Operations Research Letters 51 (2023) 234–241

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Half-cycle: A new formulation for modelling kidney exchange problems

Maxence Delorme[a,*], David Manlove[b], Tom Smeets[a]

[a] *Department of Econometrics and Operations Research, Tilburg University, the Netherlands*
[b] *School of Computing Science, University of Glasgow, United Kingdom*

ARTICLE INFO

Article history:
Received 6 October 2022
Received in revised form 17 February 2023
Accepted 21 February 2023
Available online 28 February 2023

Keywords:
Kidney exchange
Cycle formulation
Integer programming
Preprocessing
Reduced-cost variable fixing

ABSTRACT

We introduce the half-cycle formulation (HCF), a new integer programming (IP) model for the kidney exchange problem, which has life-saving applications. In HCF, a cycle (i.e., set of kidney swaps) is represented by two compatible halves. After giving several algorithmic enhancements for HCF, we show through extensive computational experiments with an IP solver that our new model outperforms existing formulations when the cycle size limit is set to 4, 5, or 6, depending on the density of the compatibility graph.

1. Introduction

Kidneys have an essential role in the human body as they filter the blood to get rid of waste products, they keep the electrolytes (such as sodium and potassium) and water content of the body constant, and they secrete a number of essential hormones [18]. While most people have two kidneys, it is possible to have a healthy and active life with only one functioning kidney. However, if someone has impaired kidney function (which is the case for almost 700 million people in the world according to recent estimates [5]), they may need specialist medical care that can either consist of dialysis or transplantation, the latter option offering both a better quality of life and a longer life expectancy while being less costly [4,23].

For many years, the kidney used in the transplantation could only come from a deceased donor or from a living donor belonging to the close family of the person in need (e.g., a partner or a blood relative). This requirement was motivated by ethical reasons, to remove any financial incentive for living kidney donors. Living donors are in high demand as (i) there are better outcomes for recipients who receive a transplant from a living donor [16] and (ii) in most countries, the pool of deceased donors is not large enough to cover the demand. However, due to a blood-type or tissue-type incompatibility between the two people involved in a transplantation, many recipients cannot receive a kidney despite having found a willing living donor.

Thanks to the seminal works of Rapaport [24] and Roth et al. [26] and after a series of legislation changes in many countries to provide a legal framework for transplants between two persons that do not know each other, kidney exchange programmes were created. Such programmes allow a set of (not necessarily compatible) recipient-donor pairs registered in the system to exchange their donor kidneys if the resulting swaps satisfy compatibility requirements. The swaps are typically determined by a centralised matching algorithm and are computed at regular intervals (e.g., every three months in the UK [17]). Compatibility relationships can easily be captured by a directed compatibility graph, where an arc is drawn from one pair to another if the donor of the first pair is compatible with the recipient of the second pair. For example, if we consider two recipient-donor pairs (r_1, d_1) and (r_2, d_2) and a complete compatibility graph, then a suitable exchange matches r_1 to d_2 and r_2 to d_1 and is called a 2-cycle ("2" because there are two kidney transplants since two recipient-donor pairs are involved in the swap, and "cycle" because the arcs associated with these swaps form a cycle in the compatibility graph). Due to practical constraints, most kidney exchange programmes impose a limit on the number of pairs that can be involved in the same cycle. The kidney exchange problem (KEP) can therefore be defined as the problem of finding a vertex-disjoint set of cycles with the maximum number of transplants, given a set of recipient-donor pairs, a compatibility graph, and a limit on the number of pairs involved in any cycle. Note that the vertex-disjoint requirement comes from the fact that each recipient and donor can be involved in at most one cycle.

* Corresponding author.
E-mail address: m.delorme@tilburguniversity.edu (M. Delorme).

https://doi.org/10.1016/j.orl.2023.02.009
0167-6377/© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Over the years, kidney exchange programmes have evolved and developed country-specific components, resulting in several KEP extensions. One of the most frequent extensions considers the inclusion of non-directed (or altruistic) donors, each of which is not paired with any recipient. An exchange triggered by a non-directed donor is called a chain because the recipient of the last recipient-donor pair receives a kidney but its paired donor is not immediately matched with any other recipient in the pool of recipient-donor pairs (this donor may initiate a chain in the next run of the programme or give their kidney to a recipient on the deceased-donor waiting list). Another frequent extension considers a set of hierarchically ordered objective functions to determine the best solution (i.e., the best set of cycles and chains). For example, a kidney exchange programme may want to maximise the number of transplants, and if several optimal solutions exist, may wish to pick the one that includes the highest number of 2-cycles. We refer the interested reader to the works of Biró et al. [6,7] for a detailed description of European kidney exchange programmes.

As far as the KEP literature is concerned, we identified three main (not necessarily disjoint) paper categories: (i) those providing new theoretical results and models for the KEP; (ii) those increasing the algorithmic performance of existing models; and (iii) those adapting existing approaches to take supplementary real-world features into account.

Starting with papers in category (i), two of the most important theoretical KEP results were obtained by Roth et al. [27] and Abraham et al. [1]. The former showed that if only 2-cycle are allowed, then an optimal solution for the KEP can be found in polynomial time as the problem can be reduced to a maximum weighted matching problem. The latter proved that the KEP becomes $\mathcal{NP}$-hard when the limit on the number of pairs involved in a cycle is greater than or equal to 3, while it can be solved in polynomial time if such a limit does not exist. The $\mathcal{NP}$-hardness result has motivated the research community to find effective techniques for coping with the complexity of KEP, which include Integer Programming (IP). Among the most efficient (i.e., fastest to solve with an IP solver) KEP models without non-directed donors introduced in the literature, we mention the *cycle formulation* and the *edge formulation* that were both introduced by Roth et al. [28], the *extended edge formulation* (EEF) introduced by Constantino et al. [9], and the *position-indexed edge formulation* (PIEF) introduced by Dickerson et al. [14]. Since a non-directed donor can be modelled as a recipient-donor pair whose recipient is compatible with every other donor, these formulations can also be used for KEP instances containing non-directed donors. We mention however that Dickerson et al. [14] introduced a very effective way to model chains in KEPs that, when coupled with the cycle formulation to model cycles, obtained state-of-the-art results for instances with at most three pairs per cycle and up to eleven pairs per chain.

In category (ii), as far as improving the algorithmic performance of existing models is concerned, we mention the work of Lam and Mak-Hau [19] who solved the cycle formulation with a branch-and-price framework, and the work of Delorme et al. [11] who used reduced-cost variable fixing together with preprocessing techniques and a diving algorithm to enhance the performance of the cycle formulation for KEPs with hierarchical optimisation. Finally, for category (iii), the literature aiming at adapting existing approaches to real-world applications is very large. We only mention a few main areas such as realistic instance generators [12,29], stochastic optimisation [2,22], and hierarchical optimisation [7,11].

