Clearing Algorithms for Barter Exchange Markets: Enabling Nationwide Kidney Exchanges

David J. Abraham
Computer Science Department
Carnegie Mellon University
dabraham@cs.cmu.edu

Avrim Blum
Computer Science Department
Carnegie Mellon University
avrim@cs.cmu.edu

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
sandholm@cs.cmu.edu

ABSTRACT

In barter-exchange markets, agents seek to swap their items with one another, in order to improve their own utilities. These swaps consist of cycles of agents, with each agent receiving the item of the next agent in the cycle. We focus mainly on the upcoming national kidney-exchange market, where patients with kidney disease can obtain compatible donors by swapping their own willing but incompatible donors. With over 70,000 patients already waiting for a cadaver kidney in the US, this market is seen as the only ethical way to significantly reduce the 4,000 deaths per year attributed to kidney disease.

The clearing problem involves finding a social welfare maximizing exchange when the maximum length of a cycle is fixed. Long cycles are forbidden, since, for incentive reasons, all transplants in a cycle must be performed simultaneously. Also, in barter-exchanges generally, more agents are affected if one drops out of a longer cycle. We prove that the clearing problem with this cycle-length constraint is NP-hard. Solving it exactly is one of the main challenges in establishing a national kidney exchange.

We present the first algorithm capable of clearing these markets on a nationwide scale. The key is *incremental problem formulation.* We adapt two paradigms for the task: constraint generation and column generation. For each, we develop techniques that dramatically improve both runtime and memory usage. We conclude that column generation scales drastically better than constraint generation. Our algorithm also supports several generalizations, as demanded by real-world kidney exchanges.

Our algorithm replaced CPLEX as the clearing algorithm of the *Alliance for Paired Donation*, one of the leading kidney exchanges. The match runs are conducted every two weeks and transplants based on our optimizations have already been conducted.

EC'07, June 11–15, 2007, San Diego, California, USA.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*; F.2 [**Analysis of Algorithms and Problem Complexity**]: [General]

General Terms

Algorithms, Economics

Keywords

Barter, Exchange, Matching, Column Generation, Branch-and-Price, Kidney, Transplant

1. INTRODUCTION

The role of kidneys is to filter waste from blood. Kidney failure results in accumulation of this waste, which leads to death in months. One treatment option is dialysis, in which the patient goes to a hospital to have his/her blood filtered by an external machine. Several visits are required per week, and each takes several hours. The quality of life on dialysis can be extremely low, and in fact many patients opt to withdraw from dialysis, leading to a natural death. Only 12% of dialysis patients survive 10 years [23].

Instead, the preferred treatment is a kidney transplant. Kidney transplants are by far the most common transplant. Unfortunately, the demand for kidneys far outstrips supply. In the United States in 2005, 4,052 people died waiting for a life-saving kidney transplant. During this time, almost 30,000 people were added to the national waiting list, while only 9,913 people left the list after receiving a deceased-donor kidney. The waiting list currently has over 70,000 people, and the median waiting time ranges from 2 to 5 years, depending on blood type.[1]

For many patients with kidney disease, the best option is to find a *living* donor, that is, a healthy person willing to donate one of his/her two kidneys. Although there are marketplaces for buying and selling living-donor kidneys, the commercialization of human organs is almost universally regarded as unethical, and the practice is often explicitly illegal, such as in the US. However, in most countries, live donation is legal, provided it occurs as a gift with no financial compensation. In 2005, there were 6,563 live donations in the US.

The number of live donations would have been much higher if it were not for the fact that, frequently, a potential donor

[1] Data from the United Network for Organ Sharing [21].

and his intended recipient are blood-type or tissue-type incompatible. In the past, the incompatible donor was sent home, leaving the patient to wait for a deceased-donor kidney. However, there are now a few regional kidney exchanges in the United States, in which patients can swap their incompatible donors with each other, in order to each obtain a compatible donor.

These markets are examples of *barter exchanges*. In a barter-exchange market, *agents* (patients) seek to swap their *items* (incompatible donors) with each other. These swaps consist of cycles of agents, with each agent receiving the item of the next agent in the cycle. Barter exchanges are ubiquitous: examples include Peerflix (DVDs) [11], Read It Swap It (books) [12], and Intervac (holiday houses) [9]. For many years, there has even been a large shoe exchange in the United States [10]. People with different-sized feet use this to avoid having to buy two pairs of shoes. Leg amputees have a separate exchange to share the cost of buying a single pair of shoes.

We can encode a barter exchange market as a directed graph $G = (V, E)$ in the following way. Construct one vertex for each agent. Add a weighted edge e from one agent v_i to another v_j, if v_i wants the item of v_j. The weight w_e of e represents the utility to v_i of obtaining v_j's item. A cycle c in this graph represents a possible swap, with each agent in the cycle obtaining the item of the next agent. The weight w_c of a cycle c is the sum of its edge weights. An *exchange* is a collection of disjoint cycles. The weight of an exchange is the sum of its cycle weights. A *social welfare* maximizing exchange is one with maximum weight.

Figure 1 illustrates an example market with 5 agents, $\{v_1, v_2, \ldots, v_5\}$, in which all edges have weight 1. The market has 4 cycles, $c_1 = \langle v_1, v_2\rangle$, $c_2 = \langle v_2, v_3\rangle$, $c_3 = \langle v_3, v_4\rangle$ and $c_4 = \langle v_1, v_2, v_3, v_4, v_5\rangle$, and two (inclusion) maximal exchanges, namely $M_1 = \{c_4\}$ and $M_2 = \{c_1, c_3\}$. Exchange M_1 has both maximum weight and maximum cardinality (i.e., it includes the most edges/vertices).

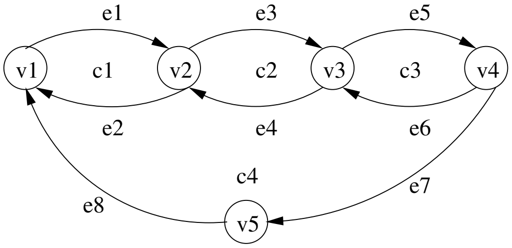

Figure 1: Example barter exchange market.

The *clearing problem* is to find a maximum-weight exchange consisting of cycles with length at most some small constant L. This cycle-length constraint arises naturally for several reasons. For example, in a kidney exchange, all operations in a cycle have to be performed simultaneously; otherwise a donor might back out after his incompatible partner has received a kidney. (One cannot write a binding contract to donate an organ.) This gives rise to a logistical constraint on cycle size: even if all the donors are operated on first and the same personnel and facilities are used to then operate on the donees, a k-cycle requires between $3k$ and $6k$ doctors, around $4k$ nurses, and almost $2k$ operating rooms.

