Manufacturing & Service Operations Management

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Kidney Exchange with Long Chains: An Efficient Pricing Algorithm for Clearing Barter Exchanges with Branch-and-Price

Kristiaan M. Glorie, J. Joris van de Klundert, Albert P. M. Wagelmans

To cite this article:
Kristiaan M. Glorie, J. Joris van de Klundert, Albert P. M. Wagelmans (2014) 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. http://dx.doi.org/10.1287/msom.2014.0496

M&SOM MANUFACTURING & SERVICE OPERATIONS MANAGEMENT
Vol. 16, No. 4, Fall 2014, pp. 498–512
ISSN 1523-4614 (print) | ISSN 1526-5498 (online)

informs

http://dx.doi.org/10.1287/msom.2014.0496

Kidney Exchange with Long Chains: An Efficient Pricing Algorithm for Clearing Barter Exchanges with Branch-and-Price

Kristiaan M. Glorie
Econometric Institute, Erasmus University Rotterdam, 3000 DR Rotterdam, The Netherlands,
glorie@ese.eur.nl

J. Joris van de Klundert
Institute of Health Policy and Management, Erasmus University Rotterdam, 3000 DR Rotterdam, The Netherlands,
vandeklundert@bmg.eur.nl

Albert P. M. Wagelmans
Econometric Institute, Erasmus University Rotterdam, 3000 DR Rotterdam, The Netherlands,
wagelmans@ese.eur.nl

Barter exchange markets are markets in which agents seek to directly trade their goods with each other. Exchanges occur in cycles or in chains in which each agent gives a good to the next agent. Kidney exchange is an important type of barter exchange market that allows incompatible patient–donor pairs to exchange kidneys so the involved patients can receive a transplant. The clearing problem is to find an allocation of donors to patients that is optimal with respect to multiple criteria. To achieve the best possible score on all criteria, long cycles and chains are often needed, particularly when there are many hard-to-match patients. In this paper we show why this may pose difficulties for existing approaches to the optimization of kidney exchanges. We then present a generic iterative branch-and-price algorithm that can deal effectively with multiple criteria, and we show how the pricing problem may be solved in polynomial time for a general class of criteria. Our algorithm is effective even for large, realistic patient–donor pools. Our approach and its effects are demonstrated by using simulations with kidney exchange data from the Netherlands and the United States.

Keywords: kidney exchange; multicriteria optimization; math programming; branch-and-price; simulation
History: Received: June 5, 2013; accepted: June 5, 2014. Published online in *Articles in Advance* September 2, 2014.

1. Introduction

Barter exchange markets are markets in which agents seek to directly trade their goods with each other. The trades in such markets consist of cycles in which each agent gives a good to the next agent in the cycle. Alternatively, the trades may consist of chains that are started by an agent who provides a good without requiring a good in return and that end with an agent who receives a good without providing one. There are numerous examples of barter exchange markets: house exchanges (in which agents seek to simultaneously buy each other's houses; e.g., see http://www.besthouseswap.com), shift exchanges (e.g., between nurses in hospitals), intraorganizational skilled worker exchanges (e.g., between projects or departments), and book exchanges (e.g., see http://www.readitswapit.co.uk). In the present paper we focus specifically on so-called kidney exchanges, but our findings are easily applicable to other types of barter exchange markets.

Kidney exchanges aim to help end-stage renal disease patients with a living and willing but medically incompatible donor to obtain a kidney transplant, which is the preferred treatment for these patients. In particular, kidney exchanges enable patients to exchange donors: if a patient's donor is compatible with some other patient, and the donor of the other patient is compatible with the first patient, the patients can switch donors so that both patients can obtain a transplant (see, e.g., Roth et al. 2004, de Klerk et al. 2011, Glorie et al. 2014). Because of the large potential for increasing the number of transplants, many countries have developed kidney exchange programs. Leading examples are the Netherlands, the United States, the United Kingdom, Australia, and South Korea (Keizer et al. 2005, Manlove and O'Malley 2012, Park et al. 1999, Delmonico et al. 2004).

Kidney exchange need not be limited to two patient–donor pairs but may involve cycles in which the donor of each pair donates, simultaneously, a kidney to the patient of the next pair in the cycle. The simultaneity is required to prevent donors from reneging after their intended recipient has received a transplant from another donor. Because of simultaneity, the length of cycles is limited to the number of logistically feasible simultaneous transplants. Alternative to cycles,

unspecified donors—i.e., donors without a specified recipient—may initiate a chain of transplants in which the last donor is allocated to the deceased donor waiting list or is preserved for a future exchange. Because in a chain no patient–donor pair needs to donate before the patient in the pair has received a kidney, donor reneging in a chain would be less harmful than in a cycle. For this reason, it is sometimes allowed to have one or more nonsimultaneous transplants in a chain, allowing chains to be longer than cycles. Chains are increasingly common and important in clinical practice (e.g., see Ashlagi et al. 2011, Glorie et al. 2013).

Presently, over 30% of living donors are incompatible with their intended recipient (Segev et al. 2005). A patient and donor are incompatible if the donor's blood contains an antigen that is not present in the patient's blood, because the patient will have antibodies against such an antigen. Two cases of incompatibility can be distinguished. The first case, known as blood type incompatibility, revolves around two major antigens: A and B (blood types are denoted as AB, A, B, and O, representing the presence of these antigens). The second case, known as crossmatch incompatibility, revolves around all other antigens against which the patient may have preformed antibodies.

The clearing problem in kidney exchange is to determine an assignment of donors to recipients that is feasible with respect to the medical compatibilities and maximizes one or more criteria such as the number of transplants. In practice, this problem is typically considered in a *static* or off-line context, in which exchanges are conducted at fixed time intervals and the assignment is optimized for the present population, as opposed to a *dynamic* or online context, in which exchanges are conducted continuously and the assignment is optimized with respect to the future evolution of the population. The difficulty of the clearing problem arises from the requirement that all transplants in a cycle must be performed simultaneously and that therefore cycles are limited in length. Whenever this limit is finite and larger than two, the static clearing problem is NP-complete (Abraham et al. 2007).

Abraham et al. (2007) present a mixed-integer programming formulation for the clearing problem with the objective of maximizing a weighted sum of transplants. They solve this formulation by a branch-and-price algorithm (see Barnhart et al. 1998), in which they identify positive price variables by depth-first search. Abraham et al. (2007) show that when each transplant has equal weight in the objective function and when exchanges are limited to cycles or chains involving at most three patients and three donors, this approach works well even when the instance size is large. The main argument for limiting cycles and chains to length 3 is that initially in many pools the maximum possible number of transplants can be achieved by using only cycles and chains up to length 3 (Roth et al. 2007). As we will show, however, when kidney exchange programs continue to evolve, cycles and chains up to length 3 are often not enough to attain the maximum possible number of transplants (see also Ashlagi et al. 2011). Moreover, when heterogeneous objective weights are used or when an objective other than maximizing the sum of transplants is desired, allowing longer cycles and chains may improve the objective function. Unfortunately, with long cycles and chains, depth-first pricing becomes a major bottleneck. In this paper we will show how this problem can be overcome.

In practice, maximizing the (weighted) sum of transplants is not the only relevant objective criterion (see, e.g., de Klerk et al. 2011). Instead of a single weighted objective criterion, several existing kidney exchange programs use a hierarchically ordered set of criteria (e.g., de Klerk et al. 2010, Manlove and O'Malley 2012, Kim et al. 2007). The Dutch national kidney exchange program, in particular, uses the following hierarchical set:

(i) maximize the number of transplants,

(ii) maximize the number of blood type–identical transplants,

(iii) match the patients in priority order based on "match probability" (see Keizer et al. 2005),

(iv) minimize the length of the longest cycle or chain,

(v) maximize the spread over transplant centers per cycle and chain, and

(vi) match the patient with the longest waiting time.

The Dutch criteria are based on European agreements governed in the Convention on Human Rights and Biomedicine (Council of Europe 2002), which determines that the allocation of organs should be both "optimal" and "fair." For this reason, the criteria include factors related to the probability of obtaining a transplant (criteria (ii) and (iii)) and waiting time (criterion (vi)). The exact aim of criterion (ii) is to help to establish a fair allocation across patient blood types by ensuring that patients of disadvantaged blood types, such as blood type O, receive as many transplants as possible (donors of the same blood type will be reserved for them whenever this is viable). Criterion (iii) establishes such fairness in a broader sense by taking into account the total match probability (as defined in Keizer et al. 2005). The priority order *within* criteria (iii) and (vi) is based on the traditional priority mechanisms for allocating deceased donor kidneys. Criteria (iv) and (v) are of a logistical nature. The hierarchy among the criteria implies that every criterion should be optimized subject to the best possible score on previous criteria. For example, the number of blood type–identical transplants (criterion (ii)) should be maximized under the condition that the total number of transplants is maximum (criterion (i)).

Because of the evolution of kidney exchange pools and the ways in which exchange can take place, and because of the advent of large multicenter exchanges and the requirement of multicriteria optimization, there is a need for new techniques for kidney exchange clearing that facilitate long chains. The work presented in this paper makes the following contributions.

1. We develop a generic iterative branch-and-price algorithm for clearing kidney exchanges with a weighted or hierarchically ordered set of objective criteria.

2. We propose a polynomial solution method for the pricing problems as they result for a general class of criteria (which includes all criteria of the Dutch exchange).

3. The presented approach accommodates long, possibly nonsimultaneous, unspecified donor chains at running times that are feasible in practice.