In this work, we introduce the *half-cycle formulation* (HCF), a new IP model for KEPs without non-directed donors. The model is inspired by the cycle formulation and represents a cycle by two compatible halves. We detail some algorithmic enhancements for HCF; these include the use of a reduced-cost variable fixing framework [11] that could also be applied to PIEF. We show that HCF together with these algorithmic enhancements obtains better empirical performance when solved with a commercial IP solver than other mathematical models when the cycle size limit is set to 4, 5, or 6, depending on the density of the compatibility graph. This improved empirical performance includes faster solution times and the ability to solve larger instances to optimality. Increasing the instance size that can be solved with IP modelling is important for the KEP community because these formulations are often considered easier to re-implement and modify than handmade algorithms such as branch-and-price algorithms [19] or matheuristics [12]. Our experimental findings are based on an extensive empirical evaluation that indicates the limits of each of the tested IP models.

The rest of the paper is organised as follows. In Section 2, we formally define the KEP and briefly mention existing IP formulations and their main features. We then introduce HCF in Section 3 together with a study of its continuous relaxation, some reduction procedures and algorithmic enhancements, a discussion about extending the model to handle non-directed donors, and some insights about a version of HCF where cycles are split into three parts or more. Section 4 gives a summary of the results obtained by extensive computational experiments on realistic instances and concluding remarks are provided in Section 5.

2. Problem statement and existing formulations

In the KEP without non-directed donors (called KEP-WNDD thereafter), we are given a set of n recipient-donor pairs together with a limit K on the maximum number of pairs that can be included in any cycle and a compatibility graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where vertex set $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ contains one node per recipient-donor pair and where arc set $\mathcal{A}$ contains arc (p_1, p_2) if the donor of pair p_1 is compatible with the recipient of pair p_2. The objective is to compute a subset of arcs $\mathcal{A}' \subseteq \mathcal{A}$ (the transplants) such that $|\mathcal{A}'|$ is maximised while being solely composed of disjoint cycles with cardinality K or below.

Let $S = \{v_1, \ldots, v_r\}$ be a set of vertices, where each recipient and donor appears in at most one vertex of S. If $(v_i, v_{i+1}) \in \mathcal{A}$ for each i $(1 \le i \le r-1)$ and $(v_r, v_1) \in \mathcal{A}$, then S gives rise to a *cycle*, denoted by $[v_1, v_2, \ldots, v_r]$. The *size* of this cycle is r. In anticipation of the description of HCF, we also introduce the concept of a *half-cycle* where a cycle can be decomposed into two halves, denoted by $\langle v_1, v_2, \ldots, v_{\lceil r/2 \rceil+1} \rangle$ and $\langle v_{\lceil r/2 \rceil+1}, v_{\lceil r/2 \rceil+2}, \ldots, v_r, v_1 \rangle$. For example if we consider the cycle $[A, B, C, D]$, where $A, B, C, D \in \mathcal{V}$ and (A, B), (B, C), (C, D), $(D, A) \in \mathcal{A}$, then this cycle can be decomposed into two half-cycles $\langle A, B, C \rangle$ and $\langle C, D, A \rangle$. The *sizes* of these half-cycles are $\lceil r/2 \rceil + 1$ and $\lfloor r/2 \rfloor + 1$, respectively. We also remark that the vertex labelling of $\{v_1, v_2, \ldots, v_n\}$ gives an ordering on the vertices that will be used in HCF.

One of the most intuitive models, the so-called cycle formulation [28], enumerates every possible cycle of size K or below, stores them in set $\mathcal{C}$, and associates one binary variable ξ_c to every feasible cycle $c \in \mathcal{C}$, where ξ_c takes value 1 if cycle c is selected in the solution and value 0 otherwise. By defining the set $V(c)$ to contain the vertices involved in cycle c $(c \in \mathcal{C})$, we can define the cycle formulation as follows:

$$\max \quad z = \sum_{c \in \mathcal{C}} |V(c)| \, \xi_c \tag{1}$$

$$\text{s.t.} \quad \sum_{c \in \mathcal{C}: v \in V(c)} \xi_c \le 1, \quad \forall v \in \mathcal{V}, \tag{2}$$

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

Objective function (1) maximises the number of transplants while constraints (2) make sure that no recipient-donor pair appears in more than one of the selected cycles. Even though it is one of the first proposed IP models, the cycle formulation still obtains the best results for KEP-WNDD instances with values of K up to 4 when compared to other IP models [9]. This can be explained by the size of the model, containing $O(n^K)$ variables and $O(n)$ constraints, which remains tractable for low values of n and K, and by the good quality of the bound derived from the Linear Programming (LP) relaxation of the model, which is the best among all existing IP models for the KEP-WNDD [14].

Another intuitive model is the so-called edge formulation [28] which associates a binary variable to every arc $a \in \mathcal{A}$ in the compatibility graph. This variable takes value 1 if the arc is selected in the solution and value 0 otherwise. The objective is therefore to maximise the number of arcs selected under the constraints that (i) there should be flow conservation at every node, (ii) every node should have at most one outgoing flow, and (iii) the selected arcs should not form a cycle of size $K+1$ or above. While the edge formulation only uses one variable per arc (so at most n^2 in total), it requires an exponential number of constraints of type (iii). One way to model these constraints is to enumerate every feasible chain of size K that does not form a cycle and forbid each of these chains with a no-good cut, resulting in a total of $O(n^K)$ constraints. It was empirically shown that the edge formulation is much less effective than the cycle formulation [9], even when solved within a constraint generation framework that only adds the relevant cycle elimination constraints (i.e., the ones that have been violated by a previous incumbent solution) to the enumeration tree.

The first compact (or fully polynomial) IP formulation for the KEP-WNDD is EEF [9]. In brief, EEF considers n copies of the compatibility graph, one for every potential cycle, and associates one variable for every arc in every copy. Such a variable takes value 1 if the arc is selected in the solution in its associated copy and value 0 otherwise. The objective is still to maximise the number of arcs selected under the constraints that (i) there should be flow conservation at every node in each of the n graph copies, (ii) every node should have a sum of outgoing flow over the n copies less than or equal to 1, and (iii) at most K arcs should be selected per graph copy. Even though EEF has $O(n^3)$ variables and $O(n^2)$ constraints, the model is not particularly competitive in practice due to the poor quality of its linear relaxation. Later on, Dickerson et al. [14] introduced PIEF, an extension of EEF that duplicates each of the n graph copies K times to keep track of the position of every arc in a cycle. They showed that PIEF has the same LP-relaxation as the cycle formulation while only having $O(Kn^3)$ variables and $O(Kn^2)$ constraints. Note that preprocessing techniques can remove many variables and constraints in EEF and PIEF.

A summary of the most prominent existing models, their respective size, and the quality of their linear relaxation is given in Table 1. For the sake of completeness, we also included our new formulation HCF in the table. While many features may impact the time required by a state-of-the-art IP solver to solve an instance to optimality, the research community tends to agree that the model size (i.e., the number of variables and constraints) and the quality of the LP-relaxation bound both play a crucial role. In that regard, the cycle formulation, PIEF, and HCF complement each other very well as each of these models has the smallest size for at least one value of K[1] while always having a tight LP-relaxation bound (for HCF, the model size and the tightness of the LP-relaxation are derived in Section 3.1).

