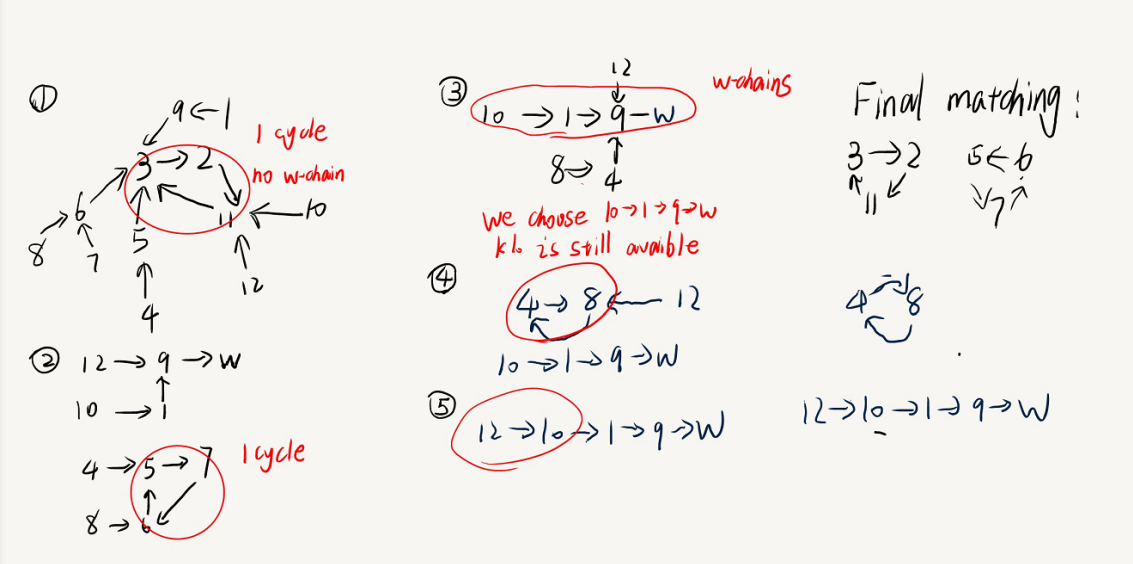

Figure 1: Illustration and final matching for Rule A

3.2 Q6

Theorem 1. *Consider a chain selection rule that keeps in the procedure any selected w-chain at a non-terminal round. With such a rule, the exchange mechanism is efficient.*

Proof. We note M the result obtained by applying the selection rules. We assume there exists another allocation M' more efficient than M. We use T to represent the ensemble of patients in M' who strictly prefer M' than M and t the patient in T with highest priority. If we look back to the procedures, t is either allocated to the remaining most preferable kidney, or allocated to waiting list.And by the hypothesis of t, we know that all the patients before t(higher priority) are allocated the same kidney in M or M'. So the kidney allocated to t in M' can not be more preferable than in M. This is absurde to the definition of t. so T is empty. Therefore the selection rule is efficient. □

3.3 Q7

Rule A is not strategy-proof.

Proof. We have an example with 6 patients and their preference list is
1:$k2 > k1$ 2:$k1 > k2$ 3:$k2 > w$
4:$k2 > k3 > k5 > k4$ 5:$k2 > k3 > k5$
6:$k2 > k5 > k6$ we have: situation 1
If Patient4 lies that his preference list is $k2 > k5 > k4$, we have situation 2

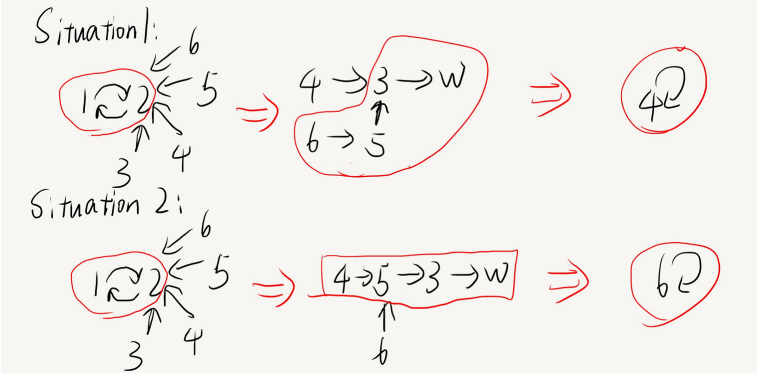

Figure 2: no lie vs lie

So Patient4 will be matched with the kidney5 rather than 4, he benefits from lying. Therefore, Rule A is not strategy-proof. □

