Erasmus University Rotterdam
Erasmus School of Economics
Bachelor thesis Econometrie en Operationele Research

Novel Third-Cycle Formulation for the Kidney Exchange Problem

Joep Kratsborn (621179jk)

Supervisor:	Roby Cremers
Second assessor:	Albert Wagelmans
Date final version:	1st July 2024

Abstract

This thesis presents a novel integer programming model for the kidney exchange problem (KEP) called the third-cycle formulation (TCF). In this model, a cycle is represented by three compatible third-cycles, aiming to enhance the existing half-cycle formulation (HCF). We provide a clear explanation of reduction procedures applicable to both the HCF and TCF. Through comparative analysis with current competitive models, the TCF demonstrates strong computational performance, although not stronger than the HCF and extended edge formulation (EE), and robustness when the maximum cycle length is increased. Additionally, we identify the most effective variable reduction techniques for the HCF and EE, addressing a gap in the current literature. Lastly, we introduce two new position-indexed chain edge formulations (PICEF) for the KEP with altruistic donors: the PICEF-Half-Cycle and PICEF-Third-Cycle. Computational analysis shows that the PICEF-Half-Cycle outperforms existing models for the KEP with altruistic donors.

1 Introduction

Kidneys play a vital role in the functioning of the human body, filtering waste products and excess fluids from the blood, while also regulating electrolyte balance, blood pressure, and red blood cell production. In general, a human is born with two kidneys. However, one could live a healthy lifestyle only having one functioning kidney. According to most recent estimates, there are hundreds of millions of people around the world who need a kidney transplant or dialysis because their kidneys have failed, often due to conditions like chronic kidney disease or acute kidney injury. Transplantation offers a better quality of life compared to dialysis while also incurring lower costs (Axelrod, 2018).

The most common source for kidney transplantation has been deceased donors, where organs are harvested from individuals who have passed away but still possessed functioning, healthy kidneys. For many years, a living kidney donor had to belong to the close family of the patient, motivated by ethical reasons to totally remove any financial incentive that could occur for these living donors (Boulware et al., 2008). Now in many countries, after a series of regulation changes, a living transplant between two persons who do not know each other has been made possible.

Despite the increase in living donations, the demand for kidney transplants continues to surpass supply. This supply deficit highlights the need for innovative solutions like paired kidney exchanges, now made possible by these legislation changes. The idea of such an exchange programme is relatively simple. A patient-donor pair might be incompatible due to mismatched blood types or immune system factors, which can cause the recipient's body to reject the donated kidney. An exchange programme enables patients with willing but incompatible donors to swap kidneys with other pairs in similar situations, thereby forming chains or cycles of donations that increase the overall number of successful transplants. These exchanges optimize the allocation of available kidneys, and reduce waiting times for patients in need of a transplant. These chains or cycles are mostly computed by algorithms over a list of patients at predetermined time intervals, for example every three months in the UK (Johnson, 2008).

Finding these exchange chains or cycles is done by solving a so-called kidney exchange problem (KEP). This KEP is usually represented by a directed compatibility graph, with each vertex representing a patient-donor pair and each arc representing compatibility between the donor and patient of two pairs. Usually, kidney exchange programmes impose a limit on the number of pairs involved in one chain or cycle. Longer cycles are logistically more complex and difficult to coordinate, requiring simultaneous surgeries and precise timing among multiple hospitals. Non-simultaneous transplantation within the same exchange cycle is risky because if a donor gives a kidney but the patient in their pair does not receive one immediately, there is a chance that another donor involved in the cycle might back out or pass away, leaving someone without a needed kidney. This situation could erode trust in the exchange program, as participants must rely on the commitment of others to ensure that every donor-recipient pair benefits from the exchange. The general KEP can therefore be seen as the problem of finding a set of vertex-disjoint [1] cycles that maximizes the number of pairs involved, given a directed compatibility graph and a limit on the number of pairs in one cycle.

One of the most common extensions to the KEP is the addition of altruistic or non-directed

[1] Each donor-patient pair can, of course, only be involved in one transplantation cycle.

donors. These altruistic donors, who are not paired with a patient and donate a kidney without expecting one in return, significantly enhance the KEP by initiating so-called chains of transplants. These donors can start a sequence of donations that would not be possible otherwise, allowing more pairs to benefit. Another common extension is to find the most desirable solution among all optimal ones. For example, a kidney exchange programme may want to pick an optimal solution that contains the most cycles of length 2. This can be done using hierarchical optimization using a sequence of hierarchically ranked objective functions.

Along implementation of a spectrum of existing formulations, this thesis proposes and implements a new third-cycle formulation for the KEP. Also new variable reduction techniques suitable for this formulation are presented. Unfortunately, results suggest that the third-cycle formulation does not outperform all existing models for any levels of the maximum cycle length or the number of patient-donor pairs that we test. However, the third-cycle formulation provides a performance more robust to an increase in the cycle length limit than the cycle formulation and recently proposed half-cycle formulation. The half-cycle and extended edge formulation show most dominant performance across the tested instances. Furthermore, we show that the extended edge formulation displays best performance for a descending vertex ordering, in contrast to claims made by Delorme, Manlove & Smeets (2023). Additionally, we clearly show which variable preprocessing techniques for the half-cycle formulation are really worth the decrease in model size in order to adress a gap in previous literature. Lastly, we introduce two new variants on the position-indexed edge formulation (PICEF) for the KEP with altruistic donors: the PICEF-Half-Cycle and PICEF-Third-Cycle. The PICEF-Half-Cycle is shown to outperform existing models for the KEP including altruistic donors.

The remainder of this thesis is structured as follows. More information on previous works on the KEP can be found in Section 2. In Section 3, a more detailed description of the problem is given. Next, the methodology is further explained in Section 4, providing a detailed description of the novel third-cycle formulation and new variable reduction techniques in Section 4.2 and Section 4.3, respectively. Furthermore, the numerical results are presented in Section 5 and the conclusion in Section 6.

2 Literature review

The KEP in itself is not a very hard problem to solve. However, the maximum cycle length complicates things. When only cycles of length 2 are allowed, an optimal solution to the KEP can be found in polynomial time, as the problem can be viewed as a maximum-weighted matching problem (Roth et al., 2005).[2] However, when the cycle length limit is greater than or equal to 3, the KEP becomes NP-hard (Abraham et al., 2007).[3] This result motivated researchers to find suitable techniques to handle the complex KEP.

First, let us discuss the papers that developed new and competitive ways to model the general KEP. Two of the most intuitive formulations, namely the cycle formulation and the

[2]The maximum weighted matching problem involves finding a matching between vertices in a weighted graph where the sum of the weights of the selected edges is maximized.

[3]NP-hard problems refer to computational problems that are unlikely to be solved to optimality in polynomial time, most of the time requiring exponential time to find an exact solution.

edge formulation were first proposed by (Roth et al., 2007). The cycle formulation has its number of variables growing exponentially with the maximum cycle length, whereas the edge formulation suffers from this characteristic in its number of constraints. Of course, this is not ideal, as some kidney exchange programmes concern hundreds of patient-donor pairs and an increased maximum cycle length could ensure multiple extra successful transplantations. Constantino et al. (2013) introduces a new extended edge formulation and an edge assignment formulation along with suitable variable reduction techniques. These formulations are often called compact as these are designed to be scalable, meaning they can handle larger instances without a significant increase in computational time. The cycle formulation performs best for maximum cycle length equal to 3 or 4, while the extended edge formulation is much more robust to an increase in maximum cycle length and outperforms the other formulations for maximum cycle length equal to 5 or 6. Dickerson et al. (2016) proposes the, also compact, position-indexed edge formulation. Next to results suggesting that this formulation can compete with the current best, this paper also introduces a novel, efficient chain structure in order to model altruistic donors. Do note that all formulations can handle altruistic donors, in a sense that a chain can very simply be modeled as a cycle. This can be done by introducing a dummy patient for each altruistic donor that is compatible with every ordinary donor. However, Dickerson et al. (2016) show that this is not the most efficient way to model these chains.

Delorme, Manlove & Smeets (2023)'s new half cycle formulation models distinct cycles by two compatible halves, resulting in a significant reduction in the number of variables compared to the ordinary cycle formulation. Their half-cycle formulation outperforms existing formulations when the cycle size limit is set to 4, 5, or 6 for large patient-donor pools, also partly depending on the density of the compatibility graph. The authors stress that the good performance might be attributable to the fact that the Linear Programming (LP) relaxation bound is as tight as that of the cycle formulation, which is the tightest known until now.

Moving on to papers that improved the algorithmic performance of existing formulations, Lam & Mak-Hau (2020) introduces the first branch-and-cut-and-price model [4] for the cycle formulation, outperforming the current solving algorithms. Furthermore, Delorme, García et al. (2023) applies a new diving algorithm, together with reduced cost variable fixing to the cycle formulation for KEPs with hierarchical optimisation. The work of Riascos-Álvarez et al. (2024) proposes a branch-and-price algorithm in which the pricing problems are solved through decision diagrams. They present a new Lagrangian-based upper bound on the optimal objective value, obtained through their master problem. Although the aim of this thesis is not to improve the algorithmic performance of implemented formulations, these or similar methods could be applied to the models discussed in this thesis.

Finally, there is the extensive literature on the adaptation of current approaches to different real-world features. This literature is not that relevant for our research, as we merely focus on the general KEP, with and without altruistic donors.

[4] A branch-and-cut-and-price model is a method that combines branch-and-bound techniques with cutting planes and column generation to efficiently solve complex instances of the problem.