To the best of our knowledge, no other mathematical model for the KEP-WNDD has been proposed in the literature, apart from two edge formulation extensions introduced in a thesis [25] that were shown to offer no competitive advantage. Note that a larger variety of models have been proposed for the KEP with non-directed donors [3,14,20]. Indeed, even though every modelling technique used to represent a cycle can also be used to represent a chain, there are also (more efficient) techniques that can be used to represent a chain that are not suitable for representing a cycle.

[1] Example values are $K=2$ for the cycle formulation, $K=4$ for HCF and $K=6$ for PIEF.

Table 1
Number of variables and constraints of IP models for the KEP-WNDD.

IP model	No. var.	No. cons.	LP-relaxation bound
Cycle formulation	$O(n^K)$	$O(n)$	tight
Edge formulation	$O(n^2)$	$O(n^K)$	not tight
EEF	$O(n^3)$	$O(n^2)$	not tight
PIEF	$O(Kn^3)$	$O(Kn^2)$	tight
HCF	$O(n^{1+\lceil K/2 \rceil})$	$O(n^2)$	tight

3. Half-cycle formulation and reduction procedures

In the last decade, several research papers introduced strategies to exploit the symmetry within combinatorial optimisation problems to derive IP models with a reduced size. For example, Delorme and Iori [13] introduced a new mathematical model for the bin packing problem that uses half of the bin capacity to model an instance, requiring significantly fewer constraints and variables than existing models (that used the full bin instead). The authors represented both the first half and the second half of a bin with the same modelling structure and showed that the resulting decrease in terms of model size outweighed – by far – the extra burden brought by the supplementary constraints that are necessary to ensure compatibility between the selected half bins. This "two halves instead of a whole" strategy was extended later on to other packing problems [10,21]. In this section, we show that a similar idea can also be used for the KEP-WNDD.

3.1. Mathematical model

In our new formulation HCF, a cycle is represented by two compatible half-cycles. To guarantee that the half-cycles selected by the model are compatible, we simply need to ensure that, for every pair of nodes $v_1, v_2 \in \mathcal{V}$, a half-cycle starting by v_1 and ending by v_2 is selected if and only if another half-cycle starting by v_2 and ending by v_1 is also selected. It is tempting to believe that splitting a cycle into two halves will lead to a mathematical model that has twice as many variables as the cycle formulation. However this need not be the case, as is illustrated in Section A of the Online Supplement.

After enumerating every possible half-cycle of size up to $1 + \lceil K/2 \rceil$ and storing them in set $\mathcal{H}$, the model associates one binary variable x_h to every feasible half-cycle $h \in \mathcal{H}$ taking value 1 if half-cycle h is selected in the solution and value 0 otherwise. Note that half-cycles containing the same node twice are not generated. By using $V^s(h)$ (resp. $V^e(h)$) to indicate the vertex starting (resp. ending) half-cycle h ($h \in \mathcal{H}$), and $V^m(h)$ to indicate the other (middle) vertices of h, we can define HCF as follows:

$$\max \quad z = \sum_{h \in \mathcal{H}} (|V^m(h)| + 1)\, x_h \tag{4}$$

$$\text{s.t.} \quad \sum_{h \in \mathcal{H}: v \in V^s(h) \cup V^e(h)} 0.5 x_h + \sum_{h \in \mathcal{H}: v \in V^m(h)} x_h \le 1, \qquad \forall v \in \mathcal{V}, \tag{5}$$

$$\sum_{h \in \mathcal{H}: v_1 \in V^s(h), v_2 \in V^e(h)} x_h = \sum_{h \in \mathcal{H}: v_1 \in V^e(h), v_2 \in V^s(h)} x_h, \qquad \forall v_1 \in \mathcal{V}, v_2 \in \mathcal{V} : v_2 > v_1 \tag{6}$$

$$x_h \in \{0, 1\}, \quad \forall h \in \mathcal{H}. \tag{7}$$

Objective function (4) maximises the number of transplants while taking into account that the starting and ending nodes of every selected half-cycle only count for halves as they both appear in two half-cycles. Constraints (5) make sure that no recipient-donor pairs appear more than once in the middle or more than twice at the start/end of the selected half-cycles, while constraints (6) ensure that every half-cycle selected in the solution can be matched with another selected half-cycle to form a complete cycle. Note that, to ensure the correctness of the model in the case where K is odd, the following set of constraints is needed in HCF:

$$x_h = 0, \quad \forall h \in \mathcal{H} : V^s(h) > V^e(h) \text{ and } |V^m(h)| = (K-1)/2. \quad (8)$$

Constraints (8) prevent any half-cycle $\langle v_1, \ldots, v_2 \rangle$ with size $1 + \lceil K/2 \rceil$ and v_1 index greater than v_2 index from being generated. This forbids two half-cycles with size $1 + \lceil K/2 \rceil$ from being merged, which would be the equivalent (when K is odd) of a cycle with size $K + 1$.

We point out that the bound obtained by the LP-relaxation of HCF is as tight as the bound obtained by the cycle formulation since the two models are equivalent, or in other words, since any continuous solution of model (4)-(8) can be transformed into a continuous solution of model (1)-(3) with the same objective value and vice versa (the proof is available in Section B of the Online Supplement).

3.2. Reduction procedures and node ordering

As in the case of other IP models for the KEP, HCF displays some symmetry. For example, a cycle $[A, B, C, D]$ can be obtained by selecting half-cycles $\langle A, B, C \rangle$ and $\langle C, D, A \rangle$ or by selecting half-cycles $\langle B, C, D \rangle$ and $\langle D, A, B \rangle$. This can be avoided by deactivating (i.e., by setting the associated variable to 0 or not generating the variable at all) every half-cycle in which the vertex with the lowest index is neither located at the beginning nor at the end of the half-cycle. This rule is not sufficient to remove all symmetry as a cycle that includes an odd number of recipient-donor pairs such as $[A, B, C, D, E]$ can still be obtained by selecting different sets of half-cycles such as $\langle A, B, C, D \rangle$ and $\langle D, E, A \rangle$ or $\langle A, B, C \rangle$ and $\langle C, D, E, A \rangle$.[2] This can be avoided in constraints (6) by only allowing a half-cycle $\langle v_1, \ldots, v_2 \rangle$ with size k to be matched with another half-cycle $\langle v_2, \ldots, v_1 \rangle$ with size k or, in case the index of v_1 is smaller than the index of v_2, with size k and $k-1$. Preprocessing removing unnecessary variables can also be applied to HCF. A simple rule is to only consider a half-cycle in the model if it can be completed by another half-cycle.

