New Integer Programming Formulations for the Stable Exchange Problem*

Virginia Costa[1], Xenia Klimentova[1], Péter Biró[2], Ana Viana[1,3], and João Pedro Pedroso[1,4]

[1] INESC TEC, Porto, Portugal
[2] Institute of Economics, Hungarian Academy of Sciences, Budapest, Hungary
[3] ISEP - School of Engineering, Polytechnic of Porto, Porto, Portugal
[4] Faculdade de Ciências, Universidade do Porto, Porto, Portugal
xenia.klimentova@inesctec.pt

Abstract. In the stable exchange problem the agents are endowed with a single good, e.g. a house or a kidney donor, and they have preferences over the others' endowments. The problem is to find an exchange of goods such that no group of agents can block the solution in an exchange cycle. An exchange is called stable if there is no blocking cycle where all the agents involved strictly prefer the new solution. An exchange is strongly stable if no weakly blocking cycle exists, where at least one agent improves and neither of them gets a worse allocation. When the lengths of the exchange cycles is not limited then a stable solution always exists and can be found efficiently by Gale's Top Trading Cycle algorithm. However, when the length of the exchange cycles is limited then a (strongly) stable solution may not exist and the problem of deciding the existence is NP-hard. This setting is particularly relevant in kidney exchange programs, where the length of exchange cycles is limited due to the simultaneity of the transplantations, e.g. the maximum length of the cycles is 3 in the UK and 4 in the Netherlands. In this work we develop several integer programming formulations to solve the (strongly) stable exchange problem, which is a novel approach for this solution concept. We compare the effectiveness of these models by conducting computational experiments on generated kidney exchange data.

Keywords: stable matching · integer programming · k-way exchange.

* This work is financed by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme, and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project "mKEP - Models and optimisation algorithms for multicountry kidney exchange programs" (POCI-01-0145-FEDER-016677), by FCT project SFRH/BPD/101134/2014 and by COST Action CA15210 ENCKEP, supported by COST (European Cooperation in Science and Technology) – http://www.cost.eu/. Biró is supported by the Hungarian Academy of Sciences under its Momentum Programme (LP2016-3/2018) and Cooperation of Excellences Grant (KEP-6/2018), and by the Hungarian Scientific Research Fund – OTKA (no. K129086).

1 Introduction

Barter exchange markets – such as kidney exchange programs – can be represented as directed graphs where agents are vertices and arcs indicate exchange opportunities. A solution consists of a set of disjoint cycles. In this paper we consider the case where agents have preferences, represented by ranks on outgoing arcs. An exchange that contains no cycle with length more than k is a *k-way exchange*. A *k-way stable exchange* is a k-way exchange such that there is no cycle where all the vertices would be better off, according to their preferences, than in the current solution. When strict preference in the blocking cycle is required only for one vertex then we speak about *strongly stable exchanges*. The problem of deciding existence is NP-hard for both problems [2, 5]. In this work, we present three novel integer programming formulations for these problems, which is a novel approach in the literature. Preliminary computational results highlight the efficiency of one formulation over the others.

1.1 Notation and definitions

Consider a digraph $G = (V, A)$, where V is the set of vertices and A is the set of arcs. Define also the *preference list* of $i \in V$ as the set $\delta(i) = \{j \mid (i, j) \in A\} \subseteq V$ where there is a strict preference order on its elements. Each $j \in \delta(i)$ is ranked with value $r \in \{1, \ldots, |\delta(i)|\}$. For $j, j' \in \delta(i)$ ranked with r, r', respectively, we say that vertex i prefers j to j', and denote by $j <_i j'$, if $r' > r$.

