Operations Research

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

New Algorithms for Hierarchical Optimization in Kidney Exchange Programs

Maxence Delorme, Sergio García, Jacek Gondzio, Jörg Kalcsics, David Manlove, William Pettersson

To cite this article:
Maxence Delorme, Sergio García, Jacek Gondzio, Jörg Kalcsics, David Manlove, William Pettersson (2024) New Algorithms for Hierarchical Optimization in Kidney Exchange Programs. Operations Research 72(4):1654-1673.
https://doi.org/10.1287/opre.2022.2374

https://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 72, No. 4, July–August 2024, pp. 1654–1673
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Methods

New Algorithms for Hierarchical Optimization in Kidney Exchange Programs

Maxence Delorme,[a,*] Sergio García,[b] Jacek Gondzio,[b] Jörg Kalcsics,[b] David Manlove,[c] William Pettersson[c]

[a] Department of Econometrics and Operations Research, Tilburg University, 5037 AB Tilburg, Netherlands; [b] School of Mathematics, University of Edinburgh, Edinburgh EH8 9YL, United Kingdom; [c] School of Computing Science, University of Glasgow, Glasgow G12 8QQ, United Kingdom
*Corresponding author

Contact: m.delorme@tilburguniversity.edu, https://orcid.org/0000-0003-3730-0894 (MD); sergio.garcia-quiles@ed.ac.uk, https://orcid.org/0000-0003-4281-6916 (SG), j.gondzio@ed.ac.uk, https://orcid.org/0000-0002-6270-4666 (JG); joerg.kalcsics@ed.ac.uk, https://orcid.org/0000-0002-5013-3448 (JK); david.manlove@glasgow.ac.uk, https://orcid.org/0000-0001-6754-7308 (DM), william.pettersson@glasgow.ac.uk, https://orcid.org/0000-0003-0040-2088 (WP)

Received: July 21, 2020
Revised: April 21, 2021; February 6, 2022
Accepted: August 21, 2022
Published Online in Articles in Advance: January 25, 2023

Area of Review: Optimization

https://doi.org/10.1287/opre.2022.2374

Abstract. Many kidney exchange programs (KEPs) use integer linear programming (ILP) based on a hierarchical set of objectives to determine optimal sets of transplants. We propose innovative techniques to remove barriers in existing mathematical models, vastly reducing solution times and allowing significant increases in potential KEP pool sizes. Our techniques include two methods to avoid unnecessary variables, and a diving algorithm that reduces the need to solve multiple complex ILP models while still guaranteeing optimality of a final solution. We also show how to transition between two existing formulations (namely, the cycle formulation and the position-indexed chain-edge formulation) when optimizing successive objective functions. We use this technique to devise a new algorithm, which, among other features, intelligently exploits the different advantages of the prior two models. We demonstrate the performance of our new algorithms with extensive computational experiments modeling the UK KEP, where we show that our improvements reduce running times by three orders of magnitude compared with the cycle formulation. We also provide substantial empirical evidence that the new methodology offers equally spectacular improvements when applied to the Spanish and Dutch KEP objectives, suggesting that our approach is not just viable, but a significant performance improvement, for many KEPs worldwide.

Funding: This work was supported by the Engineering and Physical Sciences Research Council [Grants EP/P028306/1 and EP/P029825/1].

Supplemental Material: The e-companion is available at https://doi.org/10.1287/opre.2022.2374.

Keywords: kidney exchange program • hierarchical optimization • exact algorithms • objective diving • preprocessing

1. Introduction

A recent study from Bikbov et al. (2020) has estimated that, in 2017, around 697.5 million people worldwide were impacted by impaired kidney function due to chronic kidney disease (CKD), and over 1.4 million deaths in 2019 were attributable to CKD according to the website of the Global Burden of Disease Collaborative Network and Institute for Health Metrics and Evaluation (2019). Despite best efforts by researchers, no cure exists for CKD, and for a patient in the final stage of CKD (called *end-stage renal disease*), the options are limited to either dialysis or transplantation. Of these, dialysis is more expensive and offers both a worse quality of life and a worse patient life expectancy of merely five years (see National Kidney Foundation 2020 and Axelrod et al. 2018). Transplantation is therefore the better option for a patient but requires that a donor kidney be found. Donor kidneys come from either a deceased or living donor, with better outcomes for patients who receive a transplant from a living donor (Wolfe et al. 2010, Hart et al. 2017). However, to receive a kidney from a living donor, a patient, also called a *donor recipient*, or simply *recipient*, must find a willing and medically compatible donor. A *kidney exchange program* (KEP) increases the rate of living donor kidney transplantation by alleviating the requirement that recipients find a medically compatible donor (Rapaport 1986, Roth et al. 2004). In

summary, each transplant facilitated by a KEP transforms a recipient's quality of life, and it is clear that KEPs truly are life-saving initiatives.

Recipients wishing to join a KEP will pair up with one or more willing donors, who may or may not be medically compatible with the recipient. At certain intervals, a KEP will perform a *matching run*, taking all donors and recipients that have entered the system, creating a graph with the arcs representing all potential transplants, and determining a set of *exchanges* to perform. The simplest exchange is two sets of paired donors and recipients (d_1, r_1) and (d_2, r_2), where d_1 is paired with r_1 and d_2 is paired with r_2. Such an exchange is called a *two-way exchange*, and within this, donor d_1 donates a kidney to recipient r_2, and donor d_2 donates a kidney to recipient r_1. Due to their nature, exchanges such as these are called *cycles*. A cycle containing k such pairs is said to be a cycle of size k, and is also referred to as a *k-way exchange*. Some KEPs allow the participation of *nondirected donors*. These donors are willing to donate a kidney to a recipient in the KEP despite not being paired with any recipient themselves. Such a donation can then trigger further donations from a donor whose recipient has received a kidney transplant. Such a sequence of donations is called a *chain* and usually ends with a donation to a deceased-donor waiting list.

The goal of a KEP is to perform matching runs that will provide the best outcome for the donors and recipients involved. That is, in each matching run, the KEP will need to find the "best" set of exchanges, comprising cycles and chains. An obvious desirable goal for a KEP is to maximize the number of transplants, but often many solutions are found that each attain the same maximum number of transplants. Preliminary experiments showed that even an instance with only 50 recipients and 3 nondirected donors could have over 1,000 distinct solutions with the maximum number of transplants. This number goes beyond 100,000 if we consider an instance with 100 recipients and 5 nondirected donors. Recent publications show that KEPs across Europe all have slightly differing criteria for breaking ties among solutions with the maximum number of transplants (Biró et al. 2019, Biró et al. 2020). Due to the high number of potential solutions found, often this tiebreaking involves multiple criteria. These can include the number of transplants between a donor and recipient with identical blood groups, how long a given recipient has either been on dialysis or waiting in a KEP, the human leukocyte antigen (HLA) sensitization of the recipients, or logistical objectives taking into account the number of transplantation centers that are involved, for example.

Some of these criteria can also be combined in a scoring or weighting function, with each potential transplant being given a score based on some predetermined function. In general, tiebreaking between two sets of cycles and chains that have the same overall score, or lead to the same overall number of transplants, for example, is often performed hierarchically (also known as lexicographically) relative to a prioritized sequence of objectives. These objectives are presented as an ordered list of functions $(f_1, f_2, \ldots, f_n)$ that must all be optimized in turn. That is to say, to solve such a problem, a solver will find a solution $\bar{x}$ that optimizes the first objective f_1, achieving some optimum $f_1(\bar{x}) = o_1$, and then add the constraint $f_1(x) = o_1$ before moving onto the next objective f_2. This is repeated until the last objective f_n is optimized.

If only two-way exchanges are allowed in a KEP, and all optimality criteria can be combined into a single score or weight applied to each potential transplant, then Roth et al. (2005) showed that an optimal solution can be found in polynomial time by modeling the problem as a weighted matching problem. However, even if only three-way exchanges are allowed in addition to two-way exchanges, Abraham et al. (2007) proved that it is $\mathcal{NP}$-hard to even determine a set of exchanges that maximizes the number of transplants. To determine optimal sets of exchanges, we therefore use integer linear programming (ILP) techniques, which are often used in practice to solve such hard optimization problems.

The first two ILP models for determining an optimal set of exchanges were the *edge formulation* and *cycle formulation* proposed by Roth et al. (2007). Constantino et al. (2013) introduced compact ILP formulations that Dickerson et al. (2016) used to devise the *Position-Indexed Cycle-Edge Formulation* (PICEF). Other techniques for solving such problems include the branch-and-price algorithm introduced by Abraham et al. (2007) and the recent branch-and-price-and-cut algorithm proposed by Lam and Mak-Hau (2020). A survey and comparison of ILP models for maximizing the number of transplants is given in Mak-Hau (2017). Further studies have considered models that maximize the expected number of transplants for a given failure rate of arcs (see, e.g., Klimentova et al. 2016, Alvelos et al. 2019, Chisca et al. 2019a, McElfresh et al. 2019), methods of maximizing potential fallback solutions if arcs can fail (Wang et al. 2019), and models that consider a dynamic, or online, KEP—the evolution of a KEP over time (see, e.g., Dickerson et al. 2012, Das et al. 2015, Chisca et al. 2019b, Gao 2019). Other approaches have been used for finding exchanges, including parameterized complexity (Lin et al. 2019) and randomized mechanisms (Caragiannis et al. 2015, Blum et al. 2020). However, in most, if not all, such studies, the models and techniques are compared solely on the time it takes to solve for a single objective, namely, determining a set of exchanges that maximizes the number of transplants. This is a useful measure for comparing different models but does not

reflect real-world applications that consider other metrics as well (Biró et al. 2019, Biró et al. 2020).

One avenue for increasing the number of transplants arranged by KEPs is to increase the number of donor-recipient pairs in the pool. Indeed, a solution obtained by merging two or more distinct pools is guaranteed to be globally at least as good as (and very likely better than) the union of solutions obtained on each individual pool. However, current ILP formulations for KEPs struggle to solve these larger pools, either because the formulation cannot model certain aspects of the KEP (e.g., we will see later on that PICEF cannot model one of the UK objective functions), or because the formulation creates models that are too large to be tractable by ILP solvers (e.g., the cycle formulation struggles past 300 donor-recipient pairs, as shown in our experiments).

1.1. Existing Kidney Exchange Programs

Many existing European KEPs are based on finding sets of exchanges that are optimal according to hierarchical sets of objectives. This is the case for the following three KEPs in Europe that are further detailed in Biró et al. (2019) and in Biró et al. (2020), and we now describe their objectives as follows.

- The UK KEP, run by NHS Blood and Transplant (NHSBT) and called the UK Living Kidney Sharing Scheme (UKLKSS), which optimizes over the following five objectives hierarchically: (i) maximize the number of effective two-way exchanges (i.e., the number of cycles of size 2 plus the number of cycles of size 3 containing an embedded cycle of size 2), (ii) maximize the number of transplants, (iii) minimize the number of three-way exchanges, (iv) maximize the number of cross arcs (cross arcs, which we define formally in Section 2, represent a form of "fault tolerance"), and (v) maximize the sum of the scores (see also Manlove and O'Malley 2015).
- The Spanish KEP, run by Organización Nacional de Trasplantes, which optimizes over the following five objectives hierarchically: (i) maximize the number of transplants, (ii) maximize the number of distinct exchanges selected, (iii) maximize the number of cross arcs, (iv) maximize the number of highly sensitized recipients selected (these are hard to match recipients; the precise definition varies across different KEPs), and (v) maximize the sum of the scores.
- The Dutch KEP, run by Nederlandse Transplantatie Stichting, which optimizes over the following six objectives hierarchically: (i) maximize the number of transplants, (ii) maximize the number of transplants between a donor and recipient with the same blood group, (iii) prioritize the transplants to hard-to-match recipients, (iv) minimize the length of the largest cycle selected, (v) maximize the number of distinct transplant centers involved in any one cycle (in order to spread the logistical cost of an exchange over a broader region), and (vi) maximize the longest waiting time experienced by any selected recipient (in order to give preference to those recipients who have been waiting the longest).

We note that the UKLKSS currently limits chains to have at most three donors, whereas the Dutch KEP has an upper bound of 4, and the Spanish KEP has no hard upper bound. In this paper, we will extend this cap on the number of donors in a chain in the UK to four. We do this in anticipation of likely future changes to the UKLKSS that should increase the number of kidney transplants performed in the United Kingdom. In such a scenario, the UKLKSS objectives would have to be revised. Although a new set of objectives has not yet been ratified by NHSBT, for the purposes of this paper we will consider the following set of objectives that we believe could be appropriate to the setting where longer chains are permitted: (i) maximize the number of transplants, (ii) minimize the number of chains of length 4, (iii) minimize the number of three-way exchanges and chains of length 3, (iv) maximize the number of cross arcs, and (v) maximize the sum of the scores. (We anticipate that, in the presence of longer chains, a revised set of optimality objectives will not include maximizing the number of effective two-way exchanges (which is the first criterion in the current set of objectives), as this objective can reduce the maximum number of transplants that can be found thereafter.)

Some KEPs, like the United Network for Organ Sharing in the United States, only use a single objective when determining an optimal set of exchanges but do so using a complex weighting formula to take into consideration factors such as recipient ages, blood types, and waiting times (see Organ Procurement and Transplantation Network 2020).

1.2. Our Contribution

After introducing the necessary definitions for KEPs, we review two well-known ILP models from the literature in Section 2, namely, the cycle formulation and PICEF. We point out that the former model usually involves too many variables in practice, whereas the latter model cannot efficiently handle some objective functions that are optimized in real-world KEPs.