4. The approach allows for optimization for a set of transplantation centers, such as at a national (international) level, while taking individual rationality constraints of the participating transplantation centers into account.

This paper is organized as follows. Section 2 describes the multicriteria kidney exchange problem mathematically. Section 3 details our iterative branch-and-price algorithm, used to solve the multicriteria kidney exchange clearing problem. In particular, §3.2 describes a new branching scheme and §3.3 describes how the pricing problem can be solved in polynomial time for a wide range of criteria. Section 4 discusses the setup of our simulations using actual kidney exchange data. Section 5 presents the computational results. Section 6 concludes the paper.

2. A Kidney Exchange Model

In this section we formalize the concepts used in kidney exchanges and we mathematically define the problem under consideration.

2.1. Problem Definition

Definition 1. A *kidney exchange pool* N consists of two sets, i.e., $N = N_U \cup N_S$, where N_U refers to the set of all unspecified donors and N_S refers to the set of all incompatible specified donor–recipient pairs.

Definition 2. A *kidney exchange graph* $D = (N, A)$ has as its node set a kidney exchange pool N. There is an arc $a_{i,j} = (n_i, n_j) \in A$ from node $n_i \in N$ to node $n_j \in N_S$ if the donor corresponding to node n_i is compatible with the recipient corresponding to node n_j.

Note that in any kidney exchange graph $D = (N, A)$, nodes in N_U, which correspond to donors without recipients, have no incoming arcs. We define a transplant cycle and a transplant chain as follows.

Definition 3. In any given kidney exchange graph $D = (N, A)$, a *length k cycle* is an arc traversal $\langle n_1, \ldots, n_k \rangle$ such that $\{n_1, \ldots, n_k\} \subseteq N_S$ and such that $(n_k, n_1) \in A$ and, for every $1 \le i < k$, $(n_i, n_{i+1}) \in A$.

Definition 4. A length l chain is an arc traversal $\langle n_1, n_2, \ldots, n_l \rangle$ such that $n_1 \in N_U$ and $\{n_2, \ldots, n_l\} \subseteq N_S$, and for every $1 \le i < l$, $(n_i, n_{i+1}) \in A$.

In practice, there exist limits on the number of transplants that can be performed simultaneously within a cycle or chain segment. This implies a natural bound on the maximum cycle and chain length.

Definition 5. For kidney exchange graph $D = (N, A)$ and $K, L \in \mathbb{N}$,

$$C(K,L) := \left\{ c \subseteq N : \begin{array}{l} c \text{ is a cycle in } D \text{ with length} \\ \quad \text{at most } K, \text{ or} \\ c \text{ is a chain in } D \text{ with length} \\ \quad \text{at most } L \end{array} \right\}.$$

Note that, because chains can allow for the requirement of simultaneity to be relaxed, in general $L \ge K$. Typically, chains use one or more nonsimultaneous transplants to link several simultaneous segments of transplants.

Definition 6. Let $D = (N, A)$ be a kidney exchange graph, $K, L \in \mathbb{N}$, and $C(K, L)$ be defined as above. Then, any subset $M = \{c_1, c_2, \ldots, c_{|M|}\} \subseteq C(K, L)$ is called an *exchange* if $c_i \cap c_j = \varnothing$ for all $1 \le i, j \le |M|$, $i \ne j$.

Thus, an exchange is a collection of interdependent kidney transplants that can be feasibly performed together. In the remainder of the paper, we assume a kidney exchange graph $D = (N, A)$, and $K, L \in \mathbb{N}$ are given, and refer to $\mathcal{M}$ as the *exchange set*, i.e., the set of all exchanges M as defined above. Thus, an exchange set $\mathcal{M}$ always implicitly defines a kidney exchange graph $D = (N, A)$, and $K, L \in \mathcal{N}$. Now that we have formally defined exchanges and the exchange set, we proceed by considering the criteria by which exchanges $M \in \mathcal{M}$ are evaluated.

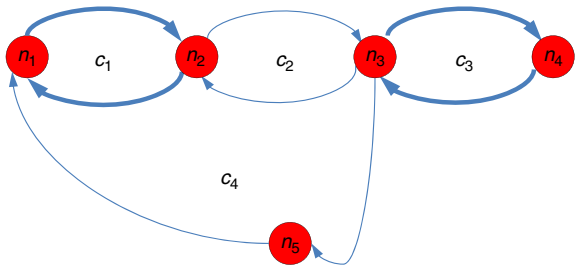
Definition 7. For any given exchange set $\mathcal{M}$, a *criterion* is a function $f: \mathcal{M} \to \mathcal{R}$.

We now arrive at the formal definition of the problem under consideration.

Definition 8. For any given exchange set $\mathcal{M}$ and ordered set of criteria $\mathcal{I} = \{f_1, \ldots, f_{|\mathcal{I}|}\}$, a *hierarchical multicriteria clearing problem* is to find an exchange $M^* \in \mathcal{M}$ such that, for each $i = 1, \ldots, |\mathcal{I}|$, $M^* \in \mathcal{M}_i$, where $\mathcal{M}_i$ is recursively defined as $\mathcal{M}_i := \{M \in \mathcal{M}_{i-1} : f_i(M) \ge f_i(M'), \forall M' \in \mathcal{M}_{i-1}\}$ with $\mathcal{M}_0 := \mathcal{M}$.

Note that the set of criteria used in the Dutch kidney exchange program is an ordered set of kidney exchange criteria that fit the above definition, as would

Figure 1 (Color online) Kidney Exchange Example



be the sole criterion of maximizing the (weighted) number of transplants. The definition also accommodates individual rationality (or participation) constraints for hospitals as is sometimes required in multihospital settings (Glorie et al. 2013).

Figure 1 illustrates an example kidney exchange clearing problem with five donor–recipient pairs, $n_1, \ldots, n_5$. The bound on the length of exchange cycles K is 4. The graph has four feasible cycles: $c_1 = \langle n_1, n_2 \rangle$, $c_2 = \langle n_2, n_3 \rangle$, $c_3 = \langle n_3, n_4 \rangle$, and $c_4 = \langle n_1, n_2, n_3, n_5 \rangle$. There are two maximal exchanges given by $M_1 = \{c_1, c_3\}$ (highlighted) and $M_2 = \{c_4\}$. Although both exchanges have the same number of transplants, in the Dutch system exchange M_1 could be preferable to exchange M_2 by, for example, criterion (iv): the maximum cycle length is 2 instead of 4.

2.2. Integer Programming Formulations

The clearing problem can be formulated as a mixed-integer linear program. For example, for the single criterion of maximizing the number of transplants, the clearing problem can be formulated by using the so-called cycle formulation, which we describe below. Although alternative mixed-integer programming formulations for the kidney exchange problem have also been investigated, these have all been proven to be dominated by the cycle formulation (Constantino et al. 2013).

2.2.1. Cycle Formulation. The cycle formulation, which was first presented in Abraham et al. (2007), uses a binary decision variable x_c for each cycle and chain $c \in C(K, L)$ that is defined as

$$x_c = \begin{cases} 1 & \text{if } c \in M^*, \\ 0 & \text{otherwise.} \end{cases}$$

Setting $x = [x_1, \ldots, x_{|C(K,L)|}]^T$, the integer program is given by

$$P_0: \quad \max \; z_0(x) = \sum_{c \in C(K,L)} |c| \cdot x_c \tag{1}$$

$$\text{s.t.} \sum_{c \in C(K,L):\, n \in c} x_c \le 1 \quad \forall n \in N, \tag{2}$$

$$x_c \in \{0, 1\} \quad \forall c \in C(K, L).$$

Table 1 Average Number of Cycles and Chains over Five Random Kidney Exchange Pools of the Indicated Size

Nodes	Arcs	Cycles ≤ 4	Chains ≤ 4	Chains ≤ 6
10	50	0	1.0e+1	5.40e+1
20	192	8.00e+1	2.21e+2	1.34e+2
50	1,087	2.04e+2	9.87e+3	2.51e+5
100	4,443	5.07e+3	1.84e+5	2.44e+7
200	16,412	1.00e+5	2.23e+6	1.34e+9
500	99,501	8.58e+6	1.02e+8	5.83e+11

Note. Sampled from historical data of the Dutch national kidney exchange program.

In P_0, the objective (1) is to select a collection of cycles and chains that maximizes the number of transplants. Constraints (2) ensure that no patient or donor is contained in more than one selected cycle or chain.

The number of variables in the cycle formulation can be very large (see Table 1, which shows the number of cycles and chains in pools based on actual data from the Dutch kidney exchange program), particularly because the number of chains grows rapidly with the number of nodes. In an exchange pool with 200 nodes, there can be over a billion chains up to length 6; thus the formulation requires at least that many variables. In contrast, Abraham et al. (2007) showed that, when dealing only with cycles up to length 3, this number of variables if often not even attained in pools of 5,000 nodes or more (see Table 2 in Abraham et al. 2007).

2.2.2. Generalized Cycle Formulation. The cycle formulation above can be generalized to allow for many other practically relevant criteria, including each of the criteria (i)–(vi) mentioned in the introduction.