Within this context, a *matching* $\mathcal{M} \subset A$ is a set of pairs (i, j) where $i \in V$ and $j \in \delta(i)$. In addition, a vertex always prefers to be matched to any of the elements in its preference list, rather than be unmatched. A vertex i is *unmatched* if there is no vertex j such that $(i, j) \in \mathcal{M}$. Let $\mathcal{C}$ be a set of cycles in G of length at most k. We denote by $V(c)$ and $A(c)$ the set of vertices and arcs, respectively, that are involved in a cycle $c \in \mathcal{C}$. We say that $c \in \mathcal{M}$ if, and only if, $A(c) \subseteq \mathcal{M}$. Let $|c|$ denote the length of cycle c, i.e., $|c| = |V(c)| = |A(c)|$. Let $\mathcal{C}(i) \subseteq \mathcal{C}$ be the set of cycles that contain vertex i. We say that vertex i *prefers* cycle $c \in \mathcal{C}(i)$ over cycle $c' \in \mathcal{C}(i)$, and denote by $c \prec_i c'$, if for $(i, j) \in A(c)$ and $(i, j') \in A(c')$, $j <_i j'$.Vertex i is *indifferent* between cycles c and c' if there exists a vertex j such that $(i, j) \in A(c) \cap A(c')$, i.e., (i, j) is both in c and c'. Finally, i *weakly prefers* c to c' if it prefers c to c' or it is indifferent between them. We define the *Stable (Strongly Stable) Exchange Problem* as the problem of finding in G a vertex-disjoint packing of directed cycles with length at most k that corresponds to a *stable (strongly stable) matching*. The definitions of stable and strongly stable matchings [2, 5] are provided below.

Definition 1. A *blocking cycle* $c \notin \mathcal{M}$ is a cycle such that every vertex i in $V(c)$ is either unmatched in $\mathcal{M}$ or prefers c to c', where $c' \in \mathcal{C}(i) \cap \mathcal{M}$. A matching $\mathcal{M}$ is called *stable* if there is no blocking cycle $c \notin \mathcal{M}$.

Definition 2. A *weakly blocking cycle* is a cycle $c \notin \mathcal{M}$ such that for every $i \in V(c)$, i is either unmatched in $\mathcal{M}$ or weakly prefers c to c', where $c' \in \mathcal{C}(i) \cap \mathcal{M}$, with strict preference for at least one vertex. A matching $\mathcal{M}$ is called *strongly stable* if there is no weakly blocking cycle $c \notin \mathcal{M}$.

2 Integer Programming Formulations

The Stable Exchange Problem can be seen as a optimization problem. In what follows we propose three integer programming formulations for it.

2.1 Stable Cycle Formulation

For each pair (i, c), $i \in V$, $c \in \mathcal{C}(i)$ we define two sets of cycles: $B_{i,c} = \{\bar{c} \in \mathcal{C}(i), \bar{c} \neq c : \bar{c} \preceq_i c\}$, which is the set of cycles that are different from c and better or equally preferable for i than c, and $S_{i,c} = \{\bar{c} \in \mathcal{C}(i) : \bar{c} \prec_i c\}$, which is the set of cycles that are strictly better for vertex i than cycle c. Consider vector $x = (x_1, \ldots, x_{|\mathcal{C}|})$ of variables such that $x_c = 1$ if all arcs in $A(c)$ are in $\mathcal{M}$, 0 otherwise. The following set of constraints will define a *stable* matching $\mathcal{M}$:

$$\sum_{c:i \in V(c)} x_c \leq 1 \qquad \forall i \in V \tag{1}$$

$$x_c + \sum_{s \in \bigcup_{i \in V(c)} B(i,c)} x_s \geq 1, \qquad \forall c \in \mathcal{C}, \tag{2}$$

$$x_c \in \{0, 1\} \qquad \forall c \in \mathcal{C}, \tag{3}$$

Constraints (1) guarantee that $\mathcal{M}$ is a set of disjoint cycles. Constraints (2) mean that either $c \in \mathcal{M}$, or, for some vertex $i \in V(c)$, there exists a cycle $c' \in B(i, c)$ such that $i \in V(c')$ and $c' \preceq_i c$. For a *strongly stable* matching, constraints (2) are replaced by:

$$x_c + \sum_{s \in \bigcup_{i \in V(c)} S(i,c)} x_s \geq 1, \forall c \in \mathcal{C}, \tag{4}$$

Constraints (4) guarantee that either c is in the matching, or otherwise one of its vertices is matched in a cycle strictly better than c.

The objective function considered maximizes the maximum number of cycles in $\mathcal{M}$ and is described as follows:

$$F(x) = \sum_{c:c \in \mathcal{C}} |c| \cdot x_c. \tag{5}$$

2.2 Stable Edge Formulation

To define the stable edge formulation, we depart from the edge formulation in [1], where $y_{i,j}$ is a binary variable denoting whether arc (i, j) is included in the solution, or not. A feasible solution with cycles of length at most k can be formalized as follows:

$$\sum_{j:(j,i) \in A} y_{j,i} - \sum_{j:(i,j) \in A} y_{i,j} = 0 \qquad \forall i \in V \tag{6}$$