3 Problem description

In the general KEP without altruistic donors, we are given a set of n incompatible patient-donor pairs, and their compatibilities with all other patient-donor pairs. This compatibility structure can be represented most easily by means of a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where vertex set $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ contains one vertex for each patient-donor pair and where arc set $\mathcal{A}$ contains arc (v_i, v_j) if and only if the donor of pair i is compatible with the patient of pair j. A weight w_{ij} can be associated to each arc $(v_i, v_j) \in \mathcal{A}$. However, in this thesis, the objective is simply to maximize the number of transplants, implying that each weight should be equal to one another. Therefore, it is most convenient to set $w_{ij} = 1$ for all arcs $(v_i, v_j) \in \mathcal{A}$. Furthermore, we are given a limit k on the number of pairs that can be included in one exchange cycle. A solution to the KEP, also called an exchange, can be represented by a subset of arcs $\mathcal{A}' \subseteq \mathcal{A}$, that solely consists of a number of vertex-disjoint cycles with a maximum length of k.

One of the most common extensions of the KEP is to include altruistic donors. Vertex set $\mathcal{V}$ can now be split into a set of patient-donor pairs $\mathcal{P}$ and a set of altruistic or non-directed donors $\mathcal{N}$. For each $v_i \in \mathcal{N}$ and each $v_j \in \mathcal{P}$, $\mathcal{A}$ contains arc (v_i, v_j) if and only if altruistic donor v_i is compatible with the patient of pair v_j. Furthermore, we are given a limit k' on the number of pairs that can be involved in one exchange chain initiated by an altruistic donor $v_i \in \mathcal{N}$.

4 Methodology

This section touches upon the methods used in this thesis. The cycle formulation (CF), edge formulation (EF), reduced edge-assignment formulation (EA) and reduced extended edge formulation (EE) are all implemented as extensively described in Constantino et al. (2013). Furthermore, the half-cyle formulation (HCF) as firstly proposed by Delorme, Manlove & Smeets (2023) is briefly described and implemented. Finally, this section proposes a novel third-cycle formulation (TCF). Even though this thesis almost exactly copies the implementations of existing formulations, the formulations and possible reduction techniques of CF and HCF are explained to set a framework for and add intuition on the new TCF. The EF, EA and EE are only described very briefly.

4.1 Existing formulations

One of the most simple and intuitive models for the KEP, is the cycle formulation as described in Constantino et al. (2013). This formulation requires a set $\mathcal{C}_k$ consisting of all possible cycles of maximum size k, in order to associate a binary variable z_c with every feasible cycle $c \in \mathcal{C}_k$. The decision variable z_c takes value 1 if this cycle c is selected in a solution and value 0 otherwise. Now let us introduce the set $V(c)$ as the set containing all vertices that are present in cycle $c \in \mathcal{C}_k$. The cycle formulation is then as follows:

$$\max \sum_{c \in \mathcal{C}_k} |V(c)| z_c \tag{1}$$

$$\text{s.t.} \sum_{c \in \mathcal{C}_k : v \in V(c)} z_c \leq 1, \qquad \forall v \in \mathcal{V}, \tag{2}$$

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

Objective function (1) represents the total number of transplants and is maximized. Constraints (2) ensure that no patient-donor pair or vertex appears more than once in the selected cycles together. Constraints (3) make sure that every possible cycle is either selected or not. Although this cycle formulation is one of the first models proposed to model the KEP, results of Constantino et al. (2013) suggest that it shows dominant performance for instances with values of k smaller or equal to 4. This can be mainly explained by the fact that the model contains $O(n^k)$ variables [5] and $O(n)$ constraints, which remains manageable for these low values of k. Furthermore, the bound of the CF's LP relaxation turns out to be the best among all relaxations of existing IP models for the KEP according to Dickerson et al. (2016).

The edge formulation associates a binary variable to every arc in the compatibility graph. The objective is to maximize the number of selected arcs under the following constraints: flow conservation at each vertex, at most one selected outgoing edge at each vertex and no path of length k can be fully covered by the selected arcs.[6]

The first two proposed compact formulations are the edge assignment and the extended edge formulation. Within the edge assignment formulation, a binary variable is associated to every edge, and to every combination of a node and potential cycle. The objective is again to maximize the number of selected arcs under the following constraints: flow conservation at each vertex, at most one selected outgoing edge at each vertex, no more than k vertices selected per potential cycle, assignment of each reached vertex to one potential cycle and assignment of v_i and v_j to the same potential cycle, when arc (v_i, v_j) is selected in the solution.

The extended edge formulation considers a number of copies of the compatibility graph, one for every potential cycle. A binary variable is associated to every edge in every copy. The objective is again to maximize the number of selected arcs under the following constraints: flow conservation at each vertex, at most one selected outgoing edge at each vertex and no more than k arcs selected per copy. Note that preprocessing techniques remove a lot of variables and constraints in both the EA and EE.

Recently, Delorme, Manlove & Smeets (2023) proposed a new so-called half-cycle formulation. The main idea is to exploit the symmetry within the problem to obtain an IP model of reduced size. The authors present an analogy with a new model for the bin packing problem by Delorme & Iori (2020) that uses half of the bin capacity to model an instance, resulting in a model of significantly reduced size. Both the first and second half of the bin would be modelled by the same structure and results suggest that decrease in terms of the number of variables outweigh by

[5] The number of actual variables in the model depends heavily on the density and structure of the compatibility graph.

[6] A fully covered k-path would imply a cycle of length at least $k + 1$ in the solution.

far the disadvantages of the extra constraints that ensure compatibility between the half-bins. The authors then show how a similar concept can be applied to the KEP.

In their HCF, a cycle is modeled by two matching half-cycles. At first glance it seems like this model might contain double the number of variables as the CF, but this is definitely not the case. Especially for large k, there may exist a lot of half-cycles starting or ending in the same two vertices. These half-cycles can be combined in a lot of ways to obtain different complete cycles, all complete cycles with an own distinct decision variable in the CF.

Let $\mathcal{H}$ be the set of all possible half-cycles up to size $\lceil k/2 \rceil$.[7] Section 4.3 discusses how this set $\mathcal{H}$ can be obtained most efficiently, assuring its size is small as possible. Then, a decision variable x_h is associated with every possible half-cycle $h \in \mathcal{H}$, being 1 if this half-cycle h is selected in a solution and 0 otherwise. Let $V^s(h)$ and $V^e(h)$ denote the set containing the starting vertex and ending vertex respectively, of each half-cycle h. Furthermore, let $V^m(h)$ be the set of all middle vertices of each half-cycle h. The HCF is then defined as follows:

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

s.t.

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

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

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

Objective function (4) maximizes the number of transplants, by modelling the size of a half-cycle by the number of middle nodes plus 1. Constraints (5) make sure that each vertex v appears a maximum of either once in the middle of a half-cycle or twice at the start or end of a half-cycle. Constraints (6) ensure that if a half cycle starting in v_1, ending in v_2 is selected in a solution, then also a half-cycle starting in v_2 and ending in v_1 must be selected. Note that $v_2 > v_1$ is imposed on these vertices v_1, v_2 to avoid double constraints. Finally, constraints (7) make sure that each half-cycle is either selected or not. This works for even values of k, when k is odd however, we need to make sure that no two half-cycles of maximum length can be matched. This can be done by adding the following constraints:

$$x_h = 0 \qquad \forall h \in \mathcal{H} : V^e(t) < V^s(h) \text{ and } |V^m(h)| = \frac{k-1}{2} \tag{8}$$

These ensure that every half-cycle of maximum length starts with a vertex indexed lower than the ending vertex. Automatically, this implies that the model will never be able to match this third-cycle of length $\lceil k/2 \rceil$, with another one of this length in a feasible solution. This ensures the maximum complete cycle length k in case that k is odd.

[7] In this thesis, the size of a half-cycle or third-cycle is defined as the number of arcs present in this half-cycle or third-cycle.

4.2 New third-cycle formulation

This thesis covers a novel TCF, in which a cycle will be constructed by three compatible third-cycles. Therefore, we need to ensure that, for every donor pair or vertex v, there must be a third-cycle starting from v if and only if there is also a third-cycle ending in vertex v. Furthermore, if we select a third-cycle starting in v_1, ending in v_2, and a third-cycle starting in v_2, ending in v_3, a third-cycle starting from v_3, ending in v_1, must also be selected. In addition, every feasible cycle of length 2 must also be eligible for selection in the solution, as these obviously can not be constructed by three third-cycles.

Now notation will be introduced for the TCF that is similar to the notation introduced for the HCF. All possible third-cycles are all paths in the graph of length up to $\lceil k/3 \rceil$. After reducing this set to a set $\mathcal{T}$ only containing actual possible third-cycles, a variable x_t is associated to each third-cycle $t \in \mathcal{T}$, being 1 if this third-cycle is selected and 0 otherwise. Again, let $V^s(t)$ and $V^e(t)$ denote the set containing the starting vertex and ending vertex respectively of each third-cycle t. Furthermore, let $V^m(t)$ be the set of all middle vertices of each third-cycle t. Now, as mentioned, we should allow for exchange cycles of length 2. This is done by obtaining the set of all length-2 cycles $\mathcal{C}_2$ and associating a variable y_c with every cycle $c \in \mathcal{C}_2$. Finally, let $V(c)$ denote the set containing both vertices that are part of cycle c. Then we can define the TCF as follows:

$$\max \sum_{t \in \mathcal{T}} (|V^m(t)| + 1)x_t + 2 \sum_{c \in \mathcal{C}_2} y_c \tag{9}$$

s.t.

$$\sum_{t \in \mathcal{T}: v \in V^s(t) \cup V^e(t)} 0.5x_t + \sum_{t \in \mathcal{T}: v \in V^m(t)} x_t + \sum_{c \in \mathcal{C}_2: v \in V(c)} y_c \leq 1, \qquad \forall v \in \mathcal{V}, \tag{10}$$

$$\sum_{t \in \mathcal{T}: v \in V^s(t)} x_t = \sum_{t \in \mathcal{T}: v \in V^e(t)} x_t, \qquad \forall v \in \mathcal{V}, \tag{11}$$

$$\sum_{t \in \mathcal{T}: v_1 \in V^s(t), v_2 \in V^e(t)} x_t + \sum_{t \in \mathcal{T}: v_2 \in V^s(t), v_3 \in V^e(t)} x_t \leq 1 + \sum_{t \in \mathcal{T}: v_3 \in V^s(t), v_1 \in V^e(t)} x_t, \tag{12}$$

$$\forall v_1, v_2, v_3 \in \mathcal{V} : v_2 > v_3, v_2 > v_1,$$

$$x_t \in \{0, 1\}, \qquad \forall t \in \mathcal{T}. \tag{13}$$

Objective function (9) maximizes the number of transplants. Then, constraints (10) make sure that each vertex v appears a maximum of either once in the middle of a third-cycle, twice at the start and end of a third-cycle or once in a cycle of length 2. Constraints (11) ensure if there is a selected third-cycle ending v, there should also be one starting in v. Constraints (12) make sure that if there is both a third-cycle starting in v_1 that ends in v_2 and one starting in v_2 that ends in v_3, there must be a third-cycle starting in v_3 that ends in v_1 . Note that each whole cycle consisting of three third-cycles has $v_2 > v_1$ and $v_2 > v_3$ for some start/end vertices v_1, v_2 and v_3, and therefore it is feasible to impose these restrictions on the triplet of vertices to avoid double constraints. Finally, constraints (13) make sure that each third-cycle is either selected or not.