Consider a criterion $f_i \in \mathcal{F}$. As before, let $x = [x_1, \ldots, x_{|C(K,L)|}]^T$ denote the vector of decision variables that indicate whether a cycle/chain $c \in C(K, L)$ is selected. In addition, for $n_i, m_i \in \mathbb{N}$, let y_i denote a $n_i \times 1$ vector of auxiliary variables which are allowed to assume values in some subspace $F_i \subseteq \mathbb{R}^{n_i}$. Then, for $w_i \in \mathbb{R}^{|C(K,L)|}$, $v_i \in \mathbb{R}^{n_i}$, $A_i \in \mathbb{R}^{m_i \times |C(K,L)|}$, $B_i \in \mathbb{R}^{m_i \times n_i}$, and $b_i \in \mathbb{R}^{m_i}$, the generalized cycle formulation is given by the following integer program:

$$P_i: \quad \max \; z_i(x, y_i) = w_i^T x + v_i^T y_i \tag{3}$$

$$\text{s.t.} \; (2),$$

$$A_i x + B_i y_i \le b_i, \tag{4}$$

$$x \in \{0, 1\}^{|C(K,L)|},$$

$$y_i \in F_i.$$

Here, the objective (3) is to maximize $z_i(x)$ with respect to f_i. As before, constraints (2) ensure that no patient or donor is contained in more than one selected cycle or chain. General constraints (4) allow for various relationships between the selected cycles x and the auxiliary variables y_i that are required to model f_i.

Ostensibly, the above formulations can also allow for multiple criteria by including a separate term in the objective function for each criterion under consideration. Each term is then multiplied with the relative weight attached to the criterion it models. As long as the weights are relatively close to each other this approach works well. However, if the criteria are hierarchically ordered, the required scaling of the weights will quickly lead to numerical instability, which renders the program to be infeasible. This is, for example, the case with the six criteria used in the Dutch national program. Therefore, in the next subsection, we present a recursive formulation that models the criteria in the hierarchy without leading to numerical instability.

2.2.3. Recursive Cycle Formulation. In this subsection we present a recursive formulation modeling hierarchical criteria that does not suffer from numerical instability: the *recursive cycle formulation*. The idea is not to capture the hierarchical multicriteria structure in a single integer program but instead recursively define multiple programs $R_1, \ldots, R_{|\mathcal{F}|}$, which are linked together by "objective propagation" constraints.

The first program in the recursion sequence is the generalized cycle formulation of criterion f_1. In the case of the Dutch criteria, we have $R_1 := P_0$, where P_0 is the program we have defined before for the maximization of the number of transplants.

Then, denoting, in addition to the notation introduced above, the optimum value of R_i by z_i^*, the programs R_i, $i = 2, \ldots, |\mathcal{F}|$, are recursively defined as

$$
\begin{aligned}
R_i: \quad \max \quad & z_i(x, y_i) = w_i^T x + v_i^T y_i && (5)\\
\text{s.t.} \quad & (2)\\
& A_j x + B_j y_j \le b_j \quad j = 1, \ldots, i, && (6)\\
& z_j(x, y_j) \ge z_j^* \quad j = 1, \ldots, i-1, && (7)\\
& x \in \{0,1\}^{|C(K,L)|},\\
& y_j \in F_j \quad j = 1, \ldots, i.
\end{aligned}
$$

As in the generalized cycle formulation, the objective (5) is to maximize the single criterion f_i. However, constraints (6) now include all the relationships required for modeling criteria $f_1, \ldots, f_i$. Constraints (7) are the objective propagation constraints, which link the program R_i to the programs $R_1, \ldots, R_{i-1}$, by propagating their corresponding objective function values.

The recursive cycle formulation naturally fits the definition of the hierarchical multicriteria kidney exchange clearing problem. Indeed, constraints (6) and (7) directly describe the sets $\mathcal{M}_i$, $i = 1, \ldots, |\mathcal{F}|$. The exchange corresponding to the solution of program $R_{|\mathcal{F}|}$ is the solution to the hierarchical multicriteria kidney exchange clearing problem.

Table 2 **Iterative Algorithm for Solving the Hierarchical Multicriteria Kidney Exchange Clearing Problem**

Step 0	Initialize $C(K, L)$ and $x = [x_1, \ldots, x_{\lvert C(K,L)\rvert}]^T$
FOR EACH	Criterion $f_i \in \mathcal{F}$ DO
Step i	Solve R_i on D: $z_i^* := \max_{x \in \{0,1\}^{\lvert C(K,L)\rvert}} f_i(x)$ s.t. (2), $f_1(x) \ge z_1^*, \ldots, f_{i-1}(x) \ge z_{i-1}^*$,
END FOR	
Output	$x^* := \arg\max_{x \in \{0,1\}^{\lvert C(K,L)\rvert}} f_i(x)$ s.t. (2), $f_1(x) \ge z_1^*, \ldots, f_{i-1}(x) \ge z_{i-1}^*$, $M^* := \{c \in C(K, L): x_c^* = 1\}$.

3. Iterative Solution Approach

In this section we will develop an iterative branch-and-price algorithm for solving the hierarchical multicriteria kidney exchange problem based on the recursive cycle formulation. The idea is to iteratively solve integer programs corresponding to the criteria in the hierarchy. If a program is solved, its objective function value is propagated to the integer program corresponding to the next criterion by means of an objective propagation constraint. Table 2 gives a schematic overview of this iterative approach, where R_i and $f_i(x) := f_i(\{c \in C(K, L): x_c = 1\})$, respectively, denote the integer program corresponding to criterion f_i and the objective function value of the exchange corresponding to $x = [x_1, \ldots, x_{|C(K,L)|}]^T$ under criterion f_i. Note that the algorithm is also valid when there is no hierarchy between the criteria and the criteria are captured into a single integer program. The algorithm then requires a single iteration.

3.1. Branch-and-Price Methodology

Because the integer programming formulations described in §2 with one variable per cycle and chain grow exponentially in the size of the exchange pool, the (recursive) integer programs $R_1, \ldots, R_{|\mathcal{F}|}$ included in the approach of Table 2 are solved by using branch-and-price methodology. The branch-and-price method starts with a limited subset $C \subseteq C(K, L)$ of cycles and chains and solves the linear program (LP) relaxation of the integer program under consideration using the corresponding restricted variable set. Whenever linear programming duality conditions imply that adding variables may improve the solution value, corresponding cycles and chains are generated and added to C. This process repeats until strong duality conditions are satisfied. By doing this repeatedly for each node in a branch-and-bound tree, an optimal integral solution can be obtained.

Generating columns for any of the LP relaxations to $R_1, \ldots, R_{|\mathcal{F}|}$ corresponds to generating cycles and chains in the kidney exchange graph. Let δ_n denote the dual value of the constraint corresponding to node $n \in N$ in (2). For the LP relaxation of R_i, $i = 1, \ldots, |\mathcal{F}|$,

we have the following reduced cost r_c^i of a cycle or chain $c \in C(K, L)\backslash C$:

$$r_c^i = w_i[c] - \sum_{n \in c} \delta_n - \sum_{j=1}^{i} \sum_{k=1}^{m_i} A_j[k, c] \cdot \mu_{j,k} - \sum_{j=1}^{i-1} w_j[c] \cdot \nu_j, \quad (8)$$

where, as before, δ_n denotes the dual value of the constraint corresponding to node $n \in N$ in (2), $\mu_{j,k}$ denotes the dual value of the kth constraint modeling criterion j in (6), and ν_j denotes the dual value of the jth objective propagation constraint in (7).

To establish LP optimality, we search for cycles with positive reduced cost in the kidney exchange graph (see §3.3 for details on how this can be accomplished). If no such cycle can be found, the LP has been solved to optimality. If the LP solution is fractional, we branch, restricting one or more variables in the values they can assume, and then resolve the LP. At each node of the branching tree, the LP solution provides an upper bound on the restricted problem of that node. An integral lower bound can be obtained by solving the integer program (IP) with the columns generated for the LP. If, at any node, the LP upper bound is no better than the best lower bound, its subtree can be pruned. If the IP lower bound matches the upper bound at the root node, the problem has been solved to optimality.

3.2. Branch-and-Bound

3.2.1. Branching. An important and integral part of any branch-and-price procedure is the branching scheme. In the best branching scheme investigated in Abraham et al. (2007), branching is performed on the cycles and chains in the kidney exchange graph. Whenever the LP solution is fractional, the cycle or chain whose corresponding variable has an LP value closest to 0.5 is selected and two branches are created, one in which the cycle's corresponding variable is set to 0, and one in which it is set to 1. Branches are then explored by using depth-first search. Because there are up to $\sum_{i=2}^{K} |N|^i$ cycles of length K or less in D, the branching tree may have exponential depth. We therefore propose to branch on the arcs, of which there can be only up to N^2. We consider two branching schemes based on the following definition:

Definition 9. An arc $a \in A$ is fractional if

$$x_a := \sum_{c \in C(K,L):\, a \in c} x_c$$

is fractional.

The existence of fractionally selected cycles need not immediately imply that a fractional arc exists. For instance, multiple fractional cycles might overlap, such that $x_a = 1$ for every arc $a \in A$. Fortunately, Theorem 1 establishes that this can never be true for all arcs whenever the LP solution is fractional.