$$\sum_{j:(i,j) \in A} y_{i,j} \leq 1 \qquad \forall i \in V \tag{7}$$

$$\sum_{(i,j) \in A(p)} y_{i,j} \leq k - 1 \qquad \forall p \in \mathcal{P}. \tag{8}$$

where $\mathcal{P}$ is a set of all non-cyclic paths p in G with k arcs, and $A(p)$ is the set of arcs of G in p. Note that sub-cycles with more than k arcs are removed from the set of feasible solutions by constraints (8). To achieve stability, according to definition 1, we introduce the following set of constraints:

$$\sum_{(i,j)\in A(c)} \left[y_{i,j} + \sum_{r:r<_i j} y_{i,r} \right] \geq 1, \qquad \forall c \in \mathcal{C}. \tag{9}$$

Strong stability can be achieved by replacing inequalities (9) by the following set of constraints:

$$|c| \cdot \left[\sum_{(i,j)\in A(c)} \sum_{r:r<_i j} y_{i,r} \right] + \sum_{(i,j)\in A(c)} y_{i,j} \geq |c|, \qquad \forall c \in \mathcal{C}. \tag{10}$$

The inequality is satisfied for cycle c by the first term if there is an agent strictly preferring her matching in the solution to what she would receive in c. The second term ensures that a cycle already in the solution cannot be a blocking cycle.

Since the sum of all binary variables $y_{i,j}$ is equal to $|\mathcal{M}|$, the objective function can be written as:

$$F(y) = \sum_{(i,j)\in A} y_{i,j}. \tag{11}$$

Note that, if the feasibility constraints from (6) to (8) and the stability constraints (9) or strong stability constrains (10) are satisfied, we obtain the maximum number of cycles in $\mathcal{M}$ by maximizing $F(y)$ in (11).

2.3 Stable Cycle-Edge Formulation

In the stable (strongly stable) cycle-edge formulation, we use the integer variables of the two formulations above in a consistent way. That is, for every cycle $c \in \mathcal{C}$, we require that $x_c = 1$ if and only if $y_{i,j} = 1$ for every $(i,j) \in A(c)$. This correspondence can be achieved by the basic feasibility cycle-constraints (1) and edge-constraints (6), and by adding the following three sets of inequalities:

$$|c| \cdot x_c \leq \sum_{(i,j)\in A(c)} y_{i,j}, \forall c \in \mathcal{C}, \tag{12}$$

$$\sum_{(i,j)\in A(c)} y_{i,j} - |c| + 1 \leq x_c, \forall c \in \mathcal{C}, \tag{13}$$

$$\sum_{j:(i,j)\in A} y_{i,j} \leq \sum_{c:i\in V(c)} x_c, \forall i \in V \tag{14}$$

Stability and strong stability are assured by constraints (9) and (10), respectively. Both (5) and (11) can be used as objective functions.

Table 1. Stable exchange problem formulations: stable cycle formulation (SCF), stable edge formulation (SEF) and stable cycle-edge formulation (SCEF).

Instances					Formulations														
					SCF					SEF					SCEF				
n	|A|	$\|\mathcal{C}\|$	$\|\mathcal{P}\|$	k	Rows	Columns	Non-zeros	Loading time (s)	Solver time (s)	Rows	Columns	Non-zeros	Loading time (s)	Solver time (s)	Rows	Columns	Non-zeros	Loading time (s)	Solver time (s)
30	165	37	3,584	3	57	37	550	0.00	0.00	3,681	165	11,617	0.0274	0.03	189	202	1,295	0.00	0.00
		153	17,477	4	177	153	14,016	0.01	0.02	17,690	165	72,772	0.1509	0.15	541	318	5,724	0.01	0.01
		269	73,636	5	294	269	51,515	0.04	0.07	73,965	165	369,782	0.7135	0.69	890	434	10,913	0.01	0.02
50	617	584	82,009	3	632	584	88,616	0.05	0.14	82,693	617	265,292	0.60	1.16	1,900	1,201	27,089	0.03	0.06
		5,236	951,322	4	5,284	5,236	10,188,648	5.80	126.70	956,658	617	4,028,087	7.25	49.64	15,856	5,853	317,803	0.23	1.56
		38,591	11,004,062	5	38,639	38,591	794,566,412	525.10	**n.m.**	11,042,753	617	56,920,039	89.02	926.14	115,921	39,208	2,852,329	1.81	24.27
70	1135	611	174,480	3	662	611	80,809	0.04	0.19	175,231	1,135	548,667	1.31	5.16	2,019	1,746	33,321	0.05	0.13
		6,700	2,135,151	4	6,753	6,700	14,035,100	7.81	150.48	2,141,991	1,135	8,876,487	15.88	191.83	20,288	7,835	458,502	0.36	5.70
		48,762	26,135,720	5	48,815	48,762	1,092,827,519	721.96	**n.m.**	26,184,622	1,135	133,510,623	229.04	2061.98	146,474	49,897	4,081,818	2.82	60.31
90	2063	3,214	884,802	3	3,298	3,214	1,846,921	1.04	13.92	888,196	2,063	2,829,076	5.91	133.75	9,904	5,277	218,618	0.21	0.94
		49,386	18,407,917	4	49,471	49,386	687,653,906	406.07	**n.m.**	18,457,483	2,063	77,174,437	141.18	1414.87	148,421	51,449	4,440,627	3.46	51.14
		710,726	382,999,769	5	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	2,132,441	712,789	78,912,742	52.86	1061.07

