Mechanism Design and Implementation for Lung Exchange*

Suiqian Luo and **Pingzhong Tang**
Institute for Interdisciplinary Information Sciences,
Tsinghua University, Beijing, China
luosq13@mails.tsinghua.edu.cn, kenshin@tsinghua.edu.cn

Abstract

We explore the mechanism design problem for lung exchange and its implementation in practice. We prove that determining whether there exists a non-trivial solution of the lung exchange problem is NP-complete. We propose a mechanism that is individually rational, strategy-proof and maximizes exchange size. To implement this mechanism in practice, we propose an algorithm based on Integer Linear Program and another based on search. Both of our algorithms for this mechanism yield excellent performances in simulated data sets.

1 Introduction

Barter exchange has been an important aspect of electronic commerce and multiagent system. Among many forms of barter exchanges, one major form is live organ exchange. Over the past decade, organ exchange has become a subject of intensive study in the AI and EC community, with the prominent example of kidney exchange [Roth *et al.*, 2004; Abraham *et al.*, 2007; Awasthi and Sandholm, 2009; Ashlagi and Roth, 2011; Dickerson *et al.*, 2012a; 2012b; 2013; Dickerson, 2014]. Nowadays, transplantation from kidney exchange accounts for about 10% of all the living donor kidney transplantations in the US [Ergin *et al.*, 2014]. Lately, such idea has been investigated across multiple different types of organs, including kidney and liver [Dickerson and Sandholm, 2014].

The idea of live organ exchange is as follow: a patient might be able to find a donor (or several donors) who is willing to donate an organ. Most often, the patient and the donor may have incompatible blood or tissue type, or fail a *cross-match test* [Blum *et al.*, 2013]. As a result, they may seek to swap organ with other pairs that suffer the same difficulty (and it is legal in some countries). After the exchange, the pair donate an organ to help some other compatible patient in the system, while obtaining a compatible organ in return.

*This work was supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the Natural Science Foundation of China Grant 61033001, 61361136003, 61303077, a Tsinghua Initiative Scientific Research Grant and a China Youth 1000-talent program.

In contrast to the rich theory and practice in kidney exchange, the exchange of lung, even though introduced two decades ago, has not been practiced so far [Ergin *et al.*, 2014].

A healthy human has five lung lobes where three lobes are in the right lung and two lobes are in the left. In a lung exchange system, a patient who suffers from lung disease needs *two* donors each to donate a lower lobe to replace the patient's dysfunctional lungs. Each donor not only need to be compatible with the patient in blood-type, but also need to donate a lobe which is as heavy (and large) as the patient's one. In addition, both operations have to be carried out simultaneously so that the two lung donors participate at the same time. To enter the exchange, each patient comes with two donors. After the exchange, as argued in [Ergin *et al.*, 2014], each patient either gets matched with two compatible donors, or remains unchanged with his own donors. In other words, it is not feasible for a patient to exchange for only one compatible donor because he does not meet the requirement for operation and must wait for a second compatible donor. This might also block his intended donor to donate for others.

In this paper, we build a model to study lung exchange from a mechanism design point of view. Our goal is to design a mechanism that maximizes the size of the exchange, subject to incentive constraints. We make the following contributions.

- We prove that, given an instance of lung exchange problem, the problem of deciding if the instance admits a feasible, non-trivial (different from the instance itself) exchange is NP-complete.
- We put forward a class of mechanisms, coined the maximum lexicographical mechanism and prove that it is individually rational, strategy-proof and maximizes exchange size.
- The description of the mechanism contains an NP-hard subroutine, thus, does not admit an efficient implementation. To mitigate this difficulty, we propose two practical implementations: one via Integer Linear Programming (ILP) and the other via search. In particular, the search is based on branch and bound as well as several useful heuristics.
- We implement the mechanism based on the algorithms above and test it on simulated data (based on population distributions and realistic problem sizes). Each al-

gorithm has its own advantage and both of them run sufficiently fast on these data.

The remainder of the paper is organized as follows. Section 2 gives the definitions and model of lung exchange problem. Section 3 proves that the problem is NP-complete. Section 4 presents the maximum lexicographical mechanism and proves several desirable properties. Section 5 describes the two algorithmic implementations and section 6 presents the experiments on simulated data. Finally, in section 7, we present our conclusion and suggest directions for future research.