Due to such resource constraints, the upcoming national kidney exchange market will likely allow only cycles of length 2 and 3. Another motivation for short cycles is that if the cycle fails to exchange, fewer agents are affected. For example, last-minute testing in a kidney exchange often reveals new incompatibilities that were not detected in the initial testing (based on which the compatibility graph was constructed). More generally, an agent may drop out of a cycle if his preferences have changed, or he/she simply fails to fulfill his obligations (such as sending a book to another agent in the cycle) due to forgetfulness.

In Section 3, we show that (the decision version of) the clearing problem is NP-complete for $L \geq 3$. One approach then might be to look for a good heuristic or approximation algorithm. However, for two reasons, we aim for an exact algorithm based on an *integer-linear program (ILP)* formulation, which we solve using specialized tree search.

- First, any loss of optimality could lead to unnecessary patient deaths.

- Second, an attractive feature of using an ILP formulation is that it allows one to easily model a number of variations on the objective, and to add additional constraints to the problem. For example, if 3-cycles are believed to be more likely to fail than 2-cycles, then one can simply give them a weight that is appropriately lower than 3/2 the weight of a 2-cycle. Or, if for various (e.g., ethical) reasons one requires a maximum cardinality exchange, one can at least in a second pass find the solution (out of all maximum cardinality solutions) that has the fewest 3-cycles. Other variations one can solve for include finding various forms of "fault tolerant" (non-disjoint) collections of cycles in the event that certain pairs that were thought to be compatible turn out to be incompatible after all.

In this paper, we present the first algorithm capable of clearing these markets on a nationwide scale. Straight-forward ILP encodings are too large to even construct on current hardware — not to talk about solving them. The key then is incremental problem formulation. We adapt two paradigms for the task: constraint generation and column generation. For each, we develop a host of (mainly problem-specific) techniques that dramatically improve both runtime and memory usage.

1.1 Prior Work

Several recent papers have used simulations and market-clearing algorithms to explore the impact of a national kidney exchange [13, 20, 6, 14, 15, 17]. For example, using Edmond's maximum-matching algorithm [4], [20] shows that a national pairwise-exchange market (using length-2 cycles only) would result in more transplants, reduced waiting time, and savings of $750 million in heath care costs over 5 years. Those results are conservative in two ways. Firstly, the simulated market contained only 4,000 initial patients, with 250 patients added every 3 months. It has been reported to us that the market could be almost double this size. Secondly, the exchanges were restricted to length-2 cycles (because that is all that can be modeled as maximum matching, and solved using Edmonds's algorithm). Allowing length-3 cycles leads to additional significant gains. This has been demonstrated on kidney exchange markets with 100 patients by using CPLEX to solve an integer-program encoding of the clearing problem [15]. In this paper, we present an alternative algorithm for this integer program that can clear markets with over 10,000 patients (and that same number of willing donors).

Allowing cycles of length more than 3 often leads to no improvement in the size of the exchange [15]. (Furthermore, in a simplified theoretical model, *any* kidney exchange can be converted into one with cycles of length at most 4 [15].) Whilst this does not hold for general barter exchanges, or even for all kidney exchange markets, in Section 5.2.3 we make use of the observation that short cycles suffice to dramatically increase the speed of our algorithm.

At a high-level, the clearing problem for barter exchanges is similar to the clearing problem (aka winner determination problem) in combinatorial auctions. In both settings, the idea is to gather all the pertinent information about the agents into a central clearing point and to run a centralized clearing algorithm to determine the allocation. Both problems are NP-hard. Both are best solved using tree search techniques. Since 1999, significant work has been done in computer science and operations research on faster optimal tree search algorithms for clearing combinatorial auctions. (For a recent review, see [18].) However, the kidney exchange clearing problem (with a limit of 3 or more on cycle size) is different from the combinatorial auction clearing problem in significant ways. The most important difference is that the natural formulations of the combinatorial auction problem tend to easily fit in memory, so time is the bottleneck in practice. In contrast, the natural formulations of the kidney exchange problem (with $L = 3$) take at least cubic space in the number of patients to even model, and therefore memory becomes a bottleneck much before time does when using standard tree search, such as branch-and-cut in CPLEX, to tackle the problem. (On a 1GB computer and a realistic standard instance generator, discussed later, CPLEX 10.010 runs out of memory on five of the ten 900-patient instances and ten of the ten 1,000-patient instances that we generated.) Therefore, the approaches that have been developed for combinatorial auctions cannot handle the kidney exchange problem.

1.2 Paper Outline

The rest of the paper is organized as follows. Section 2 discusses the process by which we generate realistic kidney exchange market data, in order to benchmark the clearing algorithms. Section 3 contains a proof that the market clearing decision problem is NP-complete. Sections 4 and 5 each contain an ILP formulation of the clearing problem. We also detail in those sections our techniques used to solve those programs on large instances. Section 6 presents experiments on the various techniques. Section 7 discusses recent fielding of our algorithm. Finally, we present our conclusions in Section 8, and suggest future research directions.

2. MARKET CHARACTERISTICS AND INSTANCE GENERATOR

We test the algorithms on simulated kidney exchange markets, which are generated by a process described in Saidman *et al.* [17]. This process is based on the extensive nationwide data maintained by the United Network for Organ Sharing (UNOS) [21], so it generates a realistic instance distribution. Several papers have used variations of this process to demonstrate the effectiveness of a national kidney exchange (extrapolating from small instances or restricting the clearing to 2-cycles) [6, 20, 14, 13, 15, 17].

Briefly, the process involves generating patients with a random blood type, sex, and probability of being tissue-type incompatible with a randomly chosen donor. These probabilities are based on actual real-world population data. Each patient is assigned a potential donor with a random blood type and relation to the patient. If the patient and potential donor are incompatible, the two are entered into the market. Blood type and tissue type information is then used to decide on which patients and donors are compatible. One complication, handled by the generator, is that if the patient is female, and she has had a child with her potential donor, then the probability that the two are incompatible increases. (This is because the mother develops antibodies to her partner during pregnancy.) Finally, although our algorithms can handle more general weight functions, patients have a utility of 1 for compatible donors, since their survival probability is not affected by the choice of donor [3]. This means that the maximum-weight exchange has maximum cardinality.