Table 2. Strongly stable exchange problem formulations: strongly stable cycle formulation (SSCF), strongly stable edge formulation (SSEF) and strongly stable cycle-edge formulation (SSCEF).

Instances					Formulations														
					SSCF					SSEF					SSCEF				
n	|A|	$\|\mathcal{C}\|$	$\|\mathcal{P}\|$	k	Rows	Columns	Non-zeros	Loading time (s)	Solver time (s)	Rows	Columns	Non-zeros	Loading time (s)	Solver time (s)	Rows	Columns	Non-zeros	Loading time (s)	Solver time (s)
30	165	37	3,584	3	57	37	490	0.00	0.00	3,681	165	11,617	0.02	0.00	189	202	1,295	0.00	0.00
		153	17,477	4	177	153	11,181	0.01	0.00	17,690	165	72,772	0.09	0.02	541	318	5,724	0.01	0.00
		269	73,636	5	294	269	40,684	0.02	0.01	73,965	165	369,782	0.41	0.10	890	434	10,913	0.01	0.00
50	617	584	82,009	3	632	584	81,497	0.05	0.02	82,693	617	265,292	0.40	0.09	1,900	1,201	27,089	0.03	0.01
		5,236	951,322	4	5,284	5,236	9,293,007	5.20	3.60	956,658	617	4,028,087	4.58	1.87	15,856	5,853	317,803	0.23	0.12
		38,591	11,004,062	5	38,639	38,591	725,505,674	437.41	385.29	11,042,753	617	56,920,039	56.87	28.57	115,921	39,208	2,852,329	1.85	1.38
70	1135	611	174,480	3	662	611	74,205	0.04	0.02	175,231	1,135	548,667	0.84	0.23	2,019	1,746	33,321	0.05	0.02
		6,700	2,135,151	4	6,753	6,700	12,928,785	7.01	5.09	2,141,991	1,135	8,876,487	10.49	4.45	20,288	7,835	458,502	0.37	0.36
		48,762	26,135,720	5	48,815	48,762	1,001,482,550	610.08	**n.m.**	26,184,622	1,135	133,510,623	134.24	67.56	146,474	49,897	4,081,818	2.81	3.56
90	2063	3,214	884,802	3	3,298	3,214	1,765,893	0.96	0.61	888,196	2,063	2,829,076	3.95	1.61	9,904	5,277	218,618	0.23	0.25
		49,386	18,407,917	4	49,471	49,386	659,470,242	389.51	341.85	18,457,483	2,063	77,174,437	91.75	44.11	148,421	51,449	4,440,627	3.35	5.97
		710,726	382,999,769	5	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	**n.m.**	2,132,441	712,789	78,912,742	53.14	104.28

3 Computational Experiments

In this section, we compare the proposed formulations in terms of time needed to find a solution, time needed to load the coefficient matrix associated with each formulation (loading time) and the length of that matrix (number of rows, columns and non-zeros elements). We consider four instances from the literature [3], with 30, 50, 70 and 90 vertices (n), and consider that the maximum length of cycles (k) allowed ranges from 3 to 5. We used C++ language and GUROBI library [4], with default options, as integer programming solver. Tests were executed in a computer with 12 cores Intel(R) Xeon(R) CPU X5675/3.07GHz, 50GB of RAM memory, Ubuntu 16.04.3 LTS operation system and g++ version 5.4.0. Preliminary tests on the (Strongly) Stable Cycle–Edge Formulation (SCEF and SSCEF), showed that by using (11) as objective function, the model was more efficient. Therefore, for the two formulations above, we only report results obtained when this objective was considered.