3.4 Q8

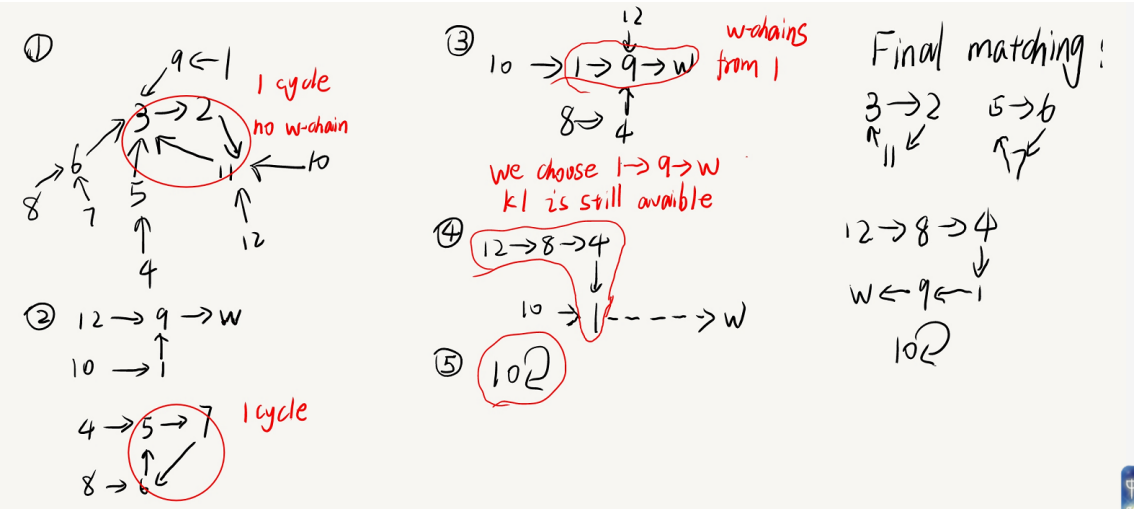

Figure 3: Illustration and final matching for Rule B

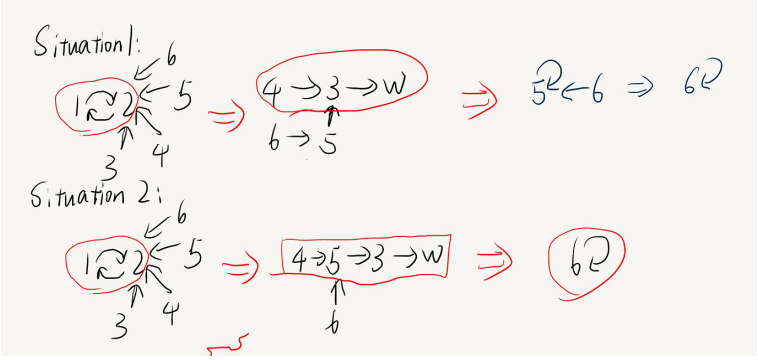

Figure 4: no lie vs lie

With the same example in Q7. This time we can see that patient 4 doesn't benefit from lying. He get 5 instead od 3.

4 Integer Linear programming

4.1 Q9

In this question,we use the BFS algorithm.