Theorem 1. *There exists a fractional arc if and only if the LP solution is fractional.*

Proof. The first implication is trivial: if $a \in A$ is a fractional arc, then by definition of x_a, there must be at least one $c \in C(K, L)$: $a \in c$ for which x_c is fractional. To prove the other implication, suppose c_1 is a fractionally selected cycle containing arcs $a_1, a_2, \ldots, a_{|c_1|}$. If any arc $a \in c_1$ is not also covered by at least one other fractionally selected cycle, then $X_a = x_c$, and hence a is fractional. Therefore suppose there are one or more other fractional cycles that have at least one arc in common with cycle c_1. Now, let c_2 be such a fractional cycle containing, without loss of generality, arc $a_1 = (n_1, n_2)$ but not arc $a_2 = (n_2, n_3)$, and let $c_3, \ldots, c_m$ be all other fractional cycles containing arc a_1. There are two options: either $\sum_{c=1}^{m} x_c = 1$ or $\sum_{c=1}^{m} x_c < 1$. In the first case, $x_{a_1} = \sum_{c=1}^{m} x_c = 1$ so arc a_1, and hence node n_2, is totally covered, implying that no positively valued cycle $c \in C(K, L)\backslash\{c_1, c_3, \ldots, c_m\}$ can cover arc $a_2 = (n_2, n_3)$, and that therefore $x_{a_2} \in [x_{c_1}, 1 - x_{c_2}]$, making arc a_2 fractional. In the second case, $x_{a_1} = \sum_{c=1}^{m} x_c < 1$ and thus arc a_1 is fractional. This completes the proof. □

In our first branching scheme, we branch on groups of multiple arcs. If the LP solution is fractional, we select the node with the largest number of fractional out-arcs and then divide its out-arcs in two subsets, S_1 and S_2, and create a branch for each subset. In each branch, all the arcs of its corresponding subset are banned. The subsets S_1 and S_2 are determined by adding arcs to S_1 in nondecreasing order of x_a value, until the sum of x_a values of arcs in S_1 is at least 0.5. The remainder of the arcs are added to S_2. Theorem 1 guarantees us that we can always find a node with at least one fractional out-arc.

In our second branching scheme, we branch on only one arc at a time. If the LP solution is fractional, we select the arc with fractional value closest to 0.5. We then create two branches: one in which the arc is banned, and one in which it is enforced. Again, Theorem 1 guarantees that a fractional arc always exists, and, moreover, that when we have branched on all fractional arcs, we have an integer solution.

In order to enforce an arc $a \in A$ in the master problem, we need to add the following constraint:

$$\sum_{c \in C(K,L):\, a \in c} x_c = 1. \quad (9)$$

Adding constraint (9) to the master problem changes the reduced cost of a cycle or chain. In particular,

if $A^* \subseteq A$ is the set of enforced arcs, the reduced cost r_c^i of a cycle or chain $c \in C(K, L)\backslash C$ in problem R_i is now given by

$$r_c^i = w_i[c] - \sum_{n \in c} \delta_n - \sum_{j=1}^{i} \sum_{k=1}^{m_i} A_j[k, c] \cdot \mu_{j,k} - \sum_{j=1}^{i-1} w_j[c] \cdot \nu_j - \sum_{a \in A^*} \mathbf{1}_{a \in c} \xi_a, \quad (10)$$

where, in addition to the previously introduced notation, ξ_a is the dual value of constraint (9) and $\mathbf{1}_{a \in c}$ is an indicator function that is 1 if $a \in c$ and 0 otherwise.

Note that banning an arc in the master problem is trivial because that arc can simply be removed from the graph.

3.2.2. Bounding. In all cases, before branching, integral upper and lower bounds can be derived from the last iteration of the algorithm. For example, in Step 2 of the iterative solution algorithm, the maximum number of blood type–identical transplants cannot be higher than the total number of transplants determined in Step 1, nor can it be lower than the number of blood type–identical transplants in Step 1's solution. These derived bounds are used to prune the irrelevant parts of the branching tree as soon as they violate the bounds.

If the objective is to maximize the number of transplants, an upper bound can be derived by determining in polynomial time the maximum number of transplants when $L = K = \infty$. If there is a low number of highly sensitized patients (i.e., patients who are crossmatch incompatible with many donors), Roth et al. (2007) have shown that this upper bound is tight.[1] As in Abraham et al. (2007), such an upper bound can be determined by finding a maximum weight matching in a bipartite graph with donors on one side and patients on the other. Let us denote this bipartite graph as $G = (U, V, E)$, with U denoting the patients, V denoting the donors, and E denoting the edges. Donors are connected to their own patients with a zero-weight edge and to all other compatible patients with an edge of weight one.

For each edge $e \in E$, let x_e be defined as

$$x_e = \begin{cases} 1 & \text{if } e \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

The maximum weight matching can then be found in polynomial time by solving the following LP:

$$\max \sum_{e \in E} w_e \cdot x_e$$
$$\text{s.t.} \sum_{e=\{u,v\} \in E} x_e = 1, \quad \forall u \in U,$$
$$\text{s.t.} \sum_{e=\{u,v\} \in E} x_e = 1, \quad \forall v \in V,$$
$$x_e \in [0, 1], \quad \forall e \in E.$$

During the branching process the initial bounds may be improved by the LP solutions (which provide an upper bound) or by a primal heuristic for constructing a feasible integer solution (which provides a lower bound). In all branching schemes, we use, as a primal heuristic, the solution to the IP with the columns generated for the LP. If, at any node of the branching tree, the LP upper bound is no better than the best lower bound, that node's subtree can be pruned. If, at any node, the IP lower bound matches the upper bound to optimality.

3.3. Pricing

In Abraham et al. (2007), the pricing problem is solved by traversing the kidney exchange graph D in search of a positive price cycle. In the worst case, this procedure enumerates all cycles in D and therefore (assuming $L \geq K$) is of order $\mathcal{O}(|N|^L)$, which is exponential in the size of the input. In this section we present a polynomial algorithm to solve the pricing problem in $\mathcal{O}(L|N||A|)$.

The algorithm requires that the reduced cost of a cycle can be expressed as a linear function of arc weights. Therefore, we first formulate the following lemma on the reduced cost of a cycle or chain in the recursive cycle formulation.

Lemma 1. *If the objective coefficients $w_j[c]$ and the constraint coefficients $A_j[k, c]$, $j = 1, \ldots, i$, $k = 1, \ldots, m_j$ for each cycle or chain $c \in C(K, L)$ in problem IR_i can be described as linear functions of arc weights, then there exist weights $\pi_a^i \in \mathbb{R}$, for all arcs $a \in A$, such that, for every cycle and chain $c \in C(K, L)$,*

$$r_c^i = \sum_{a \in c} \pi_a^i, \quad (11)$$

i.e., the reduced cost of c can also be described as a linear function of arc weights.

Proof. Let

$$w_i[c] = \sum_{a \in c} \alpha_i \omega_{i,a} \quad \text{and} \quad A_j[k, c] = \sum_{a \in c} \beta_{i,j} \omega'_{j,k,a}$$

for $j = 1, \ldots, i$ and $k = 1, \ldots, m_j$; then by (10),

$$r_c^i = w_i[c] - \sum_{n \in c} \delta_n - \sum_{j=1}^{i} \sum_{k=1}^{m_i} A_j[k, c] \cdot \mu_{j,k} - \sum_{j=1}^{i-1} w_j[c] \cdot \nu_j - \sum_{a \in A^*} \mathbf{1}_{a \in c} \xi_a$$

[1] In their simulations, Roth et al. (2007) use instances in which 10% of the patients are highly sensitized, which in their study implies that these patients are crossmatch incompatible with 90% of the donors.

$$
\begin{aligned}
&= \sum_{a\in c}\alpha_i\omega_{i,a} - \sum_{n\in c}\delta_n - \sum_{j=1}^{i}\sum_{k=1}^{m_i}\sum_{a\in c}\beta_{i,j}\omega'_{j,k,a}\cdot\mu_{j,k}\\
&\quad - \sum_{j=1}^{i-1}\sum_{a\in c}\alpha_j\omega_{j,a}\cdot\nu_j - \sum_{a\in A^*}\mathbf{1}_{a\in c}\xi_a\\
&= \sum_{a=\{n,n'\}\in c}\Bigg(\alpha_i\omega_{i,a} - \delta_{n'} - \sum_{j=1}^{i}\sum_{k=1}^{m_i}(\beta_{i,j}\omega'_{j,k,a})\cdot\mu_{j,k}\\
&\qquad\qquad - \sum_{j=1}^{i-1}(\alpha_j\omega_{j,a})\cdot\nu_j - \mathbf{1}_{a\in A^*}\xi_a\Bigg)\\
&= \sum_{a=\{n,n'\}\in c}\pi_a^i,
\end{aligned}
$$

where

$$
\begin{aligned}
\pi_a^i = \alpha_i\omega_{i,a} - \delta_{n'} - \sum_{j=1}^{i}\sum_{k=1}^{m_i}(\beta_{i,j}\omega'_{j,k,a})\cdot\mu_{j,k}\\
- \sum_{j=1}^{i-1}(\alpha_j\omega_{j,a})\cdot\nu_j - \mathbf{1}_{a\in A^*}\xi_a \qquad (12)
\end{aligned}
$$

with δ_n the dual value of constraint (2) for node n, $\mu_{j,k}$ the dual value of the kth constraint modeling criterion j in (6), ν_j the dual value of the jth objective propagation constraint in (7), and ξ_a the dual value of constraint (9). □

The linear relationship holds between the objective and constraint coefficients and the arcs in D for most criteria used in practice. In particular, all of the Dutch criteria have this property. Also, the constraints required for branching on arcs have this property. Note, however, that the constraints required to branch on cycles (as used by Abraham et al. 2007) do not satisfy this relationship, because they require constraints to enforce the inclusion of a single cycle.

Now, let us define a reversion operator as follows.