In Tables 1 and 2, $|\mathcal{C}|$ and $|\mathcal{P}|$ are the number of cycles of length at most k and the number of non-cyclic paths with k arcs, respectively. Entries "n.m." indicate that execution was halted due to insufficient memory.

Table 1 shows the experiments results for stable formulations. Notice that for $k = 3$, SCF presents better times then SEF. This fact can be explained by the number of rows and non-zero elements in the coefficient matrix. SEF has more rows because of constraints (8), that are written for all paths in $\mathcal{P}$. However, for $k = 4$ and $k = 5$, the number of non-zero elements in SCF matrices considerably increased, as well as loading times and solver times. This is due to the number of elements in sets $B_{i,c}$ that increases according to k and to the number of arcs and vertices which are common to cycles in $\mathcal{C}$. Table 1 also shows that, for all k, there is a reduction in the number of rows, columns and non-zero elements in SCEF. This happens because, in this formulation, 1) the path constraints (8) are no longer required; 2) since the stability constraints are written in terms of y_{ij}, the number of columns and non-zero elements are reduced. Table 2 shows the corresponding results for strongly stable formulations. The observations made for Table 1 also hold here.

4 Conclusion

In this work, we presented three new integer formulations for modeling k-way stable exchange problems. Computational tests were done with small instances selected from [3]. Results show that the number of rows, columns and non-zero elements of the coefficient matrix associated with each formulation increases the loading time, the solver time and the memory usage with increasing values of k. Furthermore, SCEF and SSCEF outperform the other formulations for all instances, independently of k. These formulations do also request for less memory.

References

1. David J. Abraham, Avrim Blum, and Tuomas Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, EC '07, pages 295–304, New York, NY, USA, 2007. ACM.
2. Péter Biró and Eric McDermid. Three-sided stable matchings with cyclic preferences. *Algorithmica*, 58(1):5–18, Sep 2010.
3. Miguel Constantino, Xenia Klimentova, Ana Viana, and Abdur Rais. New insights on integer-programming models for the kidney exchange problem. *European Journal of Operational Research*, 231(1):57 – 68, 2013.
4. LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018.
5. Chien-Chung Huang. Circular stable matching and 3-way kidney transplant. *Algorithmica*, 58(1):137–150, Sep 2010.

A comparison of matching algorithms for Kidney Exchange Programs

Tiago Monteiro[1], Xenia Klimentova[1], João Pedro Pedroso[1,2], and Ana Viana[1,3]

[1] INESC TEC, Porto, Portugal
[2] Faculdade de Ciências, Universidade do Porto, Porto, Portugal
[3] ISEP - School of Enginnering, Polytechnic of Porto, Porto, Portugal
tiago.p.monteiro@inesctec.pt

Abstract. Kidney Exchange Programs (KEP) allow an incompatible patient-donor pair, whose donor cannot provide a kidney to the respective patient, to have a transplant exchange with another pair in a similar situation if there is compatibility.
In this paper we propose two matching algorithms that address the waiting times of the pairs in a pool, by hierarchically maximizing the number of transplants giving preference to the pairs that have waited longer. The algorithms differ in the strategies used for finding feasible exchanges, as follows. One algorithm runs periodically (e.g. every 3 month); the other runs as soon as the pool is updated allowing for a new exchange. The two algorithms are compared to similar approaches in the literature, that aim at maximizing the number of transplants, through computational experiments.

Keywords: Greedy hierarchical algorithm · Integer programming · Kidney exchange programs · Simulation · Waiting time

1 Introduction

Kidney transplantation is currently the most effective treatment for patients with end-stage renal disease, but finding a suitable kidney can be difficult. There are two different sources for kidneys: from deceased donors and from living donors. However, for the case of living donation the patient and willing donor often do not meet compatibility requirements. This deadlock can be overcome by kidney exchange programs (KEPs) that allow incompatible pairs to perform an exchange between them if the donor in one pair is compatible with the patient in the other pair and vice versa. This is the simplest case, resulting in the so called 2-cycle exchange. However, the size of exchange cycles can be increased, following the same reasoning.

