Efficient Near-optimal Algorithms for Barter Exchange

Zhipeng Jia Pingzhong Tang Ruosong Wang Hanrui Zhang
Institute of interdisciplinary information sciences
Tsinghua University
{jzp13, wrs13, zhang-hr13}@mails.tsinghua.edu.cn, kenshinping@gmail.com

ABSTRACT

We study polynomial-time clearing algorithms for the barter exchange problem. We put forward a family of carefully designed approximation algorithms with desirable worst-case guarantees. We further apply a series of novel heuristics to implement these algorithms. We demonstrate via kidney exchange data sets that these algorithms achieve near-optimal performances while outperforming the state-of-the-art ILP based algorithms in running time by orders of magnitude.

Keywords

kidney exchange, approximation algorithm, experiments

1. INTRODUCTION

Barter exchange is a fundamental form of economic mechanism that allows agents to swap goods among each other without money transfers. Over the past decades, designing economically desirable (e.g., Pareto efficient and strategy-proof) barter exchange has been a topic of intensive research, investigated under a variety of important domains, including house allocation [50, 1, 2], kidney exchange [44, 47, 3, 53], military contract [51] and lately lung [34] and digital good exchanges [29].

When considering implementation in practice, most of these mechanisms involve a non-trivial optimization problem, making the implementation itself a challenging computation problem [3, 32, 34].

Take the design and implementation of kidney exchange for example. In the simplest kidney exchange setting (without altruistic chains) [3], a patient with kidney disease is paired with an incompatible donor. While the pair donate a kidney to help some other compatible patient in the system, they obtain a compatible kidney in return. Both patients receive a compatible kidney in the end, leading to improvements in social welfare. Nowadays, kidney exchange has been serving as an important alternative in addition to cadaver donations and has been fielded in a number of countries such as the US, UK, Netherlands and South Korea [22].

The implementation problem (aka. the clearing problem), described theoretically, is to find a set of vertex-disjoint cycles that covers the maximum number of edges in a directed graph. Each vertex in the graph represents a patient-donor pair and each arc represents compatibility between the pairs. A cycle of length L requires $2L$ people in simultaneous surgeries. For logistic feasbility imposed by simultaneous surgeries and to reduce the loss of last-minute dropping out, in practice, cycle length in the solution is required to be less than or equal to 3 [3]. Abraham et al. [3] show that finding a Pareto optimal solution under the cycle length constraint is NP-hard for any finite L. Similar hardness results are obtained in the domains of lung [34] and digital good exchanges [29].

There has been a rich literature on optimal algorithms of the clearing problem described above. Due to various hardness results [3, 13, 14, 35], existing research has been focusing on super-polynomial-time algorithms, mostly based on integer linear programming (ILP), that would return in an acceptable amount of time at practical scales. To name a few, Abraham et al. [3] propose the first clearing algorithm that enables nationwide kidney exchanges, and a recent work by Dickerson et al. [22] puts forward a compact ILP formulation that brings about drastic improvements on running time. For a comprehensive survey on ILP based clearing algorithms, we refer interested readers to [22].

While these ILP algorithms currently suffice for clearing markets at nationwide scales, there are a number of reasons to look for even faster algorithms. As Dickerson et. al. pointed out, first of all, advances in medical procedures may increase the cycle and chain caps, leading to harder problems that cannot be solved using current algorithms. Secondly, opportunities exist for cross-border databases, leading to much larger instances. Thirdly, the NP-hardness nature makes it possible to encounter hard instances that cannot be cleared in a reasonable amount of time by the ILP algorithms – this is one of our major motivations to consider polynomial time approximation algorithms. Last but not least, on exchange domains with online arrivals and departures, the algorithms for the dynamic clearing problems, e.g., [12], may make multiple calls to the static algorithms. As a result, improvements on the static algorithm may make big differences on the dynamic domain.

In this paper, orthogonal to previous approaches, we aim to tackle those challenges by design *polynomial-time, near-optimal* algorithms for the clearing problem. By near-optimality, we refer to algorithms that both have desirable worst-case guarantees in theory and yield near-optimal solutions by empirical evaluations. While it may be useful for readers to think of kidney exchange as a running example throughout the paper, we remark that the algorithmic framework

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

we propose is general and can be adapted to general barter exchange domains.

1.1 Related work

Since the introduction and clear characterization of kidney exchange [42, 46, 48, 49, 45], a great deal of work has been focusing on exact solutions of static exchange market [22, 3, 7, 18, 30, 31, 37, 41], most of which are based on *Integer Programming* formulations of the market. Anderson et al. presented a novel approach to deal with chains in [6].

Sometimes the stochastic settings are a more accurate model for reality. In a real life exchange procedure, an edge suggested by first step tests may actually be proven not to exist by more careful tests before transplantation. However, since the final test is costly and time-consuming, we cannot afford to test all patient-donor pairs beforehand. We therefore aim for algorithms which maximize the expected number of agents covered by valid cycles based on first step test results only. Stochastic settings of kidney exchange are also extensively studied [5, 16, 17, 25, 40].

Another attempt to model the reality resulted in the dynamic settings, where agents arrive and depart online, and we try to maximize the overall number of matched agents. Various concrete models are studied, and fruitful results have been acquired [4, 10, 12, 23, 27, 53, 15].

Economic aspects of kidney exchange, such as truthfulness and fairness, are considered as well. Truthful mechanisms, where agents (hospitals) have no incentive to report partial or erroneous information about the patients, are studied in [11, 48, 52, 8]. Different kinds of fairness are discussed in [32, 26], as well as corresponding algorithms.

Mak-Hau surveyed most of *Integer Programming* based methods in [36] and Dickerson presented a unified picture for various models and settings of kidney exchange [21].

1.2 Our contribution

Our contribution can be summarized as follows:

- We first revisit the so called k-Set-Packing problem, a relevant computation problem being relatively well studied, and provide a *blackbox reduction* from the clearing Problem to the k-Set-Packing problem. In particular, we show that an α-approximation algorithm for L-Set-Packing implies an $[1+(\alpha-1)L/2]$-approximation algorithm for the clearing problem when the cycle length constraint is L, based on recent efficient local search algorithms for k-Set-Packing [19]. Specifically, this reduction gives a $(3/2+\epsilon)$-approximation polynomial algorithm when $L = 3$ and a $(7/3+\epsilon)$-approximation polynomial algorithm when $L = 4$.