In Section 3, we demonstrate how these issues can be mitigated or avoided. We give two techniques that dramatically reduce the number of variables in the two models: (i) a cycle/chain deactivation algorithm that uses linear programming duality theory to remove unnecessary variables because they cannot belong to any optimal assignment, and (ii) a dominated chain detector that uses problem-specific information to remove variables that can only worsen the objective functions. We also introduce a diving algorithm that

alleviates the requirement to solve complex intermediary ILP models. We then show that these three techniques can all be utilized together. To the best of our knowledge, this is the first time that diving in hierarchical optimization and variable fixing based on reduced costs are used together. We also show for the first time that it is possible to transition from PICEF to the cycle formulation between the optimization of two objective functions, allowing hybridized algorithms using both PICEF and the cycle formulation (for solving the objective functions that cannot be formulated in PICEF).

Then in Section 4 we show that our approaches are up to three orders of magnitude faster than the basic cycle formulation on a large set of instances for the UKLKSS, and we show that our tools can be adapted to tackle other European KEPs such as those running in Spain and in the Netherlands. This is an important result, as (i) European KEPs are growing in size and many countries are participating in transnational (multicountry) KEPs (see Biró et al. 2020); and (ii) most European KEPs optimize their objective functions hierarchically, whereas a large number of improvements recently reported in the literature of the field have only addressed a single-objective KEP. In Section 5, some conclusions are drawn and possible future research directions are outlined.

1.3. Other Kidney Exchange Algorithms in the Literature

In this paper, we compare our new algorithms against the cycle formulation and PICEF. Among other algorithms for KEPs that have been proposed in the literature, we remark that (i) the *edge-assignment formulation* and the *extended edge formulation* proposed by Constantino et al. (2013) were shown by the authors to be computationally outperformed by the cycle formulation for realistic instances and were not able to solve any instance with 300 recipients or more; (ii) the formulation based on the *prize-collecting travelling salesman problem* introduced by Anderson et al. (2015) was empirically shown to be less effective than the extended edge formulation (Mak-Hau 2017); and (iii) the promising branch-and-cut-and-price from Lam and Mak-Hau (2020) did not consider nondirected donors. Other algorithms were tested in Dickerson et al. (2016), but, in almost all cases, PICEF turned out to be the most effective. It is worth observing that some of the previously mentioned methods work particularly well under specific conditions, such as when the cycle size limits or chain length limits are very large (e.g., the experiments in Constantino et al. 2013 were performed on instances with cycles of size up to 6, and those in Mak-Hau 2017 were performed on instances with chains of length up to 20, when limited), or when instances display a high proportion of compatibility (e.g., the experiments displayed in tables 1 and 2 from Lam and Mak-Hau 2020 showed that, in the tested instances, every donor was compatible with 25% of the recipients on average). These conditions, however, are not present in the real-world scenarios we study, where the cycle size is either limited to 3 or 4, and where the average proportion of compatibility is around 10% (based on UKLKSS data; see Delorme et al. 2022). We also mention that most of these algorithms were tailored to the single-objective KEP and cannot easily be extended to handle hierarchical-objective KEPs.

2. Background

2.1. Definitions

We will use *recipient*, denoted by r, to refer to a person who has chronic kidney disease and is waiting for a transplant, and *donor*, denoted by d, to refer to a donor willing to donate a kidney. This avoids the ambiguity of using the term *patient*, as both donors and recipients may, depending on which exchanges are selected, undergo surgery and be considered as *patients*. A donor in a KEP may either be paired with a recipient, forming a *recipient/donor pair* where the donor is only willing to donate if their paired recipient also receives a kidney, or they may be a *nondirected* donor (sometimes called an *altruistic* donor) who is willing to donate a kidney without having an identified paired recipient.

Given a recipient r and a donor d (possibly but not necessarily from the same recipient/donor pair), we say that r and d are *compatible* if the donation of a kidney from d to recipient r is deemed medically viable. Otherwise, we say that r and d are *incompatible*. Incompatibilities may arise from blood typing, tissue typing, or any of a number of other reasons as determined by clinicians.

We represent compatibilities by a *compatibility graph*, a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ whose vertex set $\mathcal{V}$ contains one vertex for each nondirected donor, one vertex for each recipient, and a dummy *sink* vertex S, which represents a donation being made to a deceased-donor waiting list, although it can also represent a *bridge* donation where the donor may act as a nondirected donor in the next matching run. By associating only recipients to vertices, we allow for scenarios in which a recipient is paired with multiple donors. This can occur as different donors may be compatible with a given recipient, and so pairing with multiple donors may improve a recipient's chance of being matched.

The arc set $\mathcal{A}$ of $\mathcal{G}$ contains all the arcs (u, v) where a donor corresponding to u (either the nondirected donor u, or a paired donor of the recipient u) is compatible with the recipient v. For each u ($u \in \mathcal{V} \setminus \{S\}$), as every donor corresponding to u can potentially trigger a bridge donation or a donation to a deceased-donor

waiting list, $\mathcal{A}$ also contains the arc (u, S). Where a recipient r_1 has two (or more) donors, both of whom are compatible with a common recipient r_2, we introduce parallel arcs. In certain scenarios, each arc may also be given a score or weight that is associated with the corresponding transplantation.

Given this definition of a compatibility graph, for any vertex v representing recipient r, we will use *recipient* v to refer to r, and *donor* v to refer to any of the donors paired with recipient r. Note that any ambiguity in selecting the correct donor can be ascertained if an arc leaving v is selected, or is irrelevant if no arc leaving v is selected.

A feasible *matching* M is a subgraph of $\mathcal{G}$ in which each nondirected donor has no incoming arc and one outgoing arc, and each recipient has exactly one outgoing and one incoming arc. A matching can be decomposed into connected subgraphs called *exchanges*. These exchanges come in two particular types—cycles and chains—which we define now. A *cycle* is a subgraph of M that contains recipients $r_1, r_2, \ldots, r_k$, for some $k \geq 2$, such that there is an arc from r_i to r_{i+1} for $i \in \{1, 2, \ldots, k-1\}$, as well as one arc from r_k to r_1. Such a cycle has length k and is denoted by $[r_1, r_2, \ldots, r_k]$. Upper bounds on the lengths of cycles in KEPs are common, and, to maintain the integrity of a KEP, all transplants relating to a given cycle should be performed simultaneously. A *chain* is a directed path in M starting at a nondirected donor d, containing a further k recipients $r_1, r_2, \ldots, r_k$, for some $k \geq 0$, and terminating at S. Such a chain is said to have length $k + 1$ (i.e., we count the number of arcs when determining the length of a chain) and counts as $k + 1$ transplants; it is denoted by $d \to r_1 \to r_2 \to \cdots \to r_k \to S$. Note that chains of length 1 contain a single arc from a nondirected donor directly to S. Whereas it would seem that such a chain (corresponding to a transplantation from a nondirected donor to a recipient on a deceased-donor waiting list) is not strictly related to a KEP, these chains are included in the modeling to allow a fair comparison between alternative solutions (e.g., between one solution comprising a chain of length 3 and another solution comprising a chain of length 1 and a two-way exchange). This will be particularly important for the cycle formulation that we introduce in Section 2.2.

Unlike cycles, chains can be performed nonsimultaneously while still ensuring that each donor does not donate a kidney until after their paired recipient has received a kidney, so limits on their lengths tend to be more relaxed. Limits to chain lengths are still often enforced, as the dynamic nature of a KEP means that a recipient may be better off waiting for a later matching run rather than being toward the end of a very long chain.

One important feature of the UKLKSS is the concept of a *cross arc*. Given an exchange consisting of a set of vertices $\widetilde{V}$ and a set of arcs $\widetilde{A}$, a *cross arc* is any arc that is between two vertices of $\widetilde{V}$ but is not in $\widetilde{A}$. Such an arc adds robustness to the exchange. If part of the exchange fails, either because a donor or recipient was unable or unwilling to proceed with the procedure, or an identified transplant was subsequently determined to be incompatible, then a cross arc may allow part of this exchange to continue. Such fallback options are useful, as they are easy to detect, do not require any reoptimization, and also guarantee that any recipients who are not in the failed part of an exchange are still matched. This is important for the well-being of both donors and recipients, as offering and then withdrawing a donation is traumatic for people in such situations. We detail in Figure 1 all the possible cross arcs in cycles of size 3 and chains of length 3 and 4. Gray nodes represent nondirected donors, and white nodes are used for recipient/donor pairs.

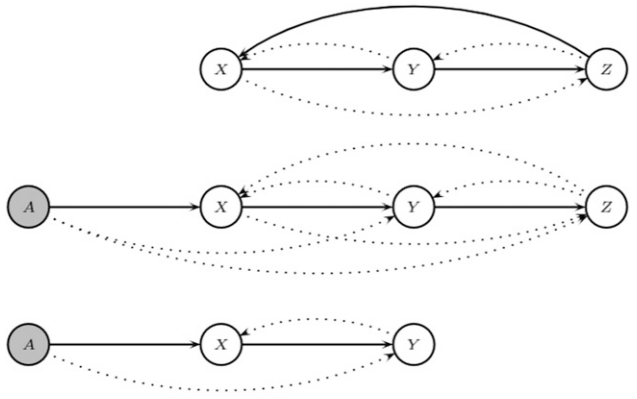
Let us consider for example that chain $A \to X \to Y$ was selected in a matching, and say that recipient/donor pair X was unable to participate in the exchange because the health of the donor would not allow him or her to go through a surgical intervention. Then, cross arc $A \to Y$ would still allow recipient/donor pair Y to be matched.

We introduce in Example EC.1 of the e-companion a small-size KEP instance that is used throughout the paper to describe the behaviors of each of our algorithms.

2.2. Cycle Formulation

The cycle formulation for KEPs was proposed by Roth et al. (2007). Even though the name refers explicitly to "cycles," the formulation can handle chains as well. In the rest of the paper, we use the original name *cycle formulation* when referring to the model, and we use *cycles/chains* when we refer to a generic variable of the model that can be either a cycle or a chain. If one of the techniques that we propose is specific to one of the

Figure 1. Example of Cross Arcs (Dotted Lined) in Cycles of Size 3 and Chains of Length 3 and 4



two structures, then we clearly identify it by only using the term *chain* or *cycle*.

In the cycle formulation, a list of every feasible cycle/chain needs to be found beforehand. As our approach is tailored primarily to the UK KEP, we assume that every feasible cycle has size at most 3 (i.e., it involves at most three recipient/donor pairs), and that every feasible chain has length at most 4 (i.e., it involves exactly one nondirected donor and at most three recipient/donor pairs). Let us consider the following notation:

- $\mathcal{N}$ and $\mathcal{R}$ are the set of nondirected donors and recipient/donor pairs, respectively.
- $\mathcal{C}$ is the set of feasible cycles/chains, $\mathcal{C}_{XL}$ is the set of feasible chains of length 4, and $\mathcal{C}_L$ is the set of feasible cycles/chains of length 3.
- For a cycle/chain c, we call $B(c)$ the number of cross arcs relative to c, $S(c)$ the sum of the arc scores included in c, and $V(c)$ the set of vertices (either in $\mathcal{N}$ or $\mathcal{R}$) that belong to c.

Let us also introduce binary decision variables x_c that take value 1 if cycle/chain c is selected, and 0 otherwise ($c \in \mathcal{C}$), and let us consider the following objective functions:

$$\max \quad f_1(x) = \sum_{c\in\mathcal{C}} |V(c)|\, x_c, \tag{1}$$

$$\min \quad f_2(x) = \sum_{c\in\mathcal{C}_{XL}} x_c, \tag{2}$$

$$\min \quad f_3(x) = \sum_{c\in\mathcal{C}_L} x_c, \tag{3}$$

$$\max \quad f_4(x) = \sum_{c\in\mathcal{C}} B(c)\, x_c, \tag{4}$$

$$\max \quad f_5(x) = \sum_{c\in\mathcal{C}} S(c)\, x_c. \tag{5}$$

Let us also denote by KEP_{f_k} the problem of optimizing objective function f_k ($k = 1,\dots,5$), and let us define $\bar{z}_1,\dots,\bar{z}_5$ to be the optimal objective values obtained when solving $\text{KEP}_{f_1},\dots,\text{KEP}_{f_5}$, respectively. The generic cycle model (called F_k^C hereafter) to solve KEP_{f_k} is as follows:

$$(F_k^C) \quad \bar{z}_k = \min/\max \quad f_k(x) \tag{6}$$

$$\text{s.t.} \sum_{c\in\mathcal{C}: v\in V(c)} x_c \le 1, \qquad \forall v \in \mathcal{N}\cup\mathcal{R}, \tag{7}$$

$$f_{k'}(x) = \bar{z}_{k'}, \qquad \forall k' = 1,\dots,k-1, \tag{8}$$

$$x_c \in \{0,1\}, \qquad \forall c \in \mathcal{C}. \tag{9}$$

The objective functions in (1)–(5) maximize the number of transplants, minimize the number of chains of size 4 selected, minimize the number of cycles/chains of size 3 selected, maximize the number of cross arcs in the selected cycles/chains, and maximize the sum of the arc scores in the selected cycles/chains, respectively. Constraints (7) ensure that donors and recipients appear in at most one of the selected cycles/chains, and Constraints (8) make sure that optimal objective values $\bar{z}_1,\dots,\bar{z}_{k-1}$ found at previous iterations are not degraded when optimizing objective function $f_k(x)$. We report in Section EC.1.1 of the e-companion the outputs obtained by the cycle formulation with an ILP solver when applied to Example EC.1.