Definition 10. For any directed cycle or chain $c = \langle n_1, n_2, \ldots, n_{|c|}\rangle$, the directed cycle (respectively, chain)

$$c^{-1} := \langle n_{|c|}, n_{|c|-1}, \ldots, n_1\rangle$$

is the reverse of c.

The pricing problems can now be solved in polynomial time through the algorithm given in Table 3. The algorithm first constructs the arc set $\tilde{A}\subseteq A$ of arcs that are not banned and then determines for each starting node $n\in N$ a shortest path up to length K or L in $\tilde{D} = (N, \tilde{A})$ (depending on whether node n corresponds to an unspecified donor or not) by using an adapted version of the Bellman–Ford method (Bellman 1958, Ford 1956). For each node $n\in N$ and $k = 0, 1, \ldots, K-1$ ($L-1$ for chains), the algorithm calculates functions $f_k^n\colon N\to\mathbb{R}\cup\{\infty\}$ and $g_k^n\colon N\to N$ that, respectively, provide the weight of the shortest path between n and any other node $n'\in N$ using at most k arcs, and the predecessor of node $n'\in N$ on such a shortest $n-n'$ path.

Table 3 Polynomial Pricing Algorithm

Step 0	Set $w'_a := -\pi_a^i\ \forall\ a\in\tilde{A}$ as in (12), $C^* = \varnothing$
FOR EACH	Node $n\in N$ DO
Step 1	Set $f_0^n(n) := 0$ and, $\forall\ n'\in N\backslash\{n\}$, $f_0^n(n') := \infty$ and $g_0^n(n') := \varnothing$
Step 2	IF $n\in N_S$ THEN set, for $k = 0, \ldots, K-2$, and for all $n'\in N$, $\hat{a} = (n'', n') := \arg\min_{a=(u,n')\in\tilde{A}}\{f_k^n(u) + w'_a\}$, $f_{k+1}^n(n') := \min\{f_k^n(n'), f_k^n(n'') + w'_{\hat{a}}\}$, $g_{k+1}^n(n') := \begin{cases} n'' & \text{if } f_k^n(n'') + w'_{\hat{a}} < f_k^n(n'),\\ g_k^n(n') & \text{otherwise.}\end{cases}$
Step 3	ELSE IF $n\in N_U$ THEN set, for $k = 0, \ldots, L-2$, and for all $n'\in N$, $\hat{a} = (n'', n') := \arg\min_{a=(u,n')\in\tilde{A}}\{f_k^n(u) + w'_a\}$, $f_{k+1}^n(n') := \min\{f_k^n(n'), f_k^n(n'') + w'_{\hat{a}}\}$, $g_{k+1}^n(n') := \begin{cases} n'' & \text{if } f_k^n(n'') + w'_{\hat{a}} < f_k^n(n'),\\ g_k^n(n') & \text{otherwise.}\end{cases}$
END FOR	
Step 4	For $n, n'\in N_S$, if $(n', n)\in\tilde{A}$ and $f_{K-1}^n(n') + w'_{\{n',n\}} < 0$, $C^*\to C^*\cup\{n', g_{K-1}^n(n'), g_{K-2}^n(g_{K-1}^n(n')), \ldots, n\}^{-1}$, and, for $n\in N_U, n'\in N_S$, if $f_L^n(n') < 0$, $C^*\to C^*\cup\{n', g_{L-1}^n(n'), g_{L-2}^n(g_{L-1}^n(n')), \ldots, n\}^{-1}$.

The algorithm consists of four main steps. Before the main steps are executed, Step 0 transforms the arc-specific weights obtained from Lemma (1) such that the pricing problem becomes a minimization problem. Then, for each node $n\in N$, Step 1 initializes the functions f_k^n and g_k^n, Step 2 calculates the function values of f_k^n and g_k^n in case of a cycle (i.e., $n\in N_S$), and Step 3 calculates the function values in case of a chain (i.e., $n\in N_U$). The final step, Step 4, checks whether there are cycles or chains with positive reduced cost and, if there are, reversely constructs them from the function values of g_k^n.

As stated in Theorem 2 below, the algorithm is exact; i.e., it always finds a positive price cycle or chain if one exists. In fact, for each starting node it finds the maximum weight cycle of length at most K (or chain of length at most L). However, it might be the case that a cycle or chain returned by the algorithm contains a subcycle (and hence is not feasible for the master problem). In the case of such a compound cycle or chain, Theorem 2 guarantees us that the subcycle will always have a positive price. We can choose to abort the algorithm as soon as a positive price cycle or chain is found or it can be run to completion, possibly resulting in multiple positive price cycles and chains being identified. (*Nota bene*, if run to completion, the algorithm will output each cycle c^* up to $|c^*|$ times; therefore it may be desirable to filter the generated cycles for duplicates.)

Before providing the theorem, we first introduce the following definition.

Definition 11. For any directed cycle c composed of simple cycles $\sigma_1, \ldots, \sigma_m$ in $D = (N, A)$, and arc weights $\pi_a^i \ \forall \ a \in A$, the maximum simple cycle $S(c)$ is the cycle given by

$$S(c) = \operatorname*{arg\,max}_{\sigma \in \{\sigma_1, \ldots, \sigma_m\}} \left\{ \sum_{a \in \sigma} \pi_a^i \right\}.$$

Theorem 2. *$C^* \neq \varnothing$, and, for all $c^* \in C^*$, $S(c^*) \in C(K, L)$ and $r^i_{S(c^*)} > 0$, if and only if $\exists\, c \in C(K, L)$: $r_c^i > 0$.*

Proof. Analogously to the Bellman–Ford method, we have, for each $n, n' \in N_S$, $k = 0, \ldots, K$, that

$$\begin{aligned} f_k^n(n') &= \min\{\textstyle\sum_{a \in P} w'_a\text{: } P \text{ is an } n - n' \text{ walk traversing at most } k \text{ arcs}\} \\ &= \max\{\textstyle\sum_{a \in P} w_a\text{: } P \text{ is an } n - n' \text{ walk traversing at most } k \text{ arcs}\}. \end{aligned}$$

Then, obviously,

$$\begin{aligned} c^*(n) &:= \{n', g^n(n'), g^n(g^n(n')), \ldots, n\}^{-1} \\ &= \operatorname{arg\,max}\{\textstyle\sum_{a \in P} \pi_a^i\text{: } P \text{ is an } n - n \text{ walk traversing at most } k \text{ arcs}\} \end{aligned} \quad (13)$$

is a possibly compound, maximum weight cycle with length at most K. Let $\sigma_1(n), \ldots, \sigma_m(n)$ be the simple cycles composing $c^*(n)$ (if $c^*(n)$ itself is a simple cycle, $m = 1$ and $\sigma_1(n) = c^*(n)$). By definition, $S(c^*(n)) \in \{\sigma_1, \ldots, \sigma_m\} \subseteq C(K, L)$. Therefore, it remains to prove that $\exists\, n \in N_S$: $c^*(n) \in C^*$ and that, for all $n \in N_S$: $c^*(n) \in C^*$, $\sum_{a \in S(c^*(n))} \pi_a^i > 0$.

To prove the first part, let $c \in C(K, L)$ be a cycle with $\sum_{a \in c} \pi_a^i > 0$ and let $n \in c$. By (13) we then have that $\sum_{a \in c^*(n)} \pi_a^i \geq \sum_{a \in c} \pi_a^i > 0$, and therefore

$$f_{K-1}^n(n') + w_{\{n', n\}} = \sum_{a \in c^*(n)} w'_a = -\sum_{a \in c^*(n)} \pi_a^i < 0,$$

which implies that $c^*(n) \in C^*$ as desired.

To prove the second part, let $n \in N_S$: $c^*(n) \in C^*$. Then

$$\sum_{a \in c^*(n)} \pi_a^i = \sum_{a \in \sigma_1(n)} \pi_a^i + \cdots + \sum_{a \in \sigma_m(n)} \pi_a^i > 0.$$

Because of this, $\exists\, \sigma \in \{\sigma_1(n), \ldots, \sigma_m(n)\}$: $\sum_{a \in \sigma} \pi_a^i > 0$, and, by Definition 11, $\sum_{a \in S(c^*(n))} \pi_a^i > 0$ as desired. The proof for chains is analogous. □

Corollary 1. *Given a kidney exchange graph $D = (N, A)$ and arc weights $\pi_a^i \ \forall\, a \in A$, a positive weight cycle up to length K or chain up to length L, if one exists, can be found in time $\mathcal{O}(\max\{K, L\}|N||A|)$.*

Proof. The proof follows directly from the description of the algorithm in Table 3. □

4. Simulations

We test our algorithm using several realistic simulators. The first is a kidney exchange simulator based on historical data from the Dutch national kidney exchange program. This simulator is described in detail in Glorie et al. (2013). The second is the simulator described in Saidman et al. (2006) (and used in Abraham et al. 2007), which is the most commonly used generator for kidney exchange pools. This second simulator is based on U.S. population data. We use the simulators to generate both static kidney exchange pools (individual pools sampled from the available patient–donor population) as well as dynamic sequences of pools and exchanges (pools that dynamically evolve by simulating arrivals sampled from the patient–donor population and by simulating removals due to exchanges and, for example, patient illness). In this section we will briefly explain the main aspects of the data and simulation procedures. Table 4 gives an overview of the pool composition under the various simulators.