This formulation works perfectly well when k is a multiple of 3. However, let us consider the case that $k = 3n + 1$, for some $n \in \mathbb{N}$. Let us introduce $\mathcal{B} \subseteq \mathcal{T}$ as the set of all third-cycles exactly of length $\lceil k/3 \rceil$. In order to ensure the correct maximum cycle length, every complete cycle can only contain a maximum of one third-cycle $t \in \mathcal{B}$. We model this by adding the following constraints to the original TCF.

$$\sum_{t \in \mathcal{B}: v \in V^s(t) \cup V^e(t)} x_t \leq 1 \qquad \forall v \in \mathcal{V} \tag{14}$$

Constraints (14) ensure that for every vertex $v \in \mathcal{V}$ there cannot be both a third-cycle $t \in \mathcal{B}$ starting and ending in this vertex v. This forces a third-cycle of maximum length to be matched with ones of strictly smaller length on both ends. Therefore, the length of the complete cycle can never exceed k.

Finally, we need to consider the case where $k = 3n + 2$, for some $n \in \mathbb{N}$. In order to ensure the correct maximum cycle length, every complete cycle can only contain a maximum of two third-cycles $t_1, t_2 \in \mathcal{B}$. We model this by adjusting constraints (12) of the original TCF.

$$\sum_{t \in \mathcal{T}: v_1 \in V^s(t), v_2 \in V^e(t)} x_t + \sum_{t \in \mathcal{T}: v_2 \in V^s(t), v_3 \in V^e(t)} x_t \leq 1 + \sum_{t \in \mathcal{T} \setminus \mathcal{B}: v_3 \in V^s(t), v_1 \in V^e(t)} x_t \tag{15}$$
$$\forall v_1, v_2, v_3 \in \mathcal{V} : v_2 > v_1, v_2 > v_3$$

This adjustment ensures that if there is both a third-cycle starting in v_1 that ends in v_2 and one starting in v_2 that ends in v_3, there must be a third-cycle $t \in \mathcal{T} \setminus \mathcal{B}$ starting in v_3 that ends in v_1, for all $v_2 > v_1, v_3$. This forces the length of the third-cycle starting from v_3, out of all start/end vertices within a complete cycle, to be strictly smaller than the maximum third-cycle length $\lceil k/3 \rceil$. Therefore, within a complete cycle there will always be at least one third-cycle that has length strictly smaller than $\lceil k/3 \rceil$. This adjustment of constraints does not rule out any solutions that should be feasible: the partition of a cycle into thirds can always be made such that the smaller third-cycle starts at the end vertex of the third-cycle that has the biggest starting vertex, namely v_2. An overview of size and LP-relaxation bound's quality of the discussed models is presented in Table 1.

Table 1: Quality of LP-relaxation bound and size of all implemented models.

Model	**No. var.**	**No. constr.**	**LP-relaxation bound**
CF	$O(n^k)$	$O(n)$	tight
EF	$O(n^2)$	$O(n^k)$	not tight
EA	$O(n^2)$	$O(n^3)$	not tight
EE	$O(n^3)$	$O(n^2)$	not tight
HCF	$O(n^{1+\lceil k/2 \rceil})$	$O(n^2)$	tight
TCF	$O(n^{1+\lceil k/3 \rceil})$	$O(n^3)$	not tight

By inspection of the table, it does not seem like either the HCF or the TCF have a competitive advantage over the other formulations for any value of k, regarding the models' size. Especially the EA and EE seem to perform best in the worst case scenario. However, in practice, for

the HCF, especially after reduction techniques, it turns out that the number of constraints and variables is much more favorable than this table suggests. Of course, we hope that the same will hold for the TCF. Section 5 provides an overview of number of constraints and variables for the different models for different values of n and k. Additionally, the fact that the HCF's LP-relaxation is as tight [8] as the CF's one could play a big role in its computational performance. The tightness of a LP-relaxation is namely often considered an indicator of how effectively an IP model will perform in practice, because the relaxation significantly influences the efficiency of modern branch-and-bound tree search algorithms. Delorme, Manlove & Smeets (2023) argue that the LP-relaxation bound of the HCF is as tight the bound obtained by the CF, because both models are equivalent. This means that any continuous solution of the HCF can be converted into a continuous solution to the CF with the same objective value and vice versa.

At first glance it might seem as if the TCF's LP-relaxation is also equivalent to the one of the CF. However, a simple counterexample is able to prove us wrong. Consider a case in which maximum cycle length k is equal to 3. For example, decision variables associated with third-cycles $\langle A, B\rangle$, $\langle B, C\rangle$, $\langle C, D\rangle$ and $\langle D, A\rangle$ all being equal to 0.5, forms a feasible solution to the LP-relaxation of the the TCF, but can not be converted into a feasible solution to the CF's LP-relaxation. Constraints (10) are satisfied, as each vertex is selected 0.5 times both at the start and end of a third-cycle, which does not violate these capacity constraints. Constraints (11) are satisfied, as each vertex is exactly selected the same number of times at the start of a third-cycle as at the end of a third-cycle, namely 0.5 times. Finally, constraints (12) are satisfied as these constraints are passive for a left hand side less or equal to 1, which is the case when each third-cycle is selected only 0.5 times. However, this continuous solution can not be converted to a continuous solution to the CF. Namely, conversion to the CF would require a variable associated to cycle $[A, B, C, D]$, which is not possible in case maximum cycle length k equals 3.

4.3 Reduction techniques

Both the HCF and TCF display some symmetry, i.e. the presence of equivalent solutions that can be obtained by rearranging the values of decision variables. Consider for example the cycle $[A, B, C, D]$. In the HCF this cycle can be obtained by choosing half-cycles $\langle A, B, C\rangle$ and $\langle C, D, A\rangle$ or by choosing half-cycles $\langle B, C, D\rangle$ and $\langle D, A, B\rangle$. In the TCF, there even exist 4 equivalent solutions that result in this very cycle. Symmetry in an IP problem is undesirable because it can lead to redundant computations, increasing solution time by causing the solver to explore these multiple equivalent solutions. For this reason, this section discusses how to minimize this symmetry, and therefore size of the model, by applying several reduction techniques.

4.3.1 Variable reduction

For the HCF, there are very convenient ways to reduce the number of variables and constraints in the model. Symmetry can be avoided by not generating any half-cycle where the vertex with the lowest index is not positioned at either the beginning or the end of the half-cycle. This is

[8] By tight, we mean the tightest known, not that the optimal objective value of the LP-relaxation is equal to the optimal objective value with integrality constraints.

equivalent to adding the following constraints:

$$x_t = 0 \qquad \forall t \in \mathcal{B} : \exists\ v \in V^m(t) \text{ s.t. } v < v_s \in V^s(t) \text{ and } v < v_e \in V^e(t) \tag{16}$$

Note that this does not really add extra constraints to the model, it just removes some variables that are redundant. This means that the size of the model decreases through the number of variables. Addition of these constraints does not remove all symmetry. Symmetry can still occur in complete cycles of odd length [9] or by splitting a complete cycle in different half-cycle lengths.[10] Delorme, Manlove & Smeets (2023) suggest that a simple adjustment of constraints (6) can handle these problems. The intended adjustment however is described ambiguously: "This can be avoided in constraints (6) by only allowing a half-cycle $\langle v_1, ..., v_2 \rangle$ with size l to be matched with another half-cycle $\langle v_2, ..., v_1 \rangle$ with size l or, in case the index of v_1 is smaller than the index of v_2, with size l and $l-1$." as stated by (Delorme, Manlove & Smeets, 2023). It remains unclear how to achieve this without multiplying the number of constraints.[11] Furthermore, (Delorme, Manlove & Smeets, 2023)'s number of variables for the HCF is more than double the number of variables in the CF for some tested instances. This gives away that their claims of including only half-cycles in the model that can be completed by another halve, and removal of all symmetry are simply wrong.

Still, there do exist some unnecessary variables that can be removed by including only half-cycles that can be completed by another halve. These removals do not necessary remove all symmetry, but they make sure that a lot of half-cycles that can not be present in a feasible solution, are not even added to the model in the first place. A half-cycle $\langle v_1, ..., v_2 \rangle$ of length l would only have to be considered if it can be completed by another half-cycle of length l or $l-1$ in case that $v_1 < v_2$ or completed by a half-cycle of length l or $l+1$ in case that $v_1 > v_2$. From now on, we will refer to this procedure as the *extra* preprocessing for the HCF. Note that after this procedure, it is not possible anymore for HCF to contain more than double the amount of variables as the CF.[12] This shows that this preprocessing was not applied in Delorme, Manlove & Smeets (2023) and is therefore new to current literature. Section 5 discusses whether the time that it takes to preprocess these half-cycles is worth the actual reduction of variables that it achieves.