2.3. Position-Indexed Chain-Edge Formulation

The Position-Indexed Chain-Edge Formulation (PICEF) for KEPs was proposed by Dickerson et al. (2016). In the following, we present a different (but equivalent) description of PICEF to the one proposed by Dickerson et al. (2016). The main idea behind PICEF is to handle cycles and chains in two separate structures. Although there is still a complete enumeration of every feasible cycle, each chain is now represented by a path in a graph $\mathcal{G}'$ with $3|\mathcal{R}|+|\mathcal{N}|+1$ nodes. We differentiate each of the three copies of set $\mathcal{R}$ with an index ℓ, where $\ell = 1,2,3$. A path is initiated by a nondirected donor and ends with the dummy node S. The graph $\mathcal{G}'$ is composed of four subgraphs:

- Subgraph 1 is a copy of the compatibility graph that only includes the arcs coming from the nondirected donors. Additional arcs link each nondirected donor to the sink node S.
- Each of subgraphs 2 and 3 is a copy of the compatibility graph that only includes the arcs coming from "activated" recipient/donor pairs. An activated recipient/donor pair in subgraph 2 (respectively, 3) is a pair that has at least one incoming arc in subgraph 1 (respectively, 2). Additional arcs link each activated recipient/donor pair to the sink node S.
- Subgraph 4 only contains arcs that link each activated pair to the sink node S.

We provide in Section EC.1.2 of the e-companion the graph required to model the chain structure in PICEF for Example EC.1.

Let us consider the following additional notation:

- $\mathcal{C}'$ is the set of feasible cycles and $\mathcal{C}'_L$ is the set of feasible cycles of size 3.
- $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$ is the graph structure required to model the chains in PICEF.
- Vertex set $\mathcal{V}'$ contains the $|\mathcal{N}|$ nondirected donors, three copies of the recipients, and the sink node S to model the end of the chain.
- $\mathcal{A}'$ is the set of arcs. We call $\delta^+(v)$ (respectively, $\delta^-(v)$) the subset of arcs emanating from (respectively, entering) vertex v.
- $\mathcal{A}'_m$ contains all the arcs emanating from m—in the first subgraph if m is an altruistic donor—in the last three subgraphs, if m is a recipient/donor pair. In other words, $\mathcal{A}'_m = \{\delta^+(v_1) \cup \delta^+(v_2) \cup \delta^+(v_3)\}$, where v_1, v_2, and v_3 are the first, second, and third copy of recipient m, respectively.

By introducing a binary variable y_{uv} taking value 1 if arc (u, v) is selected and 0 otherwise, KEP_{f_1} can be

modeled as follows with PICEF:

$$(F_1^p) \quad \max \quad f_1(x,y) = \sum_{c\in\mathcal{C}'} |V(c)|\, x_c + \sum_{(u,v)\in\mathcal{A}'} y_{uv} \tag{10}$$

$$\text{s.t.} \sum_{c\in\mathcal{C}':m\in V(c)} x_c + \sum_{(u,v)\in\mathcal{A}'_m} y_{uv} \le 1, \quad \forall m \in \mathcal{N}\cup\mathcal{R}, \tag{11}$$

$$\sum_{(v,w)\in\delta^+(v)} y_{vw} - \sum_{(u,v)\in\delta^-(v)} y_{uv} = \begin{cases} 1 & \text{if } v\in\mathcal{N}, \\ -|\mathcal{N}| & \text{if } v = S, \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

$$x_c \in \{0,1\}, \qquad \forall c \in \mathcal{C}', \tag{13}$$

$$y_{uv} \in \{0,1\}, \qquad \forall (u,v) \in \mathcal{A}'. \tag{14}$$

The objective function in (10) maximizes the number of transplants, whereas Constraints (11) make sure that nondirected donors do not initiate more than one chain, and that recipients appear in at most one of the selected cycles or chains. The constraints in (12) are the flow conservation constraints. The objective functions $f_2(x,y)$,$f_3(x,y)$, and $f_5(x,y)$, for KEP_{f_2}, KEP_{f_3}, and KEP_{f_5}, respectively, could also easily be handled by PICEF:

- If we denote by $\mathcal{A}'_{XL}$ the set of arcs in the fourth subgraph, then the second objective function in PICEF becomes $\min f_2(x,y) = \sum_{(u,v)\in\mathcal{A}'_{XL}} y_{uv}$.
- If we denote by $\mathcal{A}'_L$ the set of arcs in the third subgraph entering node S, then the third objective function in PICEF becomes $\min f_3(x,y) = \sum_{c\in\mathcal{C}'_L} x_c + \sum_{(u,v)\in\mathcal{A}'_L} y_{uv}$.
- If we denote by $S(u,v)$ the score of arc (u,v), and if that score is set to 0 for the arcs entering node S, then the fifth objective function in PICEF becomes $\max f_5(x,y) = \sum_{c\in\mathcal{C}'} S(c)\, x_c + \sum_{(u,v)\in\mathcal{A}'} S(u,v)\, y_{uv}$.

Modeling $f_4(x,y)$ with PICEF is more challenging, as there is no trivial way to count the number of cross arcs in a chain. One possibility is to generate a different graph structure for each nondirected donor, so that we can keep track of all the recipients included in a given chain and, thus, determine the number of cross arcs. However, preliminary experiments showed that such an approach is not competitive, as it dramatically increases the number of variables and constraints involved in PICEF and, by extension, the time required to solve the model to optimality.

3. New Algorithms for Hierarchical Optimization in KEPs

In this section, we describe several algorithms aimed at speeding up algorithms for KEPs that utilize hierarchical objective functions, with specific reference to the UK set of objective functions $f_1,\ldots,f_5$. Since f_4 maximizes the number of cross arcs in the selected cycles and chains, and as PICEF cannot track cross arcs in chains without adding new constraints and variables, the cycle formulation is the most appropriate starting point. Thus, the first three subsections present improvements for the cycle formulation, whereas the last subsection introduces a hybrid approach using both the improved version of the cycle formulation and an improved version of PICEF.

3.1. Cycle/Chain Deactivation

The main drawback of the cycle formulation is its large number of variables: there are $O(|\mathcal{R}|^3)$ cycles when the cycle size is at most 3, and $O(|\mathcal{N}||\mathcal{R}|^3)$ chains when the chain length is at most 4. The resulting ILP models become intractable (because of their size) for the instances we aim to solve, where $|\mathcal{R}| \in [50,1,400]$ and $|\mathcal{N}| = q|\mathcal{R}|$ with $q \in [0.01,0.20]$. After introducing the necessary theory, we describe our new cycle/chain deactivation algorithm, a technique that uses the information obtained after solving the continuous relaxation of the cycle formulation to set the value of some cycles/chains variables to 0.

3.1.1. Theoretical Foundations.

We recall first a useful property of linear programming (LP) that exploits an optimal solution of an LP problem to deliver a bound on the objective value of another closely related LP problem. Consider the LP problem given below for which an optimal solution has been found, and, therefore, its optimal basic-nonbasic partition is known (see Dantzig 1963 for an introduction to linear programming):

$$\begin{aligned} \max \quad & c_B^T x_B + c_N^T x_N \\ \text{s.t.} \quad & Bx_B + Nx_N = b, \\ & x_B \ge 0,\ x_N \ge 0. \end{aligned} \tag{15}$$

The dual problem associated with (15) has the following form:

$$\begin{aligned} \min \quad & b^T y \\ \text{s.t.} \quad & B^T y + s_B = c_B, \\ & N^T y + s_N = c_N, \\ & s_B \le 0,\ s_N \le 0, \end{aligned} \tag{16}$$

where s_B and s_N are dual slack variables. We are concerned with a solution of a new LP problem in which one of the nonbasic variables in (15), namely, x_j, has been given a new nonzero lower bound, namely, $x_j \ge \delta$:

$$\begin{aligned} \max \quad & c_B^T x_B + c_N^T x_N \\ \text{s.t.} \quad & Bx_B + Nx_N = b, \\ & x_B \ge 0,\ x_N \ge 0,\ x_j \ge \delta. \end{aligned} \tag{17}$$

The following result gives a bound on the objective value of (17), assuming that the model is feasible.

Lemma 1. *Let $\hat{z}$ and $\hat{z}_\delta$ be the optimal objective values of (15) and (17), respectively, and let $\hat{y}$ and $\hat{s}$ be the corresponding dual optimal solution of (16). The following inequality holds:*

$$\hat{z}_\delta \le \hat{z} + \hat{s}_j\delta, \tag{18}$$

where $\hat{s}_j$ is the value of the reduced cost (dual slack variable) associated with nonbasic variable x_j at the optimal solution of (15).

Proof. Observe that by substituting the variable $\widetilde{x}_j = x_j - \delta$ (and keeping all remaining nonbasic variables unchanged, i.e., $\widetilde{x}_i = x_i$ for $i \neq j$), we may rearrange (17) as follows:

$$\begin{aligned} \max \quad & c_B^T x_B + c_N^T \widetilde{x}_N + c_j \delta \\ \text{s.t.} \quad & B x_B + N \widetilde{x}_N = \widetilde{b} = b - N_j \delta, \\ & x_B \geq 0, \widetilde{x}_N \geq 0, \end{aligned} \tag{19}$$

where N_j denotes column j of matrix N, that is, the column corresponding to the variable x_j that was given a new lower bound δ.

The dual problem associated with (19) has the following form:

$$\begin{aligned} \min \quad & \tilde{b}^T y + c_j \delta \\ \text{s.t.} \quad & B^T y + s_B = c_B, \\ & N^T y + s_N = c_N, \\ & s_B \leq 0, \, s_N \leq 0 \end{aligned} \tag{20}$$

and it has the same (dual) feasibility constraints as (16). From the Weak Duality Theorem, any feasible solution (y, s) of (20) provides an upper bound for the optimal objective value of (17) (and its equivalent reformulation (19)). In particular, this holds for the (dual) optimal solution $(\hat{y}, \hat{s})$ of (16), which remains dual feasible for (20). We thus arrive at the following inequality:

$$\hat{z}_\delta \leq \tilde{b}^T \hat{y} + c_j \delta. \tag{21}$$

Substituting $\widetilde{b} = b - N_j \delta$ and using strong duality for the pair (15) and (16), which guarantees that $\hat{z} = b^T \hat{y}$, we may rewrite Inequality (21) as follows:

$$\begin{aligned} \hat{z}_\delta &\leq (b - N_j \delta)^T \hat{y} + c_j \delta \\ &= b^T \hat{y} + (c_j - \hat{y}^T N_j) \delta \\ &= \hat{z} + \hat{s}_j \delta, \end{aligned}$$

which completes the proof. □

The following lemma shows how we can use this result to facilitate our cycle/chain deactivation algorithm.

Lemma 2. *Let $\hat{z}$ be the optimal objective value of* (15)*, let $\hat{y}$ and $\hat{s}$ be the corresponding dual optimal solution of* (16)*, and let $\check{z}_\delta$ be the optimal objective value of Problem* (17) *in which the domain of variables x_B, x_N, and x_j is reduced to nonnegative integers.*

1. *If $\check{z}_\delta$ exists, (i.e., if* (17) *has at least one integer solution), then the following inequality holds:*

$$\check{z}_\delta \leq \hat{z} + \hat{s}_j \delta. \tag{22}$$

2. *For an integer value T and a nonbasic variable x_j such that $\hat{z} + \hat{s}_j < T \leq \hat{z}$, any integer solution for* (17) *with $\delta = 1$ will have an objective value $\check{z}_{\delta=1} < T$.*

Proof. The first claim holds, as the objective value of an ILP model is always bounded by the objective value of its LP-relaxation (i.e., $\check{z}_\delta \leq \hat{z}_\delta$), and Lemma 1 proved that $\check{z}_\delta \leq \hat{z} + \hat{s}_j \delta$. Concerning the second claim, if the integer version of (17) is infeasible, then we are already done, as there are no integer solutions. Otherwise, we know from the first claim that $\check{z}_{\delta=1} \leq \hat{z} + \hat{s}_j$, and from our assumption that $\hat{z} + \hat{s}_j < T$. □

In the following, for an integer $T \leq \hat{z}$, we group in a set $\mathcal{C}_R$ every variable x_j whose reduced cost $\hat{s}_j$ is strictly lower than $T - \hat{z}$. From Lemma 2, we can then be certain that any integer solution that includes one unit of a variable from $\mathcal{C}_R$ must have objective value strictly lower than T. As a result, if we denote by $\check{z}$ the optimal objective value of Problem (15) in which the domain of the variables is reduced to nonnegative integers (which corresponds to $\check{z}_{\delta=0}$, the optimal objective value of Problem (17) where δ is set to 0), and if $\check{z}$ is equal to T, then we must have $x_j = 0 \;\; \forall j \in \mathcal{C}_R$ in any optimal solution.

We point out that variants of variable fixing techniques have been mentioned in various ILP contexts before. For example, in section 4.4 of Garfinkel and Nemhauser (1972) for fixing basic variables in branch-and-bound approach, in proposition 2.1 in Nemhauser and Wolsey (1988) and in section 10.4 of Wolsey (1998), where a similar approach was applied in the context of Lagrangian relaxation for uncapacitated facility location. In those cases, however, the variable fixing technique was integrated within a particular combinatorial optimization method, and no rigorous theoretical justification of the methodology was provided.