Considering that the bound obtained by the LP-relaxation of HCF is as tight as the bound obtained by the cycle formulation, we can follow the path of Delorme et al. [11] and apply the concept of a destructive bound together with a reduced-cost variable fixing strategy to reduce the number of variables that needs to be considered in the model. To do so, we start by solving the LP-relaxation of model (4)-(8), save the objective value $\hat{z}$, and use it to derive a valid upper bound $U = \lfloor \hat{z} \rfloor$ on the maximum number of transplants. From now on, we are only looking for an integer solution with objective value U. Delorme et al. [11] proved a general result concerning reduced-cost variable fixing, a consequence of which is that any continuous solution of model (4)-(8) where a variable x_h with reduced cost $\hat{s}_h \le U - \hat{z} - \epsilon$ takes value 1 or above must have objective value strictly below U (ϵ is a very small number used to avoid precision errors). As a result, no integer solution selecting the half-cycle associated with such a variable x_h can have objective value U and x_h can therefore be deactivated. If we find a solution with value U for the reduced model, then that solution is optimal. If that is not the case, then it means that no solution with objective value U exists, so $U - 1$ becomes a valid upper bound. Therefore, U is decremented and all the variables are reactivated except the ones with reduced cost smaller than or equal to the updated $U - \hat{z} - \epsilon$ value. The algorithm is iterated until a solution with value U is found. Note that in a more efficient implementation, the algorithm terminates if the optimal objective value of the reduced model is equal to $U - 1$.

A peculiarity of HCF (which also occurs with EEF and PIEF, but not in the cycle formulation) is that the node ordering has an impact on the number of variables required by the model. To illustrate this behaviour, we consider a simple instance whose compatibility graph $\mathcal{G}$ is displayed in the left part of Fig. 1, where $\mathcal{G}$ has 4 recipient-donor pairs and $K = 4$. In the right part of the figure, we outline the half-cycles that should be merged together in order to build each of the feasible cycles of the instance, given a certain node ordering and the reduction procedures mentioned above. One can observe that using node ordering $B < C < D < A$ produces fewer half-cycles (and thus, fewer variables in the HCF model) than using node ordering $A < D < B < C$. In fact, searching for the minimum number of half-cycles that are necessary to represent every feasible cycle in a KEP-WNDD is an optimisation problem in its own right. Interestingly, such an objective cannot always be achieved by using a specific node ordering in the context of our HCF model construction process. Indeed, in our example, every feasible cycle can be obtained by using only 4 half-cycles, namely $\langle B, C \rangle$, $\langle C, B \rangle$, $\langle C, A, B \rangle$, and $\langle B, D, C \rangle$. However, these 4 half-cycles do not correspond to any valid node ordering since $\langle C, A, B \rangle$ can only be merged with $\langle B, C \rangle$ if $C < B$, while $\langle B, D, C \rangle$ can only be merged with $\langle C, B \rangle$ if $B < C$. Empirical results outlining the impact of using different node ordering rules and the impact of using a reduced-cost variable fixing strategy are provided in Section 4.

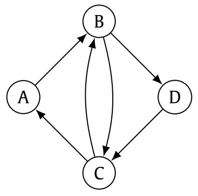
3.3. Extending HCF to the KEP with non-directed donors

Even though we introduced HCF for the KEP-WNDD, we point out that there are several ways to extend the model to take non-directed donors into account. The most intuitive (but probably not the most efficient) strategy would be to represent the chains as cycles in the compatibility graph, by introducing a dummy recipient for each non-directed donor that is compatible with every directed donor. However, we expect that better computational performance could be obtained by using instead the chain structure introduced by Dickerson et al. [14] as the latter only requires a polynomial number of variables and constraints, $O(Ln^2)$ and $O(Ln)$, respectively (where n is now the number of recipient-donor pairs plus the number of non-directed donors and L is the chain size limit).

3.4. Splitting a cycle into three or more parts

Considering the significant reduction in terms of number of variables involved in the model, one might wonder if further improvements could be obtained by splitting the cycles into three parts (or more) instead of two. Indeed, a cycle $[A, B, C, D, E, F]$ could be split, for example, into "third-cycles" $\langle A, B, C \rangle$, $\langle C, D, E \rangle$, and $\langle E, F, A \rangle$. While this would reduce the theoretical number of variables from $O(n^{1+\lceil K/2 \rceil})$ to $O(n^{1+\lceil K/3 \rceil})$, it would also increase the total number of constraints as one now needs to make sure that every third-cycle selected in the solution can be matched with two other third-cycles to form a complete cycle. In other words,

[2] Note that this example is only valid if $K \ge 6$, as at most one half-cycle among $\langle A, B, C, D \rangle$ and $\langle C, D, E, A \rangle$ would be generated if $K = 5$, since in this case, the former half-cycle would only be generated if the index of A is smaller than the index of C (as the first reduction procedure ensures that either the index of A or the index of D is smaller than the index of C, and constraints (8) force the index of A to be smaller than the index of D), while the latter half-cycle is only generated if the index of C is smaller than the index of A (again by constraints (8)).



Feasible cycles	Unique half-cycle representation given a fixed node ordering		
	$B < C < D < A$	$A < B < C < D$	$A < D < B < C$
$[A, B, D, C]$	$\langle B, D, C\rangle + \langle C, A, B\rangle$	$\langle A, B, D\rangle + \langle D, C, A\rangle$	$\langle A, B, D\rangle + \langle D, C, A\rangle$
$[A, B, C]$	$\langle B, C, A\rangle + \langle A, B\rangle$	$\langle A, B, C\rangle + \langle C, A\rangle$	$\langle A, B, C\rangle + \langle C, A\rangle$
$[B, D, C]$	$\langle B, D, C\rangle + \langle C, B\rangle$	$\langle B, D, C\rangle + \langle C, B\rangle$	$\langle D, C, B\rangle + \langle B, D\rangle$
$[B, C]$	$\langle B, C\rangle + \langle C, B\rangle$	$\langle B, C\rangle + \langle C, B\rangle$	$\langle B, C\rangle + \langle C, B\rangle$
Total	6 distinct half-cycles	7 distinct half-cycles	8 distinct half-cycles

Fig. 1. An instance outlining the importance of the node ordering rule.

one should make sure that (i) for every node v, if there is a third-cycle ending in v, there must be a third-cycle starting from v; and (ii) for every three nodes v_1, v_2, v_3, if there is both a third-cycle starting from v_1 and ending in v_2 and a third-cycle starting from v_2 and ending in v_3, there must be a third-cycle starting from v_3 and ending in v_1. This would bring the theoretical number of constraints to $O(n^3)$, which would not provide a competitive advantage with respect to the models presented in Table 1 for any value of K.

4. Computational experiments

We first tested the performance of HCF on a set of randomly generated instances with various sizes $n \in \{50, 100, 200, 400, 600, 800, 1000\}$ and maximum cycle size values $K \in \{3, 4, 5, 6, 7, 8\}$ using the instance generator introduced by Delorme et al. [12] and available at https://wpettersson.github.io/kidney-webapp. We clicked on "Use recipient blood group distributions from the paper" and left the other parameters untouched. We generated 20 instances for each value of n, resulting in 140 instances in total that can be downloaded from https://github.com/mdelorme2/Half_Cycle_Instances. These instances will be referred to as the DGGKMPT instances in the following. Note that each of the 140 DGGKMPT instances were solved using different values of K.