Unfortunately, variable reduction for the TCF cannot be performed as trivially as for the HCF. Variable reduction on the TCF can be applied to remove only a bit of symmetry. Furthermore, feasibility of variable reduction depends on the level of k.

Consider $k = 3n + 1$, for some $n \in \mathbb{N}$. Then, we can add the following constraint:

$$x_t = 0 \qquad \forall t \in \mathcal{B} : \exists\ v \in V^m(t) \cup V^e(t) \text{ s.t. } v < v_s \in V^s(t) \tag{17}$$

[9] $[A, B, C, D, E]$ can be split into $\langle A, B, C, D \rangle$ and $\langle D, E, A \rangle$ or $\langle A, B, C \rangle$ and $\langle C, D, E, A \rangle$ for $k \geq 6$

[10] $[A, B, C, D]$ can be split into $\langle A, B, C \rangle$ and $\langle C, D, A \rangle$ or $\langle A, B, C, D \rangle$ and $\langle D, A \rangle$ for $k \geq 5$

[11] The goal can be achieved by splitting $\mathcal{H}$ in smaller sets according to half-cycle length and then considering constraints (6) for each of these sets. However this would almost multiply the total number of constraints by $\lceil k/2 \rceil$.

[12] Symmetry for a complete cycle can only occur when both of the half-cycles can be matched with another half-cycle, resulting in two half-cycles being present in two possible complete cycles per case of symmetry. Additionally, half-cycles that can not be completed by another halve are not added to the model.

Table 2: Variable reduction for different values of k

Value of k	**TCs of length 1**	**TCs of length 2**	**TCs of length 3**
3	all present	×	×
4	all present	starting with lowest index	×
5	all present	starting/ending with highest index	×
6	all present	all present	×
7	all present	all present	starting with lowest index
8	all present	all present	starting/ending with highest index
9	all present	all present	all present

This simply ensures that we only consider third-cycles of maximum length $\lceil k/3 \rceil$ that start with the lowest indexed vertex present in the third-cycle. In this case for k, these third-cycles are only necessarily needed for a cycle of the maximum length k. This does not rule out any feasible solutions as the lowest index of the complete cycle of length k can always be chosen as the starting point of the longest third-cycle present in the complete cycle of length k.

Consider $k = 3n + 2$, for some $n \in \mathbb{N}$. Then, we can add the following constraint:

$$x_t = 0 \qquad \forall t \in \mathcal{B} : \exists\ v \in V^m(t) \text{ s.t. } v > v_s \in V^s(t) \text{ and } v > v_e \in V^e(t) \tag{18}$$

These constraints assures that we should only consider third-cycles of length $\lceil k/3 \rceil$ that start or end with their highest index. In this case for k, these third-cycles are only necessarily needed for a cycle of the maximum length k or length $k-1$. All other different lengths of complete cycles can be composed by exclusively using the other third-cycle lengths. Within complete cycles of length k or $k-1$ we are always able to chose the highest index as the intersection of the two largest third-cycles. Therefore, imposing the restriction that these third-cycles should either start or end in their highest indexed vertex, does not rule out any feasible solutions. Do note that these variable reductions are in line with the adjustment of constraints as described in Section 4.2, as in a complete cycle of maximum length k, the bigger third-cycles are assumed to start and end at the biggest indexed v_2 out of the triplet v_1, v_2, v_3. This configuration is always feasible.

Table 2 presents a comprehensive overview of the variable reduction for different values of k. A '×' indicates that no third-cycles of this length are present in the model because their length is strictly larger than $\lceil k/3 \rceil$.

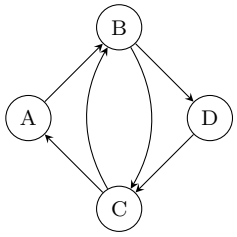
Lastly, not all constraints need to be added to the model. Within the HCF, one of constraints (6) only needs to be added when there actually exists a half-cycle starting in v_1 and ending in v_2 for $v_2 > v_1$. Similarly, in the TCF, one of constraints (12) only needs to added when there exists both a third-cycle starting in v_1 and ending in v_2 and one starting in v_2 and ending in v_3 for $v_2 > v_1, v_3$.

4.3.2 Vertex ordering

Both the HCF and TCF, but also the EA and EE have an interesting characteristic. Different ordering of vertices changes the number of variables needed in the model. Put differently, the indexing of vertices changes the number of possible half-cycles and third-cycles in the graph. Table 3 illustrates the variable reduction through vertex ordering for a specific compatibility

graph. This is the same example as presented in Delorme, Manlove & Smeets (2023).

Table 3: Variable reduction through vertex ordering



Feasible cycles	**Different ways of vertex ordering**		
	$B < C < D < A$	$A < B < C < D$	$A < D < B < C$
$[A, B, D, C]$	$\langle B, D, C\rangle + \langle C, A, B\rangle$	$\langle A, B, D\rangle + \langle D, C, A\rangle$	$\langle A, B, D\rangle + \langle D, C, A\rangle$
$[A, B, C]$	$\langle B, C, A\rangle + \langle A, B\rangle$	$\langle A, B, C\rangle + \langle C, A\rangle$	$\langle A, B, C\rangle + \langle C, A\rangle$
$[B, D, C]$	$\langle B, D, C\rangle + \langle C, B\rangle$	$\langle B, D, C\rangle + \langle C, B\rangle$	$\langle D, C, B\rangle + \langle B, D\rangle$
$[B, C]$	$\langle B, C\rangle + \langle C, B\rangle$	$\langle B, C\rangle + \langle C, B\rangle$	$\langle B, C\rangle + \langle C, B\rangle$
Total	6 different half-cycles	7 different half-cycles	8 different half-cycles

This table shows how different ordering rules lead to a different number of half-cycles that would be included in the model. Actually, finding the best vertex ordering, i.e. the one for which the number of half-cycles needed to represent every cycle is minimized, is an optimization problem in itself. However, the variable reduction is probably not worth the time such a problem would take to be solved to optimality. In order to achieve the best vertex ordering without a significant increase in computational time, Section 5 evaluates what rule of thumb for ordering achieves the greatest variable reduction.

4.4 Inclusion of altruistic donors

In Delorme, Manlove & Smeets (2023) the HCF, just as the new TCF in this thesis, is introduced for the KEP without altruistic donors. However, several approaches offer solution when it comes to incorporating altruistic donors into the model. The most intuitive way is probably to represent the chains also as cycles, by adding a dummy patient to each altruistic donor that is compatible with each of the normal, also called directed, donors. This approach however is expected to significantly increase the number of variables in the CF, HCF and TCF, as there would exist a lot more possible cycles or cycle segments within the compatibility graph. Furthermore, when the maximum chain length k' is set bigger than maximum cycle length k,[13] the mentioned formulations might struggle, as their model sizes grow with the maximum cycle length, which is related to k' in case the chains are represented as cycles. Dickerson et al. (2016) proposes a chain structure which only requires a polynomial number of variables and constraints, $O(k'n^2)$ and $O(k'n)$, respectively. This position-indexed chain-edge formulation (PICEF) models the chains by means of edges, whereas the cycles are modeled similarly to the CF, by associating a binary variable to every possible cycle. In this thesis, we introduce the new PICEF-HC and PICEF-TC, variants on the PICEF where the cycles are modelled according to the earlier discussed HCF and TCF, respectively.

[13]The maximum chain length is typically set higher than the maximum cycle length because chains do not necessarily require simultaneous surgeries, thereby reducing logistical constraints.

4.4.1 Position-indexed chain-edge formulation

First, we touch upon the PICEF as introduced by (Dickerson et al., 2016). First of all, in this section we will represent each vertex by its index, i.e.: $v_i = i$ as this benefits the readability of the different formulations. Let $\mathcal{D}(i,j)$ be the set of possible positions at which arc (i,j) may occur in a chain, i.e.:

$$\mathcal{D}(i,j) = \begin{cases} \{1\} & i \in \mathcal{N} \\ \{2,...,k'\} & i \in \mathcal{P} \end{cases}$$

Therefore, any arc leaving an altruistic donor can only be in the first position of a chain, and any arc leaving a patient donor pair may be in any position up to the chain length limit k', except for the first position. For each arc (i,j), we create a variable y_{ijd} for each $d \in \mathcal{D}(i,j)$, which takes value equal to 1 if and only if arc (i,j) is selected at position d of a chain. We use the same decision variables and sets as introduced for the CF in Section 4.1. The PICEF is then defined as follows:

$$\max \sum_{(i,j)\in\mathcal{A}} \sum_{d\in\mathcal{D}(i,j)} y_{ijd} + \sum_{c\in\mathcal{C}_k} |V(c)| z_c, \tag{19}$$

$$\text{s.t.} \sum_{j:(j,i)\in\mathcal{A}} \sum_{d\in\mathcal{D}(j,i)} y_{jid} + \sum_{c\in\mathcal{C}_k : i\in V(c)} z_c \leq 1, \qquad \forall i \in \mathcal{P}, \tag{20}$$

$$\sum_{j:(i,j)\in\mathcal{A}} y_{ij1} \leq 1, \qquad \forall i \in \mathcal{N}, \tag{21}$$

$$\sum_{j:(j,i)\in\mathcal{A} \text{ and } d\in\mathcal{D}(j,i)} y_{jid} \geq \sum_{j:(i,j)\in\mathcal{A}} y_{ij,d+1}, \qquad \forall i \in \mathcal{P}, d \in \{1,..,k'-1\}, \tag{22}$$

$$y_{ijd} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{A}, d \in \mathcal{D}(i,j), \tag{23}$$

$$z_c \in \{0,1\}, \qquad \forall c \in \mathcal{C}_k. \tag{24}$$

Objective function (19) represents the total number of transplants, assuming unitary weights, and is maximized. Constraints (20) make sure that each patient-donor pair is at most part of one selected cycle or connected to one incoming arc present in a selected chain. Furthermore, constraints (21) ensure that each altruistic donor is connected to at most one selected outgoing arc present in a selected chain. The flow inequalities (22) ensure that each patient-donor pair i has an outgoing arc at position $d+1$ of a selected chain only if i has an incoming arc at position d. Note that we use an inequality since the last patient-donor pair of a chain will have an incoming arc but no outgoing arc.