Table 1 gives lower and upper bounds on the size of a maximum-cardinality exchange in the kidney-exchange market. The lower bounds were found by clearing the market with length-2 cycles only, while the upper bounds had no restriction on cycle length. For each market size, the bounds were computed over 10 randomly generated markets. Note that there can be a large amount of variability in the markets - in one 5000 patient market, less than 1000 patients were in the maximum-cardinality exchange.

	Maximum exchange size			
	Length-2 cycles only		Arbitrary cycles	
Patients	Mean	Max	Mean	Max
100	4.00e+1	4.60e+1	5.30e+1	6.10e+1
500	2.58e+2	2.80e+2	2.79e+2	2.97e+2
1000	5.35e+2	6.22e+2	5.61e+2	6.30e+2
2000	1.05e+3	1.13e+3	1.09e+3	1.16e+3
3000	1.63e+3	1.70e+3	1.68e+3	1.73e+3
4000	2.15e+3	2.22e+3	2.20e+3	2.27e+3
5000	2.53e+3	2.87e+3	2.59e+3	2.92e+3
6000	3.26e+3	3.32e+3	3.35e+3	3.39e+3
7000	3.80e+3	3.86e+3	3.89e+3	3.97e+3
8000	4.35e+3	4.45e+3	4.46e+3	4.55e+3
9000	4.90e+3	4.96e+3	5.01e+3	5.07e+3
10000	5.47e+3	5.61e+3	5.59e+3	5.73e+3

Table 1: Upper and lower bounds on exchange size.

Table 2 gives additional characteristics of the kidney-exchange market. Note that a market with 5000 patients can already have more than 450 million cycles of length 2 and 3.

	Edges		Length 2 & 3 cycles	
Patients	Mean	Max	Mean	Max
100	2.38e+3	2.79e+3	2.76e+3	5.90e+3
500	6.19e+4	6.68e+4	3.96e+5	5.27e+5
1000	2.44e+5	2.68e+5	3.31e+6	4.57e+6
2000	9.60e+5	1.02e+6	2.50e+7	3.26e+7
3000	2.19e+6	2.28e+6	8.70e+7	9.64e+7
4000	3.86e+6	3.97e+6	1.94e+8	2.14e+8
5000	5.67e+6	6.33e+6	3.60e+8	4.59e+8
6000	8.80e+6	8.95e+6		
7000	1.19e+7	1.21e+7		
8000	1.56e+7	1.59e+7		
9000	1.98e+7	2.02e+7		
10000	2.44e+7	2.51e+7		

Table 2: Market characteristics.

3. PROBLEM COMPLEXITY

In this section, we prove that (the decision version of) the market clearing problem with short cycles is NP-complete.

THEOREM 1. *Given a graph $G = (V, E)$ and an integer $L \geq 3$, the problem of deciding if G admits a perfect cycle cover containing cycles of length at most L is NP-complete.*

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

One straightforward idea is to construct a tripartite graph with vertex sets $X \cup Y \cup Z$ and directed edges (x_a, y_b), (y_b, z_c), and (z_c, x_a) for each triple $t_i = \{x_a, y_b, z_c\} \in T$. However, it is not too hard to see that this encoding fails because a perfect cycle cover may include a cycle with no corresponding triple.

Instead then, we use the following reduction. Given an instance of 3D-Matching, construct one vertex for each element in X, Y and Z. For each triple, $t_i = \{x_a, y_b, z_c\}$ construct the gadget in Figure 2, which is a similar to one in Garey and Johnson [5, pp 68-69]. Note that the gadgets intersect only on vertices in $X \cup Y \cup Z$. It is clear that this construction can be done in polynomial time.

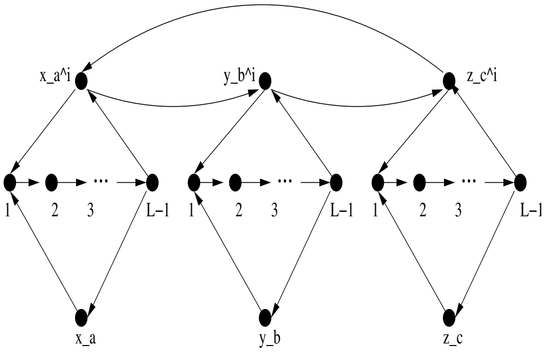

Figure 2: NP-completeness gadget for triple t_i and maximum cycle length L.

Let M be a perfect 3D-Matching. We will show the construction admits a perfect cycle cover by short cycles. If $t_i = \{x_a, y_b, z_c\} \in M$, add from t_i's gadget the three length-L cycles containing x_a, y_b and z_c respectively. Also add the cycle $\langle x_a^i, y_b^i, z_c^i \rangle$. Otherwise, if $t_i \notin M$, add the three length-L cycles containing x_a^i, y_b^i and z_c^i respectively. It is clear that all vertices are covered, since M partitions $X \times Y \times Z$.

Conversely, suppose we have a perfect cover by short cycles. Note that the construction only has short cycles of lengths 3 and L, and no short cycle involves distinct vertices from two different gadgets. It is easy to see then that in a perfect cover, each gadget t_i contributes cycles according to the cases above: $t_i \in M$, or $t_i \notin M$. Hence, there exists a perfect 3D-Matching in the original instance. □

4. SOLUTION APPROACHES BASED ON AN EDGE FORMULATION

In this section, we consider a formulation of the clearing problem as an ILP with one variable for each edge. This encoding is based on the following classical algorithm for solving the directed cycle cover problem with no cycle-length constraints.

Given a market $G = (V, E)$, construct a bipartite graph with one vertex for each agent, and one vertex for each item. Add an edge e_v with weight 0 between each agent v and its own item. At this point, the encoding is a perfect matching. Now, for each edge $e = (v_i, v_j)$ in the original market, add an edge e with weight w_e between agent v_i and the item of v_j. Perfect matchings in this encoding correspond exactly with cycle covers, since whenever an agent's item is taken, it must receive some other agent's item. It follows that the unrestricted clearing problem can be solved in polynomial time by finding a maximum-weight perfect matching. Figure 3 contains the bipartite graph encoding of the example market from Figure 1. The weight-0 edges are encoded by dashed lines, while the market edges are in bold.

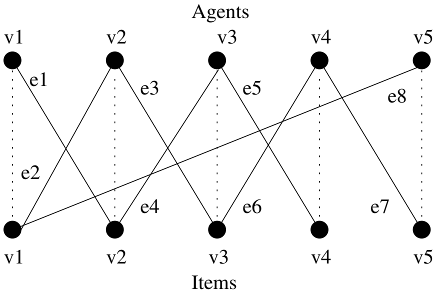

Figure 3: Perfect matching encoding of the market in Figure 1.