3.1.2. Application Using Lemma 2, our cycle/chain deactivation algorithm now looks for solutions of a given value T and deactivates (i.e., sets the associated variables to 0) all cycles/chains whose reduced cost $\hat{s}_i$ is beyond a certain threshold (i.e., below or above that threshold, depending on whether we are maximizing or minimizing the objective function). In the following, we use $L(F_k^C)$ to refer to both the continuous relaxation of F_k^C and the optimal objective value of the continuous relaxation problem, where $k = 1, \ldots, 5$, depending on the objective function $f_1, \ldots, f_5$ that we are optimizing. Recall that, for $k > 1$, model F_k^C includes the constraints $f_{k'}(x) = \bar{z}_{k'}$, where $k' = 1, \ldots, k-1$. After solving $L(F_1^C)$ with the complete set of cycles/chains, our goal is to find an integer solution of objective value $T_1 = \lfloor L(F_1^C) \rfloor$ (which is a valid upper bound because it is impossible to reach an integer solution of value $\lfloor L(F_1^C) \rfloor + 1$). Toward this aim, we gather in a set $\mathcal{C}_1$ all cycles/chains whose reduced cost is less than or equal to $T_1 - L(F_1^C) - \epsilon$ (where ϵ is set to a very small value and is used to avoid precision errors), and restrict the F_1^C model by

setting the variables associated with these cycles/chains to 0. Indeed, according to Lemma 2, selecting one or more cycles/chains in $\mathcal{C}_1$ would imply an objective value for f_1 strictly smaller than T_1. We then solve F_1^C. If no solution of value T_1 is found, then we know that it is impossible to reach an integer solution with objective value T_1, so $T_1 - 1$ becomes a valid upper bound for f_1. Thus, we decrease T_1 by one unit, update $\mathcal{C}_1$ (i.e., by removing from $\mathcal{C}_1$ any cycle/chain whose reduced cost is now greater than the new value of $T_1 - L(F_1^C) - \epsilon$), reactivate the variables associated with the cycles/chains that are no longer in $\mathcal{C}_1$, and iterate. Once a solution of objective value T_1 is found, it is guaranteed to be optimal as it matches a valid upper bound. Whereas the same approach can be used for F_5^C (provided that the scores are integers) and F_4^C, a few adaptations are required for the case of F_2^C, which is a minimization problem: we try instead to find an integer solution of objective value $T_2 = \lceil L(F_2^C) \rceil$, and we gather in $\mathcal{C}_2$ all cycles/chains whose reduced cost is greater than or equal to $T_2 - L(F_2^C) + \epsilon$. The same applies to F_3^C.

Note that such an approach can also be used to reduce the number of variables of ILP models for single-objective KEPs or for other combinatorial optimization problems such as the bin packing problem (see, e.g., Delorme and Iori 2020). It was also proven to be useful in tailored branch-and-bound algorithms (see, e.g., Balas and Christofides 1981, Carpaneto et al. 1995, Irnich et al. 2010). This technique is useful if the two following criteria are met: (i) the difference between the optimal objective value $\bar{z}_k$ and the bound derived from the continuous relaxation value $L(F_k^C)$ is small (as an ILP model needs to be solved every time the bound is updated), and (ii) the number of variables that can be deactivated because of their reduced cost is significant. Empirically, we observed that these two criteria were met when optimizing $f_1, \ldots, f_4$: in the (real-world inspired) instances we tested, $\bar{z}_1 = \lfloor L(F_1^C) \rfloor$ in 99% of the cases and 99.2% of the variables could be deactivated after optimizing f_1. As a result, we used the cycle/chain deactivation algorithm to optimize the first four objective functions, and then switched to a standard cycle formulation to optimize f_5. Preliminary tests confirmed that using the cycle/chain deactivation algorithm for f_5 was counterproductive, as $\lfloor L(F_5^C) \rfloor$ can be far away from $\bar{z}_5$, and only few cycles/chains are still activated after solving F_4^C. We remark that one could design less realistic instances in which the gap between $\bar{z}_1$ and $\lfloor L(F_1^C) \rfloor$ could also be very large (e.g., using λ exact copies of Example EC.1 with no compatibilities between nodes of different copies results in a gap of $\lambda = \lfloor L(F_1^C) \rfloor - \bar{z}_1$).

An overview of the overall cycle deactivation algorithm is presented in Algorithm 1, where $\mathrm{OPT}(F_k^C)$ denotes the optimal objective value of model F_k^C. We point out that F_k^C is a simplified notation, as the model now depends on the set of cycles/chains that were deactivated at previous steps. We save the values $\mathrm{OPT}(F_k^C)$ in a variable labeled $\ddot{z}_k$. We will prove in Theorem 1 that those are in fact the optimal objective values $\bar{z}_k$ of F_k^C without any deactivation. The outputs it obtains with an ILP solver when applied to Example EC.1 are presented in Section EC.1.3 of the e-companion.

Algorithm 1. (Cycle/Chain Deactivation Algorithm)

1: $T_1 := \lfloor L(F_1^C) \rfloor$
2: Deactivate cycles/chains $\mathcal{C}_1$ with reduced cost $\hat{s}_1 \leq T_1 - L(F_1^C) - \epsilon$ and let $\ddot{z}_1 := \mathrm{OPT}(F_1^C)$
3: **if** $\ddot{z}_1 \neq T_1$ **then** $T_1 := T_1 - 1$, reactivate cycles/chains $\mathcal{C}_1$, and go back to step 2
4: $T_2 := \lceil L(F_2^C) \rceil$
5: Deactivate cycles/chains $\mathcal{C}_2$ with reduced cost $\hat{s}_2 \geq T_2 - L(F_2^C) + \epsilon$ and let $\ddot{z}_2 := \mathrm{OPT}(F_2^C)$
6: **if** $\ddot{z}_2 \neq T_2$ **then** $T_2 := T_2 + 1$, reactivate cycles/chains $\mathcal{C}_2$, and go back to step 5
7: $T_3 := \lceil L(F_3^C) \rceil$
8: Deactivate cycles/chains $\mathcal{C}_3$ with reduced cost $\hat{s}_3 \geq T_3 - L(F_3^C) + \epsilon$ and let $\ddot{z}_3 := \mathrm{OPT}(F_3^C)$
9: **if** $\ddot{z}_3 \neq T_3$ **then** $T_3 := T_3 + 1$, reactivate cycles/chains $\mathcal{C}_3$, and go back to step 8
10: $T_4 := \lfloor L(F_4^C) \rfloor$
11: Deactivate cycles/chains $\mathcal{C}_4$ with reduced cost $\hat{s}_4 \leq T_4 - L(F_4^C) - \epsilon$ and let $\ddot{z}_4 := \mathrm{OPT}(F_4^C)$
12: **if** $\ddot{z}_4 \neq T_4$ **then** $T_4 := T_4 - 1$, reactivate cycles/chains $\mathcal{C}_4$, and go back to step 11
13: $\ddot{z}_5 := \mathrm{OPT}(F_5^C)$

We now establish the correctness of Algorithm 1.

Theorem 1. *At the termination of Algorithm 1, variable $\ddot{z}_k$ has value $\bar{z}_k$, for each $k = 1, 2, \ldots, 5$.*

Proof. First, we observe that Algorithm 1 always yields a feasible solution. Let $\bar{x}$ denote an optimal solution for the hierarchical kidney exchange problem with objective values $(\ddot{z}_1, \ddot{z}_2, \ddot{z}_3, \ddot{z}_4, \ddot{z}_5)$. Let us assume that there is an instance in which the solution found by Algorithm 1 is feasible but not optimal; that is, there exists a k such that $\ddot{z}_k \neq \bar{z}_k$. If there is more than one such k for this instance, then pick the smallest one (i.e., pick k such that $\ddot{z}_k \neq \bar{z}_k$ and $\ddot{z}_{k'} = \bar{z}_{k'}$, $k' = 1, \ldots, k-1$). This is only possible if at least one variable that was needed to reach a solution with objective value $\bar{z}_k$ was deactivated at stage k or before. Let us call that variable x_j.

Let us first assume that x_j was deactivated at stage k: according to Lemma 2, selecting one unit of any cycle/chain included in $\mathcal{C}_k$ would lead to an objective value that is strictly worse than $\ddot{z}_k$. However, since $\ddot{z}_k$ is itself strictly worse than $\bar{z}_k$, this is a contradiction.

Let us now assume that x_j was deactivated at stage k' for some $k' < k$: as k is minimal, $\bar{z}_{k'} = \ddot{z}_{k'}$. According to

Lemma 2, selecting one unit of any cycle/chain included in $\mathcal{C}_{k'}$ would prevent $f_{k'}$ from being at its optimal objective value $\bar{z}_{k'}$. Recalling that $f_{k'}(x) = \bar{z}_{k'}$ is a constraint in F_k^C, setting $x_j = 1$ would render any solution of F_k^C infeasible. This is again a contradiction. □

3.2. Diving Algorithm

After preliminary tests on Algorithm 1, we observed that steps 2 and 5 took most of the computational effort. This is not surprising, as only few cycles/chains are deactivated at these steps, and solving exactly an ILP with many variables can be time-consuming.

However, the only ILP solution that is relevant for the problem is the one solved at step 13, as it is the one that gives the set of cycles and chains that should be selected to optimize the five objective functions. The other four ILP models only give binary indications about whether a solution of objective value T_k exists, where $k = 1, 2, 3, 4$.

The idea of the diving algorithm is to first make the assumption that a solution of objective value T_k exists (where k = 1, 2) and, thus, to skip the ILP models of steps 2 and 5 necessary to obtain $\ddot{z}_1$ and $\ddot{z}_2$. The diving algorithm may backtrack and correct that assumption in case the model of step 8 is infeasible. Preliminary tests showed that it was not expedient to extend the assumption to T_3, as it involved significantly more backtracks. An overview of the diving algorithm is presented in Algorithm 2.

Algorithm 2 (Diving Algorithm)

1: $\Lambda_2 := 0, \Lambda_3 := 0$
▷ Keep track of the number of failures
2: $T_1 := \lfloor L(F_1^C) \rfloor$ ▷ Assumption 1: $\ddot{z}_1 = T_1$
3: Deactivate cycles/chains $\mathcal{C}_1$ with reduced cost $\hat{s}_1 \leq T_1 - L(F_1^C) - \epsilon$
4: $T_2 := \lceil L(F_2^C) \rceil$ ▷ Assumption 2: $\ddot{z}_2 = T_2$
5: **if** $\Lambda_2 < \bar{\Lambda}_2$ **then** ▷ If we trust Assumption 1
6: Deactivate cycles/chains $\mathcal{C}_2$ with reduced cost $\hat{s}_2 \geq T_2 - L(F_2^C) + \epsilon$
7: **else** ▷ If we doubt Assumption 1
8: $\ddot{z}_2 := \text{OPT}(F_2^C)$
▷ Solve F_2^C exactly, with no $\mathcal{C}_2$ cycle/chain deactivation
9: **if** F_2^C is infeasible **then**
▷ Assumption 1 was wrong
10: $\Lambda_2 := 0, T_1 := T_1 - 1$, reactivate cycles/chains $\mathcal{C}_1$, and go back to step 3
▷ Update Assumption 1
11: **else** ▷ Assumption 1 was right
12: Deactivate cycles/chains $\mathcal{C}_2$ with reduced cost $\hat{s}_2 \geq \ddot{z}_2 - L(F_2^C) + \epsilon$
13: **end if**
14: **end if**
15: $T_3 := \lceil L(F_3^C) \rceil$
16: **if** $\Lambda_3 < \bar{\Lambda}_3$ **then** ▷ If we trust Assumption 2
17: Deactivate cycles/chains $\mathcal{C}_3$ with reduced cost $\hat{s}_3 \geq T_3 - L(F_3^C) + \epsilon$
18: $\ddot{z}_3 := \text{OPT}(F_3^C)$
19: **if** F_3^C is infeasible **or** $\ddot{z}_3 > T_3$ **then**
20: $T_3 := T_3 + 1, \Lambda_3 := \Lambda_3 + 1$, reactivate cycles/chains $\mathcal{C}_3$, and go back to step 16
21: **end if**
22: **else** ▷ If we doubt Assumption 2
23: $\ddot{z}_3 := \text{OPT}(F_3^C)$
▷ Solve F_3^C exactly, without $\mathcal{C}_3$ cycle/chain deactivation
24: **if** F_3^C is infeasible **then**
▷ Assumption 2 was wrong
25: $\Lambda_3 := 0, \Lambda_2 := \Lambda_2 + 1, T_2 := T_2 + 1$, reactivate cycles/chains $\mathcal{C}_2$, and go back to step 5
26: **else** ▷ Assumptions 1 and 2 were right
27: Deactivate cycles/chains $\mathcal{C}_3$ with reduced cost $\hat{s}_3 \geq \ddot{z}_3 - L(F_3^C) + \epsilon$
28: **end if**
29: **end if**
30: $T_4 := \lfloor L(F_4^C) \rfloor$ ▷ Step 4 is unchanged
31: Deactivate cycles/chains with reduced cost $\hat{s}_4 \leq T_4 - L(F_4^C) - \epsilon$ and set $\ddot{z}_4 := \text{OPT}(F_4^C)$
32: **if** $\ddot{z}_4 \neq T_4$ **then** $T_4 := T_4 - 1$ and go back to step 31
33: $\ddot{z}_5 := \text{OPT}(F_5^C)$

Algorithm 2 starts by solving $L(F_1^C)$, $L(F_2^C)$, and $L(F_3^C)$, and deactivates the corresponding sets of cycles/chains $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ (steps 2–6, 15–17). Note that all three relaxations will be feasible, possibly in contrast to the respective integer problems. It then solves F_3^C and tries to find an integer solution where the first three objective function values are equal to T_1, T_2, and T_3 (step 18). If it succeeds, then we know that the three assumptions were correct. It then solves the fourth objective function, as in the cycle/chain deactivation algorithm (steps 30–32) and the fifth as in the cycle formulation (step 33). If it fails, then at least one of the three assumptions that $\ddot{z}_j = T_j$, $j \in \{1, 2, 3\}$, was wrong. The algorithm starts by trying to correct the third assumption. To that end, it records the failure in the variable Λ_3, increases T_3 (step 20), updates $\mathcal{C}_3$, and tries again (steps 17–18).