- There are a number of difficulties that prevent us from directly deploying the algorithms above to the domain of Barter exchange. To implement the algorithms, we introduce a set of powerful heuristics, including a family of greedy algorithms to provide a good initial solution for the local search algorithms and a useful sampling technique to select cycles. Combining these heuristics with a modified local search algorithm, we obtain a highly efficient, near optimal algorithm for the general barter exchange problem. Furthermore, we also show how to handle altruistic chains under our framework with negligible overheads.

- We empirically evaluate our algorithms on standard kidney exchange data sets. Comparing to the state-of-the-art ILP based algorithms, our algorithms return near-optimal solutions (no less than 98% of the optimal solutions) within running time faster by orders of magnitude.

In summary, all the evidences above suggest that our algorithms are strong candidates to be deployed in the nationwide kidney exchanges.

2. PRELIMINARIES

We formally define the abstract barter exchange problem as follows.

Definition 1 (L-Exchange Problem). *Given a directed graph $G = (V, E)$, the goal of L-Exchange is to find a set of disjoint cycles of length not exceeding L, covering the maximum number of edges.*

The k-Set-Packing problem mentioned in the introduction is formally defined as follow.

Definition 2 (k-Set-Packing problem). *Given a set U and a family $\mathcal{F} \subseteq 2^U$ of subsets of U where for each $V \in \mathcal{F}$, $|V| \leq k$, the k-Set-Packing problem is to find a maximum size subfamily of $\mathcal{F}$ consisting of pairwise disjoint sets.*

k-Set-Packing in its essence is a similar problem to L-Exchange, except that it seeks to maximize the number of disjoint cycles chosen, instead of edges covered. In [19], a family of approximation algorithms for k-Set-Packing is presented, summarized in the following theorem.

Theorem 1 (Theorem 1.2 of [19]). *For any $\epsilon > 0$ and any integer $k \geq 3$ there is a polynomial time $(k+1+\epsilon)/3$-approximation algorithm for k-Set-Packing.*

3. BLACKBOX REDUCTION VIA K-SET-PACKING

In this section, we try to construct an approximation algorithm for L-Exchange, which calls an approximation algorithm for L-Set-Packing as a subroutine, such that the better bound the latter algorithm achieves, the better bound for L-Exchange. Hence based on Theorem 1, we can construct a family of approximation algorithms for L-Exchange with desirable bounds.

The main challenge here is to carry the approximation ratio of number of cycles chosen to that of the number of edges covered without significant loss. Suppose we have an α-approximation algorithm for L-Set-Packing at hand. To solve for an L-Exchange instance, one seemingly plausible approach is to treat each cycle of length not exceeding L as a feasible set, and run the algorithm for L-Set-Packing directly. However, there are obvious hard instances, on which the optimal solution consists of t L-cycles, while our algorithm generates t/α 2-cycles. The overall approximation ratio is at least $\alpha L/2$, which is not satisfactory.

Instead of treating cycles of different lengths equally in the reduction, we enumerate the maximal number of cycles of each length, and make multiple calls to the approximation algorithm for L-Set-Packing to get the most accurate approximation. Intuitively, a limitation of the numbers of small

cycles prevents the approximation for L-SET-PACKING to return a solution full of, say, 2-cycles, and thus yields a better approximation ratio. A detailed construction is given below.

LEMMA 1. *If there is a polynomial time $\alpha(L)$-approximation for L-SET-PACKING, then there is a polynomial time $\beta(L)$-approximation for L-EXCHANGE, where $\beta(L) = 1 + (\alpha(L) - 1)L/2$.*

PROOF. Given an $\alpha(L)$-approximation algorithm for L-SET-PACKING, we construct explicitly a $\beta(L)$-approximation algorithm for L-EXCHANGE.

Suppose in an optimal solution of some L-EXCHANGE instance $G = (V, E)$, the number of l-cycles is c_l for $2 \le l \le L$, and the optimal size is therefore $s^* = \sum_{2 \le l \le L} l \cdot c_l$. We enumerate $d_2, \ldots, d_{L-1} \in [n]$, intuitively the maximal numbers of 2-, 3-, ..., $(L-1)$-cycles permitted in the solution, and construct L-SET-PACKING instances $I(d_2, \ldots, d_{L-1}) = (V(d_2, \ldots, d_{L-1}), \mathcal{F}(d_2, \ldots, d_{L-1}))$ as follows:

Let $V(d_2, \ldots, d_{L-1}) = V \cup A_{2,d_2} \cup A_{3,d_3} \cup \cdots \cup A_{L-1,d_{L-1}}$ where $A_{l,d_l} = \{a_{l,1}, \ldots, a_{l,d_l}\}$ is the set of added dummy vertices for l-cycles, with V, $A_{2,d_2}, \ldots, A_{L-1,d_{L-1}}$ mutually disjoint. We construct $\mathcal{F}(d_2, \ldots, d_{L-1})$ as follows:

- For each l-cycle $(v_1, \ldots, v_l)$ (where $l < L$) in G, we add $\{v_1, \ldots, v_l, a_{l,i}\}$ into $\mathcal{F}(d_2, \ldots, d_{L-1})$ for each $i \in [d_l]$.
- For each L-cycle $(v_1, \ldots, v_L)$, we just add $\{v_1, \ldots, v_L\}$ into $\mathcal{F}(d_2, \ldots, d_{L-1})$.

Note that the number of l-cycles will not exceed $\binom{n}{l} = O(n^l)$. The construction of $I(d_2, \ldots, d_{L-1})$ can thus be done in $O(n^L)$ time.

We then run the $\alpha(L)$-approximation for L-SET-PACKING on all these instances $I(d_2, \ldots, d_{L-1})$, and return the best outcome (i.e., with the maximal number of vertices covered).