Another possible organization for exchanges is a chain initiated by an altruistic donor, i.e. a donor with no associated patient that donates a kidney for no return. The altruistic donor donates a kidney to a patient of the first pair in the chain, his/her donor to the patient in the following pair and so forth. The last donor in the chain can either donate to the deceased donors waiting

list or act as a bridge donor for the next matching. Due to the several practical constraints the length of a cycle, and frequently of a chain have to be limited by some constant k. In most programs k is set to 3.

KEPs are managed by central or local authorities that collect the incompatible pairs or altruist's registrations and try to identify the exchanges that optimize a given objective. The dynamics of the evolving matching pool, where pairs enter and leave over time, is captured by several models. In [1] the authors conduct simulations that aim at maximizing the number of transplants performed under different time intervals between matches. In [2] authors study how dynamic policies affect the waiting times. Three different settings of feasible solutions are considered: only 2-cycles, 2 and 3-cycles, and a single unbounded chain. Average waiting time is considered as a measure of efficiency; results show that a greedy policy, where exchanges are done as soon as they are available, is nearly optimal.

In our work we propose to address the waiting time of pairs in the pool by hierarchically maximizing in the exchange the number of pairs that waited longer. We develop two matching algorithms that aim at addressing this objective. Matchings are performed either periodically, or as soon as possible (similarly to the work in [2]). We compare results with the cases where maximization of the number of transplants is the only objective considered.

2 Kidney exchange pool and hierarchical waiting times' optimization

We consider a dynamic KEP pool where pairs and altruistic donors appear over time. The pool is represented by a directed graph $G = (V, A)$. The set of vertices $V = P \cup N$ is composed by a set of incompatible pairs P and a set of altruistic donors N. The set of arcs A represent compatibility between vertices: $(i, j) \in A$ if the donor in vertex $i \in V$ and the patient in vertex $j \in V$ are compatible. A feasible exchange is a set of disjoint cycles or chains, where cycles are formed with vertices from set P and chains are initiated by an altruistic donor from set N, followed by vertices from set P. We assume that the maximum size of cycles and chains is limited by a value k (for the case of chains this limit is on the number of vertices involved in a chain, including the altruistic donor). The last donor in a chain donates to the deceased donor's waiting list.

In this sections we describe the two matching approaches proposed. The first one, presented in subsection 2.1, is referred in [2] as greedy. In this case the matching algorithm is run whenever a new incompatible pair or an altruistic donor joins the pool. In the second matching algorithm, subsection 2.2, exchanges are found periodically, i.e., the algorithm is run periodically.

2.1 Greedy hierarchical algorithm

A myopic greedy algorithm was proposed in [2] in the following way: exchanges are performed whenever a cycle or a chain is formed with an arriving altruistic

donor or incompatible pair. Possible ties are broken randomly, and the last donor in the chain acts as an altruistic donor in the following matching. We adapt this algorithm for our settings. Namely, we consider that the maximum length of cycles and chains is the same (k), and choose (randomly in case of multiple possibilities) the one with the largest number of transplants. Furthermore, we assume that the last donor in a chain donates to the deceased donors waiting list. We will refer to this version of the greedy algorithm as $Greedy_{max}$.

Furthermore, we propose a new algorithm where instead of randomly choosing a solution that maximizes the number of transplants, we aim at maximizing the number of transplants while reducing patient's waiting time in the pool. Upon arrival of a pair or of an altruistic donor to the pool, the algorithm checks if new cycles or chains are created with the new arrival. If they are, in case the arrival promotes the creation of more than one cycle or chain, preference is given to the one which contains the pair that has been in the pool for a longer time. Ties are broken by considering the pair with the second longest waiting time, and so forth. For the case of chains, the altruistic donor's waiting time is only considered after pairs are considered, i.e., when we have chains that only differ in the altruistic donor. This case happens when there is more than one altruistic donor in the pool and, upon a new pair arrival, more than one chain (initiated by different altruistic donors, but containing exactly the same pairs) is created. When a potential solution can be either a cycle or a chain and the solution only differs in the fact of the chain having the altruistic donor, priority is given to cycles. This case can happen when an unmatched altruistic donor is already in the pool when a new pair arrives. We will refer to this version of the greedy algorithm as $Greedy_{WT}$.

2.2 Hierarchical integer programming

In another approach we consider that the matching is performed periodically, within given intervals of time (this value is set to 3 months in many countries). In our work this problem is modeled and solved with integer programming (IP) using the cycle formulation [3,4].