Once it has reached a given number of failures $\bar{\Lambda}_3$, the algorithm solves F_3^C without deactivating any cycle/chain in $\mathcal{C}_3$ and without any consideration on the bound T_3 (step 23). If F_3^C is feasible, then we know that the first two assumptions were correct. The algorithm deactivates again the appropriate cycles/chains in $\mathcal{C}_3$ (step 27) and moves on to the fourth objective function. If F_3^C is infeasible, then the algorithm tries to correct the second assumption. It records the failure in the variable Λ_2, increases T_2 (step 25), updates $\mathcal{C}_2$ (step 6), and solves $L(F_3^C)$ again (step 15).

Once it has reached a given number of failures $\bar{\Lambda}_2$, the algorithm solves F_2^C without deactivating any cycle/chain in $\mathcal{C}_2$ and without any consideration on the bound T_2 (step 8). If F_2^C is feasible, then the algorithm deactivates again the appropriate cycles/chains in $\mathcal{C}_2$ (step 12) and moves on to the third objective function. If F_2^C is infeasible, then the algorithm decreases T_1 (step 10), updates $\mathcal{C}_1$ (step 3), and solves $L(F_2^C)$ again (step 4). The performance of Algorithm 2 highly depends on the allowed number of failures $\bar{\Lambda}_2$ and $\bar{\Lambda}_3$. If the parameters have values that are too small, then the algorithm may have to go to steps 8 and 23 and solve ILP models with no cycle/chain deactivation when this was not necessary. If the parameters have values that are too large, then the algorithm may have to loop many times in steps 17–21 (because of $\bar{\Lambda}_3$) and in steps 6, 15–29 (because of $\bar{\Lambda}_2$) before realizing that Assumptions 1 and 2 were wrong. Even though finding the best values for $\bar{\Lambda}_2$ and $\bar{\Lambda}_3$ is itself an (instance-dependent) optimization problem, we empirically determine that, in our case, setting both parameters to 1 gave good performance outcomes. The outputs obtained by Algorithm 2 with an ILP solver when applied to Example EC.1 are presented in Section EC.1.4 of the e-companion.

We now establish the correctness of Algorithm 2.

Theorem 2. *At the termination of Algorithm 2, variable $\ddot{z}_k$ has value $\bar{z}_k$, for each $k = 1, 2, \ldots, 5$.*

Proof. First, let us observe that Algorithm 2 behaves like Algorithm 1 when optimizing objective functions f_4 and f_5. Therefore, we only need to prove the correctness of Algorithm 2 for f_1, f_2, and f_3. Let us consider the six following cases:

- $\bar{z}_1 = \lfloor L(F_1^C) \rfloor$, $\bar{z}_2 = \lceil L(F_2^C) \rceil$, $\bar{z}_3 = \lceil L(F_3^C) \rceil$. Then Algorithm 2 behaves exactly as Algorithm 1 and terminates with an optimal solution according to Theorem 1.
- $\bar{z}_1 = \lfloor L(F_1^C) \rfloor$, $\bar{z}_2 = \lceil L(F_2^C) \rceil$, $\bar{z}_3 < \lceil L(F_3^C) \rceil + \bar{\Lambda}_3$. Then Algorithm 2 optimizes objective function f_3 exactly $\bar{z}_3 - \lceil L(F_3^C) \rceil$ times before finding an optimal solution with objective value $\bar{z}_3$. The algorithm cannot terminate with a suboptimal solution as (i) if $T_3 = \bar{z}_3$, then only the cycles/chains that would prevent reaching objective value $f_3(x) = \bar{z}_3$ or fulfilling constraints $f_1(x) = \bar{z}_1$ or $f_2(x) = \bar{z}_2$ are deactivated; and (ii) if $T_3 < \bar{z}_3$, then it is impossible to reach a solution with objective value $f_3(x) = T_3$, so the algorithm has to update T_3 until it is equal to $\bar{z}_3$.
- $\bar{z}_1 = \lfloor L(F_1^C) \rfloor$, $\bar{z}_2 = \lceil L(F_2^C) \rceil$, $\bar{z}_3 \geq \lceil L(F_3^C) \rceil + \bar{\Lambda}_3$. Then Algorithm 2 optimizes objective function f_3 exactly $\bar{\Lambda}_3 + 1$ times before finding an optimal solution with objective value $\bar{z}_3$ ($\bar{\Lambda}_3$ unsuccessful attempts with deactivated cycles/chains $\mathcal{C}_3$ and one attempt without any deactivated cycles/chains in the set). The algorithm cannot terminate with a suboptimal solution as (i) if $T_3 < \bar{z}_3$, then it is impossible to reach a solution with objective value $f_3(x) = T_3$; and (ii) if there is no consideration on the bound T_3, then only the cycles/chains that would prevent fulfilling constraints $f_1(x) = \bar{z}_1$ or $f_2(x) = \bar{z}_2$ are deactivated.
- $\bar{z}_1 = \lfloor L(F_1^C) \rfloor$ and $\bar{z}_2 < \lceil L(F_2^C) \rceil + \bar{\Lambda}_2$. Then Algorithm 2 updates T_2 exactly $\bar{z}_2 - \lceil L(F_2^C) \rceil$ times before reaching optimal objective value $\bar{z}_2$. Each of these updates requires optimizing objective function f_3 exactly $\bar{\Lambda}_3 + 1$ times to find out that no solution with objective value $T_2 < \bar{z}_2$ exists. The algorithm cannot terminate with a suboptimal solution as (i) if $T_2 = \bar{z}_2$, then only the cycles/chains that would prevent fulfilling constraints $f_1(x) = \bar{z}_1$ or $f_2(x) = \bar{z}_2$ are deactivated; and (ii) if $T_2 < \bar{z}_2$, then it is impossible to reach a solution satisfying constraint $f_2(x) = T_2$, so the algorithm has to update T_2 until it is equal to $\bar{z}_2$.
- $\bar{z}_1 = \lfloor L(F_1^C) \rfloor$ and $\bar{z}_2 \geq \lceil L(F_2^C) \rceil + \bar{\Lambda}_2$. Then Algorithm 2 updates T_2 exactly $\bar{\Lambda}_2$ times (each of these updates requiring $\bar{\Lambda}_3 + 1$ optimizations of objective function f_3) before optimizing f_2 without making any assumptions on the bound T_2 to eventually find a solution with objective value $f_2(x) = \bar{z}_2$. The algorithm cannot terminate with a suboptimal solution as (i) if $T_2 < \bar{z}_2$, then it is impossible to reach a solution satisfying $f_2(x) = T_2$; and (ii) if there is no consideration on the bound T_2, then only the cycles/chains that would prevent fulfilling constraints $f_1(x) = \bar{z}_1$ are deactivated.
- $\bar{z}_1 < \lfloor L(F_1^C) \rfloor$. Then Algorithm 2 updates T_1 exactly $\lfloor L(F_1^C) \rfloor - \bar{z}_1$ times before reaching optimal objective value $\bar{z}_1$. Each of these updates requires optimizing objective function f_3 exactly $\bar{\Lambda}_2 \times (\bar{\Lambda}_3 + 1)$ times and objective function f_2 once with no consideration on T_2 to find out that no solution with objective value $T_1 > \bar{z}_1$ exists. The algorithm cannot terminate with a suboptimal solution as (i) if $T_1 = \bar{z}_1$, then only the cycles/chains that would prevent fulfilling constraint $f_1(x) = \bar{z}_1$ are deactivated; and (ii) if $T_1 > \bar{z}_1$, then it is impossible to reach a solution satisfying $f_1(x) = T_1$, so the algorithm has to update T_1 until it is equal to $\bar{z}_1$. □

3.3. Dominated Chains

Let us consider a chain c. If there exists a combination s of one chain and one cycle involving the same pairs and nondirected donor as the ones involved in c that improves objective function f_2 (respectively, f_3) while preserving f_1 (respectively, f_1 and f_2) with respect to c, then we call c a *dominated* chain and s a *dominating* set. We detail in Section EC.1.5 of the e-companion four generic cases of dominated chains, their respective dominating set, and the dominated chains of Example EC.1. Variables associated with dominated chains can be removed from the model, as any feasible solution containing a dominated chain can be replaced by another (better) feasible solution where the dominated chain is replaced by its dominating set.

Note that the concept of dominated chains could also be extended to take the cycles into account in order to detect some sets of dominated cycles. For example, three cycles of length 2 of the form $[U,V],[W,X]$, and $[Y,Z]$ should always be selected over two cycles of length 3 of the form $[U,V,W]$ and $[X,Y,Z]$. However, the detection of such dominated sets of cycles is not trivial, and while removing a dominated chain fixes a variable to 0, removing a dominated set of cycles requires an additional constraint, which goes against our model size reduction paradigm.

3.4. Hybrid Algorithm

So far, we have used the cycle formulation because objective function f_4 maximizes the number of cross arcs in the selected cycles and chains, and no easy modification of PICEF would allow us to count the number of cross arcs in the chain structure. In the hybrid algorithm, we propose the use of PICEF for the first three objective functions f_1,f_2, and f_3, and transition to the cycle formulation for the two last objective functions f_4 (with cycle/chain deactivation) and f_5. Note that returning to PICEF when optimizing f_5 is not an option since we now have a constraint on the number of cross arcs that needs to be in the solution.

The cycle/chain deactivation algorithm can be extended to PICEF and is now called the *cycle/arc deactivation algorithm*, as chains are represented in a graph structure in PICEF. The diving algorithm can also be used with PICEF following Algorithm 2 (until step 29). Forbidding dominated chains in PICEF is more challenging than it is in the cycle formulation. Indeed, removing a dominated chain in the cycle formulation simply sets its corresponding decision variable to 0. In PICEF, we need to add a constraint for each dominated chain to force the sum of the decision variables associated with each of its arcs to be less than or equal to the length of the chain minus one. Let us recall that since each chain is uniquely defined by a path from a nondirected donor to the sink node S, forbidding chain $A \to B_1 \to E_2 \to S$ does not forbid $A \to B_1 \to E_2 \to F_3 \to S$, for example. Preliminary tests indicated that adding such constraints dramatically increases the time to solve PICEF, so dominated chains were not forbidden in our PICEF implementation.

Regarding the transition between f_3 and f_4, we need to do a complete graph exploration of the PICEF chain structure and transform every feasible path in PICEF into a chain in the cycle formulation, provided that the resulting chain is not dominated. The outputs obtained by the extension of Algorithm 2 to PICEF (until line 29) with an ILP solver when applied to Example EC.1 are detailed in Section EC.1.6 of the e-companion.

4. Experimental Results

We report in this section the outcome of extensive computational experiments aimed at testing the effectiveness of our new algorithms for the UKLKSS. We also show that our methods can be adapted to take into account the objective functions of other countries that employ hierarchical optimization (see Biró et al. 2019), as illustrated by the application of our approach to the Spanish and Dutch KEPs.

All our algorithms were coded in C++ and can be downloaded from https://github.com/mdelorme2/Hierarchical_Optimisation_Kidney_Exchange_Programmes_Codes. The experiments were run on an Intel Xeon E5-2680W v3, with 2.50 gigahertz and 192 gigabytes of memory, running under Scientific Linux 7.5, and Gurobi 7.5.2 was used to solve the ILP models. Parameter ϵ was set to 0.001, and parameters $\bar{\Lambda}_2$ and $\bar{\Lambda}_3$ were set to 1. Each instance was run using a single core and no time limit was imposed, unless specified otherwise. We used the following parameters for Gurobi:

- `Method = 2`, meaning that the barrier algorithm was used to solve both the LP models and the root nodes of the ILP models. Preliminary tests showed that using the barrier algorithm was faster than letting Gurobi choose the algorithm (`Method = -1`).
- `MIPGap = 0` for the ILP models, meaning that the solver terminates with an optimal solution if the gap between the lower bound and the upper bound is equal to 0.
- `Crossover = 0` for the LP models, meaning that the solver does not try to transform the interior solution produced by the barrier algorithm into a basic solution. Our approaches only need the LP optimal objective values and the reduced costs associated with each variable, not a basic solution. Disabling crossover saved a significant amount of time in our tests.

4.1. Instance Generation

In order to generate instances similar to real-world cases, we used the data generator written by Trimble (2020). The generator is derived from the work of Saidman et al. (2006) and was used previously in the literature by Klimentova et al. (2016) and Blum et al. (2020). The generator can be used to replicate any KEP pool provided that the user has an accurate estimation of the following key population parameters:

- donor blood types: proportion of donors in each blood group {O, A, B, AB};
- recipient blood types: proportion of recipients in each blood group {O, A, B, AB};
- donors per recipient: proportion of recipients who have 1, 2, 3, or 4 donors; and
- calculated reaction frequency (or cPRA): the proportion of donors who would not be tissue-type compatible with a given recipient.

We used a large set of data provided by the UKLKSS to create instances with a similar distribution to those solved in their quarterly runs. Note that the aforementioned generator also used "Additional fields for compatibility with Saidman generator," which we left untouched since the information for these fields were not available in our data, and we added an inner routine to attribute a score to each compatibility following the score distribution observed in the UKLKSS data.