Now we analyze the approximation ratio of the algorithm mentioned above. Let s' be the outcome of our algorithm and $s(d_2, \ldots, d_{L-1})$ be that on $I(d_2, \ldots, d_{L-1})$. Since $c_2, \ldots, c_{L-1} \in [n]$, clearly $s' \ge s(c_2, \ldots, c_{L-1})$. Denote the numbers of edges covered by the cycles in the optimal solution by $(x_1, \ldots, x_c)$, sorted in ascending order, where $c = \sum_{2 \le l \le L} c_l$. So $s^* = \sum_{i \in [c]} x_i$. Note that a natural one-to-one mapping between $(c_2, \ldots, c_L)$ and $(x_1, \ldots, x_c)$ follows directly from the defintion. Let

$$X(c_2, \ldots, c_L) = \sum_{1 \le i \le \lceil c/\alpha(L) \rceil} x_i,$$

then $s(c_2, \ldots, c_{L-1}) \ge X(c_2, \ldots, c_L)$, and so we have

$$\beta(L) \le \max_{c_2, \ldots, c_L} \left\{ \frac{s^*}{X(c_2, \ldots, c_L)} \right\}.$$

Obviously the extreme case is when $x_i = 2$ for all $i \le \lceil c/\alpha(L) \rceil$ and $x_i = L$ otherwise, which gives

$$\beta(L) \le \frac{2c/\alpha(L) + Lc(1 - 1/\alpha(L))}{2c/\alpha(L)} = 1 + \frac{(\alpha(L) - 1)L}{2}.$$

In total, we make $O(n^L)$ calls to the polynomial time approximation algorithm for L-SET-PACKING, and spend $O(n^L)$ time to construct the instance each time. The overall running time of our constructed algorithm is therefore polynomial. □

THEOREM 2. *For any $\epsilon > 0$, there is a polynomial time $(1 + L(L-2)/6 + \epsilon)$-approximation for L-EXCHANGE.*

PROOF. Theorem 1 suggests that for any $\epsilon > 0$, there is a polynomial time $((L + 1 + \epsilon)/3)$-approximation for L-SET-PACKING. It follows directly from Lemma 1 that for any $\epsilon > 0$, there is a polynomial time $(1 + L(L-2)/6 + \epsilon)$-approximation for L-EXCHANGE. □

COROLLARY 1. *For any $\epsilon > 0$, there is a polynomial time $(3/2 + \epsilon)$-approximation for* 3-EXCHANGE.

COROLLARY 2. *For any $\epsilon > 0$, there is a polynomial time $(7/3 + \epsilon)$-approximation for* 4-EXCHANGE.

4. IMPLEMENTING THE CLEARING ALGORITHMS

As mentioned, the kidney exchange marketplace looks like an instance of 3-EXCHANGE, as defined in Definition 1, except for the existence of altruistic donors. Altruistic donors, first discussed in [39, 49], behave just like normal patient-donor pairs, except that they do not expect an incoming kidney. They may thus initiate chains of donations instead of cycles. Unlike cycles, operations involved in chains can be performed sequentially, as observed in [43]. Also according to Ashlagi et al. [9] and Ding et al. [28], such chains are very effective in the market, especially when the chance for exchange is sparse. For these practical reasons, we consider chains with much larger sizes in real life kidney exchange. So the real problem is: How to find a family of 2- and 3-cycles, and chains initiated by altruistic donors of reasonable lengths, to cover as many edges (i.e., transplants) as possible?

In the remainder of the section, we show how to combine a number of novel heuristics with the algorithm constructed in the previous section, in order to obtain an applicable clearing algorithm for the real life kidney exchange problem described above, fast and near-optimal.

4.1 Overview of the framework

The algorithm for k-SET-PACKING stated in Theorem 1, which we call as a subroutine in the approximation algorithm for L-EXCHANGE, proceeds in a manner well known as *Local Search*. Basically, it maintains a solution and tries to augment it constantly, by replacing up to t existing cycles in the current solution with $t+1$ cycles (where t is a parameter controlling the approximation ratio), by adding some unused cycles to the solution and possibly removing some used ones. More precisely, it picks an unused cycle, adds it to the solution, and removes all other cycles currently in the solution which overlap with the added one. If no cycle is removed, clearly it successfully improved the solution, and the algorithm iterates to the next round. Otherwise there would be some vertices freed from cycles removed from the solution. It then continues searching for unused cycles to cover these freed vertices, till it sees an improvement of the solution, or reaches the limitation that t cycles have been removed, so this branch of searching fails and it retracts. The algorithm terminates when it becomes impossible to replace t cycles with $t + 1$.

Several issues make a straightforward application of this algorithm practically unfavorable:

- The *Local Search* process can actually start from any feasible solution. Presumably it is much more efficient to generate a heuristic initial solution first, and apply

Local Search to augment it, than to search everything out from scratch.

- A single round of searching takes $\Omega(n^t)$ time, which is unacceptable (in spite of being polynomial in n) when t is large, not to mention that we have to enumerate the maximum number of cycles and make a call for each group of limitation.
- The *Local Search* scheme does not specify any order of searching while an appropriate order might make a huge difference in practice.
- Chains cannot be effectively handled by a straightforward implementation of *Local Search*.

We therefore retain the framework of *Local Search*, as well as adopting manifold heuristics in order to tackle the issues above. We expound these heuristics in the rest of this section, and give explanations why they work. The overall framework is roughly as follows: We first try to handle chains and remove altruistic agents, such that we consider only cycles henceforth. Then we generate a coarse solution as fast as possible, and subsequently apply a modified version of *Local Search* procedure to make the solution more accurate. See Algorithm 1.

Algorithm 1: The *Local Search* framework

Data: a compatibility graph G, subroutines **ClearChains**, **InitSolution**, **LocalSearch**
Result: a near optimal solution to the clearing problem defined by G

1 Let $\mathcal{C} = \textbf{ClearChains}(G)$.
2 Let $G' = G \setminus \bigcup_{C \in \mathcal{C}} C$.
3 Let $\mathcal{S}_0 = \textbf{InitSolution}(G')$.
4 Let $\mathcal{S} = \textbf{LocalSearch}(G', \mathcal{S}_0)$.
5 Return $\mathcal{C} \cup \mathcal{S}$.