4.4.2 Reduced PICEF

As proposed by (Dickerson et al., 2016) we can reduce the number of variables included in the model. Let $r(i)$ be the shortest distance to any altruistic donor $j \in \mathcal{N}$, for all $i \in \mathcal{P}$. Now we know that any outgoing arc from i cannot appear at a position smaller than $r(i)+1$ in a chain and therefore we replace $\mathcal{D}(i,j)$ by $\mathcal{D}^{red}(i,j)$ which is defined as follows:

$$\mathcal{D}^{red}(i,j) = \begin{cases} \{1\} & i \in \mathcal{N} \\ \{r(i)+1, ..., k'\} & i \in \mathcal{P} \end{cases}$$

4.4.3 New PICEF-HC and PICEF-TC

Combining the PICEF as described in Section 4.4.1 with the HCF as described in Section 4.1 and TCF as described in Section 4.2, yields the new PICEF-HC and PICEF-TC, respectively. The complete formulations and interpretation of constraints and objectives can be found in Appendix A.

5 Results

This section first addresses the performance on computational experiments of all implemented methods for the KEP without altruistic donors. We tested performance on a variety of randomly generated instances with number of vertices $n \in \{50, 70, 100, 200, 400, 600\}$ and maximum cycle length $k \in \{3, 4, 5, 6, 7, 8, 9\}$ using my supervisor's instance generator. This generator creates compatibilities based on bloodtype and cPRA,[14] using data of Erasmus School of Economics. For each value of n, 10 instances were generated, resulting in a total of 60 instances. All experiments were run on a computer with following specifications: Intel Core i7-1255U processor, 4.7 GHz and 16GB of RAM. All models were coded in Java and a guide to reproduce all results can be found in Appendix B. All IP-models were solved with the use of Gurobi 11.0.0.

In Table 4 we report results on a first experiment evaluating impact of two vertex ordering rules. In the first approach, the vertices are re-indexed in order of descending total degree as suggested by Dickerson et al. (2016), abbreviated by 'desc.'. In the second approach, the vertices are re-indexed in order of ascending total degree , abbreviated by 'asc.'. The values reported in the table are the number of variables or constraints included in the model after the vertex ordering, divided by the number of variables or constraints included in the model after random vertex ordering. Therefore, a value below 1 means that the node ordering resulted in a decrease in the number of variables or constraints. The random vertex ordering is carried out 10 times per instance to create a more accurate comparison.

Table 4: Impact of node ordering in HCF, TCF, EA and EE for two ordering rules for $k = 3$ and 4.

		HCF				**TCF**				**EA**				**EE**			
k	n	desc.		asc.		desc.		asc.		desc.		asc.		desc.		asc.	
		var.	con.	var.	con.	var.	con.	var.	con.	var.	con.	var.	con.	var.	con.	var.	con.
3	50	0.73	0.82	0.90	1.13	1.00	1.04	1.00	1.30	0.96	0.66	1.02	1.42	0.86	0.86	1.08	1.08
	100	0.66	0.71	0.95	1.20	1.00	0.99	1.00	1.36	0.93	0.53	1.04	1.54	0.87	0.76	1.08	1.16
	200	0.64	0.67	0.91	1.19	1.00	0.98	1.00	1.42	0.91	0.47	1.04	1.64	0.88	0.69	1.06	1.16
4	50	0.86	0.74	0.98	1.09	0.72	0.62	0.88	1.48	0.86	0.50	1.07	1.54	0.60	0.65	1.24	1.21
	100	0.82	0.64	1.00	1.10	0.67	0.51	0.95	1.63	0.76	0.40	1.11	1.60	0.62	0.50	1.20	1.26
	200	0.80	0.63	1.01	1.10	0.63	0.44	0.91	1.68	0.70	0.37	1.11	1.59	0.58	0.43	1.19	1.22

[14] cPRA (calculated Panel Reactive Antibody) is a measure used in kidney exchange to estimate the likelihood that a recipient will react negatively to potential donor organs due to pre-existing antibodies, reflecting the percentage of the donor pool that would be incompatible with the recipient.

These results suggest that sorting the vertices in descending order of total degree reduces the model size for every instance of the HCF, EA and EE. This matches the suggestion of Dickerson et al. (2016). Very interesting to note is that, for the TCF, it turns out that for $k = 3$ and $n = 50$, this node ordering rule does actually slightly increase the model's size, while for all other instances the descending order reduces the size of the model. It does make a lot of sense that the number of variables in the TCF for $k = 3$ stays the same after any ordering rule as the third-cycles just consist of the single edges without any variable reduction being performed. The descending ordering rule results in a model with significantly smaller size for EE and EA, however it is not quite clear whether this will reduce computational time. The LP-relaxation bound would be of poorer quality after the descending vertex ordering, according to Delorme, Manlove & Smeets (2023). The authors even claim, but do not proof with any results, that the ascending ordering rule benefits computational times of the EE more. Later on in this section their claim will be tested, but for now, the ordering rule that results in smallest size will be used: ordering in descending degree.

Now we will move on to the computational performance of all implemented approaches. For each combination of n and k, performance is evaluated on the set of 10 instances. In the following tables, #opt denotes the number of instances within the set that were solved to optimality within the time limit of 1800 seconds. Between brackets is the number of instances for which the solving process was terminated early due to an out-of-memory error. T denotes the average total time needed to reach optimal solutions; this includes finding cycles, cycle-segments and paths, preprocessing, variable reduction and setting up the model. Note that average solving time T only includes instances that were actually solved to optimality. Furthermore, #var. and #cons. denote the average number of variables and constraints after vertex ordering and reduction techniques for each formulation, respectively.

In Table 5 and Table 6 we present the results obtained on the CF, HCF, TCF, EF, EA and EE for every $n \in \{50, 70, 100, 200\}$ and $k \in \{3, 4, 5, 6\}$. For the HCF, TCF, EA and EE, vertices are ordered using the descending ordering rule. Extra preprocessing is not applied to the HCF.

Table 5: Results of the CF, HCF and TCF for $k = 3, 4, 5$ and 6.

			CF				**HCF**				**TCF**			
k	n	obj	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.
3	50	14.0	10	0	95	50	10	0	1025	562	10	0	505	1105
	70	25.2	10	0	268	70	10	0	2425	1042	10	6	936	2775
	100	39.8	10	0	810	100	10	0	6433	2199	10	1	1982	8172
	200	90.2	10	0	4940	200	10	0	40232	8928	9	53	7584	55513
4	50	15.6	10	0	415	50	10	0	2017	610	10	0	1049	1381
	70	27.3	10	0	1884	70	10	0	5032	1168	10	1	2473	3735
	100	42.3	10	0	7139	100	10	0	14506	2457	10	2	6532	11781
	200	94.1	10	6	86398	200	10	1	95612	10270	8	34	40611	90587
5	50	16.0	10	0	1883	50	10	0	3466	618	10	0	2314	6335
	70	27.8	10	1	15064	70	10	0	11616	1192	10	2	5825	20495
	100	43.1	10	3	68289	100	10	1	40820	2521	10	16	17711	89239
	200	94.7	10	100	$1.6 * 10^6$	200	10	14	460740	10820	10	422	120643	959012
6	50	16.2	10	0	9898	50	10	0	6729	622	10	0	3296	6499
	70	27.9	10	5	139666	70	10	0	24724	1210	10	4	8412	20789
	100	43.1	10	24	709577	100	10	2	101079	2570	10	21	25584	90407
	200	95.0	0 (10)	-	-	-	10	68	$1.2 * 10^6$	11182	10	307	175377	971287

Table 6: Results of the EF, EA and EE for $k = 3, 4, 5$ and 6.

			EF				**EA**				**EE**			
k	n	obj	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.
3	50	14.0	10	0	481	16379	10	0	584	1152	10	0	148	305
	70	25.2	9	1	888	66018	10	12	1083	2846	10	0	393	530
	100	39.8	9	143	1883	297097	10	8	2329	8266	10	0	1157	1092
	200	90.2	1	928	7205	$4.0 * 10^6$	6	54	8924	57538	10	0	6459	3838
4	50	15.6	10	1	481	91932	10	0	608	1392	10	0	309	353
	70	27.3	9	7	888	346644	10	1	1170	4010	10	0	1042	704
	100	42.3	9	424	1883	$3.6 * 10^6$	10	57	2581	12750	10	0	3655	1595
	200	94.1	0(10)	-	-	-	9	239	10452	110069	10	5	27645	6893
5	50	16.0	10	11	481	503235	10	0	622	1511	10	0	447	381
	70	27.8	9	125	888	$5.8 * 10^6$	10	0	1218	4664	10	0	1686	799
	100	43.1	1 (9)	448	1883	$8.6 * 10^6$	10	3	2733	15521	10	1	6665	1901
	200	94.7	0 (10)	-	-	-	9	160	11565	143657	10	27	57960	9121
6	50	16.2	10	64	481	$2.7 * 10^6$	10	0	628	1574	10	0	520	393
	70	27.9	8 (2)	188	888	$7.1 * 10^6$	10	0	1235	4916	10	0	2130	833
	100	43.1	0 (10)	-	-	-	10	1	2798	16717	10	1	8763	2031
	200	95.0	0 (10)	-	-	-	10	47	12224	161089	10	39	81558	10437

The CF demonstrates dominant performance for $k = 3$ or 4. CF achieved optimal solutions for all instances within minimal computational time, indicating its efficiency in handling smaller maximum cycle lengths. However, a large n in combination with an increasing k, makes the computational burden grow, due to sharp increases in the number of variables. An increase in k exponentially increases the number of possible cycles that need to be considered in the model after all. Therefore, it cannot solve any of the instances for $n = 200$ and $k = 6$. This result clearly highlights the shortcoming of the CF: its sensitivity to an increase in k for large n.