Alternatively, we can solve the problem by encoding it as an ILP with one variable for each edge in the *original market graph G*. This ILP, given below, has the advantage that it can be extended naturally to deal with cycle length constraints. Therefore, for the rest of this section, this is the approach we will pursue.

$$\max \sum_{e \in E} w_e e$$

such that for all $v_i \in V$, the conservation constraint

$$\sum_{e_{out}=(v_i,v_j)} e_{out} - \sum_{e_{in}=(v_j,v_i)} e_{in} = 0$$

and capacity constraint

$$\sum_{e_{out}=(v_i,v_j)} e_{out} \leq 1$$

are satisfied.

If cycles are allowed to have length at most L, it is easy to see that we only need to make the following changes to the ILP. For each length-L path (throughout the paper, we do not include cycles in the definition of "path") $p = \langle e_{p_1}, e_{p_2}, \ldots, e_{p_L} \rangle$, add a constraint

$$e_{p_1} + e_{p_2} + \ldots + e_{p_L} \leq L - 1,$$

which precludes path p from being in any feasible solution.

Unfortunately, in a market with only 1000 patients, the number of length-3 paths is in excess of 400 million, and so we cannot even construct this ILP without running out of memory.

Therefore, we use a tree search with an *incremental formulation* approach. Specifically, we use CPLEX, though we add constraints as cutting planes during the tree search process. We begin with only a small subset of the constraints in the ILP. Since this ILP is small, CPLEX can solve its LP relaxation. We then check whether any of the missing constraints are violated by the fractional solution. If so, we generate a set of these constraints, add them to the ILP, and repeat. Even once all constraints are satisfied, there may be no integral solution matching the fractional upper bound, and even if there were, the LP solver might not find it.

In these cases, CPLEX branches on a variable (we used CPLEX's default branching strategy), and generates one new search node corresponding to each of the children. At each node of the search tree that is visited, this process of solving the LP and adding constraints is repeated. Clearly, this approach yields an optimal solution once the tree search finishes.

We still need to explain the details of the constraint seeder (i.e., selecting which constraints to begin with) and the constraint generation (i.e., selecting which violated constraints to include). We describe these briefly in the next two subsections, respectively.

4.1 Constraint Seeder

The main constraint seeder we developed forbids any path of length $L-1$ that does not have an edge closing the cycle from its head to its tail. While it is computationally expensive to find these constraints, their addition focuses the search away from paths that cannot be in the final solution. We also tried seeding the LP with a random collection of constraints from the ILP.

4.2 Constraint Generation

We experimented with several constraint generators. In each, given a fractional solution, we construct the subgraph of edges with positive value. This graph is much smaller than the original graph, so we can perform the following computations efficiently.

In our first constraint generator, we simply search for length-L paths with value sum more than $L-1$. For any such path, we restrict its sum to be at most $L-1$. Note that if there is a cycle c with length $|c| > L$, it could contain as many as $|c|$ violating paths.

In our second constraint generator, we only add one constraint for such cycles: the sum of edges in the cycle can be at most $\lfloor |c|(L-1)/L \rfloor$.

This generator made the algorithm slower, so we went in the other direction in developing our final generator. It adds one constraint per violating path p, and furthermore, it adds a constraint for each path with the same interior vertices (not counting the endpoints) as p. This improved the overall speed.

4.3 Experimental performance

It turned out that even with these improvements, the edge formulation approach cannot clear a kidney exchange with 100 vertices in the time the cycle formulation (described later in Section 5) can clear one with 10,000 vertices. In other words, column generation based approaches turned out to be drastically better than constraint generation based approaches. Therefore, in the rest of the paper, we will focus on the cycle formulation and the column generation based approaches.

5. SOLUTION APPROACHES BASED ON A CYCLE FORMULATION

In this section, we consider a formulation of the clearing problem as an ILP with one variable for each cycle. This encoding is based on the following classical algorithm for solving the directed cycle cover problem when cycles have length 2.

Given a market $G = (V, E)$, construct a new graph on V with a weight w_c edge for each cycle c of length 2. It is easy to see that matchings in this new graph correspond to cycle covers by length-2 cycles in the original market graph. Hence, the market clearing problem with $L = 2$ can be solved in polynomial time by finding a maximum-weight matching.

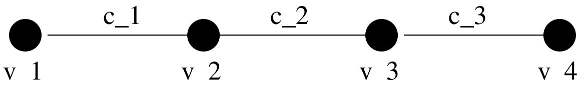

Figure 4: Maximum-weight matching encoding of the market in Figure 1.

We can generalize this encoding for arbitrary L. Let $C(L)$ be the set of all cycles of G with length at most L. Then the following ILP finds the maximum-weight cycle cover by $C(L)$ cycles:

$$\begin{aligned} &\max \sum_{c \in C(L)} w_c c \\ &\text{subject to } \sum_{c: v_i \in c} c \leq 1 \qquad \forall v_i \in V \\ &\text{with } c \in \{0,1\} \quad \forall c \in C(L) \end{aligned}$$

5.1 Edge vs Cycle Formulation

In this section, we consider the merits of the edge formulation and cycle formulation. The edge formulation can be solved in polynomial time when there are no constraints on the cycle size. The cycle formulation can be solved in polynomial time when the cycle size is at most 2.

We now consider the case of short cycles of length at most L, where $L \geq 3$. Our tree search algorithms use the LP relaxation of these formulations to provide upper bounds on the optimal solution. These bounds help prune subtrees and guide the search in the usual ways.

THEOREM 2. *The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.*

PROOF. Consider an optimal solution to the LP relaxation of the cycle formulation. We show how to construct an equivalent solution in the edge formulation. For each edge in the graph, set its value as the sum of values of all the cycles of which it is a member. Also, define the value of a vertex in the same manner. Because of the cycle constraints, the conservation and capacity constraints of the edge encoding are clearly satisfied. It remains to show that none of the path constraints are violated.

Let p be any length-L path in the graph. Since p has $L-1$ interior vertices (not counting the endpoints), the value sum of these interior vertices is at most $L-1$. Now, for any cycle c of length at most L, the number of edges it has in p, which we denote by $e_c(p)$, is at most the number of interior vertices it has in p, which we denote by $v_c(p)$. Hence, $\sum_{e \in p} e = \sum_{c \in C(L)} c * e_c(p) \leq \sum_{c \in C(L)} c * v_c(p) = \sum_{v \in p} v = L-1$. □

The converse of this theorem is not true. Consider a graph which is simply a cycle with n edges. Clearly, the LP relaxation of the cycle formulation has optimal value 0, since there are no cycles of size at most L. However, the edge formulation has a solution of size $n/2$, with each edge having value $1/2$.