2 Preliminary

In a lung exchange market, each patient comes with two donors, seeks to swap with other patients [Ergin *et al.*, 2014]. After the exchange, each patient must either be matched with two compatible donors, or remain staying with its original donors. In other words, it is infeasible for a patient to exchange for one donor.

2.1 The lung exchange problem

We formulate a lung exchange problem as follows.

Definition 1. *A lung exchange problem consists of a set of donor-patient triples:*

$In = \{(p_1, d_{11}, d_{12}), (p_2, d_{21}, d_{22}), \ldots\ldots, (p_n, d_{n1}, d_{n2})\}$

There are n patients and $2n$ donors in total. Each patient p_i comes with two donors d_{i1} and d_{i2} and seeks for two compatible donors. Each patient reports his compatible set D_i of all the donors (may be compatible with one of his own donors) according to his blood-type and lung size. The outcome of this problem is also a set of donor-patient triples:

$Out = \{(p_1, r_{11}, r_{12}), (p_2, r_{21}, r_{22}), \ldots\ldots, (p_n, r_{n1}, r_{n2})\}$

Each patient p_i either is assigned with two different compatible donors $r_{i1}, r_{i2} \in D_i, r_{i1} \neq r_{i2}$ or remains the same: $r_{i1} = d_{i1}, r_{i2} = d_{i2}$. Every donor must be assigned to exactly one patient.

Definition 2. *The exchange size is the number of the patients who obtain two compatible donors.*

Definition 3. *We call the outcome with exchange size* 0 *a* trivial solution *while the others are called* non-trivial solutions*, since all patients remaining unchanged is always a feasible but trivial outcome.*

3 Problem complexity

In this section, we prove that even to decide whether there exists a non-trivial solution is computationally hard. Note that, the same problem is clearly in P for kidney exchange. So our result cannot be implied from the hardness result in [Abraham *et al.*, 2007].

Theorem 1. *Given an instance G of lung exchange, the problem of deciding if G admits a non-trivial solution is NP-complete.*

Proof. It is clear that this problem is in NP. For NP-hardness, we reduce from perfect 3D-Matching problem which is the problem of given disjoint sets X, Y, Z of size m, and a set of triples $T \subseteq X \times Y \times Z$ as shown in Figure 1, deciding if there exists a disjoint subset M of T with size m.

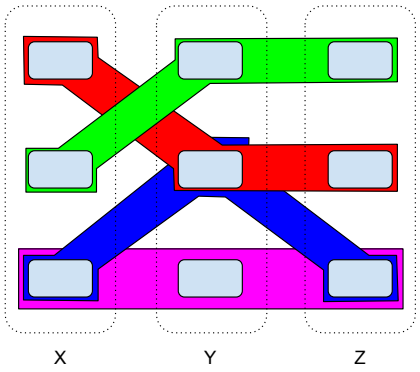

Figure 1: An instance of 3D-Matching problem

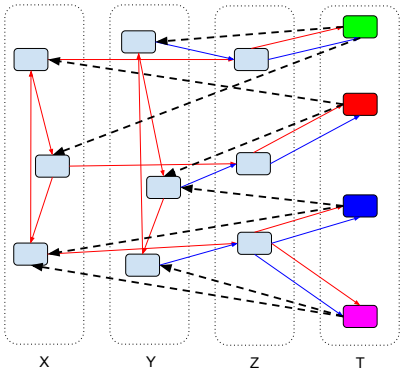

Figure 2: The construction of lung exchange instance. The red lines indicate that it is compatible with the first donor. The blue and black lines indicate that it is compatible with the second donor. The black lines are constructed by the triples of 3D-Matching.