The HCF shows a much more robust performance over different values of k and n. For $k = 3$ or 4, HCF can match the dominant performance of the CF. When k increases, the number of variables that need to be considered in the HCF grows way less fast than for the CF and therefore computational times remain tractable. This supports the basic goal of the HCF, which is to alter the CF in a way such that supplementary constraints dampen the exponential growth in the number of variables. These results suggest that the benefit of the variable reduction outweighs, by far, the addition of the supplementary constraints in the HCF as also concluded by Delorme, Manlove & Smeets (2023). Furthermore, the HCF and EE are the only formulations that are able to solve all 40 instances for each level of k and their performance seems very similar, both needing more than half a second for the exact same combinations of k and n. It is interesting to note that the size of the EE model is significantly smaller than the size of the HCF model for every combination of k and n, both for the number of variables and constraints. The HCF contains for $k = 6$, 10 times as much variables as the EE, while also having more constraints. The tight LP-relaxation of the HCF in comparison to the worse relaxation of the EE is probably the reason why the HCF still is able to match the performance of the EE.

The TCF turns out to struggle solving instances of larger size, showing significant increase in computational time for $n = 200$. This does not come as a total surprise, as the number of constraints is $O(n^3)$. Furthermore, an increase in k, which increases the number of variables, turns out to increase the number of constraints that need to be added to the model even more. The TCF is dominated for every combination of n and k by the HCF and EE. We conclude that

the new TCF does not outperform existing methods. However, it does show a significantly more robust performance to increase in k than CF.

The EF performs very poorly, as it manages to solve all instance only for $n = 50$. This can be explained by the fact that the number of constraints grows exponentially with k and by the poor quality of its LP-relaxation bound. For $n = 100$ or 200 and $k = 5$ or 6 only one instance was solved to optimality, out of 40, as for all other instances the solver ran out of memory. This is not very surprising, as in these cases the number of k-paths probably exceeds 10^7 which is simply too much to handle for the computer. Note that only for the EF, the number of variables is equal over all k, as this simply equals the number of edges in the graph. The # cons. column represents the average number of constraints over all instances for which the model was actually set up.[15]

Moving on to the EA, we see that performance on large instances ($n = 200$) is very poor, only solving 6 out of 10 for $k = 3$. However, when k increases, and therefore the number of variables, the computational times decrease and more instances are solved to optimality. One reason could be that the increased variable reduction for small values of k, even amplified by vertex ordering, deteriorates the LP-relaxation bound, as also pointed out by Delorme, Manlove & Smeets (2023). In general it holds over all n that performance improves for an increase in k. Even though the EA's performance is relatively poor, this compact formulation shows how a small increase in number of variables with respect to the EF, can result in a huge cut in the number of constraints. The number of constraints grows polynomially and this results in the performance not being negatively impacted by an increase in k.

Lastly, the EE shows dominant performance over all levels of n and k just as the HCF. An increase in k only results in a minor increase in the model's size, only slightly increasing the computational times. The EE seems to be a bit more sensitive to an increase in n while the HCF seems to be more sensitive to an increase in k, which makes sense regarding the theory behind the models' sizes.

The TCF is designed to be robust to an increase in k and therefore its performance is evaluated along the two best performing formulations EE and HCF for $n \in \{50, 70, 100\}$ and $k \in \{7, 8, 9\}$. Results are presented in Table 7.

Table 7: Results of the HCF, TCF and EE for $k = 7, 8$ and 9.

			HCF				**TCF**				**EE**			
k	n	obj	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.
7	50	16.2	10	0	11732	624	10	1	4745	6715	10	0	562	396
	70	27.9	10	1	64103	1212	10	4	14997	21813	10	0	2349	846
	100	43.1	10	8	300991	2589	10	32	51898	95535	10	1	9845	2089
8	50	16.2	10	0	24089	626	10	2	12269	12143	10	0	583	398
	70	27.9	10	3	139977	1218	10	24	44814	37778	10	0	2441	851
	100	43.1	10	31	808403	2617	10	191	192260	168676	10	1	10281	2118
9	50	16.2	10	1	44190	627	10	3	19574	12740	10	0	594	399
	70	27.9	10	9	426961	1220	10	41	74290	38378	10	0	2493	853
	100	43.1	6 (4)	42	$1.6 * 10^6$	2438	10	301	322480	170356	10	1	10433	2129

The results suggest that for large values of k the EE dominates the HCF and TCF. However, do note that no objectives, in 30 instances, increase when k is increased from 6 to 9. This makes

[15]Some instances resulted in an out-of-memory error before the model was set up.

one wonder if such large values of k would even be useful in practice.

Now, the three best performing models, the CF, HCF and EE are evaluated for larger instances $n \in \{400, 600\}$ and $k \in \{3, 4, 5\}$. Results are presented in Table 8.

Table 8: Results of the CF, HCF and EE for $k = 3, 4$ and 5 evaluated on larger instances.

			CF				**HCF**				**EE**			
k	n	obj	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.
3	400	219.0	10	5	40051	400	10	2	239624	38911	10	3	48306	18110
	600	362.7	10	29	120905	600	10	6	675321	91546	10	16	142087	44764
4	400	227.0	10	185	$1.5 * 10^6$	400	10	27	834709	45127	10	427	297763	36502
	600	367.5	7	1113	$6.7 * 10^6$	600	10	199	$2.7 * 10^6$	106149	0	-	-	-
5	400	-	0 (10)	-	-	-	1 (9)	460	$3.7 * 10^6$	44994	4	988	677844	49120
	600	-	0 (10)	-	-	-	0 (10)	-	-	-	0	-	-	-

The HCF is the only formulation able to solve all instances for $k = 3$ or 4 and shows dominant computational times in doing so. This certifies its robustness to an increase in n. However, when $k = 5$, the HCF is only able to solve 1 instance to optimality due to the sharp increase in the number of variables. The number of variables for the EE remains somewhat tractable for $k = 5$ and the EE is therefore able to still solve 4 instances for $n = 400$. This can be explained by its strong robustness to increase in k. The CF is dominated by the HCF for all the tested instances. Interesting to note is that, for $k = 3$, the model sizes of the CF are significantly smaller than the sizes of the HCF. Still, the HCF shows superior computational times.[16]

Additionally, we evaluate whether an ascending or descending vertex ordering benefits the EE. In all previous conducted experiments, descending vertex ordering was used. Table 9 presents these results along the computational performance of the EE after ascending vertex ordering for $n \in \{100, 200, 400, 600\}$ and $k \in \{3, 4, 5\}$.

Table 9: Ascending and descending vertex ordering for EE.

			Ascending				**Descending**			
k	n	obj	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.
3	100	39.8	10	0	1437	1654	10	0	1157	1092
	200	90.2	10	1	7770	6460	10	0	6459	3838
	400	219.0	10	6	54472	30442	10	3	48306	18110
	600	362.7	10	18	154848	70287	10	16	142087	44764
4	100	42.3	10	1	7047	4010	10	0	3655	1595
	200	94.1	10	19	56255	19418	10	5	27645	6893
	400	227.0	10	716	525079	93707	10	427	297763	36502
	600	367.5	0	-	-	-	0	-	-	-
5	100	43.1	10	4	21221	5675	10	1	6665	1901
	200	95.0	10	150	206766	26437	10	27	57960	9121
	400	-	0	-	-	-	4	988	677844	49120
	600	-	0	-	-	-	0	-	-	-

These results refute the claim that the EE would show better computational performance after ascending vertex ordering, made by (Delorme, Manlove & Smeets, 2023). The descending vertex ordering shows dominant computational times and is able to solve more instances. This is probably due to the smaller size of the model after descending ordering, obtaining the maximum

[16]More constraints can sometimes decompose the problem into smaller, more manageable subproblems that the solver can handle more efficiently. Furthermore, extra constraints could improve the solver's ability to prune the search tree, eliminating infeasible or sub optimal branches earlier in the process. Also, the finding of all cycles might take more time than finding all half-cycles.

reduction with respect to ascending ordering for $n = 200$ and $k = 5$. In these instances the descending ordering rule leads to a model containing only 28% of variables and 35% of constraints of those of the model after the ascending ordering rule: a huge reduction in size that turns out to have a big impact on computational times.

Moving on, we present results on our extra HCF preprocessing technique as proposed in Section 4.3.1. This reduction procedure ensures that only half-cycle are included that can be matched with at least one other half-cycle. Results are presented in Table 10, compared to the normal procedure, as applied in all previous experiments.

Table 10: Extra preprocessing for HCF in comparison to the normal procedure.

			Extra preprocessing				**Normal procedure**			
k	n	obj	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.
3	100	39.8	10	0	1035	325	10	0	6433	2199
	200	90.2	10	0	5840	1100	10	0	40232	8928
	400	219.0	10	2	45685	6033	10	2	239624	38911
	600	362.7	10	8	135569	15265	10	6	675321	91546
4	100	42.3	10	0	3912	518	10	0	14506	2457
	200	94.1	10	1	30106	2282	10	1	95612	10270
	400	227.0	10	33	325107	12587	10	27	834709	45127
	600	367.5	10	237	$1.2 * 10^6$	33794	10	199	$2.7 * 10^6$	106149
5	100	43.1	10	1	19292	668	10	1	40820	2521
	200	95.0	10	17	247623	3003	10	14	460740	10820
	400	-	1 (9)	539	$2.6 * 10^6$	14731	1 (9)	460	$3.7 * 10^6$	44994
	600	-	0 (10)	-	-	-	0 (10)	-	-	-

These results suggest a similar computational performance across both approaches. The larger solving times for the extra processing are due to the time such extra preprocessing methods take. Although, especially for low values of n and k, the extra preprocessing results in significant reduction of model's size, the actual solving time of the model does not get affected much. A reason could be that the HCF is able to prune solutions containing redundant variables early on in the process, not demanding any preprocessing that makes sure these variables are not even added in the first place.