4.1. Static Simulation with Dutch Clinical Data

The data for our first simulator is obtained from the Dutch Transplant Foundation (NTS 2012, 2014) and originates from the empirical registry of the Dutch national kidney exchange program. It includes 438 incompatible patient–donor pairs who participated in Dutch kidney exchanges between October 2003 and January 2011. In addition it contains 109 unspecified donors who were screened at one of the seven Dutch transplant centers during that period. Each patient and donor has a blood type as well as a registration center. Donors also have a record of their antigen types, while patients have a record of the antigen types that are medically unacceptable to them. Patient and donors are marked as blood type or crossmatch (in)compatible based on the data. A static kidney exchange pool is generated at random from the data using sampling with replacement.

Using Table 4, a pool with similar characteristics as the Dutch pool can be easily constructed by generating pairs and unspecified donors by sampling randomly from the categories listed in the table. Each category should then have a probability of being sampled equal to the percentage listed in the table.

4.2. Static Simulation with U.S. Population Data

We also perform simulations with U.S. population data using the simulator described in Saidman et al. (2006).

Table 4 Pool Composition in the Dutch Simulator, the Saidman Simulator, and the Modified Saidman Simulator

	Donor blood type							
	O		A		B		AB	
Dutch simulator								
Patient blood type								
O	% PRA 0–80:	10.6	% PRA 0–80:	24.1	% PRA 0–80:	3.8	% PRA 0–80:	0.6
	% PRA 80–95:	0.9	% PRA 80–95:	1.1	% PRA 80–95:	0.1	% PRA 80–95:	0.0
	% PRA 95–100:	0.7	% PRA 95–100:	1.3	% PRA 95–100:	0.2	% PRA 95–100:	0.0
A	% PRA 0–80:	5.8	% PRA 0–80:	6.8	% PRA 0–80:	5.3	% PRA 0–80:	0.4
	% PRA 80–95:	0.9	% PRA 80–95:	2.0	% PRA 80–95:	0.3	% PRA 80–95:	0.2
	% PRA 95–100:	0.6	% PRA 95–100:	1.4	% PRA 95–100:	0.3	% PRA 95–100:	0.0
B	% PRA 0–80:	2.3	% PRA 0–80:	6.1	% PRA 0–80:	1.0	% PRA 0–80:	0.2
	% PRA 80–95:	0.4	% PRA 80–95:	0.5	% PRA 80–95:	0.0	% PRA 80–95:	0.0
	% PRA 95–100:	0.6	% PRA 95–100:	0.6	% PRA 95–100:	0.2	% PRA 95–100:	0.0
AB	% PRA 0–80:	0.2	% PRA 0–80:	1.0	% PRA 0–80:	0.0	% PRA 0–80:	0.0
	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0
	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0
—	% unspecified	10.3	% unspecified	7.7	% unspecified	0.8	% unspecified	0.8
Saidman simulator								
O	% PRA 0–80:	4.5	% PRA 0–80:	29.1	% PRA 0–80:	12.3	% PRA 0–80:	2.7
	% PRA 80–95:	4.0	% PRA 80–95:	3.2	% PRA 80–95:	1.4	% PRA 80–95:	0.0
	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0
A	% PRA 0–80:	3.2	% PRA 0–80:	2.2	% PRA 0–80:	8.5	% PRA 0–80:	2.2
	% PRA 80–95:	2.7	% PRA 80–95:	2.0	% PRA 80–95:	0.9	% PRA 80–95:	0.2
	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0
B	% PRA 0–80:	1.3	% PRA 0–80:	8.6	% PRA 0–80:	0.4	% PRA 0–80:	1.0
	% PRA 80–95:	1.1	% PRA 80–95:	1.0	% PRA 80–95:	0.4	% PRA 80–95:	0.1
	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0
AB	% PRA 0–80:	0.4	% PRA 0–80:	0.3	% PRA 0–80:	0.1	% PRA 0–80:	0.0
	% PRA 80–95:	0.3	% PRA 80–95:	0.2	% PRA 80–95:	0.1	% PRA 80–95:	0.0
	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0	% PRA 95–100:	0.0
—	% unspecified	2.2	% unspecified	1.5	% unspecified	0.6	% unspecified	0.2
Modified Saidman simulator								
O	% PRA 0–80:	2.4	% PRA 0–80:	16.3	% PRA 0–80:	7.1	% PRA 0–80:	2.6
	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0
	% PRA 95–100:	11.9	% PRA 95–100:	9.1	% PRA 95–100:	3.1	% PRA 95–100:	1.1
A	% PRA 0–80:	1.3	% PRA 0–80:	1.1	% PRA 0–80:	5.6	% PRA 0–80:	1.5
	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0
	% PRA 95–100:	7.8	% PRA 95–100:	5.4	% PRA 95–100:	2.3	% PRA 95–100:	0.7
B	% PRA 0–80:	0.5	% PRA 0–80:	5.7	% PRA 0–80:	0.2	% PRA 0–80:	0.6
	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0
	% PRA 95–100:	3.2	% PRA 95–100:	2.4	% PRA 95–100:	1.0	% PRA 95–100:	0.3
AB	% PRA 0–80:	0.1	% PRA 0–80:	0.1	% PRA 0–80:	0.0	% PRA 0–80:	0.0
	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0	% PRA 80–95:	0.0
	% PRA 95–100:	0.9	% PRA 95–100:	0.6	% PRA 95–100:	0.3	% PRA 95–100:	0.1
—	% unspecified	2.1	% unspecified	1.5	% unspecified	0.6	% unspecified	0.2

Notes. This table gives averages over 10 instances. % PRA refers to the percentage of the pool with which the patient is crossmatch incompatible.

The simulation is based on data from the United Network for Organ Sharing (UNOS) in the United States. The simulator generates patients with a random blood type, sex, and probability of being crossmatch incompatible (this probability is called the percentage panel reactive antibody (PRA) level) with a randomly chosen donor. Each patient is assigned a potential donor with a random blood type and relation to the patient. If the patient and the potential donor are incompatible, they are added to the kidney exchange pool. Blood types and probabilities of crossmatch failure are then used to determine the compatibilities in the pool. Table 5 summarizes the probabilities as described in Saidman et al. (2006). Because the original simulator did not include unspecified donors, we add to each pool a fixed percentage of unspecified donors (generated as above but without assignment to a patient).

Table 5 Probabilities in the Saidman Simulator

Prob. blood type A	0.3373
Prob. blood type B	0.1428
Prob. blood type AB	0.0385
Prob. blood type O	0.4814
Prob. low PRA (5%)	0.7019
Prob. medium PRA (10%)	0.2000
Prob. high PRA (90%)	0.0981
Prob. female	0.409
Prob. spousal donor[a]	0.4897
% unspecified donor[b]	4.5000

[a]Applies to female patients only. Spousal PRA := 1 − 0.75 (1 − PRA)
[b]Original simulator did not have altruistic donors.

4.3. Sparse Pools

Ashlagi and Roth (2012) recently found that the percentage of highly sensitized patients (i.e., patients with a high probability of crossmatch incompatibility with a randomly chosen donor) in practice can be significantly higher than assumed in Saidman et al. (2006). A possible reason for a higher percentage of highly sensitized patients could be the use of more sensitive crossmatching techniques. Ashlagi and Roth (2012) describe an empirical distribution in which half of the patients has a very high PRA—between 95% and 100%—and the other half has a very low PRA—between 0% and 5% (see Table 1 in Ashlagi and Roth 2012). Because of this, a kidney exchange pool can in practice be much sparser than the pools generated by the Saidman simulator. We make some modifications to the Saidman simulator to reflect this phenomenon of sparse pools. Table 6 summarizes the probabilities in the modified simulator.

4.4. Dynamic Simulations

We use the static simulators described above to perform dynamic kidney exchange simulations as described in Glorie et al. (2013). The dynamic simulation procedure consists of repeated simulated arrivals and exchanges.

For the Dutch simulator, we generate a population of size 547 using sampling with replacement and then assign each pair and each unspecified donor in the population a random arrival date. Arrival dates are drawn uniformly, corresponding to a Poisson arrival process. In each exchange round, the optimization algorithm described in §3 implemented with the Dutch hierarchical criteria identifies a match. The last donor of each chain in an exchange round donates to the waiting list (hence, this donor is not available for future exchange rounds). Proposed matches may fail with a probability depending on patient and donor characteristics. If matches fail, this information is incorporated in the compatibility matrix and the optimization algorithm is rerun for the present exchange round. This process is repeated until a feasible match is found. Patients and donors may leave the pool over time due to simulated attrition and reneging. For the precise probabilities, we refer to Glorie et al. (2013).

When using the simulator with U.S. data, we generate a population of size 10,000 and then, for every exchange, we generate a fixed number of arrivals by sampling with replacement from this population. In each exchange round, the optimization algorithm described in §3 implemented with the maximum number of transplants criterion identifies a matching. Match failure is simulated as above. We use this dynamic simulation to study the clearing time of pools in a dynamic state. We do this by considering the clearing time of the 10th exchange round.

Table 6 Probabilities in the Modified Saidman Simulator

Prob. blood type A	0.3373
Prob. blood type B	0.1428
Prob. blood type AB	0.0385
Prob. blood type O	0.4814
Prob. low PRA (2.5%)	0.7000
Prob. high PRA (97.5%)	0.3000
Prob. female	0.4090
Prob. spousal donor[a]	0.4897
% unspecified donor	4.5000

[a]Applies to female patients only. Spousal PRA := 1 − 0.75 (1 − PRA).

5. Computational Results

Our experiments were performed on a Windows 7 64-bit computer with a 3 GHz AMD Athlon II X2 processor and 4 GB of RAM. The iterative branch-and-price algorithm has been implemented in C# and run on the .NET framework, and LPs and IPs are solved by using CPLEX 12.5.

