Cutting Plane Approaches for the Robust Kidney Exchange Problem

Citation for published version (APA):

Document license:
CC BY

DOI:
10.1016/j.cor.2023.106470

Document status and date:
Published: 01/02/2024

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at: openaccess@tue.nl
providing details and we will investigate your claim.
Cutting Plane Approaches for the Robust Kidney Exchange Problem

Danny Blom, Christopher Hojny, Bart Smeulders

Department of Mathematics and Computer Science, Eindhoven University of Technology, Groene Loper 5, 5612 AZ Eindhoven, The Netherlands

ARTICLE INFO

Keywords:
- Interdiction games
- Robust optimization
- Kidney exchange

ABSTRACT

Renal patients who have a willing but incompatible donor can decide to participate in a kidney exchange program (KEP). The goal of a KEP is to identify sets of incompatible pairs that can exchange donors, leading to compatible transplants for each recipient. Significant uncertainty is involved in this process, as planned transplants may be canceled for many reasons. It is therefore crucial to take into account failures while planning exchanges. In this paper, we consider a robust variant of this problem with recourse studied in the literature that takes into account failures of donors or recipients. This problem belongs to the class of defender–attacker–defender (DAD) models. We propose a cutting plane method for solving the attacker–defender subproblem based on two commonly used mixed-integer programming formulations for kidney exchange problems. Our results imply a running time improvement of one order of magnitude compared to the state-of-the-art. Moreover, our cutting plane methods can solve many previously unsolved instances. Furthermore, we propose a new practical policy for recourse in KEPs and show that the robust optimization problem concerning this policy is tractable for small to mid-size KEPs in practice.

1. Introduction

In the final stage of chronic kidney disease, patients suffer from end-stage renal disease. The most preferred treatment option for this disease is a kidney transplant. However, the supply of healthy kidneys from deceased donors does not adequately meet the needs of these patients. An alternative is living-donor donation, e.g., by a relative or friend who is willing to donate one of their two kidneys to a specific patient. This is possible since only one healthy kidney is necessary to ensure sufficient kidney function. Transplants from living donors also offer better long-term outcomes for the recipient compared to deceased donor grafts, with limited risk to the donor (Davis and Delmonico, 2005). However, to allow for donations, recipients must be medically compatible with the donor. Incompatibilities can be caused, among other reasons, by conflicting blood or tissue type. Recipients might thus be unable to receive a transplant from their intended donor.

To overcome incompatibilities between a patient and its related donor, Rapaport (1986) introduced kidney exchange programs (KEPs), which have been further popularized by a series of seminal papers in the field (Roth et al., 2004, 2005, 2007). In a KEP, a set of incompatible patient–donor pairs is given, and every donor is willing to give one of their kidneys to any patient as long as their paired recipient receives a transplant in return from some other donor. In the simplest case, one is thus looking for two incompatible donor–recipient pairs (p1, p2) such that the kidney of p1’s donor is compatible with the recipient of p2 and vice versa. Once such a match is identified, the patients from p1 and p2 can receive the kidney from the other pair’s donor. We call such an exchange of donors a 2-way exchange as it includes two patient–donor pairs. Of course, this idea can be generalized to k-way exchanges, where the exchange of donors occurs in a cyclic way between k patient–donor pairs. In many real-life KEPs, an upper bound K is used for the size of a cycle, see Biró et al. (2019). As transplants within an exchange are usually performed simultaneously to remove the risk of donors reneging once their paired patient has received a kidney, long exchanges are logistically challenging and thus avoided.

A kidney exchange program can also include non-directed donors (NDDs) as studied by Morrissey et al. (2005) and Roth et al. (2006). These are donors who are willing to donate to any recipient, while not having a paired patient and thus not requiring a return transplant. An NDD can start a chain of transplants, in which the donor of the rth pair in the chain donates a kidney to the recipient of the (i + 1)th pair. The donor of the last pair in the chain can donate their kidney to a recipient on the deceased donor waiting list, or become a bridge donor, functioning as a non-directed donor in a future KEP run. In practice, it is often also assumed that the maximum chain length is bounded by some integer L.

The basic KEP model can be regarded as a directed graph G = (V, A) whose vertices v ∈ V correspond to recipient–donor pairs and non-directed donors, and an arc (i, j) ∈ A corresponds to compatibility
between the donor of \( i \) and the recipient of \( j \). Notice that each feasible \( k \)-way exchange corresponds to a cycle in this graph \( G \). Therefore, we refer to \( k \)-way exchanges as \( k \)-cycles. A feasible KEP solution is then a set of pairwise vertex-disjoint cycles and chains in \( G \) of lengths respecting the upper bounds. A common objective in kidney exchange problems is to maximize the number of transplants that can be realized. Fig. 1 illustrates an example of a kidney exchange program and a feasible KEP solution.