We construct a lung exchange instance in the following way. For each element in X, Y and Z, we construct one patient as shown in Figure 2. For each triple in T, we also construct a corresponding patient. The directed edge (a, b, i) means that patient a is compatible with the i^{th} donor of patient b ($i = 1$ or $i = 2$). In the set X and Y, we construct two super cycles $(x_i, x_{i+1}, 1)$ and $(y_i, y_{i+1}, 1)$ which mean that the patient is compatible with the first donor of the next patient. For patient z_i in Z, we add two edges $(x_i, z_i, 1)$ and $(y_i, z_i, 2)$. For each triple $t_i = \{x_a, y_b, z_c\} \in T$, we construct four edges $(z_c, t_i, 1)$, $(z_c, t_i, 2)$, $(t_i, x_a, 2)$ and $(t_i, y_b, 2)$. The construction can be done in polynomial time.

Let M be a perfect 3D-Matching. We will show that the construction admits a non-trivial solution. For all the triple t_i which $t_i \notin M$, t_i will not participate in the exchange as getting his own donors back. Since the m triples in M are disjoint, all the remaining vertices will exactly get two compatible donors.

Conversely, suppose we have a non-trivial solution in the

construction. It is easy to see that all the vertexes in X, Y and Z must be in the exchange, for any vertex in X or Y within the exchange will lead to the super cycles. Since each donor will be assigned to one patient, there will be exactly m triples which get all the second donors of X and Y in the non-trivial solution. Hence, the m triples constitute a perfect 3D-Matching in the original instance. □

4 Mechanism design

In this section, we first introduce three desiderata for barter exchange [Abdulkadiroğlu and Sönmez, 1999; Roth *et al.*, 2004]. From the perspective of mechanism design, [Sönmez, 1999] searches for the foundations on designing such a mechanism that satisfies all the three economic desiderata, and also provides some positive and negative results. Due to the special structure of agents preferences in our model, we can have and we also propose a mechanism, coined maximum lexicographical mechanism, that satisfies all these desiderata.

Definition 4. *[Pareto efficient] An outcome of lung exchange problem is Pareto efficient if there is no other outcome that make all patients weakly better off and at least one patient strictly better off (i.e., from unmatched to matched).*

Definition 5. *[Individually rational] An outcome of lung exchange problem is individually rational if no existing patient strictly prefers his endowment to his assignment.*

Definition 6. *[Strategy-proof] A mechanism of lung exchange problem is strategy-proof if whenever a patient can not get two compatible donors by reporting truthfully his compatible set, he can not get two true compatible donors by misreporting.*

4.1 Maximum lexicographical Mechanisms

We now describe a class of mechanisms that satisfy all these properties. In such mechanisms, we first generate all the feasible exchange outcomes. Use the notation $o = (s, a_1, a_2, \ldots\ldots, a_n)$ to denote an outcome where s represents the exchange size and a_i represents whether patient p_i gets two compatible donors: $a_i = 1$ if yes while $a_i = 0$ if no. Our mechanism picks the largest outcome with respect to the lexicographical order defined by o. That is, between two outcomes, o_1 and o_2, we first compare their first number s. If they are the same, then compare the second number, so on and so forth. In other words, among all the outcomes of the maximum size, we give priorities to the ones with smaller indices. Note that, the indices can be any permutations on the set of agents. So what we describe here is a class of mechanisms. Note also that our mechanisms differ from the celebrated serial dictatorship mechanism [Abdulkadiroğlu and Sönmez, 1998] in that we first optimize for each agent one by one, subject to the constraint that the exchange size is maximized.

We use the following example to illustrate our mechanism.
Example. There are 3 patients with 6 donors. The set of donor-patient triples is $\{(1,1,2),(2,3,4),(3,5,6)\}$ which means that each patient i brings two donors $i\times 2-1$ and $i\times 2$ to participate in the exchange.

The compatible set for each patient is $D_1 = \{3,4\},\ D_2 = \{1,2,5,6\},\ D_3 = \{3,4\}$ as shown in Figure 3.

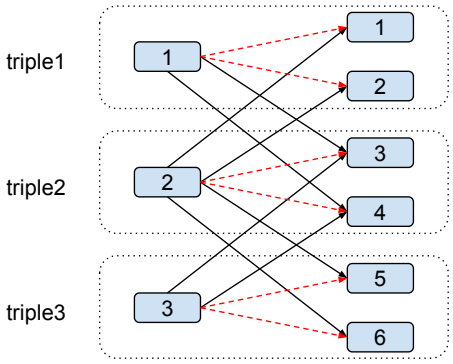

Figure 3: The graph of the example. The black lines represent the compatibility between patient and donor.