5.1 Inclusion of altruistic donors

This subsection addresses the performance of all implemented methods for the KEP including altruistic donors. We tested performance on a variety of randomly generated instances with number of patient-donor pairs $n \in \{50, 100, 200, 400, 600\}$ and maximum cycle length $k \in \{3, 4, 5\}$ again using my supervisor's instance generator. Each instance contains $0.1 * n$ altruistic donors, i.e. 1 altruistic donor for every 10 patient-donor pairs. Maximum chain length $k' = 10$ for all conducted computational experiments.

Within the National Kidney Registry (NKR) from 2008 through May 2016, 94% of all initiated kidney exchange chains had length smaller or equal to 10, and all exchange cycles had length smaller or equal to 5 (Cowan et al., 2017). The longest chain initiated during this period involved 35 transplants. Due to the risk on a chain interruption, we decide that $k' = 10$ is an appropriate, and in practice applicable, maximum chain length. Although broken chains turn out to happen infrequent (Cowan et al., 2017), they can have a big impact when they break chains still involving a lot of patient-donor pairs.

First, performance of the PICEF-C, PICEF-HC and PICEF-TC is evaluated for relatively smaller instances with $n \in \{50, 100, 200\}$ and $k \in \{3, 4, 5\}$. Vertices are ordered according to descending degree for all instances. [17] Results are presented in Table 11.

Table 11: Results of the PICEF-C, PICEF-HC and PICEF-TC for $k = 3, 4$ and 5.

			PICEF-C				**PICEF-HC**				**PICEF-TC**			
k	n	obj	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.
3	50	22.0	10	0	2484	505	10	0	3323	1035	10	0	2893	1430
	100	50.2	10	0	12382	1010	10	0	17226	3325	10	1	13877	9605
	200	123.1	10	2	62663	2020	10	2	91376	12427	10	4	65890	69486
4	50	22.0	10	0	2904	505	10	0	4382	1096	10	0	3346	1802
	100	50.2	10	0	17597	1010	10	0	27119	3657	10	2	17322	13522
	200	123.1	10	9	176666	2020	10	3	183038	14147	10	21	91798	124783
5	50	22.0	10	0	5336	505	10	0	5719	1104	10	0	4903	6762
	100	50.2	10	2	63627	1010	10	1	44331	3739	10	7	29863	85554
	200	123.1	10	189	$2.5 * 10^6$	2020	10	22	515890	14883	10	361	218904	991998

These results clearly show similarities to the results presented in Table 5. The PICEF-C, just as the CF shows the best performance for $k = 3$ or 4, with computational times increasing sharply for $k = 5$ in combination with $n = 200$. The PICEF-HC shows dominance across all instances, while PICEF-TC struggles when $n = 200$. However, it is interesting to note that the PICEF-TC shows improved performance over the TCF on the KEP without altruistic donors, as the PICEF-TC solves all instances, also within a shorter timeframe. It seems like the problem structure becomes more favorable to the PICEF-TC when altruistic donors are added to the problem. Performance of the PICEF-C and PICEF-HC is similar, but a bit worse, compared to their counterparts CF and HCF. Interestingly enough, optimal objective values do not increase with k. This could be explained by the fact that the longer paths or cycles in the compatibility graph can now already be presented in a solution by a chain, initiated by one of the altruistic donors. It is therefore quite unlikely that an increase in k offers better solutions.

Results on the PICEF-C and PICEF-HC for $n \in \{400, 600\}$ and $k \in \{3, 4, 5\}$ are presented in Table 12.

Table 12: Results of the PICEF-C and PICEF-HC for $k = 3, 4$ and 5, evaluated on larger instances.

			PICEF-C				**PICEF-HC**			
k	n	obj	# opt	T	# var.	# cons.	# opt	T	# var.	# cons.
3	400	257.9	10	15	277696	4040	10	17	456685	46832
	600	404.7	10	65	688619	6060	10	62	$1.2 * 10^6$	108221
4	400	257.9	10	228	$1.7 * 10^6$	4040	10	58	$1.2 * 10^6$	55260
	600	404.7	2	1425	$8.1 * 10^6$	6060	10	239	$3.7 * 10^6$	128707
5	400	-	0 (10)	-	-	-	0 (10)	-	-	-
	600	-	0 (10)	-	-	-	0 (10)	-	-	-

The PICEF-HC is able to solve all instances for $k = 3$ or 4, while the PICEF-C fails to do so. For $k = 3$, both methods show similar performance, now quite a bit worse than their counterparts CF and HCF as can be seen in Table 8. Computational times and number of solved

[17] PICEF-C, just as PICEF-HC and PICEF-TC, also has a different model size for different orderings of vertices.

instances for $k = 4$ again show that the half-cycle modelling structure in the PICEF-HC is more robust to an increase in k than the full cycle structure in PICEF-C.

6 Conclusion

This thesis presents a new IP model for the KEP called the third-cycle formulation (TCF). In this model, a cycle is represented by three compatible third-cycles, aiming to improve performance of the existing half-cycle formulation (HCF). We provide a clear explanation of the reduction procedures, including symmetry reduction and vertex ordering, applicable to both the HCF and TCF. Performance of the HCF and TCF is evaluated along a spectrum of existing models: the cycle (CF), edge, edge-assignment (EA) and extended edge formulation (EE). The performance of EE and EA is improved by using different vertex ordering rules, something that has not been empirically tested before.

The TCF, despite being designed to improve upon HCF, struggles with larger instance sizes. TCF's number of constraints grows cubically with n, and an increase in maximum cycle length k amplifies this growth. Consequently, TCF shows significant computational times for larger instances, failing to outperform HCF and EE. Nevertheless, TCF shows much more robustness to increasing k compared to the CF.

Furthermore, HCF and EE stand out as the most robust and dominant formulations for the KEP without altruistic donors. HCF's ability to solve larger instances efficiently emphasizes its practicality for real-world applications. While TCF shows potential, further improvement is needed to solve instances of larger size. However, we think that it would be very hard to obtain an improved TCF. The CF, despite its limitations, remains effective for smaller cycle lengths and instance sizes. In general, if problem size becomes larger through an increase in n, the HCF with descending vertex ordering is preferred, and if through k, the EE with descending vertex ordering is preferred. Results show that this descending vertex ordering rule reduces model's sizes for the HCF, TCF, EA, EE drastically. Computational expirements refute the claim made by Delorme, Manlove & Smeets (2023) that an ascending vertex ordering would benefit the EE. For the HCF, new preprocessing techniques, i.e. only incorporating each half-cycle that can be matched with another one, turns out to deteriorate its performance. The reduction in size of the model does not outweigh the additional preprocessing time needed to set up the model.

We observed that increasing k to a value higher than 6 does not increase the number of transplants for all instances up to size of 100 patient-donor pairs. This suggests that if such a large k would be allowed, it could be worth attempting to solve the instance for a smaller value of k first, and asses afterwards by means of an upper bound whether the possible missed number of transplants is worth the extra logistic challenges for an increase in k. This upper bound can be obtained either through dropping the maximum cycle length, for which the problem becomes solvable in polynomial time, or relaxation of the integrality constraints for this higher value of k.

Furthermore, we introduced two new position-indexed chain edge formulations (PICEF) for the KEP to incorporate altruistic donors: the PICEF-Half-Cycle (PICEF-HC) and PICEF-Third-Cycle (PICEF-TC). The PICEF-Cycle, PICEF-HC and PICEF-TC show similar performance to their counterparts CF, HCF, and TCF for the KEP without altruistic donors. The

PICEF-HC shows dominance across all instances, being much more robust to an increase in k than the PICEF-C. Optimal objective values do not increase with k, probably due to the longer maximum length on a chain, initiated by one of the altruistic donors. This argues for a similar solving strategy as discussed in the previous paragraph.

A suggestion for further research would be to investigate whether the HCF and TCF are applicable to the stochastic KEP, which takes possible vertex and arc failures into account, as modeled by Alvelos et al. (2019) and McElfresh et al. (2019). Also, it would be interesting to examine their ability to handle hierarchical objectives, as is already efficiently done for the CF (Delorme, García et al., 2023). Furthermore, performance of the EE could maybe be improved by using more sophisticated vertex ordering rules. Lastly, it would be worth investigating whether there exist ways to remove absolutely all symmetry within the HCF, avoiding multiplication of the model's majority of constraints.

References

Abraham, D. J., Blum, A. & Sandholm, T. (2007). Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th acm conference on electronic commerce* (pp. 295–304).

Alvelos, F., Klimentova, X. & Viana, A. (2019). Maximizing the expected number of transplants in kidney exchange programs with branch-and-price. *Annals of Operations Research*, *272*(1), 429–444.

Axelrod, X., Schnitzler. (2018). An economic assessment of contemporary kidney transplant practice. *American Journal of Transplantation, 18(5), 1168-1176*.

Boulware, L., Troll, M., Plantinga, L. & Powe, N. R. (2008). The association of state and national legislation with living kidney donation rates in the united states: a national study. *American Journal of Transplantation*, *8*(7), 1451–1470.

Constantino, M., Klimentova, X., Viana, A. & Rais, A. (2013). New insights on integer-programming models for the kidney exchange problem. *European Journal of Operational Research*, *231*(1), 57–68.

Cowan, N., Gritsch, H., Nassiri, N., Sinacore, J. & Veale, J. (2017). Broken chains and reneging: a review of 1748 kidney paired donation transplants. *American Journal of Transplantation*, *17*(9), 2451–2457.

Delorme, M., García, S., Gondzio, J., Kalcsics, J., Manlove, D. & Pettersson, W. (2023). New algorithms for hierarchical optimization in kidney exchange programs. *Operations Research*.

Delorme, M. & Iori, M. (2020). Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. *INFORMS Journal on Computing*, *32*(1), 101–119.

Delorme, M., Manlove, D. & Smeets, T. (2023). Half-cycle: A new formulation for modelling kidney exchange problems. *Operations Research Letters*, *51*(3), 234-241.