Considering the compatibility graph $G = (V, A)$, let $\mathcal{C}$ be a set of cycles and chains with at most k vertices in G. Let $V(c)$ denote the set of vertices that belong to cycle/chain c. By associating the variable x_c for each $c \in \mathcal{C}$, where $x_c = 1$ if cycle/chain c is selected, 0 otherwise, we can write the following IP model:

$$\text{Maximize} \quad \sum_{c \in \mathcal{C}(k)} w_c x_c \tag{1a}$$

$$\text{Subject to:} \quad \sum_{c : i \in V(c)} x_c \leq 1 \qquad \forall i \in V \tag{1b}$$

$$x_c \in \{0, 1\} \qquad \forall c \in \mathcal{C}(k). \tag{1c}$$

The objective function (1a) maximizes the weighted sum of transplants to be performed and constraints (1b) ensure that each vertex is in at most one of the

selected cycles (i.e., each donor may donate, and each patient may receive only one kidney). We will refer to this IP model as IP_{max}.

Similarly to the previous greedy algorithms, we will also consider an IP model where the waiting times of the patients in the pool are addressed. For doing so, for each matching period t we first find exchanges that maximize the number of pairs that waited for t running periods. This can be done by replacing w_c in formulation (1a)-(1c) by w_c^t, where w_c^t is the number of pairs in cycle c that have been in the pool for t matching periods. The optimal value of this problem is denoted by v_t^*. Then, in case there are multiple optimal solutions that provide v_t^*, we choose the ones that maximize the number of patients that waited for $t-1$ periods, similarly, by considering coefficient w_c^{t-1}, that will provide optimum value v_{t-1}^*, in the objective function (1a) and imposing the following additional constraints:

$$\sum_{c \in \mathcal{C}} w_c^t x_c \geq v_t^* \tag{2}$$

The process is repeated until $t = 0$ by, adding to the IP problem constraints (2) in each iteration, with w_c^t replaced by $w_c^{t-1}, \ldots, w_c^0$.

3 Computational analysis

In this section we validate and compare the four matching policies described in section 2: $Greedy_{max}$, $Greedy_{WT}$ and IP_{max}, IP_{WT}. Computational results were obtained for 100 instances, generated with the simulator developed in [5], considering an horizon of 6 years. For the periodic matching algorithms, the interval between matchings was set to 90 days (3 months). We assumed that each patient has only one associated donor, and that pairs and altruistic donors do only leave the pool when matched. Furthermore, if more than one pair or altruistic donor enters the pool in the same day, we prioritize pairs that have O-blood type patients and, if necessary, with a higher value of Panel-reactive antibody (PRA). The maximum length of cycles and chains was set to 3.

Figure 1 illustrates the average waiting times within each period of 90 days for: 1) matched pairs (lower part of the graph), 2) pairs remaining in the pool after matching is performed (upper part of the graph) and, 3) all pairs in the program (matched and not matched). As shown, average waiting times of matched pairs for approaches that prioritize pairs that have been in the pool for a longer time is higher when compared to the other algorithms ($Greedy_{max}$ and IP_{max}). The reason for this is the fact, from all potential solutions, the selected solution is the one that has pairs with longer waiting times. We can also observe that at the beginning of the simulation the IP algorithms have higher waiting times. That can be justified by the time pairs have to wait until the match day arrives. The average waiting time for the pairs remaining in the pool is lower for approaches that take into account the waiting time. This can be justified by the reason presented before. The same happens for the average time of pairs in the program.