All the feasible outcomes and the corresponding representation are

$t_1 = \{(1,1,2),(2,3,4),(3,5,6)\},\ o_1 = (0,0,0,0)$
$t_2 = \{(1,3,4),(2,1,2),(3,5,6)\},\ o_2 = (2,1,1,0)$
$t_3 = \{(1,1,2),(2,5,6),(3,3,4)\},\ o_3 = (2,0,1,1)$

With comparison that $o_2 > o_3 > o_1$, the outcome $\{(1,3,4),(2,1,2),(3,5,6)\}$ will the output of this mechanism as patient 1 and 2 swap their donors while patient 3 gets his own back.

Theorem 2. *The maximum lexicographical mechanism is individually rational, strategy-proof and maximizes exchange size (hence Pareto efficient as well).*

Proof. It is clear that the mechanism will give one of the outcome with maximum exchange size which will directly lead to the Pareto efficient property and the lung exchange problem guarantees that all patients will get back his own donors in the worst case by definition, so the mechanism with maximum size must be Pareto efficient and individually rational.

It remains to show that the mechanism is strategy-proof. Suppose that patient p_i who reports true compatible set can not get two compatible donors, and the outcome is $o_1 = (s, a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n)$. If patient p_i reports fake compatible set D_i' and get two true compatible donors as a result, we record the new outcome as $o_2 = (s', a_1', \ldots, a_{i-1}', 1, a_{i+1}', \ldots, a_n')$. First of all, note that o_2 is still feasible when patient p_i reports truthfully. In other words, both o_1 and o_2 are valid candidate outcomes when all patients are truthful. It is clear that $o_1 \neq o_2$. If $o_1 > o_2$, then when patient p_i reports fake set D_i', o_2 could not be the output which contradicts with the definition of maximum lexicographical mechanism. If $o_1 < o_2$, then when patient p_i reports real set, o_1 could not be the output. Thus, it is impossible for patient p_i to get two true compatible donors by misreporting. □

5 Algorithms to implement the mechanism

Clearly, in the statement of our mechanism, we access as a blackbox for generating all feasible exchange outcomes. This is clearly a computationally infeasible task. In this section,

we propose an ILP algorithm and a search algorithm [Sandholm, 2006] to implement the mechanism in practice.

5.1 ILP algorithm

We consider a formulation of the problem as an ILP with one variable for each edge. Given an instance of lung exchange $In = \{(p_1, d_{11}, d_{12}), (p_2, d_{21}, d_{22}), \ldots\ldots, (p_n, d_{n1}, d_{n2})\}$, construct a bipartite graph with one vertex for each patient, and one vertex for each donor. Add two edge $(p_i, d_{i1})_1$, $(p_i, d_{i2})_1$ between each patient and its two own donors as the red lines, as shown in Figure 3. For each donor t in the compatible set D_i, add an edge $(p_i, t)_2$ between patient p_i and the corresponding donor as the black lines. If a patient is compatible with his donor, there will be two edges that link him with his compatible donor. All the variables of edge are binary, taking values in $\{0, 1\}$. There are three kinds of constraints for the graph.

The first is the consistency constraint, for all $i \in [1, n]$

$$(p_i, d_{i1})_1 = (p_i, d_{i2})_1$$

which means that each patient either gets two compatible donors or remains matched with his own donors.

The second is the patient constraint, for all $i \in [1, n]$

$$(p_i, d_{i1})_1 + (p_i, d_{i2})_1 + \sum_{t \in D_i} (p_i, t)_2 = 2$$

which means that each patient will get exactly two donors in any outcome.

The third is the donor constraint, for all $i \in [1, n]$

$$(p_i, d_{i1})_1 + \sum_{p} (p, d_{i1})_2 = (p_i, d_{i2})_1 + \sum_{p} (p, d_{i2})_2 = 1$$

which means that each donor must be assigned to one patient.