Table 7 displays the running time performance of our algorithm with the single objective of maximizing the number of transplants (all transplants have equal weight) on instances constructed by the simulator with Dutch clinical data described in §4.1. The performance of the different pricing and branching strategies described in §3 is compared on instances of various sizes. The cycle length limit K is set to either 3 (short cycles) or 4 (long cycles), and the chain length limit L is set to either 3 (short chains) or 6 (long chains).

In our comparisons we include the depth-first pricing algorithm with cycle branching as described in Abraham et al. (2007). In this algorithm, the kidney exchange graph is traversed for positive price cycles by exploring nodes in nondecreasing dual value order. Intermittently, the search path is pruned based on the fact that new nodes will have dual value as least as large as the current node.

In all instances the master problem is seeded with a starting collection of 10,000 random cycles and chains

Table 7 Average Performance Characteristics over 10 Randomly Sampled Static Instances from Historical Data from the Dutch National Kidney Exchange Program

Pool size	Total time (s)	LP time (s)	IP time (s)	# nodes proc./ # nodes	Pricing time (s)
Depth-first pricing with cycle branching, $K=3, L=3$					
10	0.61	0.04	0.00	1/1	0.00
20	0.21	0.04	0.00	1/1	0.00
50	1.11	0.26	0.00	7/13	0.00
100	0.78	0.09	0.28	1/1	0.06
200	1.81	0.52	0.47	1/1	0.36
500	54.92	32.59	0.47	28/ 55	12.17
Depth-first pricing with cycle branching, $K=4, L=6$					
10	0.28	0.05	0.00	2/3	0.00
20	0.83	0.17	0.00	5/9	0.00
50	0.42	0.13	0.00	2/3	0.00
100	7.52	0.14	0.27	1/1	6.81
200	730.65	9.30	0.20	16/31	713.17
500	>10,800	—	—	—	—
Polynomial pricing with arc branching, $K=3, L=3$					
10	0.33	0.03	0.00	1/1	0.00
20	0.20	0.03	0.00	1/1	0.00
50	2.97	0.91	0.00	20/39	0.00
100	0.80	0.09	0.25	1/1	0.13
200	1.58	0.17	0.30	1/1	0.64
500	21.98	1.09	0.69	1/1	18.08
Polynomial pricing with arc branching, $K=4, L=6$					
10	0.27	0.06	0.00	2/3	0.00
20	0.94	0.19	0.00	6/11	0.00
50	1.61	0.53	0.00	10/19	0.00
100	0.83	0.13	0.31	1/1	0.08
200	2.78	0.67	0.30	1/1	1.33
500	103.67	21.44	4.53	24/47	35.72
Polynomial pricing with subset arc branching, $K=3, L=3$					
10	0.16	0.05	0.00	1/1	0.00
20	0.16	0.03	0.00	1/1	0.00
50	0.50	0.13	0.00	3/5	0.00
100	0.70	0.08	0.22	1/1	0.11
200	1.59	0.17	0.30	1/1	0.64
500	22.06	1.08	0.67	1/1	18.13
Polynomial pricing with subset arc branching, $K=4, L=6$					
10	0.42	0.09	0.00	3/5	0.00
20	0.91	0.22	0.00	6/11	0.00
50	1.20	0.39	0.00	7/13	0.00
100	0.88	0.14	0.31	1/1	0.08
200	2.78	0.67	0.30	1/1	1.31
500	95.02	15.59	7.64	12/23	35.61

(generated by random walks from a randomly chosen node in the kidney exchange graph until a feasible cycle or chain is found). The collection of cycles and chains is managed such that whenever the problem contains more than 400,000 cycles and chains, the cycles and chains with the lowest reduced cost are deleted (except those that are branched on or have a nonzero LP value). Per pricing iteration up to 100 new cycles and chains is added (except in the depth-first pricing algorithm, where we adhered to the setting of one new cycle or chain per iteration, as advised in Abraham et al. 2007 and which, after tuning, we found to work best for this pricing algorithm).

The first column in Table 7 indicates the pool size. The second column contains the total running time in seconds. The third and fourth columns, respectively. contain the time spent on solving LPs and IPs for the master problem. When branching is applied, the fifth column reports the number of processed nodes in the branch-and-bound tree over the total number of nodes in the tree; the sixth column reports the total time required for solving pricing problems.

As can be seen from the table, our algorithm is able to find optimal solutions in instances with 500 nodes—which contain around 5.83e+11 chains up to length 6 (see Table 1)—within two minutes. In almost all instances the polynomial pricing algorithm performs better than the depth-first pricing algorithm. In fact, using depth-first pricing, the algorithm is not able to solve the larger instances within the imposed time limit of three hours (see the instance with 500 nodes) because the pricing takes too much time. By using polynomial pricing, all instances can be solved fast. Subset arc branching appears to require the least amount of branching decisions of the various branching strategies, although the difference in performance is small. Often the optimal solution is already found in the root of the branch-and-bound tree.

Next, we perform experiments with instances constructed by the simulator with the U.S. population data described in §§4.2–4.4. When we consider only cycles and chains up to length 3, all algorithms perform similarly; therefore we directly proceed and report results for cycles up to length four and chains up to length 6 ($K=4$ and $L=6$). We generate various instances using both static and dynamic simulation with the Saidman simulator and the modified Saidman simulator.

Tables 8 and 9 summarize the average performance characteristics over, respectively, the static and the dynamic instances. The columns in Tables 8 and 9 are similar to the columns in Table 7, except that the percentage of solved instances is reported in the last column, because not all versions of the algorithm are able to solve all the sparse instances.

The findings reported in Tables 8 and 9 are in line with the findings reported in Table 7 for static instances generated with Dutch data. For both the instances generated by the Saidman simulator and the sparser instances generated by the modified Saidman simulator, the polynomial pricing algorithm performs much better than the depth-first algorithm, regardless of whether they are simulated as static or dynamic. In fact, when using depth-first pricing with cycle branching, many of the larger instances cannot be solved within the imposed time limit of three hours (this is the case for 20% of the instances with 500 nodes, and 80% of

Table 8 **Average Performance Characteristics over 10 Randomly Generated Static Instances Generated with U.S. Population Data**

Pool size	Total time (s)	LP time (s)	IP time (s)	# nodes proc./ # nodes	Pricing time (s)	% solved
Saidman simulator, depth-first pricing with cycle branching						
100	3.67	2.16	0.11	1.8/2.6	1.26	100
200	180.26	24.41	0.39	3.0/5.0	154.51	100
500	1,270.82	191.80	0.43	25.0/49.0	1,067.03	80
1,000	4,109.97	556.64	1.81	58.5/116.0	3,383.05	40
Modified Saidman simulator, depth-first pricing with cycle branching						
100	2.14	1.59	0.07	1.0/1.0	0.40	100
200	121.01	31.76	0.30	1.0/1.0	87.98	100
500	2,948.91	311.19	1.18	2.4/3.8	2,634.55	80
1,000	1,760.15	120.33	15.36	11.0/6.0	416.34	20
Saidman simulator, polynomial pricing with arc branching						
100	1.40	0.84	0.15	5.6/10.2	0.09	100
200	1.58	0.53	0.53	1.0/1.0	0.36	100
500	9.66	3.43	1.29	3.6/6.2	3.46	100
1,000	102.36	23.78	13.94	10.8/20.5	52.07	100
Modified Saidman simulator, polynomial pricing with arc branching						
100	0.87	0.35	0.11	1.0/1.0	0.31	100
200	92.08	34.36	0.87	1.0/1.0	56.43	100
500	740.31	131.38	1.46	1.0/1.0	606.38	100
1,000	1,111.98	705.45	29.99	301.5/602	76.28	100
Saidman simulator, polynomial pricing with subset arc branching						
100	0.93	0.48	0.15	3.4/5.8	0.09	100
200	1.59	0.52	0.56	1.0/1.0	0.36	100
500	8.50	3.04	1.26	2.8/4.6	3.32	100
1,000	180.55	43.06	21.20	30.0/59.0	79.07	100
Modified Saidman simulator, polynomial pricing with subset arc branching						
100	0.90	0.36	0.12	1.0/1.0	0.31	100
200	33.92	9.94	0.47	1.0/1.0	23.17	100
500	24.83	12.18	1.84	1.0/1.0	10.21	100
1,000	1,717.42	857.50	16.14	363.0/725.0	203.52	100

Note. Cycle limit is 4, chain limit is 6.

Table 9 **Average Performance Characteristics over 10 Randomly Generated Dynamic Instances Generated with U.S. Population Data**

Pool size	Total time (s)	LP time (s)	IP time (s)	# nodes proc./ # nodes	Pricing time (s)	% solved
Saidman simulator, depth-first pricing with cycle branching						
100	52.57	1.82	0.25	4.8/8.6	49.32	100
200	117.55	3.97	0.58	4.4/7.8	110.70	100
500	6,644.83	20.40	0.77	1.0/1.0	6,609.75	50
Modified Saidman simulator, depth-first pricing with cycle branching						
100	1.37	0.76	0.04	4.0/7.0	0.06	100
200	790.68	3.45	0.56	1.0/1.0	785.65	80
500	2,449.93	187.54	2.18	93.0/47.0	2,258.44	80
Saidman simulator, polynomial pricing with arc branching						
100	4.98	0.65	0.24	3.8/6.6	3.41	100
200	9.95	0.68	0.38	1.0/1.0	7.68	100
500	1,731.53	18.39	2.31	19.3/37.7	1,685.6	100
Modified Saidman simulator, polynomial pricing with arc branching						
100	1.24	0.33	0.14	2.6/4.2	0.35	100
200	13.78	0.82	0.64	1.0/1.0	11.33	100
500	54.47	3.62	1.49	1.0/1.0	40.98	100

Note. Cycle limit is 4, chain limit is 6.