4.2 *Greedy* by Product of Degrees

One nice property of the *Local Search* framework is that it may start from any existing solution instead of having to build one from scratch. Exploiting this property, we try to find an appropriate way to generate an initial solution, in order to speed up the entire algorithm. Naturally, we want the generation to be fast, but not necessarily so accurate, since we always count on the *Local Search* procedure to accomplish that part of the task. For such a purpose, we introduce the *Greedy* paradigm that turns out to be extremely effective in practice.

The *Greedy* paradigm is indeed simple as it sounds: We simply order all cycles and try them one by one. We add each cycle into the solution whenever it does not bring a conflict. Here, the key of *Greedy* is to choose a good ordering. Here we present two orderings that work well in practice: Product of Degrees (PoD) ordering and LP ordering, together with an important sampling heuristic which notably improves the running time and accuracy of *Greedy* with LP ordering.

Let $d_{\text{in}}(v)$ and $d_{\text{out}}(v)$ denote the in-degree and out-degree of vertex v. The Product of Degrees ordering, as its name indicates, sorts all vertices in ascending order of the product of their in-degree (plus 1) and out-degree (plus 1), i.e., $(d_{\text{in}}(v)+1)\cdot(d_{\text{out}}(v)+1)$, and tries to cover them one by one. An ordering of cycles is implicitly defined by the covering procedure. The PoD ordering is first implicitly applied in [35]. It is based on the following simple idea: It is harder to find a cycle covering a vertex with a smaller product of in-degree and out-degree, so if we do not cover it as early as possible, very likely it would never be covered.

In practice, there are two ways of covering in *Greedy* by PoD: first searching for 2-cycles and then 3-cycles, and vice versa. The better outcome is chosen as the output of *Greedy* with PoD.

Greedy with the PoD ordering runs extremely fast, yet its performance is surprisingly good.

On the theoretical side, the *Greedy* paradigm, realized with any ordering, is a 3-approximation for 3-Exchange, as stated in the following theorem.

Theorem 3. *The* Greedy *paradigm realized with any ordering is a 3-approximation for* 3-Exchange.

Proof. Denote the number of 2- and 3-cycles in the market graph G by $c = |\mathcal{C}|$. Let $\pi : [c] \to \mathcal{C}$ be any permutation of 2- and 3-cycles of G. Recall that the *Greedy* paradigm proceeds by checking $\pi(1), \ldots, \pi(c)$ one by one and adding a cycle whenever possible. Let $\mathcal{C}^*$ be an optimal solution for G. We show that in the solution generated by *Greedy*, there is at least one vertex in each $C \in \mathcal{C}^*$ covered. Suppose not, i.e., there is some C, none of whose vertices is covered. Clearly there is some $i_C \in [c]$ such that $\pi(i_C) = C$. When *Greedy* reaches $C = \pi(i_C)$, clearly none of its vertices is covered, so the algorithm would add the entire C into the solution, a contradiction. Now since there are at most 3 vertices in each cycle, the *Greedy* paradigm is a 3-approximation. □

As a corollary, we note that *Greedy* with the PoD ordering is itself a 3-approximation for 3-Exchange, which becomes more powerful combined with *Local Search*.

4.3 *Greedy* by solutions of LP relaxation

Another natural yet powerful ordering is the LP relaxation ordering. That is, we solve the relaxed LP of the instance, and order cycles in descending order of the corresponding LP variables. The LP ordering is first implemented by Dickerson [20].

The relaxed LP of an instance $G = (V, E)$ is:

$$\begin{array}{lll} \text{maximize}: & \sum_{C\in\mathcal{C}} |C| \cdot x_C & \\ \text{subject to}: & \sum_{C: v\in C} x_C \le 1, & \text{for all } v \in V, \\ & x_C \ge 0, & \text{for all } C \in \mathcal{C}, \end{array}$$

where $\mathcal{C}$ is the set of all feasible cycles and $|C|$ denotes the size of a cycle $C \in \mathcal{C}$.

Clearly the solution to the relaxed LP of an instance is greater than or equal to the solution to the barter exchange formulation itself, but sure the LP solution provides evidences about which cycles are more "important" than others. In fact, at least two messages can be derived from the fact that the variable corresponding to a particular cycle is large: A large solution to the relaxed LP relies heavily on the cycle, and the cycle overlaps with few other important cycles in the relaxed LP. Both messages indicate that we should assign the cycle a relatively high priority.

We use **Gurobi** to solve the relaxed LP. A simple trick here is to solve the dual form of the LP instead of the primal form, for it is harder for most LP solvers to deal with huge number of variables.

Experiments show that LP relaxation ordering often yields slightly better initial solutions than that with PoD ordering, with a notably larger consumption of time, since we have to solve the relaxed LP first to run the *Greedy* procedure based. Also, as a corollary of Theorem 3, *Greedy* by LP relaxation itself is a 3-approximation for 3-EXCHANGE.

4.4 Sampling heuristic

LP ordering is not yet perfectly preferable, in that the time consumption is too great (though negligible compared to that needed by an exact algorithm), especially considering we are just generating an initial solution. We hence further adopt the sampling heuristic to reduce the running time, while boosting the accuracy as a surprisingly byproduct.

The sampling heuristic is just as simple as the two orderings. We pick 3-cycles at random, and consider only 3-cycles we picked in the relaxed LP. In other words, we fix the LP variables of other cycles to be 0, that is to say, practically we do not create variables for them at all. (We still consider all 2-cycles, since the number of 2-cycles is always reasonable.)