The objective function is the exchange size, formulated as

$$maxSize = \max(n - \sum_{i=1}^{n} (p_i, d_{i1})_1)$$

The first step of the ILP algorithm is to use the formulation above to figure out the maximum exchange size. The next step is to figure out the maximum outcome according to the lexical order. The method is that we determine, for the each patient p_i, whether he can participate in the exchange or not. Suppose the current partial outcome is $o_{cur} = (maxSize, a_1, \ldots, a_{i-1}, a_i = ?, \times, \ldots, \times)$, and all the variables before a_i has been determined. In order to fix the maximum exchange size and the previous results, we add the following constraint

$$n - \sum_{i=1}^{n} (p_i, d_{i1})_1 = maxSize$$

$$a_k = 1 - (p_k, d_{k1})_1 \quad for\ all\ 1 \leq k < i$$

and maximize the new objective function

$$a_i = \max(1 - (p_i, d_{i1})_1)$$

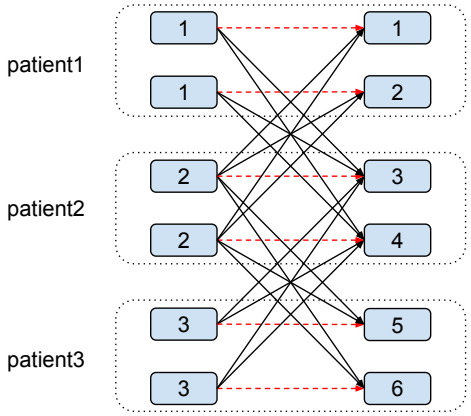

Figure 4: The maximum matching graph of the Example

The number of variables in ILP algorithm is the number of edges as shown in Figure 3, which is $2n + \sum_{i=1}^{n} |D_i| = O(n^2)$, and the number of constraints is $4n + i = O(n)$. Running the ILP algorithm for $n + 1$ times, finally we can figure out the largest outcome for the instance and return the corresponding solution.

5.2 A search algorithm

Alternatively, we can search the maximum outcome directly. In the next two sections, we describe our search algorithm. In this section, we describe how we introduce the basic flow of the search algorithm and in the next section, we describe our optimization techniques.

Our idea here is to make a set of decisions for each patient, deciding whether he is in the solution or not. In principle, the search algorithm works by simulating all possible ways of making the decisions using depth-first search. At any node, the question to branch on is which patient should be in or out. Once the search reaches a leaf, we use the maximum matching algorithm to verify whether the corresponding solution exists.

The construction of the graph for maximum matching is as follows. Given the patients who have been determined to participate in the exchange, construct a bipartite graph with two vertexes for each patient, and one vertex for each donor. Add an edge with weight 0 between each patient and his own donor as the red lines in Figure 4. At this point, the encoding forms a perfect matching. Now, for each donor t in the compatible set D_i, add two edges with weight 1 between the vertexes of patient p_i to donor t as the black lines.

We use the Kuhn-Munkres algorithm [Mills-Tettey *et al.*, 2007] to solve the maximum matching. The algorithm assigns labels α_i to each node in the left side, and labels β_i to each node in the right while maintaining the values for all i, j that $\alpha_i + \beta_j \geq w_{ij}$. An edge in the bipartite graph is admissible if and only if $\alpha_i + \beta_j = w_{ij}$. The subgraph which consists of the currently admissible edges is called the equality subgraph. Finding the augmenting paths, the algorithm does not stop adjusting the labels until there exists a perfect matching in the equality subgraph. The perfect matching in equal-