In addition to the above key population parameters, the generator also requires some information about the number of recipients and the proportion of altruistic donors. In order to have a complete overview of the performance of our algorithms and their scalability, we generated instances with a large range of number of recipients ($|\mathcal{R}| \in \{50, 100, 200, 400, 600, 800, 1,000, 1,200, 1,400\}$) and a few different proportions of nondirected donors ($q \in \{0.01, 0.05, 0.10, 0.15, 0.20\}$, where $|\mathcal{N}| = q|\mathcal{R}|$). For each pair $(|\mathcal{R}|, q)$, we generated 30 instances resulting in $9 \times 5 \times 30 = 1,350$ instances in total. All the instances can be downloaded from the online repository http://researchdata.gla.ac.uk/id/eprint/1016.

4.2. Randomly Generated Instances with the UKLKSS Objectives

We first tested each of our new approaches on a subset of the randomly generated instances and we compared their results with those obtained by the cycle formulation, which is currently in use by the UKLKSS. The subset contains 450 instances with a number of recipients $|\mathcal{R}| \in \{50, 100, 200, 400, 600\}$ and a proportion of nondirected donors $q \in \{0.01, 0.05, 0.10\}$. These values were chosen so that each algorithm could solve all the instances to optimality in a reasonable time. We then tested the hybrid algorithm on the remaining 900 instances to show its scalability. When not specified otherwise, we display the results by number of recipients (i.e., instances with different values of q are merged).

Table 1 contains the results of the five algorithms for instances with 50 and 100 recipients, which are comparable in size to medium-sized European KEPs such as the Spanish KEP (around 110 recipients per run; see Biró et al. 2019) or the Dutch KEP (around 40 recipients per run). The "Algorithm" column specifies the algorithm used, the attribute "− DC" indicates that dominated chains were removed from the model (for all the objective functions for the diving algorithm and for f_4 and f_5 for the hybrid algorithm), columns "TT" give the average total time required by the algorithm to solve an instance, and columns "Tk" indicate the time required to optimize objective function f_k, where $k = 1, \ldots, 5$. The last line of the table reports the optimal value of each objective function $f_1, \ldots, f_5$ averaged over the 90 instances. As no time limit was imposed, all the algorithms solved the instances to optimality. Note that (i) "TT" also incorporates the time to generate the variables of the model and, thus, might be a few seconds away from the sum of the "Tk" for large size instances; (ii) the time spent by the hybrid algorithm to perform the transition between PICEF and the cycle formulation is included in T4; and (iii) for algorithms including a diving component, the time displayed in T2 only incorporates the time spent optimizing f_2 once the optimal value of f_1 was found. The time spent optimizing f_2 for wrong values of f_1 is incorporated in T1. The same comments apply for the time displayed in T3.

The table shows that instances with up to 100 recipients can be solved to optimality with the cycle formulation in less than one second on average. Even though our new approaches are faster (on average 0.06 second [s] for the hybrid algorithm vs 0.81 s for the cycle formulation for instances with 100 recipients), the time saved is not enough to motivate a switch from the cycle formulation for such instances. Table 2 contains the same information for instances with 200 and 400 recipients, which is of comparable size to a large European KEP such as the UKLKSS (around 300 recipients per matching run).

The table shows a number of interesting facts:

- Whereas the cycle formulation can solve to optimality instances with 200 recipients in 26 seconds on average, it takes almost 15 minutes on average to solve instances with 400 recipients.
- The cycle formulation spends a significant amount of time optimizing each objective function, with f_3 and f_5 being the longest to optimize.

Table 1. Average Time (in Seconds) Required by the Algorithms to Solve Instances with 50 and 100 Recipients

	50 recipients, $q = \{0.01, 0.05, 0.10\}$						100 recipients, $q = \{0.01, 0.05, 0.10\}$					
Algorithm	**TT**	T1	T2	T3	T4	T5	**TT**	T1	T2	T3	T4	T5
Cycle	**0.07**	0.01	0.02	0.02	0.01	0.02	**0.81**	0.09	0.15	0.15	0.17	0.25
Cycle/chain deactivation	**0.04**	0.01	0.01	0.01	0.01	0.01	**0.11**	0.06	0.02	0.01	0.01	0.01
Diving algorithm	**0.03**	0.00	0.00	0.01	0.01	0.01	**0.08**	0.04	0.01	0.01	0.01	0.01
Diving algorithm − DC	**0.03**	0.00	0.00	0.01	0.01	0.01	**0.06**	0.02	0.01	0.01	0.01	0.01
Hybrid algorithm − DC	**0.03**	0.00	0.00	0.01	0.01	0.00	**0.06**	0.01	0.01	0.01	0.02	0.01
Optimal objective values	—	18.63	0.94	2.94	3.77	1407	—	47.19	2.39	8.73	9.6	3852

Table 2. Average Time (in Seconds) Required by the Algorithms to Solve Instances with 200 and 400 Recipients

	200 recipients (90 instances)						400 recipients (90 instances)					
Algorithm	**TT**	T1	T2	T3	T4	T5	**TT**	T1	T2	T3	T4	T5
Cycle	**25.58**	2.54	4.82	5.34	4.35	8.49	**866.88**	60.39	191.05	227.79	125.64	261.44
Cycle/chain deactivation	**1.29**	1.08	0.11	0.03	0.02	0.01	**34.83**	31.37	2.40	0.24	0.16	0.07
Diving algorithm	**1.20**	1.00	0.09	0.03	0.02	0.02	**32.95**	28.42	2.64	0.29	0.14	0.08
Diving algorithm – DC	**0.44**	0.31	0.04	0.03	0.02	0.01	**9.86**	7.79	1.05	0.25	0.12	0.07
Hybrid algorithm – DC	**0.20**	0.06	0.02	0.03	0.06	0.02	**1.18**	0.45	0.14	0.17	0.28	0.09
Optimal objective values	—	122.94	5.48	26.06	25.06	10 235	—	310.49	12.82	72.46	58.88	25 686

• All our new approaches spend the majority of their time optimizing f_1; the other objective functions are optimized almost instantly.

• Removing the dominated chains has a significant impact on the model performances: it takes 32.95 s on average to solve an instance with 400 recipients by the diving algorithm if we allow dominated chains, whereas it only takes 9.86 s if we remove them. As additional information, we also mention that on average, for instances with 400 recipients, there are 12,469 dominated chains of length 3 (out of 46,021) and 970,050 dominated chains of length 4 (out of 1,468,336).

• The hybrid algorithm is by far the fastest among all the tested algorithms and obtains outstanding results. It solves instances with 400 recipients three orders of magnitude faster, on average, than the cycle formulation. Its PICEF component saves a significant amount of time while optimizing f_1, f_2, and f_3 with respect to the other methods. The time spent during the transition between PICEF and the cycle formulation is noticeable (+0.16 s on average to optimize f_4 for instances with 400 recipients with respect to the diving algorithm without dominated chains) but irrelevant when compared with the time saved optimizing the first three objective functions.

Table 3 gives detailed information about the models for instances with 200 and 400 recipients. In particular, it gives the number of variables, constraints, and nonzero elements in the last ILP model (i.e., when optimizing f_5) in columns "Nb. var.," "Nb. cons.," and "Nb. nzs.," respectively. It also indicates in columns "Nb. Fk" the average number of times the bound T_k ($k = 1, \ldots, 4$) had to be updated, meaning that a backtracking step was necessary. We remind the reader that there is no backtracking in the cycle formulation.

We observe the following:

• Our new approaches have significantly fewer variables and nonzero elements when optimizing the last objective function f_5, indicating that the cycle/chain deactivation is extremely effective.

• The number of backtracking steps is relatively small for objective functions f_1 and f_2, supporting the hypothesis that the diving algorithm can reasonably assume that the bounds obtained after solving the linear relaxation of the first two objective functions are accurate.

• The number of variables when optimizing the last objective function f_5 is similar when comparing the hybrid algorithm and the diving algorithm without dominated chains. Out of 180 instances with 200 and 400 recipients, we counted respectively 164, 8, and 8 instances in which the hybrid algorithm had the same, larger, or smaller number of variables as the diving algorithm without dominating chains.

Table 4 contains the results of the five algorithms for instances with 600 recipients, which could be realistic size instances if half a dozen European countries were merging their pools.

Not surprisingly, the table shows that the cycle formulation is much slower than our other methods, since it required more than four hours on average to solve an instance with 600 recipients and 60 nondirected donors, whereas it only took our hybrid algorithm 4.9 seconds on average to solve the same instances. It is also interesting to observe that the proportion of nondirected donors q has a dramatic impact

Table 3. Detailed Information on the Algorithms for Instances with 200 and 400 Recipients

	200 recipients (90 instances)							400 recipients (90 instances)						
Algorithm	Nb. var.	Nb. cons.	Nb. nzs.	Nb. F1	Nb. F2	Nb. F3	Nb. F4	Nb. var.	Nb. cons.	Nb. nzs.	Nb. F1	Nb. F2	Nb. F3	Nb. F4
Cycle	90,861	174	623,669	—	—	—	—	1,526,232	400	10,571,785	—	—	—	—
Cycle/chain deactivation	319	143	1,892	0	0.01	0.08	0	958	338	5,636	0.02	0.03	0.26	0.24
Diving algorithm	319	143	1,892	0	0.01	0.10	0	958	338	5,636	0.02	0.03	0.26	0.24
Diving algorithm – DC	319	143	1,892	0	0.01	0.10	0	955	338	5,615	0.02	0.03	0.26	0.24
Hybrid algorithm – DC	318	143	1,887	0	0.01	0.10	0	939	338	5,523	0.02	0.03	0.26	0.24

Table 4. Average Time (in Seconds) Required by Each of the Proposed Algorithms to Solve Instances with 600 Recipients for Each Value of q

	$q = 0.01$ (30 instances)						$q = 0.05$ (30 instances)						$q = 0.10$ (30 instances)					
Algorithm	**TT**	T1	T2	T3	T4	T5	**TT**	T1	T2	T3	T4	T5	**TT**	T1	T2	T3	T4	T5
Cycle	**576**	44	109	150	88	184	**7,328**	343	1,494	1,927	1,611	1,951	**14,805**	907	4,550	2,951	2,375	4,016
Cycle/chain deactivation	**24.1**	22.8	0.4	0.2	0.1	0.1	**196.8**	157.5	24.6	4.5	5.7	2.1	**486.2**	424.7	54.4	1.7	0.5	0.2
Diving algorithm	**21.7**	20.4	0.1	0.2	0.1	0.1	**163.3**	107.9	42.7	5.7	2.4	2.0	**280.9**	223.2	50.3	1.6	0.5	0.2
Diving algorithm − DC	**8.4**	7.4	0.1	0.2	0.1	0.1	**68.7**	40.2	17.3	4.5	2.6	2.3	**103.3**	73.7	23.7	1.4	0.3	0.2
Hybrid algorithm − DC	**2.0**	1.0	0.1	0.2	0.5	0.1	**11.0**	2.1	2.7	1.8	2.7	1.7	**4.9**	1.9	1.2	0.6	0.8	0.2

on the average time required to solve an instance for most approaches (the average times for $q = 0.10$ are between 10 and 30 times larger than the average times for $q = 0.01$), with the exception of the hybrid algorithm, which seems to scale well as q increases. This can be explained by the fact that $O(|\mathcal{R}|^3)$ new chains (and thus, new variables) are created in the cycle formulation for each new altruistic donor, whereas only $O(|\mathcal{R}|)$ new arcs are created in PICEF for each new altruistic donor (in most cases).

Table 5 displays the time spent on average by the hybrid algorithm to solve instances with up to 1,400 recipients and with a proportion of nondirected donors up to $q = 0.20$. Results from previous tables are included for the sake of comparison.

We note that the hybrid algorithm can solve instances with up to 1,400 recipients and up to 280 nondirected donors in slightly more than a minute on average, indicating that our algorithm scales well with both the number of recipients and the proportion of nondirected donors. As expected, there is a strong correlation between the number of recipients and the time required to solve an instance. For example, instances with 800 recipients and 80 nondirected donors are solved in 13 seconds on average, whereas instances with 1,400 recipients and 140 nondirected donors are solved in 275 seconds on average.

The correlation between the proportion of nondirected donors and the time required to solve an instance is not as straightforward. Instances with $q = 0.01$ are always the fastest to solve, but we observe that instances with $q = 0.20$ do not necessarily take the longest to solve. For example, instances with 1,400 recipients and 280 nondirected donors are solved in 78 seconds on average, whereas instances with the same number of recipients and half the number of nondirected donors are solved in 275 seconds on average. A possible explanation for this phenomenon is that instances with $q = 0.20$ do not necessarily produce larger mathematical models compared with instances with $q = 0.10$ because of the cycle/arc deactivation. Table 6 gives information about the size of the model solved when optimizing the last objective function f_5, and, indeed, we observe that the last model has 72,167 variables on average for instances with 1,400 recipients and 280 nondirected donors, whereas it has 139,962 variables on average for instances with the same number of recipients and 140 nondirected donors.