Algorithm4.1: Find Minimal Infeasible Paths(FMIP)

Input: Seuil K, A directed graph G. Output:All the minimal infeasible paths. /* */

[1] P=[] //P is a vector contains all the paths of length K
Map= is a hashmap containing all the k,P couples.
if *K is in Map.keys* **then**
    P=Map.find(K)
end
else if *K=1* **then**
    Add all the directed edges to P.;
    Add 1,P to Map
end
else if *K%2=0* **then**
    P1=FMIP(K/2);
    add k/2,P1 to Map;
    P=all the possible combination of paths in P1
end
else
    P1=FMIP(k/2),P2=FMIP(K-k/2);
    add k/2,P1 k-K/2,P2to Map;
    P=all the possible combination of paths in P1,P2
end
Return P;

Algorithm 4.2: DFS$_l imit$

Input: seuil,patient, A directed graph G Output:All the minimal infeasible paths.
/* */

[1] P=[],P is a vector contains all the paths of length K
if *seuil==1* **then**
    for *all the neighbers n for patient* **do**
        if n hasn't been visited; Q=DFS$_l imit(seuil-1, p, g)$;
        connect the paths in Q to patient to formulate P
    end
    return P;
end
else if *seuil*≥ 2 **then**
    for *all the neighbers n for patient* **do**
        if n hasn't been visited;
        P.add(the path containing only n);
    end
    return P;
end

```
for patient p:G.vertices do
    Q=DFS_limit(data.seuil, p, g);
    for Path pa:Q do
        P.add(pa);
    end
end
Return P;
```

4.2 Q10

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

We have the constraints as follows:
1.$max \sum_{(i_0,j\in E)} X_{i_0,j} \leq 1$ for all $i_0 \in V$.
2.$max \sum_{(i,j_0\in E)} X_{i,j_0} \leq 1$ for all $j_0 \in V$.
3.$\sum_{(i_0,j\in E)} X_{i0,j} = \sum_{(i,j_0\in E)} X_{j,i0}$ for all $i_0 \in V$.
4.$\sum_{e\in p} X_e \leq K-1$ for all $p \in P[K+1]$.
5.$X_{i,j} \in 0, 1$.

4.3 Q11

Algorithm4.2: Branch-and Bound Algorithm

```
Input: lpsole problem, edges Q, depth
Output:Non because the reslut is stored in problem(pass by adress /* E=ensemble of
   edges in the graph                                                    */
/* depth=the depth of problem in the BB tree(the number of extra-constraints we
   suppose on the edges.                                                 */
/* problem= we create a class containing the basic constraints in question 10
   mainly using the lpsolve class                                        */
/*                                                                       */
if the problem has no solution or depth≥ Q.size then
    return;
end
else if obj¿lowerBound then
    return;
end
else if depth==Q.size()-1 then
    lowerBound=obj;
    note the solution;
    solver.solve();
    return;
```

```
80 end
81 create two new constraints for the depth-th edge.
82 cons1:depth-th edge=0;
83 cons2:depth-th edge=1;
84 problem1=problem.add(cons1);
85 problem2=problem.add(cons2);
86 Branch-and Bound(problem1,edges,depth+1);
87 Branch-and Bound(problem2,edges,depth+1);
88 return;
```

4.4 13

```
<terminated> ExamResults [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe
The transplantation rate for direct donation: 0.725
The transplantation rate for 2-way exchange 0.754
The transplantation rate for 2-way exchange without direct donation 0.886
The transplantation rate for rule B:0.7806666666666666
The transplantation rate for rule B without direct donation:0.778
The transplantation rate for rule A:0.781
The transplantation rate for rule A without direct donation:0.7826666666666
```

Figure 5: results for different algorithms

We can see that the transplantation rate becomes higher and higher from algorithm1 to algorithm2 to algorithm3A and 3B. So with cycle and w-chain we can conduct to better results. Moreover we can see that running the direct donation algorithm as a preprocessing step doesn't help other algorithms because direct donation will also be seen as a cycle for Algo3A and 3B, as quote(Option ki either means that ki is compatible with ti, in which case ki is the most preferred option) . For algo2 without propossessing, we see that the compatible rate is much higher, that's because those who have a compatible donor sacrfice someway in there preference. Finally we can see the gain allowing multiple-way change is about 78%-75%= 3% as shown in resluts.