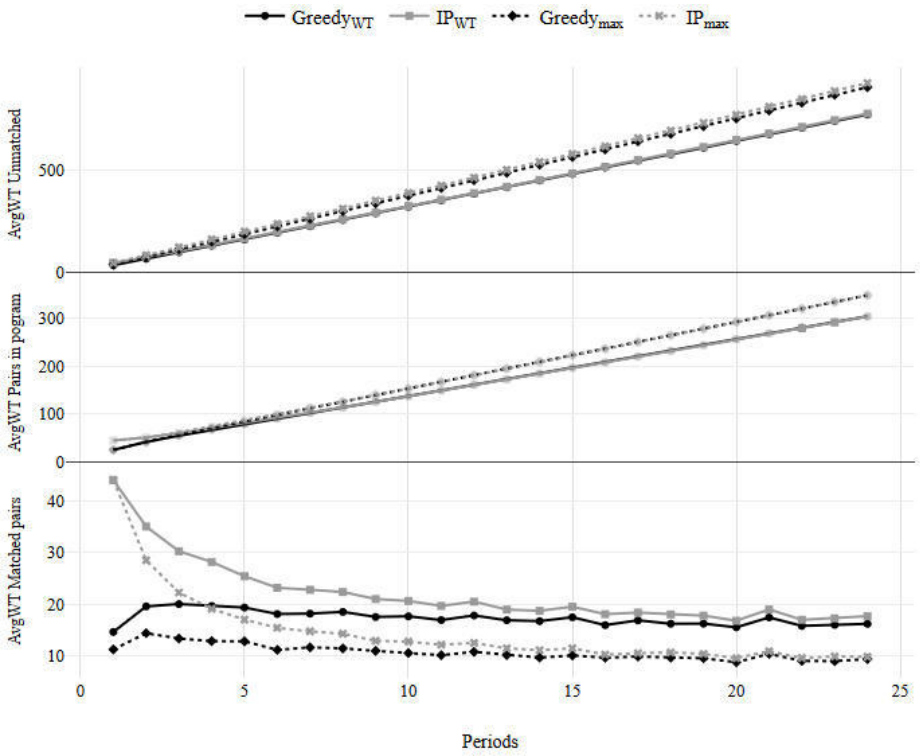

Fig. 1. Average waiting times (AvgWT) for matched pairs, unmatched pairs in the pool and all pairs in the program (matched and unmatched) using four different approaches. Greedy approaches are represented by black lines and IP approaches by grey. Solid lines represent algorithms that consider pairs' waiting times and dashed lines represent those that only consider the maximization of the number of transplants.

In table 1 we present the average number of matched pairs at the end of each simulation year. We can observe that, at the end of the simulation, the application of the *Greedy* algorithms result in less transplants when compared with *IP* algorithms. Moreover, the approaches that consider waiting times present in average a lower number of transplants when compared with those that neglect waiting times.

Table 1. Average number of pairs matched at the end of each simulation for each approach.

	Year						
	1	2	3	4	5	6	**Total**
$Greedy_{WT}$	153.53	163.92	165.06	165.71	163.89	166.53	**978.64**
$Greedy_{max}$	154.1	164.74	166.06	166.21	164.67	167.38	**983.16**
IP_{WT}	159.88	164.84	165.07	165.77	163.79	165.92	**985.27**
IP_{max}	163.32	166.4	166.48	167.03	165.48	168.49	**997.2**

4 Conclusions

In this work we simulate different matching policies in order to identify how they affect pairs waiting time. We propose approaches based on greedy and periodic matching and as evaluation criteria we consider the maximization of the number of transplants and minimization of waiting times of patients. Preliminary computational results show that average waiting times of unmatched patients are reduced for approaches that consider pairs waiting time. This is achieved by slightly sacrificing the total number of transplants performed and increasing average waiting times of matched pairs. As future work we intend to implement a lexicographic procedure that first maximizes the number of transplants and, in a second stage, select the solution involving the pair(s) with longer waiting times. Furthermore, we intend to extend computational results by considering graphs of different density and different frequencies for pair and altruistic donor's arrival.

Acknowledgements This work is financed by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme, and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project "mKEP - Models and optimisation algorithms for multicountry kidney exchange programs" (POCI-01-0145-FEDER-016677), by FCT project SFRH/BPD/101134/2014 and by COST Action CA15210, ENCKEP, supported by COST (European Cooperation in Science and Technology) – http://www.cost.eu/.

References

1. M. Beccuti, V. Fragnelli, G. Franceschinis, and S. Villa. Dynamic simulations of kidney exchanges. In Bo Hu, Karl Morasch, Stefan Pickl, and Markus Siegle, editors, *Operations Research Proceedings 2010*, pages 539–544, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
2. R. Anderson, I. Ashlagi, D. Gamarnik, and Y. Kanoria. Efficient dynamic barter exchange. *Operations Research*, 65(6):1446–1459, 2017.
3. D.J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for Barter exchange markets: Enabling nationwide kidney exchanges. *In: Proceedings of the 8th ACM conference on Electronic commerce, June 13-16*, pages 295–304, 2007.
4. M. Constantino, X. Klimentova, A. Viana, and A. Rais. New insights on integer-programming models for the kidney exchange problem. *European Journal of Operational Research*, 231(1):57–68, 2013.
5. N. Santos, P. Tubertini, A. Viana, and J. P. Pedroso. Kidney exchange simulation and optimization. *Journal of the Operational Research Society*, 68(12):1521–1532, 2017.