It is kind of surprising that such a simple heuristic reduces significantly the running time of LP approximation while even improving the accuracy. Here is our explanation: Suppose there are c 3-cycles in total, and we are to approximate the optimal solution of the original graph by the optimal solution in a subgraph generated by picking 3-cycles uniformly at random. Conceivably, to get a perfectly accurate solution with probability of 0.5, we may need $0.9c$ uniformly random 3-cycles, while, say, a 0.9-approximation of the optimum requires only $0.1c$ uniformly random 3-cycles, as the marginal utility of 3-cycles decreases intuitively. Also, as the number of 3-cycles decreases, the graph becomes sparser, which implies that the solution to the relaxed LP should become more concentrated, as there are less conflicts between 3-cycles. Clearly, the more concentrated the solution is, the better approximation we will get for the sampled subgraph. As the rate at which the approximation becomes more accurate overruns that at which the optimum of the sampled subgraph decreases, the overall accuracy rises, in a certain interval of density of the sampled graph.

In practice, we pick 3-cycles in a more delicate way instead of just picking them uniformly at random. In picking each cycle, we first pick a uniformly random patient (vertex), and then pick one of unselected cycles covering it, uniformly at random. We try sampling with $0.1c$ and $0.01c$ 3-cycles. Both parameters work well.

With the sampling heuristic, the speed of *Greedy* by solutions of LP relaxation becomes comparable with that of *Greedy* with PoD ordering, though the latter is still about 10 times faster. Given the utter efficiency of *Greedy* with PoD ordering, such a running time seems totally acceptable, even under the most challenging requirements of practical purposes.

4.5 Speedups of *Local Search*

To craft a faster version of *Local Search*, first we note:

- Our objective is to maximize the number of edges (or vertices) covered, but not the number of cycles selected as in the original version in [19].
- The enumeration of numbers of cycles, as stated in the proof of Lemma 1, amplifies the running time to an overwhelming extent.

The two facts above lead directly to a first step in our modification of *Local Search*: In the search procedure, we seek to replace at most t covered vertices, but not cycles, with $t + 1$. And subsequently, we no longer need to enumerate the numbers of cycles. Recall that the enumeration is introduced to handle the issue, that cycles of different "weights" are treated equally in the original version of *Local Search*, but we are now directly maximizing the number of covered vertices.

In each round of the search, we choose a starting cycle not in the current solution, start the search by adding the cycle and removing overlapping cycle(s) from the solution, and try to append other cycles into the solution to cover the vertices freed from the removed cycle(s), and hopefully some more vertices previously not covered. The choice of the starting cycle follows several rules, listed in descending order of priority:

1. All cycles covering no non-free vertices should be added into the solution immediately.
2. When all cycles cover at least one non-free vertex, cycles covering the least number of non-free vertices should be tried first.
3. If there are multiple such cycles, ones undergone the least number of former attempts should be tried first.
4. If there are multiple such cycles, and they all cover 1 non-free vertices, ones whose covered non-free vertex is currently covered by a 2-cycle (instead of a 3-cycle) should be tried first.
5. If there are multiple such cycles, under the condition of rule 4, ones whose covered non-free vertex, say v, has the minimum $d_{\text{in}}(v) \cdot d_{\text{out}}(v)$ (no "plus 1" here) should be tried first.
6. If there are still multiple such cycles, then pick any one of them.

Once we have chosen the starting cycle and begin to search, we keep track of a set of freed vertices from removed cycles, to be covered by unused cycles. We consider cycles that cover 3 freed vertices first, then those which cover 2 freed vertices. Those that cover only 1 freed vertex are ignored in the current stage. We try each cycle in the order described above, remove all (if any) overlaping cycles, and continue searching to cover the remaining (and possibly some new) freed vertices. We count the number of vertices freed from cycles removed in the search. Once the number reaches a predefined threshold, which means we have freed too many vertices and gone too far, we declare an immediate failure and return. Also, for similar reasons, we give up and return when the depth of the search exceeds some predetermined threshold. An attempt fails if it reaches the above two thresholds, or it fails to cover all freed vertices. On the global level, when a certain number of consecutive attempts of starting cycles fail, we assert that a further improvement is extremely difficult to find, and terminate the entire *Local Search* procedure.

4.6 Dealing with chains

One remaining issue is to deal with chains initiated by altruistic donors. Our approach is similar to what was proposed in [28]. The approach is quite simple: We try all altruistic vertices one by one, and pick the longest chains possible.

After chain selection, all vertices covered by chosen chains, as well as altruistic vertices, are removed from the graph, and so we consider only 2- and 3-cycles from now on.

To improve efficiency and accuracy, we further adopt a heuristic inspired by the ideas discussed in previous sections. In choosing chains, we consider only patients with small PoDs.[1] More specifically, we sort all vertices in ascending order of their PoDs, and consider only the first r fraction of the vertices, where r is a predetermined parameter.

Equipped with the heuristic above, our algorithm handles chains in negligible time, with negligible sacrifice of overall accuracy compared to exact solutions. See the following section for experimental results.

5. EMPIRICAL EVALUATIONS

5.1 Experiment setup

We evaluated our algorithms on three kidney datasets: the kidney data available on PrefLib.org [38], [24]; the US data and China data used in [33].

For comparison of running time and evaluation of approximation ratio, we also ran the state-of-the-art **PICEF** algorithm [22] which can give the optimal solution.

All the experiments are conducted on a server with 4 CPU cores and 28GB RAM.

5.2 Results

Figure 1 and Figure 3 show running time and approximation ratio of the proposed *Local Search* framework with initial solutions via *Greedy* by solutions of LP relaxation. Figure 2 and Figure 4 show running time and approximation ratio of the proposed *Local Search* framework with initial solutions via *Greedy* by PoD.

From the figures we see that both of our algorithms are clearly faster by orders of magnitude than **PICEF**. Both algorithms achieve better than 90%-approximation under all settings. Particularly, when initial solutions are produced via *Greedy* by solutions of LP relaxation, the *Local Search* framework achieves 99%-approximation without chains, and 98%-approximation with chains. On the other hand, *Local Search* with initial solutions via *Greedy* by PoD is notably faster, yet attaining an acceptably satisfactory approximation ratio. In words, *Greedy* by solutions of LP relaxation accomplishes a balance between running time and approximation ratio, and fits for most practical purposes, especially when accuracy is desired. *Greedy* by PoD, which further speeds the algorithm up with a reasonable loss of accuracy, is more favorable when speed is the most significant concern.