We then tested the performance of HCF on instances from the "Kidney Data (00036)" dataset [15], which can be downloaded from the PrefLib library at https://www.preflib.org/dataset/00036 and which will be referred to as the PrefLib instances in the following. We selected the 10 PrefLib instances without non-directed donors for each size $n \in \{64, 128, 256, 510, 1024\}$. PrefLib instances have been widely used by the research community [12,19] and are characterised by a high graph density (around 25%), which usually increases the number of variables that needs to be generated in IP models. The graph density for DGGKMPT instances is close to 10%, a proportion that is similar to what is observed in the UK kidney exchange programme, which is one of the reasons why it was argued in [12] that the generator from [12] produced more realistic instances than those obtained by the well-known Saidman generator [29], which was used to create the PrefLib instances. We tested the following models:

- **CYCLE**, the cycle formulation [28] with symmetry reduction (only the cycles starting with the lowest indexed pair are generated);
- **EDGE**, the edge formulation [28] implemented within a branch-and-cut framework;
- **EEF**, the extended edge formulation [9] with symmetry reduction (in copy v of the compatibility graph, only the arcs with head and tail indices equal to v or above that can be included in a cycle of size up to K containing node v are considered; nodes are sorted according to a pre-defined node ordering);
- **PIEF**, the position-indexed edge formulation [14] with the same symmetry reduction and node ordering as EEF;
- **HCF**, the half-cycle formulation introduced in Section 3 with symmetry reduction and node ordering as described in Section 3.2;
- **CYCLE-RCVF**, algorithm CYCLE with reduced-cost variable fixing (as described by Delorme et al. [11] and summarised in Section 3.2);
- **PIEF-RCVF**, algorithm PIEF with reduced-cost variable fixing;
- **HCF-RCVF**, algorithm HCF with reduced-cost variable fixing.

We point out that we did not use reduced-cost variable fixing on the edge formulation and on EEF because the LP-relaxation of those models is often more than one unit away from the optimal solution value.

Our algorithms were all coded in C++ and can be downloaded from https://github.com/mdelorme2/Half_Cycle_Codes. The experiments were run on an Intel(R) Core(TM) i5-1135G7, 2.40 GHz with 32 GB of memory, running under Ubuntu 22.04.1 LTS, and Gurobi 10.0.0 was used to solve the IP models. A single core was used for the tests (i.e., parameter `Threads` was set to 1), the barrier algorithm was used to solve the root nodes of the IP models (i.e., parameter `Method` was set to 2), and callbacks were used for the constraint generation (and thus, parameter `LazyConstraints` was set to 1 for EDGE). All the instances were first solved with a 16 GB memory limit for the environment (i.e., parameter `MemLimit` was set to 16). Those instances that ended prematurely due to being out of memory were rerun with the memory limit increased to 30 GB. For the algorithms using reduced-cost variable fixing, we deactivated the crossover operation when solving the LP models (i.e., parameter `Crossover` was set to 0). This means that the solver did not try to transform the interior solution produced by the barrier algorithm into a basic solution. For these last three approaches, we also changed the focus of the solver so that more time was spent on finding a good-quality solution (i.e., parameter `MIPFocus` was set to 1). For every run, a time limit of 3600 seconds was imposed. Note that, for a given K, when an approach could not solve any instance with size n, we decided to not test the approach on instances larger than n and filled the associated rows with the symbol "-".

Results for the DGGKMPT instances are presented in Section 4.1, whilst those for the PrefLib instances are given in Section 4.2.

4.1. DGGKMPT instances

We report in Table 2 the results of a first set of experiments aimed at evaluating the impact of two node ordering rules: one where the vertices are sorted in descending order of total degree (denoted by the suffix "-D") that was suggested by Dickerson et al. [14] for PIEF, and one where the vertices are sorted in ascending order of total degree (denoted by the suffix "-A"). For the sake of conciseness, we only report the results for models PIEF and HCF on instances with $K = 4$. The first two columns identify the limit K on the number of pairs that can be included in a cycle and the total number of pairs n. The following columns provide, for each of the tested models, the number of optimal solutions found, the average CPU time over the 20 runs (including the ones terminated by the time limit), and the average number of variables and constraints involved.

The results suggest that sorting the vertices in descending order of total degree reduces the model size for both PIEF and HCF, which results in more instances being solved to optimality over-

Table 2
Results of PIEF and HCF with two node ordering rules for DGGKMPT instances with $K = 4$.

K	n	PIEF-A				PIEF-D				HCF-A				HCF-D			
		# opt	T(s)	no. var.	no. cons.	# opt	T(s)	no. var.	no. cons.	# opt	T(s)	no. var.	no. cons.	# opt	T(s)	no. var.	no. cons.
4	50	20	0	174	403	20	0	127	279	20	0	117	95	20	0	86	77
	100	20	0	1499	2681	20	0	1127	1620	20	0	1068	470	20	0	821	312
	200	20	2	16 489	20 942	20	1	11 492	10 735	20	1	13 788	3762	20	0	9546	1953
	400	20	54	153 932	131 649	20	38	109 629	67 788	20	23	141 066	26 025	20	15	99 605	13 241
	600	19	1010	638 524	397 533	20	396	454 042	197 518	20	91	606 854	83 422	20	69	429 021	41 236
	800	6	3324	1 672 574	800 293	13	2789	1 190 571	398 566	20	515	1 616 706	174 170	20	300	1 144 460	85 692
	1000	0	3600	3 658 273	1 413 525	1	3589	2 621 062	699 596	17	1811	3 566 763	312 648	20	966	2 544 940	153 453

Table 3
Results of the tested approaches for DGGKMPT instances with $K = 3$ and $K = 4$.

K	n	CYCLE		EDGE		EEF		PIEF		HCF		CYCLE-RCVF		PIEF-RCVF		HCF-RCVF	
		#opt	T(s)	#opt	T(s)	#opt	T(s)	#opt	T(s)	#opt	T(s)	#opt	T(s)	#opt	T(s)	#opt	T(s)
3	50	20	0	20	0	20	0	20	0	20	0	20	0	20	0	20	0
	100	20	0	20	0	20	0	20	0	20	0	20	0	20	0	20	0
	200	20	0	5	2865	20	0	20	0	20	0	20	0	20	0	20	0
	400	20	0	0	3600	20	1	20	2	20	0	20	0	20	3	20	0
	600	20	1	-	-	20	4	20	10	20	1	20	1	20	12	20	1
	800	20	2	-	-	20	20	20	30	20	3	20	2	20	32	20	3
	1000	20	5	-	-	20	56	20	79	20	8	20	4	20	68	20	6
4	50	20	0	20	0	20	0	20	0	20	0	20	0	20	0	20	0
	100	20	0	18	719	20	0	20	0	20	0	20	0	20	0	20	0
	200	20	0	0	3600	20	2	20	1	20	0	20	0	20	1	20	0
	400	20	7	-	-	20	118	20	38	20	15	20	4	20	23	20	7
	600	20	80	-	-	20	1474	20	396	20	69	20	40	20	161	20	41
	800	20	387	-	-	0	3600	13	2789	20	300	20	110	19	1237	20	107
	1000	20	1358	-	-	-	-	1	3589	20	966	20	325	14	2611	20	294

all. This is in line with the suggestion of Dickerson et al. [14]. Note that we also tested the impact of these two ordering rules on PIEF-RCVF and HCF-RCVF, and similar conclusions could be drawn. As far as EEF is concerned, the descending ordering rule also led to smaller size models, but it also led to a deterioration of the LP-relaxation bound (we remind the reader that the LP-relaxation bound of EEF is not tight). Empirically, we observed that instances were solved faster for EEF when using the ascending ordering rule.