Hence, the cycle formulation is tighter than the edge formulation. Additionally, for a graph with m edges, the edge formulation requires $O(m^3)$ constraints, while the cycle formulation requires only $O(m^2)$.

5.2 Column Generation for the LP

Table 2 shows how the number of cycles of length at most 3 grows with the size of the market. With one variable per cycle in the cycle formulation, CPLEX cannot even clear markets with 1,000 patients without running out of memory (see Figure 6). To address this problem, we used an incremental formulation approach.

The first step in LP-guided tree search is to solve the LP relaxation. Since the cycle formulation does not fit in memory, this LP stage would fail immediately without an incremental formulation approach. However, motivated by the observation that an exchange solution can include only a tiny fraction of the cycles, we explored the approach of using column (i.e., cycle) generation.

The idea of column generation is to start with a restricted LP containing only a small number of columns (variables, i.e., cycles), and then to repeatedly add columns until an optimal solution to this partially formulated LP is an optimal solution to the original (aka master) LP. We explain this further by way of an example.

Consider the market in Figure 1 with $L = 2$. Figure 5 gives the corresponding master LP, P, and its dual, D.

$$\begin{array}{llllll|l}
 & & \textsc{Primal } P & & & & \\
\max & 2c_1 & +2c_2 & +2c_3 & & & \\
\text{s.t.} & c_1 & & & & \le 1 & (v_1) \\
 & c_1 & +c_2 & & & \le 1 & (v_2) \\
 & & +c_2 & +c_3 & & \le 1 & (v_3) \\
 & & & +c_3 & & \le 1 & (v_4) \\
\text{with} & c_1, c_2, c_3 & & & & \ge 0 &
\end{array}$$

$$\begin{array}{llllll|l}
 & & \textsc{Dual } D & & & & \\
\min & v_1 & +v_2 & +v_3 & +v_4 & & \\
\text{s.t} & v_1 & +v_2 & & & \ge 2 & (c_1) \\
 & & +v_2 & +v_3 & & \ge 2 & (c_2) \\
 & & & +v_3 & +v_4 & \ge 2 & (c_3) \\
\text{with} & v_1, v_2, v_3, v_4 & & & & \ge 0 &
\end{array}$$

Figure 5: Cycle formulation.

Let P' be the restriction of P containing columns c_1 and c_3 only. Let D' be the dual of P', that is, D' is just D without the constraint c_2. Because P' and D' are small, we can solve them to obtain $OPT(P') = OPT(D') = 4$, with $c_{OPT(P')} = \langle c_1 = c_3 = 1\rangle$ and $v_{OPT(D')} = \langle v_1 = v_2 = v_3 = v_4 = 1\rangle$.

While $c_{OPT(P')}$ must be a feasible solution of P, it turns out (fortunately) that $v_{OPT(D')}$ is feasible for D, so that $OPT(D') \ge OPT(D)$. We can verify this by checking that $v_{OPT(D')}$ satisfies the constraints of D not already in D' — i.e. constraint c_2. It follows that $OPT(P') = OPT(D') \ge OPT(D) = OPT(P)$, and so $v_{OPT(P')}$ is provably an optimal solution for P, even though P' is contains a only strict subset of the columns of P.

Of course, it may turn out (unfortunately) that $v_{OPT(D')}$ is not feasible for D. This can happen above if $v_{OPT(D')} = \langle v_1 = 2, v_2 = 0, v_3 = 0, v_4 = 2\rangle$. Although we can still see that $OPT(D') = OPT(D)$, in general we cannot *prove* this because D and P are too large to solve. Instead, because constraint c_2 is violated, we add column c_2 to P', update D', and repeat. The problem of finding a violated constraint is called the *pricing* problem. Here, the *price* of a column (cycle in our setting) is the difference between its weight, and the dual-value sum of the cycle's vertices. If any column of P has a positive price, its corresponding constraint is violated and we have not yet proven optimality. In this case, we must continue generating columns to add to P'.

5.2.1 Pricing Problem

For smaller instances, we can maintain an explicit collection of all feasible cycles. This makes the pricing problem easy and efficient to solve: we simply traverse the collection of cycles, and look for cycles with positive price. We can even find cycles with the most positive price, which are the ones most likely to improve the objective value of restricted LP [1]. This approach does not scale however. A market with 5000 patients can have as many as 400 million cycles of length at most 3 (see Table 2). This is too many cycles to keep in memory.

Hence, for larger instances, we have to *generate* feasible cycles while looking for one with a positive price. We do this using a depth-first search algorithm on the market graph (see Figure 1). In order to make this search faster, we explore vertices in non-decreasing value order, as these vertices are more likely to belong to cycles with positive weight. We also use several pruning rules to determine if the current search path can lead to a positive weight cycle. For example, at a given vertex in the search, we can prune based on the fact that every vertex we visit from this point onwards will have value at least as great the current vertex.

Even with these pruning rules, column generation is a bottleneck. Hence, we also implemented the following optimizations.

Whenever the search exhaustively proves that a vertex belongs to no positive price cycle, we mark the vertex and do not use it as the root of a depth-first search until its dual value decreases. In this way, we avoid unnecessarily repeating our computational efforts from a previous column generation iteration.

Finally, it can sometimes be beneficial for column generation to include several positive-price columns in one iteration, since it may be faster to generate a second column, once the first one is found. However, we avoid this for the following reason. If we attempt to find more positive-price columns than there are to be found, or if the columns are far apart in the search space, we end up having to generate and check a large part of the collection of feasible cycles. In our experiments, we have seen this occur in markets with hundreds of millions of cycles, resulting in prohibitively expensive computation costs.

5.2.2 Column Seeding

Even if there is only a small gap to the master LP relaxation, column generation requires many iterations to improve the objective value of the restricted LP. Each of these iterations is expensive, as we must solve the pricing problem, and re-solve the restricted LP. Hence, although we could begin with no columns in the restricted LP, it is much faster to seed the LP with enough columns that the optimal objective value is not too far from the master LP. Of course, we cannot include so many columns that we run out of memory.

We experimented with several column seeders. In one class of seeder, we use a heuristic to find an exchange, and then add the cycles of that exchange to the initial restricted LP. We implemented two heuristics. The first is a greedy algorithm: for each vertex in a random order, if it is uncovered, we attempt to include a cycle containing it and other uncovered vertices. The other heuristic uses specialized maximum-weight matching code [16] to find an optimal cover by length-2 cycles.