Dickerson, J. P., Manlove, D. F., Plaut, B., Sandholm, T. & Trimble, J. (2016). Position-indexed formulations for kidney exchange. In *Proceedings of the 2016 acm conference on economics and computation* (p. 25–42). New York, NY, USA: Association for Computing Machinery.

Johnson, F., Allen. (2008). Early experience of paired living kidney donation in the united kingdom. *Transplantation 86(12):p 1672-1677*.

Lam, E. & Mak-Hau, V. (2020). Branch-and-cut-and-price for the cardinality-constrained multi-cycle problem in kidney exchange. *Computers & Operations Research*, *115*, 104852.

McElfresh, D. C., Bidkhori, H. & Dickerson, J. P. (2019). Scalable robust kidney exchange. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 1077–1084).

Riascos-Álvarez, L. C., Bodur, M. & Aleman, D. M. (2024). A branch-and-price algorithm enhanced by decision diagrams for the kidney exchange problem. *Manufacturing & Service Operations Management*, *26*(2), 485–499.

Roth, A. E., Sönmez, T. & Ünver, M. U. (2007). Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review*, *97*(3), 828–851.

Roth, A. E., Sönmez, T. & Utku Ünver, M. (2005). Pairwise kidney exchange. *Journal of Economic Theory*, *125*(2), 151-188.

A PICEF-HC and PICEF-TC

Combining the PICEF as described in Section 4.4.1 with the HCF as described in Section 4.1 yields the following new PICEF-HC:

$$\max \sum_{(i,j)\in\mathcal{A}} \sum_{d\in\mathcal{D}(i,j)} y_{ijd} + \sum_{h\in\mathcal{H}} (|V^m(h)|+1)x_h, \tag{25}$$

$$\text{s.t.} \sum_{j:(j,i)\in\mathcal{A}} \sum_{d\in\mathcal{D}(j,i)} y_{jid} + \sum_{h\in\mathcal{H}:i\in V^s(h)\cup V^e(h)} 0.5x_h + \sum_{h\in\mathcal{H}:i\in V^m(h)} x_h \leq 1, \qquad \forall i\in\mathcal{P}, \tag{26}$$

$$\sum_{j:(i,j)\in\mathcal{A}} y_{ij1} \leq 1, \qquad \forall i\in\mathcal{N}, \tag{27}$$

$$\sum_{j:(j,i)\in\mathcal{A} \text{ and } d\in\mathcal{D}(j,i)} y_{jid} \geq \sum_{j:(i,j)\in\mathcal{A}} y_{ij,d+1}, \qquad \forall i\in\mathcal{P}, d\in\{1,..,k'-1\}, \tag{28}$$

$$\sum_{h\in\mathcal{H}:i\in V^s(h),j\in V^e(h)} x_h = \sum_{h\in\mathcal{H}:j\in V^s(h),i\in V^e(h)} x_h, \qquad \forall i,j\in\mathcal{V}: j>i, \tag{29}$$

$$y_{ijd}, \in \{0,1\} \qquad \forall (i,j)\in\mathcal{A}, d\in\mathcal{D}(i,j), \tag{30}$$

$$x_h \in \{0,1\}, \qquad \forall h\in\mathcal{H}. \tag{31}$$

Objective function (25) equals objective function (19) of PICEF, with objective function of the CF (1) replaced by objective function (4) of the HCF. This objective maximizes the number of transplants. Then, constraints (26) are the adjusted capacity constraints on each directed donor, combining constraints (5) and (20). All the other constraints are copied exactly from the PICEF and HCF. Interpretation of these constraints can be easily deducted from the description in Section 4.4.1 and Section 4.1, respectively. Furthermore, combining the PICEF as described in Section 4.4.1 with the TCF as described in Section 4.2 results in the following new PICEF-TC:

$$\max \sum_{(i,j)\in\mathcal{A}} \sum_{d\in\mathcal{D}(i,j)} y_{ijd} + \sum_{t\in\mathcal{T}} (|V^m(t)|+1)x_t + 2\sum_{c\in\mathcal{C}_2} z_c, \tag{32}$$

$$\text{s.t.} \sum_{j:(j,i)\in\mathcal{A}} \sum_{d\in\mathcal{D}(j,i)} y_{jid} + \sum_{t\in\mathcal{T}:i\in V^s(t)\cup V^e(t)} 0.5x_t + \sum_{t\in\mathcal{T}:i\in V^m(t)} x_t + \sum_{c\in\mathcal{C}_2:i\in V(c)} z_c \leq 1, \qquad \forall i\in\mathcal{P}, \tag{33}$$

$$\sum_{j:(i,j)\in\mathcal{A}} y_{ij1} \leq 1, \qquad \forall i\in\mathcal{N}, \tag{34}$$

$$\sum_{j:(j,i)\in\mathcal{A} \text{ and } d\in\mathcal{D}(j,i)} y_{jid} \geq \sum_{j:(i,j)\in\mathcal{A}} y_{ij,d+1}, \qquad \forall i\in\mathcal{P}, d\in\{1,..,k'-1\}, \tag{35}$$

$$\sum_{t\in\mathcal{T}:i\in V^s(t)} x_t = \sum_{t\in\mathcal{T}:i\in V^e(t)} x_t, \qquad \forall i\in\mathcal{V}, \tag{36}$$

$$\sum_{t\in\mathcal{T}:i\in V^s(t),j\in V^e(t)} x_t + \sum_{t\in\mathcal{T}:j\in V^s(t),l\in V^e(t)} x_t \leq 1 + \sum_{t\in\mathcal{T}:l\in V^s(t),i\in V^e(t)} x_t, \qquad \forall i,j,l\in\mathcal{V}: j>l, j>i, \tag{37}$$

$$x_t \in \{0,1\}, \qquad \forall t\in\mathcal{T}, \tag{38}$$

$$y_{ijd} \in \{0,1\}, \qquad \forall (i,j)\in\mathcal{A}, d\in\mathcal{D}(i,j), \tag{39}$$

$$z_c \in \{0,1\}, \qquad \forall c\in\mathcal{C}_2. \tag{40}$$

Now, objective function (32) equals objective function (19) of PICEF, with objective function of the CF (1) replaced by objective function (9) of the TCF. This objective maximizes the number of transplants. Furthermore, constraints (33) are the adjusted capacity constraints on each directed donor, combining constraints (10) and (20). All the other constraints are copied exactly from the PICEF and TCF. Interpretation of these constraints can be easily deducted from the description in Section 4.4.1 and Section 4.2, respectively.

B Programming Code

This appendix contains a guide to reproduce all results stated in Section 5. To solve a set of instances for the KEP without altruistic donors, run the method: *produceResults*(String size, int k, String method, double timeLimit, boolean isAscending), in which:

- String size determines the size of the set of instances. String S is associated to $n = 50$, String M is associated to $n = 70$, String L is associated to $n = 100$ and String XL is associated to $n = 200$. Finally, String 400 is associated to $n = 400$ and String 600 is associated to $n = 600$.
- int k sets maximum cycle length k.
- String method sets the solving method. String Cycle is associated to CF, String HalfCycle is associated to the HCF, String ThirdCycle is associated to the TCF, String Edge is asso-

ciated to the EF, String EdgeAssignment is associated to the EA and String ExtendedEdge is associated to the EE.

- double timeLimit sets the time limit on total solving time in seconds.
- boolean isAscending sets the vertex ordering to ascending if true, if false to descending.

This method prints the wanted results: average objective, average solving time, proportion of unsolved instances, average number of variables and average number of constraints. Average solving time includes the solving times of models that were terminated due to reaching the time limit or an out-of-memory error (returned solving time is then set to the time limit). Together with the proportion of unsolved instances, it is easy to determine the average solving time over all solved instances. Proportion of unsolved instances is given as a number $x.y$, where x denotes the number of instances terminated due to an out-of-memory error and y denotes the number of instances not solved due to the time limit. Average number of constraints and variables are calculated over all instances for which the model was set up. If the model was not set up due to an out-of-memory error, both the number of variables and constraints equal 0.

To obtain results for the KEP with altruistic donors, run the method: *produceResultsPICEF*(String size, int maxCycle, int maxChain, String method, int timeLimit) for which

- String size determines the size of the set of instances. For the different levels of n, the input String should be n.
- int maxCycle sets maximum cycle length k
- int maxChain sets maximum chain length k'
- String method sets the solving method. String Cycle is associated to PICEF-C, String HalfCycle is associated to the PICEF-HC, String ThirdCycle is associated to the PICEF-TC.
- double timeLimit sets the time limit on total solving time in seconds.

For all formulations a descending node ordering is used and results are printed in the same manner.

To obtain number of variables for random vertex ordering run the following method: *produceResultsRandomVertexOrdering*(String size, int k, String method, int numIterations)

- String size, determines the size of the set of instances. String S is associated to $n = 50$, String M is associated to $n = 70$, String L is associated to $n = 100$ and String XL is associated to $n = 200$. Finally, String 400 is associated to $n = 400$ and String 600 is associated to $n = 600$.
- int k sets maximum cycle length k.
- String method sets the solving method. String Cycle is associated to CF, String HalfCycle is associated to the HCF, String ThirdCycle is associated to the TCF, String Edge is associated to the EF, String EdgeAssignment is associated to the EA and String ExtendedEdge is associated to the EE.

- int numIterations sets the number of times the random vertex ordering is carried out per instance.

Average number of variables and constraints over all instances and iterations are clearly printed by this method. Lastly, to obtain results on the HCF with extra preprocessing, run the method *solveHalfCycle*(int[][] compatibility, int maxExchange, double timeLimit, boolean isAscending, boolean isRandom, boolean extraPreprocessing) where:

- int[][] compatibility is a two-dimensional matrix representing the compatibility graph.
- int maxExchange sets maximum cycle length k.
- double timeLimit sets the time limit in seconds.
- boolean isAscending sets vertex ordering to ascending if true.
- boolean isRandom sets vertex ordering to random if true.
- boolean extraPreprocessing, if true the extra preprocessing is applied.

Results are again printed in the same manner as for previous methods.