We report in Table 3 the results obtained by each of the tested approaches (with its best ordering rule, i.e., descending for PIEF, HCF, PIEF-RCVF, and HCF-RCVF and ascending for EEF) on DGGKMPT instances with $K = 3$ and $K = 4$.

For $K = 3$, we observe that every approach apart from EDGE could solve all the instances. CYCLE-RCVF obtained the best results, followed by CYCLE, HCF-RCVF, and HCF. We point out that there is no theoretical interest in using HCF (or HCF-RCVF) over CYCLE (or CYCLE-RCVF) for these instances as every half-cycle of size 3 can only be completed by one single half-cycle of size 2. We also remark that the version of the model using reduced-cost variable fixing always outperforms the corresponding version that does not use it.

For $K = 4$, we observe that only CYCLE-RCVF and HCF-RCVF can solve all the tested instances, but neither of the two approaches clearly outperforms the other. We mention that CYCLE has around 2.8 million variables and 800 constraints on average for instances with 800 recipient-donor pairs while HCF has around 1.1 million variables and 86 thousand constraints. This confirms the general paradigm behind HCF, which is to reduce the number of variables at the cost of supplementary constraints compared to CYCLE. For the same instances, CYCLE-RCVF has around 1 million variables and 800 constraints on average while HCF-RCVF had 581 thousand variables and 66 thousand constraints, which indicates that reduced-cost variable fixing can be very effective.

Based on these results, we continue our tests with larger values of K for the methods EDGE, CYCLE-RCVF, PIEF-RCVF, and HCF-RCVF. We decided to include EDGE because it is expected that the branch-and-cut framework becomes more effective as K increases since the cycles found in incumbent solutions are less likely to have size $K + 1$ or above (and therefore, are less likely to require a cut).

We report in Table 4 the results obtained by the four remaining approaches on DGGKMPT instances with $K = 5$ and $K = 6$. For each model, we added the average number of variables and constraints involved in the model after reduced-cost variable fixing (if any) and for EDGE, the average number of cuts added within the branch-and-cut framework. As early termination due to memory limit occurred in these experiments (even after increasing that limit to 30 GB), the number of instances that were prematurely ended is now reported within brackets after the number of optimal solutions found in column "# opt" (if that number was not 0). Typically, an instance would run out of memory within the first hundred seconds of running time, when solving the LP-relaxation of the model (so before reduced-cost variable fixing). As a result, the instances terminated because the memory limit was exceeded were neither used to compute the average computation time of the approach nor the average number of variables and constraints involved in the model.

For $K = 5$, we observe that CYCLE-RCVF starts running out of memory for instances with 600 recipients. This can be explained by the very large model sizes before reduced-cost variable fixing (40 million variables and 199 million non-zero elements in the coefficient matrix on average for the instances that were prematurely terminated versus respectively 23 million and 114 million for the instances that were not). While we cannot guarantee that CYCLE-RCVF would not be able to solve any of the instances ended prematurely due to memory limitation, our results already show that the model is not the most competitive anymore as it could not solve two instances within the time limit while HCF-RCVF could. Indeed, HCF-RCVF is the only approach able to solve all instances with $n = 600$ and is also the fastest to solve instances with $n = 400$. In addition, the model could solve 13 instances with $n = 800$, while the second best approach, PIEF-RCVF, could only solve 3 instances in that class. Note that, even though HCF-RCVF ran out of memory for 6 instances with $n = 1000$, we do not expect that memory limitation was a major issue since the model could not solve any of the other 14 instances within the time limit. PIEF-RCVF is the sec-

Table 4
Results of the tested approaches for DGGKMPT instances with $K = 5$ and $K = 6$.

K	n	EDGE					CYCLE-RCVF				PIEF-RCVF				HCF-RCVF			
		# opt	T(s)	no. var.	no. cons.	no. cuts	# opt	T(s)	no. var.	no. cons.	# opt	T(s)	no. var.	no. cons.	# opt	T(s)	no. var.	no. cons.
5	50	20	0	95	100	59	20	0	13	50	20	0	110	326	20	0	46	82
	100	17	706	690	200	9003	20	0	445	100	20	0	1279	2446	20	0	708	398
	200	0	3600	3202	400	95 206	20	2	35 505	200	20	5	21 113	20 290	20	2	20 113	2941
	400	-	-	-	-	-	20	171	1 007 642	400	20	189	253 464	129 307	20	76	325 892	18 985
	600	-	-	-	-	-	11(7)	1654	7 650 138	600	18	1594	1 152 842	358 123	20	623	1 764 656	53 045
	800	-	-	-	-	-	0(20)	-	-	-	3	3546	2 965 211	688 450	13	2735	5 351 740	99 204
	1000	-	-	-	-	-	-	-	-	-	0	3600	6 442 985	1 165 448	0(6)	3600	11 125 680	160 809
6	50	20	0	108	100	92	20	0	18	50	20	0	168	467	20	0	57	87
	100	18	903	748	200	19 741	20	0	2332	100	20	0	2949	4513	20	0	2253	550
	200	0	3600	3226	400	143 308	20	35	330 733	200	20	26	53 639	35 862	20	15	87 849	4510
	400	-	-	-	-	-	1(19)	1049	5 840 474	400	19	1035	645 935	211 977	20	680	1 759 561	28 726
	600	-	-	-	-	-	0(20)	-	-	-	1	3597	2 782 083	561 026	0(2)	3600	10 133 013	73 822
	800	-	-	-	-	-	-	-	-	-	0	3600	6 869 893	1 064 200	-	-	-	-

Table 5
Results of the tested approaches for DGGKMPT instances with $K = 7$ and $K = 8$.