These heuristics perform extremely well, especially taking into account the fact that they only add a small number of columns. For example, Table 1 shows that an optimal cover by length-2 cycles has almost as much weight as the exchange with unrestricted cycle size. However, we have enough memory to include hundreds-of-thousands of additional columns and thereby get closer still to the upper bound.

Our best column seeder constructs a random collection of feasible cycles. Since a market with 5000 patients can have as many as 400 million feasible cycles, it takes too long to generate and traverse all feasible cycles, and so we do not include a uniformly random collection. Instead, we perform a random walk on the market graph (see, for example, Figure 1), in which, after each step of the walk, we test whether there is an edge back onto our path that forms a feasible cycle. If we find a cycle, it is included in the restricted LP, and we start a new walk from a random vertex. In our experiments (see Section 6), we use this algorithm to seed the LP with 400,000 cycles.

This last approach outperforms the heuristic seeders described above. However, in our algorithm, we use a combination that takes the union of all columns from all three seeders. In Figure 6, we compare the performance of the combination seeder against the combination without the random collection seeder. We do not plot the performance of the algorithm without any seeder at all, because it can take hours to clear markets we can otherwise clear in a few minutes.

5.2.3 Proving Optimality

Recall that our aim is to find an optimal solution to the master LP relaxation. Using column generation, we can prove that a restricted-primal solution is optimal once all columns have non-positive prices. Unfortunately though, our clearing problem has the so-called *tailing-off effect* [1, Section 6.3], in which, even though the restricted primal is optimal in hindsight, a large number of additional iterations are required in order to *prove* optimality (i.e., eliminate all positive-price columns). There is no good general solution to the tailing-off effect.

However, to mitigate this effect, we take advantage of the following problem-specific observation. Recall from Section 1.1 that, almost always, a maximum-weight exchange with cycles of length at most 3 has the same weight as an unrestricted maximum-weight exchange. (This does not mean that the solver for the unrestricted case will find a solution with short cycles, however.) Furthermore, the unrestricted clearing problem can be solved in polynomial time (recall Section 4). Hence, we can efficiently compute an upper bound on the master LP relaxation, and, whenever the restricted primal achieves this upper bound, we have proven optimality without necessarily having to eliminate all positive-price columns!

In order for this to improve the running time of the overall algorithm, we need to be able to clear the unrestricted market in less time than it takes column generation to eliminate all the positive-price cycles. Even though the first problem is polynomial-time solvable, this is not trivial for large instances. For example, for a market with 10,000 patients and 25 million edges, specialized maximum-weight matching code [16] was too slow, and CPLEX ran out of memory on the edge formulation encoding from Section 4. To make this idea work then, we used column generation to solve the edge formulation.

This involves starting with a small random subset of the edges, and then adding positive price edges one-by-one until none remain. We conduct this secondary column generation not in the original market graph G, but in the perfect matching bipartite graph of Figure 3. We do this so that we only need to solve the LP, not the ILP, since the integrality gap in the perfect matching bipartite graph is 1—i.e. there always exists an integral solution that achieves the fractional upper bound.

The resulting speedup to the overall algorithm is dramatic, as can be seen in Figure 6.

5.2.4 Column Management

If the optimal value of the initial restricted LP P' is far from the the master LP P, then a large number of columns are generated before the gap is closed. This leads to memory problems on markets with as few as 4,000 patients. Also, even before memory becomes an issue, the column generation iterations become slow because of the additional overhead of solving a larger LP.

To address these issues, we implemented a column management scheme to limit the size of the restricted LP. Whenever we add columns to the LP, we check to see if it contains more than a threshold number of columns. If this is the case, we selectively remove columns until it is again below the threshold[2]. As we discussed earlier, only a tiny fraction of all the cycles will end up in the final solution. It is unlikely that we delete such a cycle, and even if we do, it can always be generated again. Of course, we must not be too aggressive with the threshold, because doing so may offset the per-iteration performance gains by significantly increasing the number of iterations required to get a suitable column set in the LP at the same time.

There are some columns we never delete, for example those we have branched on (see Section 5.3.2), or those with a non-zero LP value. Amongst the rest, we delete those with the lowest price, since those correspond to the dual constraints that are *most* satisfied. This column management scheme works well and has enabled us to clear markets with 10,000 patients, as seen in Figure 6.

5.3 Branch-and-Price Search for the ILP

Given a large market clearing problem, we can successfully solve its LP relaxation to optimality by using the column generation enhancements described above. However, the solutions we find are usually fractional. Thus the next

[2] Based on memory size, we set the threshold at 400,000.

step involves performing a *branch-and-price* tree search [1] to find an optimal integral solution.

Briefly, this is the idea of branch-and-price. Whenever we set a fractional variable to 0 or 1 (branch), both the master LP, and the restriction we are working with, are changed (constrained). By default then, we need to perform column generation (go through the effort of pricing) at each node of the search tree to prove that the constrained restriction is optimal for constrained master LP. (However, as discussed in Section 5.2.3, we compute the integral upper bound for the root node based on relaxing the cycle length constraint completely, and whenever any node's LP in the tree achieves that value, we do not need to continue pricing columns at that node.)

For the clearing problem with cycles of length at most 3, we have found that there is rarely a gap between the optimal integral and fractional solutions. This means we can largely avoid the expensive per node pricing step: whenever the constrained restricted LP has the same optimal value as its parent in the tree search, we can prove LP optimality, as in Section 5.2.3, without having to include any additional columns in the restricted LP.

Although CPLEX can solve ILPs, it does not support branch-and-price (for example, because there can be problem-specific complications involving the interaction between the branching rule and the pricing problem). Hence, we implemented our own branch-and-price algorithm, which explores the search tree in depth-first order. We also experimented with the A* node selection order [7, 2]. However, this search strategy requires significantly more memory, which we found was better employed in making the column generation phase faster (see Section 5.2.2). The remaining major components of the algorithm are described in the next two subsections.

5.3.1 Primal Heuristics

Before branching on a fractional variable, we use primal heuristics to construct a feasible integral solution. These solutions are lower bounds on the final optimal integral solutions. Hence, whenever a restricted fractional solution is no better than the best integral solution found so far, we prune the current subtree. A primal heuristic is effective if it is efficient and constructs tight lower bounds.

We experimented with two primal heuristics. The first is a simple rounding algorithm [8]: include all cycles with fractional value at least 0.5, and then, ensuring feasibility, greedily add the remaining cycles. Whilst this heuristic is efficient, we found that the lower bounds it constructs rarely enable much pruning.