Also, as shown in Table 1, the sampling heuristic does make a huge difference in practice. As the sampling ratio decreases from 1 to 0.01, the approximation ratio rises from 0.981 to 0.995, and the running time decreases from 59.419 to 1.132. However, an even smaller sampling ratio, 0.001, decreases the approximation ratio. We choose 0.01 as the sampling ratio in practice to achieve the best approximation ratio.

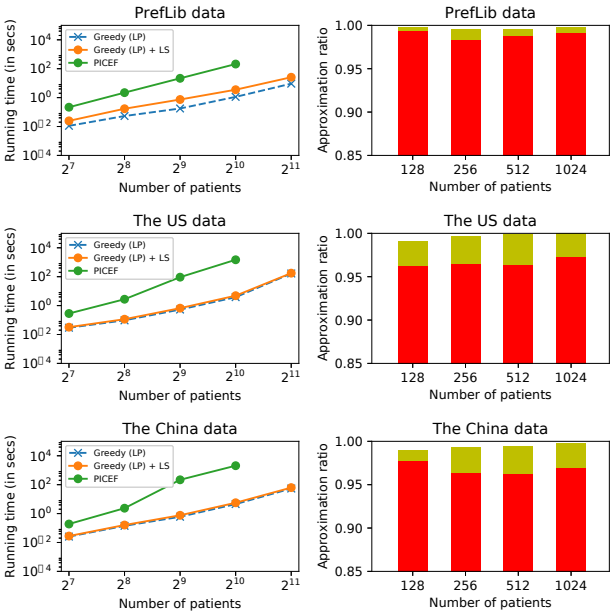

Figure 1: Running time (the charts on the left) and approximation ratio (the charts on the right) of the proposed *Local Search* framework with initial solutions via *Greedy* by solutions of LP relaxation in the settings of having no chains. In the charts showing approximation ratio, the red part indicates the approximation ratio of initial solutions, and the yellow part indicates the improvement after applying *Local Search*.

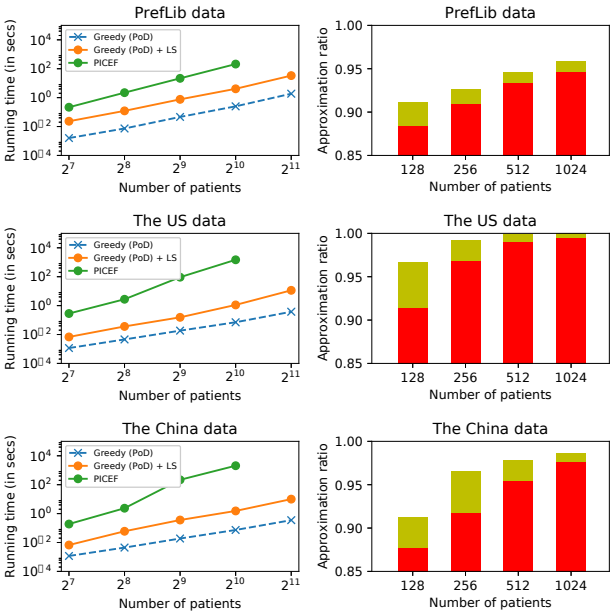

Figure 2: Running time and approximation ratio of the proposed *Local Search* framework with initial solutions via *Greedy* by PoD in the settings of having no chains. Drawn in the same way as Figure 1.

Acknowledgement

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, 61561146398, a Tsinghua Initiative

[1]Recall that the PoD for a vertex v is defined as $(d_{\text{in}}(v) + 1) \cdot (d_{\text{out}}(v) + 1)$ in Section 4.2.

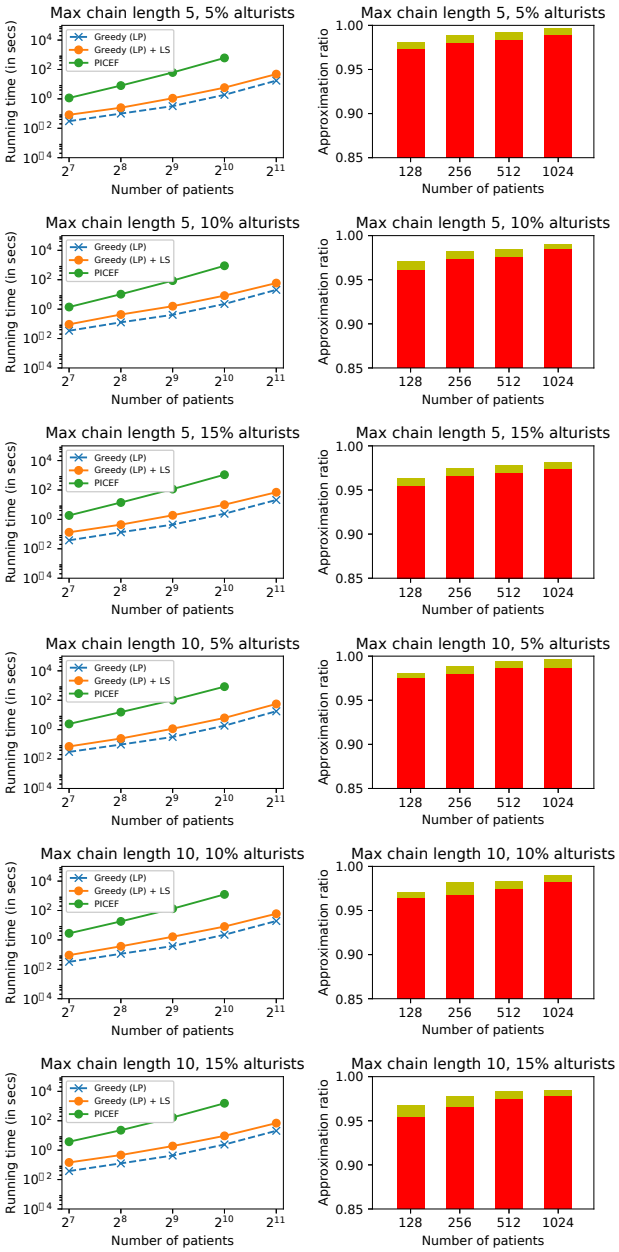

Figure 3: Running time and approximation ratio of the proposed *Local Search* framework with initial solutions via *Greedy* by solutions of LP relaxation in the settings of having chains on PrefLib data. Drawn in the same way as Figure 1.