K	n	EDGE					CYCLE-RCVF				PIEF-RCVF				HCF-RCVF			
		# opt	T(s)	no. var.	no. cons.	no. cuts	# opt	T(s)	no. var.	no. cons.	# opt	T(s)	no. var.	no. cons.	# opt	T(s)	no. var.	no. cons.
7	50	20	0	114	100	95	20	0	23	50	20	0	233	627	20	0	71	92
	100	18(1)	460	770	200	21 563	20	1	8225	100	20	1	4876	6578	20	0	4634	673
	200	0	3600	3226	400	178 638	17(3)	336	2 112 290	200	20	77	101 433	52 152	20	57	267 377	5618
	400	-	-	-	-	-	0(20)	-	-	-	16	2039	1 142 383	296 453	10(1)	3138	6 900 079	30 996
	600	-	-	-	-	-	-	-	-	-	0	3600	4 673 736	768 008	0(20)	-	-	-
8	50	20	0	116	100	52	20	0	41	50	20	0	317	807	20	0	95	95
	100	19(1)	79	767	200	11 908	20	6	30 453	100	20	2	7178	8480	20	1	10 843	746
	200	0	3600	3226	400	200 502	1(19)	528	3 589 926	200	19	293	157 346	68 675	20	250	1 026 008	6637
	400	-	-	-	-	-	0(20)	-	-	-	7	3368	1 681 013	382 001	0(20)	-	-	-
	600	-	-	-	-	-	-	-	-	-	0	3600	6 675 680	975 641	-	-	-	-

ond best performing algorithm for instances with $n \geq 600$ and the third best for instances with $n \leq 400$. We point out that PIEF-RCVF never ran out of memory in these experiments. EDGE struggles to solve instances with $n = 100$ or more, indicating that the approach is not competitive for low K values.

For $K = 6$, HCF-RCVF obtains again the best results as it was the only approach able to solve all instances with $n \leq 400$, and its computation times were the fastest on average. However, for $n = 600$, we observe that the model could not solve any of the 20 instances while PIEF-RCVF could solve one, indicating the start of a performance shift between the two models. We note that the number of variables grows at a lower pace with n for PIEF-RCVF than for HCF-RCVF, but the former model involves more constraints on average than the latter. For this K value also, we note that EDGE struggles to solve instances with $n = 100$ or more and that CYCLE-RCVF experiences difficulties due to memory limits.

We report in Table 5 the results obtained by the same four approaches on instances with $K = 7$ and $K = 8$.

For $K = 7$ and $K = 8$, it is interesting to observe that PIEF-RCVF now obtains the best results as it is able to solve 16 out of 20 instances with size $n = 400$ for $K = 7$ (versus 10 for HCF-RCVF) and it is the only approach able to solve instances with size $n = 400$ for $K = 8$. Even though HCF-RCVF is ahead when it comes to solving instances with size $n \leq 200$ for $K = 7$ and 8, we observe that the number of variables involved in the model increases too quickly with n compared to PIEF-RCVF, indicating that the model loses its competitive advantage for $K \geq 7$ over PIEF-RCVF. We also note that we start to see a shift in the performance of EDGE, as it becomes faster to solve instances with size $n = 100$ as K increases, which seems to indicate that when K becomes very large, EDGE might obtain the best results.

4.2. PrefLib instances

Our second set of experiments is intended to give evidence that the competitive advantage displayed by HCF over the cycle formulation and PIEF also occurs for KEP instances produced by other generators. To do so, we tested EDGE, CYCLE-RCVF, PIEF-RCVF, and HCF-RCVF on PrefLib instances and report in Section C of the Online Supplement the results obtained by the four tested approaches on instances with $K = 3, 4, 5$ and 6.

For $K = 3$, the observations made in our previous experiments still hold: CYCLE-RCVF obtained the best results, followed by HCF-RCVF, and PIEF-RCVF. For $K = 4$, we observe an interesting shift as HCF-RCVF now clearly outperforms CYCLE-RCVF. This is mainly due to the model size: as the graph density is larger, the number of variables increases at a very fast pace in CYCLE-RCVF with respect to what is observed for HCF-RCVF. PIEF-RCVF is not yet competitive as it requires almost as many variables as HCF-RCVF while also requiring five times as many constraints. For $K = 5$, PIEF-RCVF and HCF-RCVF obtain similar results: the former requires more constraints but the latter requires more variables. For $K = 6$, the number of variables in HCF-RCVF now increases too fast with the instance size and the best performance is now obtained by PIEF-RCVF. This observation was confirmed by similar experiments using $K = 7$ and $K = 8$. Overall, PrefLib instances appear to be harder for the tested models, but one should keep in mind that ad hoc strategies to solve instances with high density compatibility graphs to optimality have been proposed in the literature and are particularly effective for PrefLib instances [12].

5. Conclusions

We introduced the half-cycle formulation (HCF), an IP model to solve the KEP-WNDD that represents a cycle using two halves. After detailing a few symmetry reduction procedures, we showed that it was possible to use reduced-cost variable fixing to enhance the performance of the model. We showed through extensive computational experiments that HCF, when solved with symmetry reduction, reduced-cost variable fixing, and descending-degree node ordering, obtained better results than the cycle formulation with symmetry reduction and reduced-cost variable fixing when the cycle size limit is set to $K = 5$ or above ($K = 4$ or above for instances with a dense compatibility graph like PrefLib instances), but we also observed that it was behind the position-indexed edge formulation (PIEF) with symmetry reduction, reduced-cost variable fixing, and descending-degree node ordering when the cycle size limit is set to $K = 7$ or above ($K = 5$ or above for instances with a dense compatibility graph). A set of rules outlining which formulation to choose depending on the value of K and the density of the compatibility graph is given in Section D of the Online Supplement.

Our work focused on evaluating IP models and their performance as the instance size grows both in terms of recipient-donor pairs and in terms of cycle size limit. We point out, however, that other algorithmic procedures could also be used to enhance the performance of the tested approaches. For example, HCF could be solved within a branch-and-price framework to be more effective on instances where $K \geq 4$, but this goes beyond the scope of our study as our primary focus is on mathematical programming, which is often regarded as an accessible tool by the research community. Also, we observed that increasing K for instances with more than 200 recipient-donor pairs does not necessarily increase the maximum number of transplants (e.g., for DGGKMPT instances with 200 recipients and $K = 5$, the maximum number of transplants was 124.20 on average versus 124.25 for $K = 6$). This phenomenon is even more pronounced for instances with a dense compatibility graph, as the maximum number of transplants for every PrefLib instance with size $n \geq 128$ never increased when K was set to a value greater than or equal to 4. This indicates that for large K values, it could be worth trying to solve the instance with a smaller K (resulting in faster approaches) and assess afterwards by mean of a valid upper bound (e.g., obtained by relaxing the integrality constraints or the limitation on K) whether or not better solutions can be obtained by considering larger cycles. We did not empirically evaluate the performance of the models on KEP instances with non-directed donors, but we pointed out that every formulation could take non-directed donors into account by using the chain structure introduced for PICEF by Dickerson et al. [14]. In other words, one could introduce PICEF-HC, an algorithm that uses the efficient chain structure of PICEF together with our new HCF model to represent the cycles. Depending on the cycle size limit, it could even be relevant to introduce PICEF-PIE, a version where cycles are represented with PIEF.

For future work, it would be interesting to study whether or not HCF can handle hierarchical objectives as well as the cycle formulation, even though we expect that a few objectives such as maximising the number of cross-arcs [11] will be difficult to model with HCF. Another future research direction could focus on assessing whether or not HCF can be useful for the stochastic or robust version of the KEP [8]. Finally, it would also be worth investigating whether the “two halves instead of a whole” idea can be extended to other combinatorial optimisation problems.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments that have helped to improve the presentation of this paper.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.orl.2023.02.009.