We also tried using CPLEX as a primal heuristic. At any given node of the search tree, we can convert the restricted LP relaxation back to an ILP by reintroducing the integrality constraints. CPLEX has several built-in primal heuristics, which we can apply to this ILP. Moreover, we can use CPLEX's own tree search to find an optimal integral solution. In general, this tree search is much faster than our own.

If CPLEX finds an integral solution that matches the fractional upper bound at the root node, we are done. Otherwise, no such integral solution exists, or we don't yet have the right combination of cycles in the restricted LP. For kidney-exchange markets, it is usually the second reason that applies (see Sections 5.2.2 and 5.2.4). Hence, at some point in the tree search, once more columns have been generated as a result of branching, the CPLEX heuristic will find an optimal integral solution.

Although CPLEX tree search is faster than our own, it is not so fast that we can apply it to every node in our search tree. Hence, we make the following optimizations. Firstly, we add a constraint that requires the objective value of the ILP to be as large as the fractional target. If this is not the case, we want to abort and proceed to generate more columns with our branch-and-price search. Secondly, we limit the number of nodes in CPLEX's search tree. This is because we have observed that no integral solution exists, CPLEX can take a very long time to prove that. Finally, we only apply the CPLEX heuristic at a node if it has a sufficiently different set of cycles from its parent.

Using CPLEX as a primal heuristic has a large impact because it makes the search tree smaller, so all the computationally expensive pricing work is avoided at nodes that are not generated in this smaller tree.

5.3.2 Cycle Brancher

We experimented with two branching strategies, both of which select one variable per node. The first strategy, branching by certainty, randomly selects a variable from those whose LP value is closest to 1. The second strategy, branching by uncertainty, randomly selects a variable whose LP value is closest to 0.5. In either case, two children of the node are generated corresponding to two subtrees, one in which the variable is set to 0, the other in which it is set to 1. Our depth-first search always chooses to explore first the subtree in which the value of the variable is closest to its fractional value.

For our clearing problem with cycles of length at most 3, we found branching by uncertainty to be superior, rarely requiring any backtracking.

6. EXPERIMENTAL RESULTS

All our experiments were performed in Linux (Red Hat 9.0), using a Dell PC with a 3GHz Intel Pentium 4 processor, and 1GB of RAM. Wherever we used CPLEX (e.g., in solving the LP and as a primal heuristic, as discussed in the previous sections), we used CPLEX 10.010.

Figure 6 shows the runtime performance of four clearing algorithms. For each market size listed, we randomly generated 10 markets, and attempted to clear them using each of the algorithms.

The first algorithm is CPLEX on the full cycle formulation. This algorithm fails to clear any markets with 1000 patients or more. Also, its running time on markets smaller than this is significantly worse than the other algorithms.

The other algorithms are variations of the incremental column generation approach described in Section 5. We begin with the following settings (all optimizations are switched on):

Category	Setting
Column Seeder	Combination of greedy exchange and maximum-weight matching heuristics, and random walk seeder (400,000 cycles).
Column Generation	One column at a time.
Column Management	On, with 400,000 column limit.
Optimality Prover	On.
Primal Heuristic	Rounding & CPLEX tree search.
Branching Rule	Uncertainty.

The combination of these optimizations allows us to easily clear markets with over 10,000 patients. In each of the next two algorithms, we turn one of these optimizations off to highlight its effectiveness.

First, we restrict the seeder so that it only begins with 10,000 cycles. This setting is faster for smaller instances, since the LP relaxations are smaller, and faster to solve. However, at 5000 vertices, this effect starts to be offset by the additional column generation that must be performed. For larger instance, this restricted seeder is clearly worse.

Finally, we restore the seeder to its optimized setting, but this time, remove the optimality prover described in Section 5.2.3. As in many column generation problems, the tailing-off effect is substantial. By taking advantage of the properties of our problem, we manage to clear a market with 10,000 patients in about the same time it would otherwise have taken to clear a 6000 patient market.

7. FIELDING THE TECHNOLOGY

Our algorithm and implementation replaced CPLEX as the clearing algorithm of the *Alliance for Paired Donation*, one of the leading kidney exchanges, in December 2006. We conduct a match run every two weeks, and the first transplants based on our solutions have already been conducted.

While there are (for political/inter-personal reasons) at least four kidney exchanges in the US currently, everyone understands that a unified unfragmented national exchange would save more lives. We are in discussions with additional kidney exchanges that are interested in adopting our technology. This way our technology (and the processes around it) will hopefully serve as a substrate that will eventually help in unifying the exchanges. At least computational scalability is no longer an obstacle.

8. CONCLUSION AND FUTURE RESEARCH

In this work we have developed the most scalable exact algorithms for barter exchanges to date, with special focus on the upcoming national kidney-exchange market in which patients with kidney disease will be matched with compatible donors by swapping their own willing but incompatible donors. With over 70,000 patients already waiting for a cadaver kidney in the US, this market is seen as the only ethical way to significantly reduce the 4,000 deaths per year attributed to kidney disease.

Our work presents the first algorithm capable of clearing these markets on a nationwide scale. It optimally solves the kidney exchange clearing problem with 10,000 donor-donee pairs. Thus there is no need to resort to approximate solutions. The best prior technology (vanilla CPLEX) cannot handle instances beyond about 900 donor-donee pairs because it runs out of memory. The key to our improvement is *incremental problem formulation*. We adapted two paradigms for the task: constraint generation and column generation. For each, we developed a host of techniques that substantially improve both runtime and memory usage. Some of the techniques use domain-specific observations while others are domain independent. We conclude that column generation scales dramatically better than constraint generation. For column generation in the LP, our enhancements include pricing techniques, column seeding techniques, techniques for proving optimality without having to bring in all positive-price columns (and using another column-generation process in a different formulation to do so), and column removal techniques. For the branch-and-price search in the integer program that surrounds the LP, our enhancements include primal heuristics and we also compared branching strategies. Undoubtedly, further parameter tuning and perhaps additional speed improvement techniques could be used to make the algorithm even faster.

Our algorithm also supports several generalizations, as desired by real-world kidney exchanges. These include multiple alternative donors per patient, weighted edges in the market graph (to encode differences in expected life years added based on degrees of compatibility, patient age and weight, etc., as well as the probability of last-minute incompatibility), "angel-triggered chains" (chains of transplants triggered by altruistic donors who do not have patients associated with them, each chain ending with a left-over kidney), and additional issues (such as different scores for saving different altruistic donors or left-over kidneys for future match runs based on blood type, tissue type, and likelihood that the organ would not disappear from the market by the donor getting second thoughts). Because we use an ILP methodology, we can also support a variety of side constraints, which often play an important role in markets in practice [19]. We can also support forcing part of the allocation, for example, "This acutely sick teenager has to get a kidney if possible."