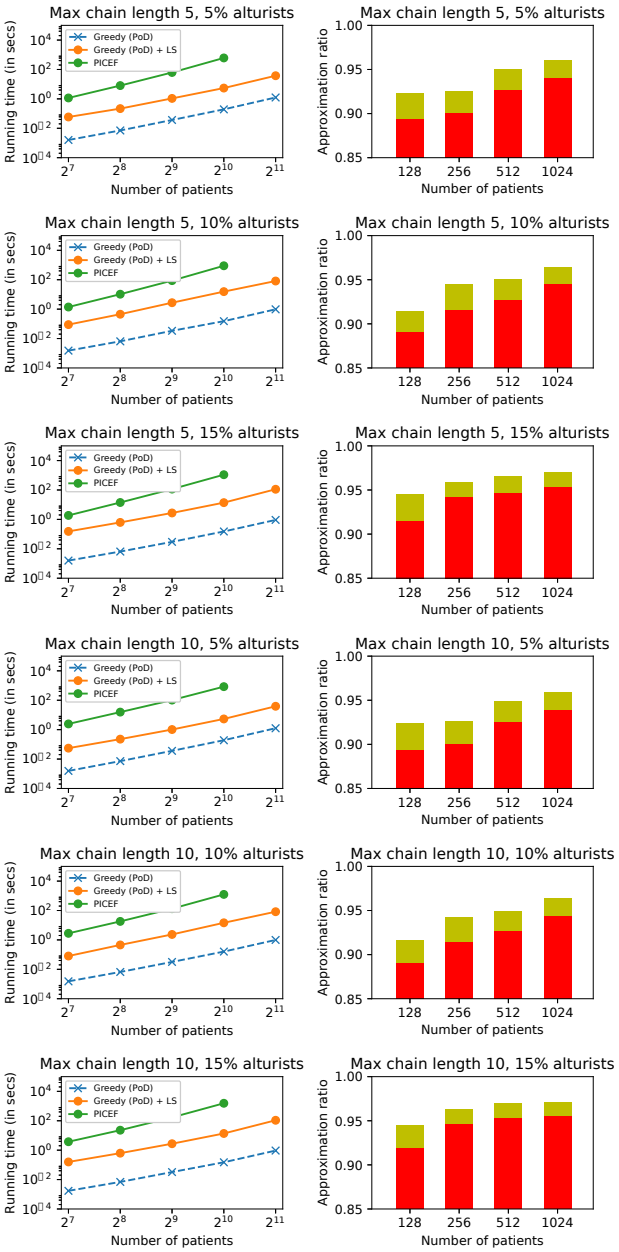

Figure 4: Running time and approximation ratio of the proposed *Local Search* framework with initial solutions via *Greedy* by PoD in the settings of having chains on PrefLib data. Drawn in the same way as Figure 1.

Sampling ratio	Approximation ratio	Running time (in seconds)
1	0.981	59.419
0.1	0.987	4.283
0.01	0.995	1.132
0.001	0.994	0.561

Table 1: Comparison of different sampling ratios for *Greedy* by solutions of LP relaxation. Run on PrefLib data of 1024 patients in the settings of having no chains.

Scientific Research Grant and a China Youth 1000-talent program.

REFERENCES

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

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

[3] D. J. Abraham, A. Blum, and T. 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.

[4] M. Akbarpour, S. Li, and S. O. Gharan. Dynamic matching market design. *arXiv preprint arXiv:1402.3643*, 2014.

[5] F. Alvelos, X. Klimentova, A. Rais, and A. Viana. A compact formulation for maximizing the expected number of transplants in kidney exchange programs. In *Journal of Physics: Conference Series*, volume 616, page 012011. IOP Publishing, 2015.

[6] R. Anderson, I. Ashlagi, D. Gamarnik, and A. E. Roth. Finding long chains in kidney exchange using the traveling salesman problem. *Proceedings of the National Academy of Sciences*, 112(3):663–668, 2015.

[7] R. M. Anderson. *Stochastic models and data driven simulations for healthcare operations*. PhD thesis, Massachusetts Institute of Technology, 2014.

[8] I. Ashlagi, F. Fischer, I. A. Kash, and A. D. Procaccia. Mix and match: A strategyproof mechanism for multi-hospital kidney exchange. *Games and Economic Behavior*, 91:284–296, 2015.

[9] I. Ashlagi, D. Gamarnik, M. A. Rees, and A. E. Roth. The need for (long) chains in kidney exchange. Technical report, National Bureau of Economic Research, 2012.

[10] I. Ashlagi, P. Jaillet, and V. H. Manshadi. Kidney exchange in dynamic sparse heterogenous pools. *arXiv preprint arXiv:1301.3509*, 2013.

[11] I. Ashlagi and A. E. Roth. Free riding and participation in large scale, multi-hospital kidney exchange. *Theoretical Economics*, 9(3):817–863, 2014.

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

[13] P. Biro and K. Cechlarova. Inapproximability of the kidney exchange problem. *Institute of Mathematics, PJ Safarik University, Slovakia. Mimeo*, 2006.

[14] P. Biro, D. F. Manlove, and R. Rizzi. Maximum weight cycle packing in directed graphs, with application to kidney exchange programs. *Discrete Mathematics, Algorithms and Applications*, 1(04):499–517, 2009.

[15] A. Blum, I. Caragiannis, N. Haghtalab, A. D. Procaccia, E. B. Procaccia, and R. Vaish. Opting into optimal matchings.

[16] A. Blum, J. P. Dickerson, N. Haghtalab, A. D. Procaccia, T. Sandholm, and A. Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 325–342. ACM, 2015.

[17] A. Blum, A. Gupta, A. Procaccia, and A. Sharma. Harnessing the power of two crossmatches. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 123–140. ACM, 2013.