References

[1] D.J. Abraham, A. Blum, T. Sandholm, Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges, in: Proceedings of EC'07: the 8th ACM Conference on Electronic Commerce, ACM, 2007, pp. 295–304.

[2] F. Alvelos, X. Klimentova, A. Viana, Maximizing the expected number of transplants in kidney exchange programs with branch-and-price, Ann. Oper. Res. 272 (2019) 429–444.

[3] R. Anderson, I. Ashlagi, D. Gamarnik, A.E. Roth, Finding long chains in kidney exchange using the traveling salesman problem, Proc. Natl. Acad. Sci. USA 112 (3) (2015) 663–668.

[4] D.A. Axelrod, M.A. Schnitzler, H. Xiao, W. Irish, E. Tuttle-Newhall, S.-H. Chang, B.L. Kasiske, T. Alhamad, K.L. Lentine, An economic assessment of contemporary kidney transplant practice, Am. J. Transplant. 18 (2018) 1168–1176.

[5] B. Bikbov, et al., Global, regional, and national burden of chronic kidney disease, 1990–2017: a systematic analysis for the global burden of disease study 2017, Lancet 395 (2020) 709–733.

[6] P. Biró, B. Haase-Kromwijk, T. Andersson, E.I. Ásgeirsson, T. Baltesová, I. Boletis, C. Bolotinha, G. Bond, G. Böhmig, L. Burnapp, K. Cechlárová, P. Di Ciaccio, J. Fronek, K. Hadaya, A. Hemke, C. Jacquelinet, R. Johnson, R. Kieszek, D.R. Kuypers, R. Leishman, M.-A. Macher, D. Manlove, G. Menoudakou, M. Salonen, B. Smeulders, V. Sparacino, F.C.R. Spieksma, M.O. Valentín, N. Wilson, J. van der Klundert, Building kidney exchange programmes in Europe–an overview of exchange practice and activities, Transplantation 103 (2019) 1514–1522.

[7] P. Biró, J. van de Klundert, D. Manlove, W. Pettersson, T. Andersson, L. Burnapp, P. Chromy, P. Delgado, P. Dworczak, B. Haase, A. Hemke, R. Johnson, X. Klimentova, D. Kuypers, A.N. Costa, B. Smeulders, F. Spieksma, M.O. Valentín, A. Viana, Modelling and optimisation in European kidney exchange programmes, Eur. J. Oper. Res. 291 (2020) 447–456.

[8] D. Blom, C. Hojny, B. Smeulders, A Benders-type approach for robust optimization of kidney exchanges under full recourse, Preprint, available from https://arxiv.org/abs/2105.08565. (Accessed 9 September 2021).

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

[10] M. Dell'Amico, M. Delorme, M. Iori, S. Martello, Mathematical models and decomposition methods for the multiple knapsack problem, Eur. J. Oper. Res. 274 (2019) 886–899.

[11] M. Delorme, S. García, J. Gondzio, J. Kalcsics, D. Manlove, W. Pettersson, New algorithms for hierarchical optimisation in kidney exchange programs, Oper. Res. (2023), in press. Available from https://doi.org/10.1287/opre.2022.2374. (Accessed 4 February 2023).

[12] M. Delorme, S. García, J. Gondzio, J. Kalcsics, D. Manlove, W. Pettersson, J. Trimble, Improved instance generation for kidney exchange programmes, Comput. Oper. Res. 141 (2022) 105707.

[13] M. Delorme, M. Iori, Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems, INFORMS J. Comput. 32 (2020) 101–119.

[14] J.P. Dickerson, D.F. Manlove, B. Plaut, T. Sandholm, J. Trimble, Position-indexed formulations for kidney exchange, in: Proceedings of EC'16: the 17th ACM Conference on Economics and Computation, ACM, 2016, pp. 25–42.

[15] J.P. Dickerson, A.D. Procaccia, T. Sandholm, Optimizing kidney exchange with transplant chains: theory and reality, in: Proceedings of AAMAS'12: the 11th International Conference on Autonomous Agents and Multiagent Systems, IFAAMAS, 2012, pp. 711–718.

[16] A. Hart, J.M. Smith, M.A. Skeans, S.K. Gustafson, D.E. Stewart, W.S. Cherikh, J.L. Wainright, A. Kucheryavaya, M. Woodbury, J.J. Snyder, B.L. Kasiske, A.K. Israni, OPTN/SRTR 2015 annual data report: Kidney, Am. J. Transplant. 17 (2017) 21–116.

[17] R.J. Johnson, J.E. Allen, S.V. Fuggle, J.A. Bradley, C. Rudge, Early experience of paired living kidney donation in the United Kingdom, Transplantation 86 (2008) 1672–1677.

[18] Kidney Research UK. Kidney Health Information, https://www.nhs.uk/Livewell/Kidneyhealth/Documents/kidney%20guide.pdf. (Accessed 21 September 2022).

[19] E. Lam, V. Mak-Hau, Branch-and-cut-and-price for the cardinality-constrained multi-cycle problem in kidney exchange, Comput. Oper. Res. 115 (2020) 104852.

[20] V.H. Mak-Hau, On the kidney exchange problem: cardinality constrained cycle and chain problems on directed graphs: a survey of integer programming approaches, J. Comb. Optim. 33 (2017) 35–59.

[21] J. Martinovic, M. Delorme, M. Iori, G. Scheithauer, N. Strasdat, Improved flow-based formulations for the skiving stock problem, Comput. Oper. Res. 113 (2020) 104770.

[22] D.C. McElfresh, H. Bidkhori, J.P. Dickerson, Scalable robust kidney exchange, in: Proceedings of AAAI'19: the 33rd AAAI Conference on Artificial Intelligence, AAAI Press, 2019, pp. 1077–1084.

[23] National Kidney Foundation. Dialysis, https://www.kidney.org/atoz/content/dialysisinfo. (Accessed 21 September 2022).

[24] F.T. Rapaport, The case for a living emotionally related international kidney donor exchange registry, Transplant. Proc. 18(3) (Suppl. 2) (1986) 5–9.

[25] L.C. Riascos-Álvarez, Formulations and algorithms for the kidney exchange problem, Master's thesis, Universidad Autonoma de Nuevo León, 2017. Available from https://eprints.uanl.mx/18121/2/tesisM_2017_Caro.pdf. (Accessed 21 September 2022).

[26] A.E. Roth, T. Sönmez, M.U. Ünver, Kidney exchange, Q. J. Econ. 119 (2) (2004) 457–488.

[27] A.E. Roth, T. Sönmez, M.U. Ünver, Pairwise kidney exchange, J. Econ. Theory 125 (2) (2005) 151–188.

[28] A.E. Roth, T. Sönmez, M.U. Ünver, Efficient kidney exchange: coincidence of wants in a market with compatibility-based preferences, Am. Econ. Rev. 97 (3) (2007) 828–851.

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