Table 10 **Percentage of Instances Showing Improvement at Step *i* in the *i*th Objective Criterion of the Dutch Hierarchical Criteria**

Criterion	i	ii	iii	iv	v	vi
% of instances showing improvement at step *i*	99.3	24.5	52.1	0.3	0.3	34.5

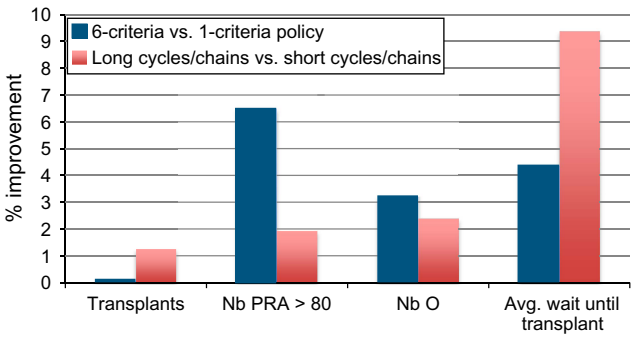
Note. Cycle limit is 4, chain limit is 6.

the instances with 1,000 nodes), whereas all of these instance can be solved within a reasonable time when using polynomial pricing. As before, many instances can be solved in the root of the branching tree, but, in total, branching is now required for more instances. When branching is required, arc branching appears to be slightly more effective than subset arc branching because it leads to fewer branches on average (see the 1,000 node instances).

As Ashlagi et al. (2011) have shown, allowing longer cycles and chains is important to increasing the number of transplants for the hardest-to-match patients. This is especially true if the pool is sparse. Allowing long cycles and chains may also be important for other objectives than the number of transplants, whether they are captured through a hierarchical objective function or through a single weighted objective function. By allowing longer cycles and chains, these objectives may be improved.

Table 10 and Figure 2 display the long-term effects of using the multiple hierarchical criteria used in the Netherlands. In particular, Table 10 shows the percentage of instances showing improvement in the *i*th objective criterion, and Figure 2 shows the relative difference in the total number of transplants, the average wait time, the number of highly sensitized patients (patients with PRA > 80) transplanted, and O patients transplanted versus (a) a policy using only the single criterion of maximizing the number of transplants and (b) a policy using only short cycles and chains.

Figure 2 **(Color online) Long-Term Effects Using Dutch Hierarchical Criteria**



Notes. Cycle limit is 4, chain limit is 6. Dark bar: comparing the 6-criteria policy versus the 1-criteria "maximum transplants" policy. Light bar: comparing the 6-criteria policy versus the 6-criteria policy with cycles and chains limited to 3.

As we can see from Table 10, additional criteria often make a difference, even if they have a low hierarchical ranking. For instance, in our simulations, Dutch criterion (vi) constitutes an improvement in 34.5% of the instances. Although the long-run improvement in the total number of transplants versus a single criterion policy aimed at just maximizing the number of transplants is small (see Figure 2), the improvement in terms of highly sensitized patients transplanted is significant. This (normally disadvantaged) group can receive up to 4.5% more transplants when using the Dutch allocation criteria. Also O type patients, who are another disadvantaged group, benefit by over 3%. Furthermore, Figure 2 shows how long cycles and chains may lead to a 9% improvement in average waiting time.

6. Conclusions

In this paper we have shown how to clear large multicriteria kidney exchanges with long chains using a general and scalable exact algorithm. This is particularly important because, over the last years, kidney exchange has quickly increased as a modality for transplanting for end-stage renal disease patients with an incompatible living donor, and long exchange chains have turned out to be increasingly important to help the most disadvantaged patients. Most kidney exchange programs not only seek to maximize directly the number of transplants but also to optimize other objectives, such as fairness prescribed in international treaties (e.g., Council of Europe 2002). For this reason, many programs use a set of multiple, often hierarchical, optimization criteria. Using our algorithm, we can effectively deal with such criteria, even in large and sparse exchange pools that now begin to arise in practice.

To maximize the benefits from exchange, it should be coordinated at a national (international) level. However, participation barriers for transplant centers may prevent such nationally coordinated kidney exchange from being established. To make such coordination possible, participation constraints must be included. Our algorithm can also deal with such constraints by including them as a hierarchical objective at the highest level.

Mathematically, the algorithm consists of an iterative branch-and-price procedure. By using a general but effective class of integer programming formulations, we are able to optimally clear exchange pools with billions of cycles and chains within minutes. The key part of our algorithm is a polynomial pricing procedure for this class of formulations in combination with a branching strategy that branches on arcs or on subsets of arcs. These elements allow us to efficiently deal with long chains that would not be possible with depth-first pricing techniques suggested in previous research.

We hope that our algorithm may serve as a reference solution framework for other researchers, so that solution methods and data can be shared, to the benefit of the patients suffering from end-stage renal disease across the globe. Our approach is also easily applicable to other types of barter exchange markets besides kidney exchange and can therefore have implications for a broader class of allocation issues, particularly where, for various reasons, market prices do not exist.

References

Abraham D, Blum A, Sandholm T (2007) Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. Parkes D, Tennenholtz M, eds. *Proc. 8th ACM Conf. Electronic Commerce* (ACM, New York).

Ashlagi I, Roth AE (2012) New challenges in multi-hospital kidney exchange. *Amer. Econom. Rev., Papers Proc.* 102(3):354–359.

Ashlagi I, Gilchrist DS, Roth AE, Rees MA (2011) Nonsimultaneous chains and dominos in kidney paired donation—Revisited. *Amer. J. Transplantation* 11(5):984–994.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* 46(3):316–329.

Bellman R (1958) On a routing problem. *Quart. Appl. Math.* 16:87–90.

Constantino M, Klimentova X, Viana A, Rais A (2013) New insights on integer-programming models for the kidney exchange problem. *Eur. J. Oper. Res.* 231(1):57–68.

Council of Europe (2002) Additional protocol to the convention on human rights and biomedicine concerning transplantation of organs and tissues of human origin. *Eur. Treaty Series* (No. 186).

de Klerk M, van Gestel JK, Haase-Kromwijk B, Claas F, Weimar W (2011) Eight years of outcomes of the dutch living donor kidney exchange program. *Clinical Transplants* 2011:287–290.

de Klerk M, Van der Deijl WM, Witvliet MD, Haase-Kromwijk BJJM, Claas FHJ, Weimar W (2010) The optimal chain length for kidney paired exchanges: An analysis of the dutch program. *Transplant Int.* 23(11):1120–1125.

Delmonico F, Morrissey P, Lipkowitz G, Stoff J, Himmelfarb J, Harmon W, Pavlakis M, et al.(2004) Donor kidney exchanges. *Amer. J. Transplantation* 4(10):1628–1634.

Ford LR (1956) Network flow theory. Paper P-923, RAND Corporation, Santa Monica, CA.

Glorie K, Haase-Kromwijk B, van de Klundert J, Wagelmans A, Weimar W (2014) Allocation and matching in kidney exchange programs. *Transplant Internat.* 27(4):333–343.

Glorie KM, de Klerk M, Wagelmans APM, van de Klundert JJ, Zuidema WC, Claas FHJ, Weimar W (2013) Coordinating unspecified living kidney donation and transplantation across the blood-type barrier in kidney exchange. *Transplantation* 96(9): 814–820.

Keizer KM, de Klerk M, Haase-Kromwijk BJJM, Weimar W (2005) The Dutch algorithm for allocation in living donor kidney exchange. *Transplantation Proc.* 37(2):589–591.

Kim BS, Kim YS, Kim SI, Kim MS, Lee HY, Kim YL, Kim CD, et al. (2007) Outcome of multipair donor kidney exchange by a web-based algorithm. *J. Amer. Soc. Nephrology* 18(3):1000–1006.

Manlove D, O'Malley G (2012) Paired and altruistic kidney donation in the UK: Algorithms and experimentation. Klasing R, ed. *Experimental Algorithms: Proc. SEA 2012* (Springer, Berlin), 271–282.

NTS (Nederlandse Transplantatie Stichting) (2012) Annual report 2012. http://www.transplantatiestichting.nl/sites/default/files/product/downloads/nts_jaarverslag_2012_web.pdf.

NTS (Nederlandse Transplantatie Stichting) (2014) Website, http://www.transplantatiestichting.nl.

Park K, Moon JI, Kim SI, Kim YS (1999) Exchange donor program in kidney transplantation. *Transplantation* 67(2):336–338.

Roth AE, Sönmez T, Ünver MU (2004) Kidney exchange. *Quart. J. Econom.* 119(2):457–488.

Roth AE, Sönmez T, Ünver MU (2007) Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *Amer. Econom. Rev.* 97(3):828–851.

Saidman S, Roth A, Sönmez T, Ünver U, Delmonico F (2006) Increasing the opportunity of live kidney donation by matching for two- and three-way exchanges. *Transplantation* 81(5): 773–782.

Segev DL, Gentry SE, Warren DS, Reeb B, Montgomery RA (2005) Kidney paired donation and optimizing the use of live donor organs. *J. Amer. Medical Assoc.* 293(15):1883–1890.