[18] M. Constantino, X. Klimentova, A. Viana, and A. Rais. New insights on integer-programming models for the kidney exchange problem. *European Journal of Operational Research*, 231(1):57–68, 2013.

[19] M. Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 509–518. IEEE, 2013.

[20] J. Dickerson. Personal communication.

[21] J. P. Dickerson. *A Unified Approach to Dynamic Matching and Barter Exchange*. PhD thesis, Stanford University, 2015.

[22] J. P. Dickerson, D. F. Manlove, B. Plaut, T. Sandholm, and J. Trimble. Position-indexed formulations for kidney exchange. *arXiv preprint arXiv:1606.01623*, 2016.

[23] J. P. Dickerson, A. D. Procaccia, and T. Sandholm. Dynamic matching via weighted myopia with application to kidney exchange. In *AAAI*, 2012.

[24] J. P. Dickerson, A. D. Procaccia, and T. Sandholm. Optimizing kidney exchange with transplant chains: Theory and reality. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 711–718. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[25] J. P. Dickerson, A. D. Procaccia, and T. Sandholm. Failure-aware kidney exchange. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 323–340. ACM, 2013.

[26] J. P. Dickerson, A. D. Procaccia, and T. Sandholm. Price of fairness in kidney exchange. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1013–1020. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[27] J. P. Dickerson and T. Sandholm. Futurematch: Combining human value judgments and machine learning to match in dynamic environments. In *AAAI*, pages 622–628, 2015.

[28] Y. Ding, D. Ge, S. He, and C. T. Ryan. A non-asymptotic approach to analyzing kidney exchange graphs. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 257–258. ACM, 2015.

[29] W. Fang, P. Tang, and S. Zuo. Digital good exchange. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 264–270, 2016.

[30] K. M. Glorie, J. J. van de Klundert, and A. P. Wagelmans. Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manufacturing & Service Operations Management*, 16(4):498–512, 2014.

[31] X. Klimentova, F. Alvelos, and A. Viana. A new branch-and-price approach for the kidney exchange problem. In *International Conference on Computational Science and Its Applications*, pages 237–252. Springer, 2014.

[32] J. Li, Y. Liu, L. Huang, and P. Tang. Egalitarian pairwise kidney exchange: fast algorithms vialinear programming and parametric flow. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 445–452. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[33] Y. Liu, P. Tang, and W. Fang. Internally stable matchings and exchanges. In *AAAI*, pages 1433–1439, 2014.

[34] S. Luo and P. Tang. Mechanism design and implementation for lung exchange. In *IJCAI*, 2015.

[35] S. Luo, P. Tang, C. Wu, and J. Zeng. Approximation of barter exchanges with cycle length constraints. *arXiv preprint arXiv:1605.08863*, 2016.

[36] V. Mak-Hau. On the kidney exchange problem: cardinality constrained cycle and chain problems on directed graphs: a survey of integer programming approaches. *Journal of Combinatorial Optimization*, pages 1–25, 2015.

[37] D. F. Manlove and G. O'Malley. Paired and altruistic kidney donation in the uk: Algorithms and experimentation. In *International Symposium on Experimental Algorithms*, pages 271–282. Springer, 2012.

[38] N. Mattei and T. Walsh. Preflib: A library for preferences http://www. preflib. org. In *International Conference on Algorithmic DecisionTheory*, pages 259–270. Springer, 2013.

[39] R. A. Montgomery, S. E. Gentry, W. H. Marks, D. S. Warren, J. Hiller, J. Houp, A. A. Zachary, J. K. Melancon, W. R. Maley, H. Rabb, et al. Domino paired kidney donation: a strategy to make best use of live non-directed donation. *The Lancet*, 368(9533):419–421, 2006.

[40] J. P. Pedroso. Maximizing expectation on vertex-disjoint cycle packing. In *International Conference on Computational Science and Its Applications*, pages 32–46. Springer, 2014.

[41] B. Plaut, J. P. Dickerson, and T. Sandholm. Fast optimal clearing of capped-chain barter exchanges. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[42] F. T. Rapaport. The case for a living emotionally related international kidney donor exchange registry. In *Transplantation proceedings*, volume 18, pages 5–9, 1986.

[43] M. A. Rees, J. E. Kopke, R. P. Pelletier, D. L. Segev, M. E. Rutter, A. J. Fabrega, J. Rogers, O. G. Pankewycz, J. Hiller, A. E. Roth, et al. A nonsimultaneous, extended, altruistic-donor chain. *New England Journal of Medicine*, 360(11):1096–1101, 2009.

[44] A. E. Roth, T. Sönmez, and M. Ünver. Kidney exchange. *The Quarterly Journal of Economics*, 119(2):457–488, 2004.

[45] A. E. Roth, T. Sönmez, and M. Ünver. Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *The American economic review*, 97(3):828–851, 2007.

[46] A. E. Roth, T. Sonmez, and M. U. Unver. Kidney exchange. Technical report, National Bureau of Economic Research, 2003.

[47] A. E. Roth, T. Sönmez, and M. U. Ünver. A kidney exchange clearinghouse in new england. *American Economic Review*, pages 376–380, 2005.

[48] A. E. Roth, T. Sönmez, and M. U. Ünver. Pairwise kidney exchange. *Journal of Economic theory*, 125(2):151–188, 2005.

[49] A. E. Roth, T. Sönmez, M. U. Ünver, F. L. Delmonico, and S. L. Saidman. Utilizing list exchange and nondirected donation through 'chain' paired kidney donations. *American Journal of transplantation*, 6(11):2694–2705, 2006.

[50] L. Shapley and H. Scarf. On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37, 1974.

[51] T. Sönmez and T. B. Switzer. Matching with (branch-of-choice) contracts at the united states military academy. *Econometrica*, 81(2):451–488, 2013.

[52] P. Toulis and D. C. Parkes. Design and analysis of multi-hospital kidney exchange mechanisms using random graphs. *Games and Economic Behavior*, 91:360–382, 2015.

[53] M. U. Ünver. Dynamic kidney exchange. *The Review of Economic Studies*, 77(1):372–414, 2010.