Our work has treated the kidney exchange as a batch problem with full information (at least in the short run, kidney exchanges will most likely continue to run in batch mode every so often). Two important directions for future work are to explicitly address both online and limited-information aspects of the problem.

The online aspect is that donees and donors will be arriving into the system over time, and it may be best to not execute the myopically optimal exchange now, but rather save part of the current market for later matches. In fact, some work has been done on this in certain restricted settings [22, 24].

The limited-information aspect is that even in batch mode, the graph provided as input is not completely correct: a number of donor-donee pairs believed to be compatible turn out to be incompatible when more expensive last-minute tests are performed. Therefore, it would be desirable to perform an optimization with this in mind, such as outputting a low-degree "robust" subgraph to be tested before the final match is produced, or to output a contingency plan in case of failure. We are currently exploring a number of questions along these lines but there is certainly much more to be done.

Acknowledgments

We thank economists Al Roth and Utku Unver, as well as kidney transplant surgeon Michael Rees, for alerting us to the fact that prior technology was inadequate for the clearing problem on a national scale, supplying initial data sets, and discussions on details of the kidney exchange process. We also thank Don Sheehy for bringing to our attention the idea of shoe exchange. This work was supported in part by the National Science Foundation under grants IIS-0427858 and CCF-0514922.

9. REFERENCES

[1] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance.

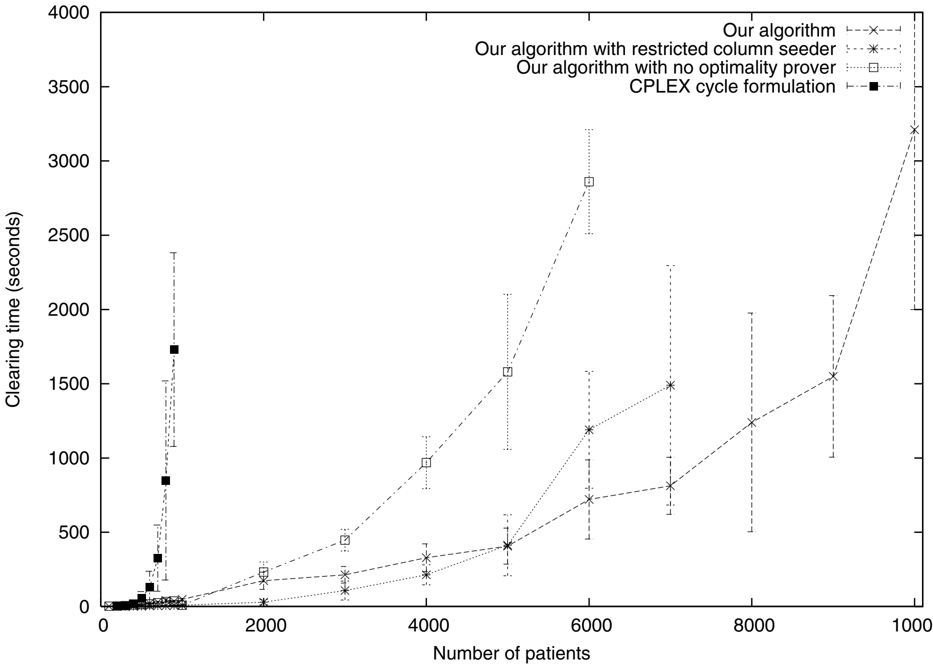

Figure 6: Experimental results: average runtime with standard deviation bars.

Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, May-June 1998.

[2] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32(3):505–536, 1985.

[3] F. L. Delmonico. Exchanging kidneys - advances in living-donor transplantation. *New England Journal of Medicine*, 350:1812–1814, 2004.

[4] J. Edmonds. Path, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. 1990.

[6] S. E. Gentry, D. L. Segev, and R. A. Montgomery. A comparison of populations served by kidney paired donation and list paired donation. *American Journal of Transplantation*, 5(8):1914–1921, August 2005.

[7] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[8] K. Hoffman and M. Padberg. Solving airline crew-scheduling problems by branch-and-cut. *Management Science*, 39:657–682, 1993.

[9] Intervac. *http://intervac-online.com/.*

[10] National odd shoe exchange. *http://www.oddshoe.org/.*

[11] Peerflix. *http://www.peerflix.com.*

[12] Read it swap it. *http://www.readitswapit.co.uk/.*

[13] A. E. Roth, T. Sonmez, and M. U. Unver. Kidney exchange. *Quarterly Journal of Economics*, 119(2):457–488, May 2004.

[14] A. E. Roth, T. Sonmez, and M. U. Unver. A kidney exchange clearinghouse in New England. *American Economic Review*, 95(2):376–380, May 2005.

[15] A. E. Roth, T. Sonmez, and M. U. Unver. Efficient kidney exchange: Coincidence of wants in a market with compatibility-based preferences. *American Economic Review*, forthcoming.

[16] E. Rothberg. Gabow's n^3 maximum-weight matching algorithm: an implementation. *The First DIMACS Implementation Challenge*, 1990.

[17] S. L. Saidman, A. E. Roth, T. Snmez, M. U. Unver, and F. L. Delmonico. Increasing the opportunity of live kidney donation by matching for two and three way exchanges. *Transplantation*, 81(5):773–782, 2006.

[18] T. Sandholm. Optimal winner determination algorithms. In *Combinatorial Auctions*, Cramton, Shoham, and Steinberg, eds. MIT Press, 2006.

[19] T. Sandholm and S. Suri. Side constraints and non-price attributes in markets. In *IJCAI-2001 Workshop on Distributed Constraint Reasoning*, pages 55–61, Seattle, WA, 2001. To appear in *Games and Economic Behavior.*

[20] D. L. Segev, S. E. Gentry, D. S. Warren, B. Reeb, and R. A. Montgomery. Kidney paired donation and optimizing the use of live donor organs. *Journal of the American Medical Association*, 293(15):1883–1890, April 2005.

[21] United Network for Organ Sharing (UNOS). *http://www.unos.org/.*

[22] M. U. Unver. Dynamic kidney exchange. *Working paper.*

[23] United States Renal Data System (USRDS). *http://www.usrds.org/.*

[24] S. A. Zenios. Optimal control of a paired-kidney exchange program. *Management Science*, 48(3):328–342, March 2002.