4.3. Randomly Generated Instances with the Spanish and Dutch KEP Objectives

We tested our best approach, the hybrid algorithm, on the Spanish KEP objectives. As we did not have any information regarding the score distribution, we only optimized the four first objective functions, which are as follows: (i) maximize the number of transplants, (ii) maximize the number of distinct exchanges selected, (iii) maximize the number of cross arcs, and (iv)

Table 5. Average Time (in Seconds) Required by the Hybrid Algorithm to Solve Each Set of Instances

	$q = 0.01$						$q = 0.05$						$q = 0.10$						$q = 0.15$						$q = 0.20$					
$\lvert\mathcal{R}\rvert$	**TT**	T1	T2	T3	T4	T5	**TT**	T1	T2	T3	T4	T5	**TT**	T1	T2	T3	T4	T5	**TT**	T1	T2	T3	T4	T5	**TT**	T1	T2	T3	T4	T5
50	**0**	0	0	0	0	0	**0**	0	0	0	0	0	**0**	0	0	0	0	0	**0**	0	0	0	0	0	**0**	0	0	0	0	0
100	**0**	0	0	0	0	0	**0**	0	0	0	0	0	**0**	0	0	0	0	0	**0**	0	0	0	0	0	**0**	0	0	0	0	0
200	**0**	0	0	0	0	0	**0**	0	0	0	0	0	**0**	0	0	0	0	0	**0**	0	0	0	0	0	**0**	0	0	0	0	0
400	**1**	0	0	0	0	0	**1**	0	0	0	0	0	**1**	1	0	0	0	0	**2**	1	0	0	0	0	**2**	1	0	0	0	0
600	**2**	1	0	0	1	0	**11**	2	3	2	3	2	**5**	2	1	1	1	0	**6**	2	1	1	1	1	**7**	2	2	1	1	0
800	**7**	3	1	1	2	1	**15**	4	2	3	3	3	**13**	4	2	3	2	1	**16**	4	3	3	3	2	**13**	4	3	2	2	1
1,000	**18**	7	3	4	3	2	**32**	7	4	10	6	4	**29**	7	5	6	7	4	**23**	7	5	5	4	1	**26**	8	6	6	4	1
1,200	**26**	9	2	8	4	3	**49**	10	7	20	6	4	**53**	12	9	16	11	5	**42**	13	9	9	7	3	**46**	13	11	10	8	4
1,400	**77**	16	4	28	12	17	**163**	17	13	77	24	32	**275**	18	14	49	69	125	**138**	16	14	28	28	51	**78**	17	16	26	13	6

Table 6. Model Size of the Hybrid Algorithm When Optimizing f_5 on Each Set of Instances

	$q = 0.01$			$q = 0.05$			$q = 0.10$			$q = 0.15$			$q = 0.20$		
$\lvert\mathcal{R}\rvert$	Nb. var.	Nb. cons.	Nb. nzs.	Nb. var.	Nb. cons.	Nb. nzs.	Nb. var.	Nb. cons.	Nb. nzs.	Nb. var.	Nb. cons.	Nb. nzs.	Nb. var.	Nb. cons.	Nb. nzs.
50	11	20	58	19	26	103	12	21	65	52	37	223	79	44	341
100	40	46	220	87	60	523	105	73	545	179	85	846	268	98	1,197
200	156	122	898	252	139	1,542	546	167	3,223	730	194	3,907	956	218	4,653
400	527	289	3,137	838	340	5,100	1,453	385	8,333	3,054	429	16,100	3,109	477	5,233
600	802	473	4,629	5,155	547	32,985	3,098	608	16,780	8,027	673	42,098	9225	734	45,017
800	2,516	678	15,095	4,861	759	28,395	13,908	835	77,037	13,955	916	69,877	17,213	990	80,736
1,000	5,281	892	30,787	14,174	971	79,483	30,670	1,064	168,546	20,200	1,152	96,636	24,496	1,248	109,174
1,200	8,148	1,081	45,779	21,596	1,182	119,370	38,887	1,294	206,681	37,641	1,394	182,593	42,065	1,501	187,638
1,400	16,595	1,283	92,957	64,875	1,401	363,897	139,962	1,535	770,552	116,531	1,638	622,379	72,167	1,760	332,253

maximize the number of highly sensitized recipients selected. We used the same instances as the ones created for the UKLKSS objectives, and we used a binary indicator defining a recipient as highly sensitized if their cPRA value was at 0.85 or above.

According to Biró et al. (2019), only cycles of size 2 and 3 are allowed in the Spanish KEP. There is no theoretical restriction on the maximum allowed length of chains, but given that the longest chain that has proceeded in Spain to date had length 6, we imposed this as an upper bound on the chain length for our experiments. All the techniques presented in Section 3 can be trivially extended to the Spanish KEP with the exception of the dominated chains, for which we detail in Figures 2 and 3 the most common generic cases of dominated chains of length 5 and 6 and their respective dominating sets. Note that there exist more complex dominated chains of length 5 and 6 (e.g., chain $A \to W \to X \to Y \to Z$ with cross arcs (A, X) (Z, W), and (W, Z) is dominated by chain $A \to X \to Y$ and cycle $[W, Z]$).

We first tested the adapted hybrid algorithm on a subset of the randomly generated instances, and we compared its results with those obtained by the cycle formulation. The subset contains 270 instances with the number of recipients $|\mathcal{R}| \in \{50, 100, 200\}$ and the proportion of nondirected donors $q \in \{0.01, 0.05, 0.10\}$. These values were chosen so that each algorithm could solve all the instances to optimality in a reasonable time. We then tested the hybrid algorithm on the remaining 1,080 instances to show its scalability. We added a time limit of 3,600 seconds to this set of experiments because some instances could not be solved, even after 10 hours of computation time. We observed some memory issues occurring when optimizing f_3 due to the high number of chains of size 5 and 6 that were generated during the transition between PICEF and the cycle formulation (above two billion for some instances with more than 1,200 recipients). Hence, we also imposed a limit for the model size: if at any stage the solver was dealing with an integer model containing more than 75 million variables, then the algorithm stopped and the instance was considered as unsolved in 3,600 seconds. Table 7 displays the time spent on average by the cycle and the hybrid algorithms optimizing each objective function and the model size when optimizing f_4 for instances with 50, 100, and 200 recipients.

The table shows that, also for the Spanish KEP, the hybrid algorithm is significantly faster than the cycle formulation. This is even true for small instances, as it takes more than one minute on average for the cycle

Figure 2. Dominated Chains of Length 5 for the Spanish KEP and Their Dominating Sets

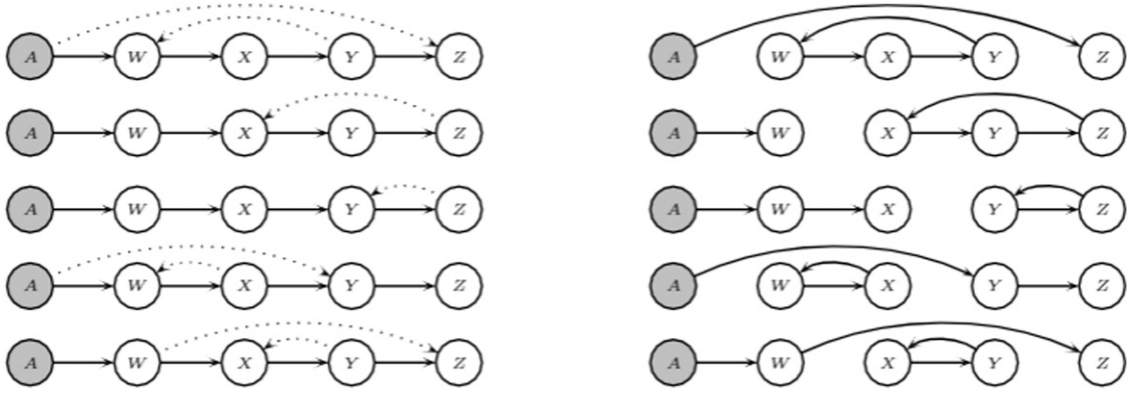
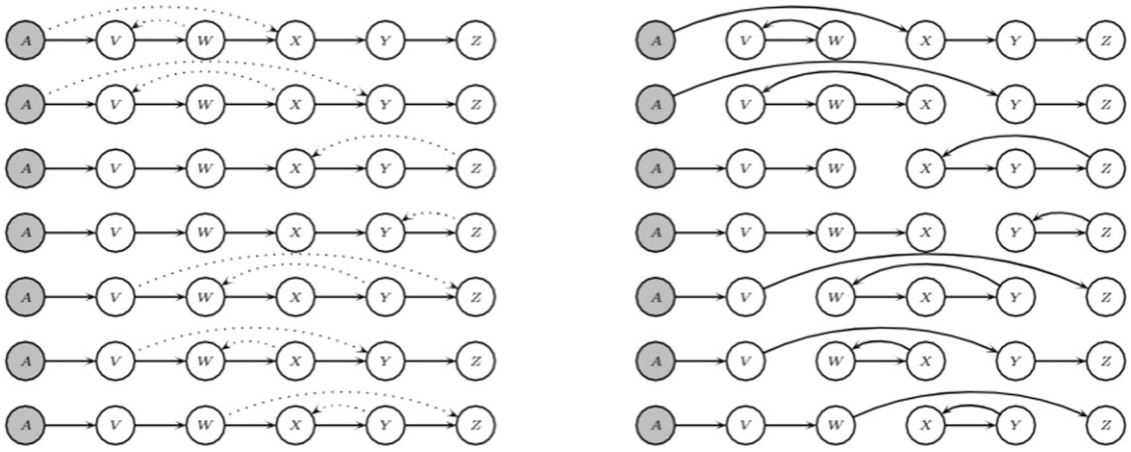


Figure 3. Dominated Chains of Length 6 for the Spanish KEP and Their Dominating Sets



formulation to solve an instance with 100 recipients, whereas it only takes 0.1 second on average to solve the same instance using the hybrid algorithm. This is due to the very large number of variables involved in the cycle formulation, which is even higher in the Spanish KEP than it is in the UKLKSS because the chain length is capped at 6 instead of 4.

For each set of instances with up to 1,400 recipients and with a proportion of nondirected donors up to $q = 0.20$, Table 8 displays the number of instances solved by the hybrid algorithm in less than one hour, the average time spent solving each instance, and the average number of times the bound T_k ($k = 1, \ldots, 4$) had to be updated.

We observe that the hybrid algorithm does not scale as well for the Spanish KEP as it did for the UKLKSS since only one instance with 1,400 recipients and 280 nondirected donors could be solved in less than an hour. It is worth observing that the instances treated here are very challenging since there are theoretically up to $O(10^{18})$ possible chains of length 6 (280×1400^5), which means up to $O(10^{18})$ variables if the cycle formulation was used, which is not solvable by any state-of-the-art ILP solvers. It seems that, for the Spanish KEP, both the number of recipients and the proportion of nondirected donors have a strong impact on the time required to solve an instance. It is not surprising, considering that each nondirected donor can potentially initiate $O(|\mathcal{R}|^5)$ chains, which may result in an excessive number of variables when the hybrid algorithm switches from PICEF to the cycle formulation (between f_2 and f_3). Interestingly, we also notice that there are more backtracking steps in the Spanish KEP than in the UKLKSS. This is probably due to a weaker continuous relaxation caused by longer chains. We tried several extensions of the hybrid algorithms, either without any cycle/chain deactivation for f_3 and f_4, or with at most one iteration before reactivating every cycle/chain and removing bounds T_3 and T_4, but we obtained results that were significantly worse than those presented in the table.

We also tested our best algorithm on the Dutch KEP objectives, which allows cycles and chains up to size 4 (Biró et al. 2019). We extended most of the techniques introduced in Section 3, including the cycle/chain deactivation algorithm and the diving algorithm, and we introduced additional techniques specifically tailored for the Dutch KEP, namely, dominated cycles and warm starts. We compared the results of our tailored approach with those obtained by PICEF (as no objective function involves maximizing the number of cross arcs) on a subset of 750 instances where the number of recipients $|\mathcal{R}| \in \{50, 100, 200, 400, 600\}$ and the proportion of nondirected donors $q \in \{0.01, 0.05, 0.10, 0.15, 0.20\}$ and found that our algorithm was faster than PICEF by one order of magnitude on average. More detailed outcomes are displayed in Section EC.2 of the e-companion.

Table 7. Information on the Hybrid and Cycle Algorithms for Instances with 50, 100, and 200 Recipients

$\lvert\mathcal{R}\rvert$	Algorithm	**TT**	T1	T2	T3	T4	Nb. var.	Nb. cons.	Nb. nzs.
50	Cycle	**0.55**	0.08	0.13	0.16	0.18	5,425	32	46,399
	Hybrid algorithm − DC	**0.04**	0.00	0.02	0.01	0.01	17	25	101
100	Cycle	**81.28**	8.91	20.31	22.83	29.11	299,108	75	2,642,251
	Hybrid algorithm − DC	**0.10**	0.02	0.03	0.03	0.01	68	58	495
200	Cycle	**13,305.51**	998.1	3,090.59	3,510.77	5,698.48	19,735,661	181	176,175,415
	Hybrid algorithm − DC	**0.61**	0.18	0.16	0.21	0.05	407	145	3,320

Table 8. Information on the Hybrid Algorithm for Instances With up to 1400 Recipients for Each Value of q