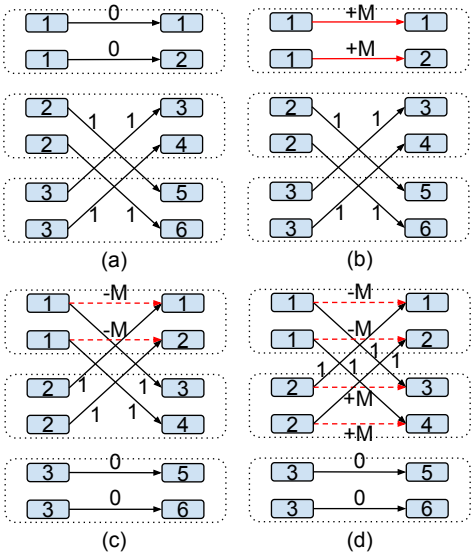

Figure 5: The graph for weight adjustment. For clarity, we designate matched edges with solid lines and unmatched edges with dotted lines. The red lines indicate that the weight of edge has been adjusted. (a) It is the original graph, which total weight is 4. (b) Patient 1 is forced to leave, and the total weight is $2M + 4$ so that the upper bound is 2. (c) Patient 1 is forced to participate, and the total weight is 4. (d) Patient 1 is forced to participate while patient 2 is forced to leave. The total weight is 4, but $4 - 2M < 0$, so it is impossible to come out a feasible solution in this case.

ity subgraph is the maximum matching of the original graph. If the sum of weights in maximum matching is exactly the number of vertex in each side, it means that all patients can get two compatible donors successfully. Otherwise, it is impossible to satisfy every patient in the exchange. In this way, since we have found out all the feasible solutions, we compare them in terms of their lexicographical order. We choose the largest one as the output of the search algorithm.

5.3 Optimization

Upper bound

In the search process, the value of best solution so far is stored globally. In any search node, if the upper bound of current solution is not as large as the best solution so far, we prune the node immediately.

In a non-leaf node, the patients are classified into three classes: there are g patients forced to participate, h patients forced to leave and the others have not been decided yet. Using the bipartite graph constructed above, we adjust the edges with weight 0. We increase the weight of edges between patient p_i and his own donors to a large number M (In implementation, $2n + 1$ is enough) in order to indicate that patient p_i is forced to get his own donors, and the weight of matching will increase $2g \times M$ in total. We decrease the weight to $-M$ so that the patient is forced to participate. Suppose the sum of weights in the maximum matching is sum. If $sum - 2g \times M < 0$, then there is at least a patient, who is forced to participate, has got his own donor or a patient, who is forced to leave, has got a compatible donor. There will be no feasible solution under this search node as shown in Figure 5(d). Otherwise, the value $\frac{1}{2}(sum - 2g \times M)$ is served as a upper bound of the exchange size.

In the maximum matching, if it happens that no patient gets one compatible donor and one of his own donor, then the matching yields a feasible solution with maximum exchange size. We compare it to the best solution found so far and decide whether to update the current solution.

Techniques for deciding which patient to branch on

In the non-leaf node, the question which patient should be branched on is critical to the efficiency of the search algorithm. We propose three strategies to improve the search algorithm and the simulation confirms that it is very helpful.

1. Invalid patient: We prefer to branch on the patient who get one compatible donor and one his own at the current maximum matching. We prefer to eliminate this kind of patients. The intuition is clear: if there is no such patients, the maximum matching has already formed a feasible solution.
2. KM labels: In the Kuhn-Munkres algorithm for solving maximum matching, each patient has 4 KM labels. We prefer to choose the one whose sum of the labels is high. The fact is that, if the patient is forced to leave, the upper bound will decrease at least for the sum of the labels according to Theorem 3.
3. Degree: For each patient, we calculate the out-degree of the patient times the two in-degrees of the donors. We prefer to branch on the patient with small degree since it is difficult for him to participate the exchange. Branching on the small degree nodes first will have the effect that the maximum matching returns a feasible solution fast.

Theorem 3. *In the Kuhn-Munkres algorithm, suppose that $w_{ij} = 0$ and α_i, β_j are the current labels in a maximum matching. If we adjust the weight of edge w_{ij} to a big positive value M ($M > \alpha_i + \beta_j$), and record the previous and new total weights of maximum matching as W_{old}, W_{new} respectively, then we have*

$$W_{new} - M \leq W_{old} - (\alpha_i + \beta_j)$$

Proof. In the graph of maximum matching, we first change the weight w_{ij} to $\alpha_i + \beta_j$. This step will not alter the labels of maximum matching and the total weight. After that, we change the weight from $\alpha_i + \beta_j$ to M. The total weight of matching will increase at most $M - (\alpha_i + \beta_j)$ so we have $W_{new} \leq W_{old} + M - (\alpha_i + \beta_j)$. □

Since $W_{new} - M$ will be the new upper bound of the search node and $\alpha_i + \beta_j \geq w_{ij} = 0$, we have successfully decreased the bound in the searching paths. In the implementation, we first find out all the invalid patients in the maximum matching. Then we calculate the sum of labels for each invalid patient and pick up the highest one. If there is more than one patient, we finally choose the one with smallest degree and branch on this patient.

Dynamic Kuhn-Munkres algorithm

In a searching node, once pick up a patient and branch on, we adjust the weight of two edges in maximum matching and go into a child searching node. The difference of graphs between a node and its child node is only the weight of two edges. The time complexity of Kuhn-Munkres algorithm for maximum matching is $O(n^3)$. Using the dynamic Hungarian algorithm [Mills-Tettey *et al.*, 2007], we can reduce the time complexity to $O(n^2)$ from the previous matching result.

Once the weight of one edge is changed, at most one matching edge has been broken up in the equality subgraph. We adjust the labels of this broken edge, and re-do the process of finding an augmenting path for another time. If an augmenting path is found, we flip the matched and unmatched edges along this path. After that, we have found the maximum matching of the new weight. The time complexity of finding an augmenting path is only $O(n^2)$. This technique has improved the speed of search algorithm significantly.

Search twice

In the searching process, we need to figure out the maximum exchange size as well as the largest outcome. Separating the two targets can reduce the number of searching nodes. We divide the search into two steps. In the first step, we focus on finding the maximum exchange size so that we can prune in the search when the upper bound of current exchange size is not greater than the best size found so far and we reduce the time of searching the largest outcome. In the second step, since we have already known the best exchange size, we focus on finding the solution with the largest outcome so that we can prune the search when exchange size is smaller than the best. This time, we have reduced the time for searching the maximum exchange size. This technique also has improved the searching speed significantly.

6 Experimental results

In this section, we implement the ILP algorithm and search algorithm proposed in the previous section.

6.1 Experiments setup

All our experiments are performed in Linux (openSUSE 13.1), using a PC with four 3.2GHz Intel i5-3470 processors, and 4GB of RAM. We use the CPLEX12.6 software which can take benefit of multiple processors to serve as an ILP solver in our experiment. The running time is recorded as the time actually passed instead of the total time of which the four processors use. In the search algorithm, we just use one processor to implement and run our algorithm. Our experimental data is carefully simulated based on the statistics of US populations. Live donor lobar transplantation is especially common for these who suffer from cystic fibrosis or pulmonary hypertension [Ergin *et al.*, 2014]. Based on this, we simulate all the patients come from these two classed with ratio 2.66 to 1, according to the statistics. The patient blood type distribution in total is 45.7% O, 40.4% A, 10.5% B, 3.4% AB. The donor blood type distribution is 44% O, 42% A, 10% B, 4% AB. We generate the lung size of each patient uniformly. Compatibility test is based on the blood type compatibility and size compatibility. Besides the donation can be between the same blood-type, blood-type O can donate to any type and blood-type AB can accept any type. The size compatibility is that a patient can only receive from a donor as heavy as himself. After the generation of patients and donors, we randomly group them into n donor-patient triples.

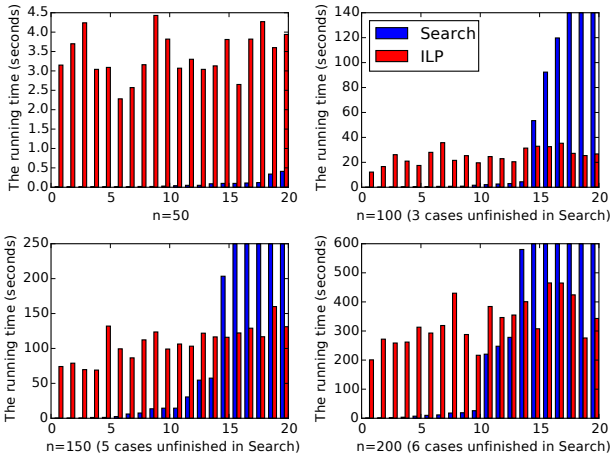

Figure 6: The test results of ILP and search algorithm

6.2 Experimental results

For $n = 50, 100, 150, 200$, we randomly generate 20 copies of simulated instance and test on them the ILP algorithm and search algorithm respectively. The running time is shown in Figure 6 in which the test cases are ordered by the running time of search algorithm. There is a fraction of cases which does not finish searching on different size. The ILP algorithm is very stable and finish in all cases within a reasonable time. In more than half of the cases, the search algorithm is much faster than ILP algorithm and finishes in just a few seconds.