$q = 0.01$						
$\|\mathcal{R}\|$	Opt	TT	Nb. F1	Nb. F2	Nb. F3	Nb. F4
50	30	**0**	0	0	0	0
100	30	**0**	0	0	0	0
200	30	**0**	0	0	0	0
400	30	**1**	0	0.1	0.1	0.1
600	30	**7**	0	0.1	0.9	0.9
800	30	**32**	0.1	0.4	1.8	2.1
1,000	30	**90**	0	0.2	2.3	2.6
1,200	30	**112**	0	0.1	0.9	1.1
1,400	24	**950**	0	0.2	2	2.4
$q = 0.05$						
$\|\mathcal{R}\|$	Opt	**TT**	Nb. F1	Nb. F2	Nb. F3	Nb. F4
50	30	**0**	0	0	0	0
100	30	**0**	0	0	0.1	0.1
200	30	**1**	0.1	0.2	0.2	0.2
400	30	**12**	0.2	0.5	1.8	1
600	27	**586**	0	0.2	2.2	2.5
800	26	**992**	0	0	1.7	1.9
1,000	19	**1,711**	0	0.1	1.4	1.7
1,200	16	**2,322**	0	0.1	0.4	0.7
1,400	7	**3,022**	0	0.1	0.3	0.4
$q = 0.10$						
$\|\mathcal{R}\|$	Opt	**TT**	Nb. F1	Nb. F2	Nb. F3	Nb. F4
50	30	**0**	0	0	0	0
100	30	**0**	0	0	0.1	0
200	30	**1**	0	0	0	0.1
400	30	**37**	0	0	0.5	0.3
600	29	**339**	0	0	0.7	1
800	28	**596**	0	0	0.6	1.1
1,000	23	**1,795**	0	0	0.6	0.9
1,200	7	**3,189**	0	0	0.2	0.5
1,400	7	**3,110**	0	0	0.1	0.5
$q = 0.15$						
$\|\mathcal{R}\|$	Opt	**TT**	Nb. F1	Nb. F2	Nb. F3	Nb. F4
50	30	**0**	0	0	0	0
100	30	**0**	0	0	0	0
200	30	**1**	0	0	0.1	0.1
400	30	**49**	0	0	0.4	0.5
600	26	**697**	0	0.1	0.6	0.7
800	26	**1,123**	0	0	0.5	0.6
1,000	16	**2,525**	0	0	0.3	0.8
1,200	8	**2,878**	0	0	0.2	0.4
1,400	5	**3,368**	0	0	0	0.1
$q = 0.20$						
$\|\mathcal{R}\|$	Opt	**TT**	Nb. F1	Nb. F2	Nb. F3	Nb. F4
50	30	**0**	0	0	0	0
100	30	**0**	0	0	0	0
200	30	**1**	0	0	0.2	0.1
400	30	**31**	0	0	0.1	0.1
600	29	**407**	0	0	0.4	0.4
800	25	**1,395**	0	0	0.3	0.4
1,000	14	**2,481**	0	0	0.4	0.3
1,200	2	**3,504**	0	0	0	0.1
1,400	1	**3,504**	0	0	0.2	0

5. Conclusion

Hierarchical optimization is an important feature of many kidney exchange programs globally. KEPs have established the set of objective functions that they wish to optimize, and many use the cycle formulation to model their problem. Such models are manageable when the number of recipients is limited, but they quickly become impractical when programs reach 400–600 recipients. These numbers may seem high at the current time, but some European countries with large and successful KEPs such as the United Kingdom already have around 300 recipients in their pool at each run. It is also known that cooperation among European countries for transnational KEPs could increase the number of transplants (see European Network for Collaboration on Kidney Exchange Programmes 2020), so it is not impossible to envisage merged pools of more than 1,000 recipients in the foreseeable future. It is important for algorithmic techniques to handle such pool sizes to be developed well ahead of the time that they may be required. Most countries do not completely agree on the set of objective functions that should be optimized, but all agree that the number of transplants should be maximized at some level (Biró et al. 2019, Biró et al. 2020). As a result, they generally prefer to use exact approaches since an additional unit in the objective function may be associated with an additional transplant, and thus, an additional life saved.

We described three tools to improve the performances of both the cycle formulation and PICEF: a cycle/chain deactivation algorithm and a dominated chain detector to remove unnecessary variables, and a diving algorithm to remove the necessity to solve complex integer linear programming models. We also showed that it was possible to transition from PICEF to the cycle formulation between the optimization of two objective functions. As a result, PICEF can be used until a critical objective function (such as cross arcs maximization) needs to be optimized.

We tuned our approaches for the UKLKSS and showed that they could be up to 1,000 times faster

than the cycle formulation. We could even solve instances with up to 1,400 recipients and 280 nondirected donors in less than two minutes on average. Problems of such scale have never been solved before by the algorithms used for the UKLKSS. We also demonstrated that our approaches could be adapted for KEPs from other countries such as Spain and the Netherlands. For the Spanish KEP, our best approach was also several orders of magnitude faster than the cycle formulation, and we could solve an instance with 1,400 recipients and 280 nondirected donors in less than an hour, which is significant considering that chains of length 6 are allowed. For the Dutch KEP, our best approach was almost up to 10 times faster than PICEF, and we could solve instances with up to 600 recipients and 120 nondirected donors, which is also significant since cycles of size 4 are allowed.

Naturally, we do not claim that the algorithms proposed in this work are ready-to-use tools for all the European KEPs. Among the possible limitations of our work, we remark that the instances we solved were generated with UK-specific parameters, that some objective functions in the Spanish and Dutch KEP were not optimized because we did not have any information regarding the data distribution, and that the tuning of our algorithms was focused on the UKLKSS and could be improved for the other countries. In addition, we used a commercial solver, which is sometimes too expensive (or simply unnecessary) for some KEPs. Lastly, our proposed methods are specifically designed for hierarchical optimization, as is prevalent in Europe; thus, only dominated chain detection and the cycle/chain deactivation algorithm (to a lesser extent) may be applicable to KEPs that optimize a single objective, which is common in USA-based KEPs.

Still, we successfully showed the effectiveness of our tools for the UKLKSS and their adaptability to other countries' requirements. Some of the techniques we propose could even be used for hierarchical optimization in other areas, provided that the bounds given by the continuous relaxations are close to the optimal objective values. We leave as future work the adaptation of our techniques to multicountry KEPs, a problem in which (i) pools of several countries are merged, (ii) a consensus on the objective function has to be found, and (iii) a minimum number of transplants per country is imposed, so that no country loses any transplants by participating in the program.

Acknowledgments

The authors thank the two anonymous reviewers for their valuable comments that have helped to improve the presentation of this paper. The authors would like to thank Joris van de Klundert for helpful insights regarding objective (iii) used in the Dutch KEP and Julian Hall for his insights regarding linear programming theory.

References

Abraham D, Blum A, Sandholm T (2007) Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. *Proc. Eighth ACM Conf. Electronic Commerce* (ACM, New York), 295–304.

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

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

Axelrod DA, Schnitzler MA, Xiao H, Irish W, Tuttle-Newhall E, Chang SH, Kasiske BL, Alhamad T, Lentine KL (2018) An economic assessment of contemporary kidney transplant practice. *Amer. J. Transplantation* 18:1168–1176.

Balas E, Christofides N (1981) A restricted Lagrangean approach to the traveling salesman problem. *Math. Programming* 21:19–46.

Bikbov B, Purcell CA, Levey AS, Smith M, Abdoli A, Abebe M, Adebay OM, et al. (2020) Global, regional, and national burden of chronic kidney disease, 1990–2017: A systematic analysis for the Global Burden of Disease Study 2017. *Lancet* 395:709–733.

Biró P, Haase-Kromwijk B, Andersson T, Ásgeirsson EI, Baltesová T, Boletis I, Bolotinha C, et al. (2019) Building kidney exchange programmes in Europe—an overview of exchange practice and activities. *Transplantation* 103:1514–1522.

Biró P, van de Klundert J, Manlove D, Pettersson W, Andersson T, Burnapp L, Chromy P, et al. (2020) Modelling and optimisation in European kidney exchange programmes. *Eur. J. Oper. Res.* 291:447–456.

Blum A, Dickerson J, Haghtalab N, Procaccia A, Sandholm T, Sharma A (2020) Ignorance is almost bliss: Near-optimal stochastic matching with few queries. *Oper. Res.* 68:16–34.

Caragiannis I, Filos-Ratsikas A, Procaccia AD (2015) An improved 2-agent kidney exchange mechanism. *Theoret. Comput. Sci.* 589: 53–60.

Carpaneto G, Dell'Amico M, Toth P (1995) Exact solution of large-scale, asymmetric traveling salesman problems. *ACM Trans. Math. Software* 21:394–409.

Chisca D, Lombardi M, Milano M, O'Sullivan B (2019a) Logic-based Benders decomposition for super solutions: An application to the kidney exchange problem. Schiex T, de Givry S, eds. *Principles and Practice of Constraint Programming* (Springer, Cham, Switzerland), 108–125.

Chisca DS, Lombardi M, Milano M, O'Sullivan B (2019b) A sampling-free anticipatory algorithm for the kidney exchange problem. Rousseau LM, Stergiou K, eds. *Integration of Constraint Programming, Artificial Intelligence, and Operations Research* (Springer, Cham, Switzerland), 146–162.

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

Dantzig GB (1963) *Linear Programming and Extensions* (Princeton University Press, Princeton, NJ).

Das S, Dickerson JP, Li Z, Sandholm T (2015) Competing dynamic matching markets. *Proc. Conf. Auctions Market Mechanisms Appl.* (AMMA), vol. 112 (ACM, New York), 245.

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

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

Dickerson JP, Procaccia AD, Sandholm T (2012) Dynamic matching via weighted myopia with application to kidney exchange. *Proc. 26th AAAI Conf. Artificial Intelligence* (AAAI Press, Palo Alto, CA), 1340–1346.

Dickerson J, Manlove D, Plaut B, Sandholm T, Trimble J (2016) Position-indexed formulations for kidney exchange. *Proc. 2016 ACM Conf. Econom. Comput.* (ACM, New York), 25–42.

European Network for Collaboration on Kidney Exchange Programmes (2020) ENCKEP. Retrieved June 25 from http://www.enckep-cost.eu.

Gao I (2019) Fair matching in dynamic kidney exchange. Preprint, submitted December 23, https://doi.org/10.48550/arXiv.1912.10563.

Garfinkel R, Nemhauser G (1972) *Integer Programming* (Wiley, New York).

Global Burden of Disease Collaborative Network and Institute for Health Metrics and Evaluation (2019) Global Burden of Disease Study. Retrieved March 25, 2021, from http://ghdx.healthdata.org/gbd-results-tool?params=gbd-api-2019-permalink/bb32e062360440c605520145ecd096b2.

Hart A, Smith JM, Skeans MA, Gustafson SK, Stewart DE, Cherikh WS, Wainright JL, et al. (2017) OPTN/SRTR 2015 annual data report: Kidney. *Amer. J. Transplantation* 17:21–116.

Irnich S, Desaulniers G, Desrosiers J, Hadjar A (2010) Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS J. Comput.* 22:297–313.

Klimentova X, Pedroso JP, Viana A (2016) Maximising expectation of the number of transplants in kidney exchange programmes. *Comput. Oper. Res.* 73:1–11.

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

Lin M, Wang J, Feng Q, Fu B (2019) Randomized parameterized algorithms for the kidney exchange problem. *Algorithms* 12(2):50.

Mak-Hau VH (2017) On the kidney exchange problem: Cardinality constrained cycle and chain problems on directed graphs: A survey of integer programming approaches. *J. Combin. Optim.* 33(1):35–59.

Manlove D, O'Malley G (2015) Paired and altruistic kidney donation in the UK: Algorithms and experimentation. *ACM J. Experiment. Algorithmics* 19:1–21.

McElfresh DC, Bidkhori H, Dickerson JP (2019) Scalable robust kidney exchange. *Proc. 33rd AAAI Conf. Artificial Intelligence* (AAAI Press, Palo Alto, CA), 1077–1084.

National Kidney Foundation (2020) Dialysis. Retrieved June 25 from https://www.kidney.org/atoz/content/dialysisinfo.

Nemhauser G, Wolsey L (1988) *Integer and Combinatorial Optimization* (Wiley, New York).

Organ Procurement and Transplantation Network (2020) Organ Procurement and Transplantation Network policies. Retrieved June 25 from https://optn.transplant.hrsa.gov/media/1200/optn_policies.pdf.

Rapaport F (1986) The case for a living emotionally related international kidney donor exchange registry. *Transplantation Proc.* 18(3, Suppl. 2)5–9.

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

Roth A, Sönmez T, Ünver M (2005) Pairwise kidney exchange. *J. Econom. Theory* 125(2):151–188.

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

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

Trimble J (2020) Data set generator. Retrieved June 25 from https://jamestrimble.github.io/kidney-webapp/#/generator.

Wang W, Bray M, Song PX, Kalbfleisch JD (2019) An efficient algorithm to enumerate sets with fallbacks in a kidney paired donation program. *Oper. Res. Health Care* 20:45–55.

Wolfe RA, Roys EC, Merion RM (2010) Trends in organ donation and transplantation in the United States, 1999–2008. *Amer. J. Transplantation* 10:961–972.

Wolsey L (1998) *Integer Programming* (Wiley, New York).

Maxence Delorme is an assistant professor in the Department of Econometrics and Operations Research at Tilburg University. His research covers a large range of combinatorial optimization problems, especially in the packing, scheduling, and matching areas. He is particularly interested in pushing the boundaries of exact algorithms.

Sergio García is a reader in operational research at the School of Mathematics, University of Edinburgh. His research is focused on the development of models and algorithms for integer programming and combinatorial optimization problems, particularly on areas such as facility location, vehicle routing, clustering with feature selection, matching problems, and airport logistics.

Jacek Gondzio is a professor of optimization at the University of Edinburgh. His research interests include the theory and implementation of algorithms for very large-scale optimization. He is best known for his contributions in the area of interior point methods. He is an associate editor of the *European Journal of Operational Research* and several leading optimization journals.

Jörg Kalcsics is a reader in operational research at the School of Mathematics, University of Edinburgh. His research interests lie in developing exact and heuristic models and algorithms for solving combinatorial optimization problems, especially in the areas of facility location, network design, sales districting, and service scheduling.

David Manlove is professor of algorithms and complexity at the School of Computing Science, University of Glasgow. He has worked with the National Health Service since 2007 on developing algorithms for the UK Living Kidney Sharing Scheme. He was awarded the Lord Kelvin Medal from the Royal Society of Edinburgh in 2019 for this work.

William Pettersson is a research associate of the Formal Analysis, Theory and Algorithms section in the School of Computing Science at the University of Glasgow. His research focuses on graph algorithms and their applications to resource allocation problems.