Our overall recommendation is that, first run the search algorithm. If it does not return a solution in reasonable amount of time, switch to the ILP algorithm. Since the search algorithm runs faster than the ILP algorithm in a majority of cases, this method can speed up for solving lung exchange problem.

7 Conclusion and future research

We have proposed a mechanism that is optimal, individually rational and strategy-proof for the lung exchange problem. Since the problem is NP-hard, we present two practical implementation of our mechanism: one based on ILP formulation and the other on search. Both algorithms are carefully designed to speed up the search. Our experiments show that the algorithms scale up to reasonably large instances and each algorithm has its own advantages.

There are a number of future directions to expand the current research. First, we would like to further speed up our search algorithm. Ideas include branching on a set of patients simultaneously. Second, we would also like to take into considerations chains and sensitivity of the patients. Last but not least, we would like to consider dynamic version of the problem and design desirable mechanisms for the dynamic settings.

References

[Abdulkadiroğlu and Sönmez, 1998] Atila Abdulkadiroğlu and Tayfun Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, pages 689–701, 1998.

[Abdulkadiroğlu and Sönmez, 1999] Atila Abdulkadiroğlu and Tayfun Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, 88(2):233–260, 1999.

[Abraham *et al.*, 2007] David J Abraham, Avrim Blum, and Tuomas Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 295–304. ACM, 2007.

[Ashlagi and Roth, 2011] Itai Ashlagi and Alvin Roth. Individual rationality and participation in large scale, multi-hospital kidney exchange. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 321–322. ACM, 2011.

[Awasthi and Sandholm, 2009] Pranjal Awasthi and Tuomas Sandholm. Online stochastic optimization in the large: Application to kidney exchange. In *IJCAI*, volume 9, pages 405–411, 2009.

[Blum *et al.*, 2013] Avrim Blum, Anupam Gupta, Ariel D. Procaccia, and Ankit Sharma. Harnessing the power of two crossmatches. In *ACM Conference on Electronic Commerce, EC '13, Philadelphia, PA, USA, June 16-20, 2013*, pages 123–140, 2013.

[Dickerson and Sandholm, 2014] John P. Dickerson and Tuomas Sandholm. Multi-organ exchange: The whole is greater than the sum of its parts. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1412–1418, 2014.

[Dickerson *et al.*, 2012a] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Dynamic matching via weighted myopia with application to kidney exchange. In *AAAI*, 2012.

[Dickerson *et al.*, 2012b] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Optimizing kidney exchange with transplant chains: theory and reality. In *AAMAS*, pages 711–718, 2012.

[Dickerson *et al.*, 2013] John P Dickerson, Ariel D Procaccia, and Tuomas Sandholm. Failure-aware kidney exchange. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 323–340. ACM, 2013.

[Dickerson, 2014] John P. Dickerson. Robust dynamic optimization with application to kidney exchange. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 1701–1702, 2014.

[Ergin *et al.*, 2014] Haluk Ergin, Tayfun Sönmez, and M UtkuUnver. Lung exchange. Technical report, 2014.

[Mills-Tettey *et al.*, 2007] G Ayorkor Mills-Tettey, Anthony Stentz, and M Bernardine Dias. The dynamic hungarian algorithm for the assignment problem with changing costs. 2007.

[Roth *et al.*, 2004] Alvin E Roth, Tayfun Sönmez, and M Ünver. Kidney exchange. *The Quarterly Journal of Economics*, 119(2):457–488, 2004.

[Sandholm, 2006] Tuomas Sandholm. Optimal winner determination algorithms. *Combinatorial auctions*, pages 337–368, 2006.

[Sönmez, 1999] Tayfun Sönmez. Strategy-proofness and essentially single-valued cores. *Econometrica*, 67(3):677–689, 1999.