Basic KEP models assume that all identified matches proceed to transplant. In practice, however, this is usually not the case. For example, Dickerson et al. (2019) reports that 93\% of proposed matches failed in the initial years of the UNOS program. In the NHS Living Kidney Sharing Scheme, 31\% of identified matches did not proceed to transplant in the period from 2008 until 2022 (Robb and Shanmugaranjan, 2023).

Such failures can occur for a variety of reasons, e.g., when additional medical incompatibilities are identified during the time between the matching run and the actual surgeries. Furthermore, recipients and donors may decide to (temporarily) withdraw from KEPs due to health issues. There exists a growing literature taking into account failures, which we discuss in more detail in Section 1.2. We focus in particular on Carvalho et al. (2021), who introduced a robust optimization approach that addresses such donor or recipient (i.e., vertex) failures and aims to optimize a minimum guarantee for recipients that were selected for transplant in advance of the failure event. They consider a three-stage model that belongs to the class of defender-attacker–defender (DAD) models, as optimizing worst-case behavior can be interpreted as playing a sequential game against an informed adversary.

In the first stage, the KEP owner (defender) identifies a feasible kidney exchange solution, which we refer to as the initial solution. Subsequently, it is anticipated that up to a specified number \( B \) of pairs and/or NDDs might withdraw from the KEP. In terms of DAD models, this corresponds to an adversarial player selecting those pairs and NDDs that disrupt the initial solution as much as possible. Therefore we refer to these withdrawals as an attack pattern or attack.

In the third and final stage, a new kidney exchange solution is identified using only the remaining pairs and NDDs. This kidney exchange solution is called the recourse solution and it is selected such that it includes the largest number of patients also included in the initial solution. Additional restrictions on the set of possible recourse solutions are captured through recourse policies.

The objective of the model is to select the initial solution that maximizes the worst-case number, concerning all attacks, of patients involved in both the initial and recourse solution. The idea behind this objective is to diminish the risk that recipients selected for transplant in the initial stage will end up without a transplant due to vertex withdrawals.

1.1. Our contributions

This paper builds on the work of Carvalho et al. (2021), and the algorithmic literature on (D)AD models. Our main contributions are the following.

- We propose a new approach based on cutting planes to solve the second-stage problem, i.e., the interdiction problem faced by the attacker. Our computational results show that this new approach outperforms the branch-and-bound algorithm of Carvalho et al. (2021) by an order of magnitude. This improvement can be observed both with regard to the average computation times and the number of instances that could be solved within the time limit of one hour.
- We make use of the position-indexed chain-edge formulation (PICEF) for kidney exchange that is compact concerning the set of chains. We show that the resulting formulation of the second stage is significantly stronger than those obtained when using the traditional cycle-chain formulation. Specifically, we show the strength of interdiction cuts depends on the underlying formulation, provided we consider the form proposed by Fischetti et al. (2019).
- We introduce a new policy for recourse in kidney exchange programs, the Fix Successful Exchanges (FSE) policy, where cycles and chains of the initial solution that are not affected by the subsequent attack will also be selected for the recourse solution. This recourse policy further protects recipients involved in a selected cycle or chain of the initial solution and has additional logistical benefits. We further motivate the FSE policy in Section 2. Our computational results show that imposing the FSE policy for our recourse options leads to only a small loss in the attainable objective value.

The remainder of the paper is structured as follows. We review relevant literature in Section 1.2. The algorithmic framework for solving the attacker–defender subproblem is laid out in Section 2, in which we also consider two policies for recourse for the third stage of the defender–attacker–defender model. The implementation of this framework with respect to the cycle-chain and position-indexed chain edge formulations is made more concrete in Section 3. We finish by presenting our computational efforts in Section 4 and a conclusion in Section 5.

1.2. Literature review

In this section, we position our paper in the literature on failure in kidney exchanges and discuss some formulations for the basic kidney exchange problem which we adapt to our setting. Finally, we mention relevant literature on (Defender-)Attacker–Defender models.

Failure in kidney exchange was first considered by Manlove and O’Malley (2012). They investigate a hierarchy of tiebreaking objectives to decide between multiple kidney exchange solutions based on minimizing the risk of failure within an exchange. For instance, a solution in which many of the 3-cycles have a subcycle is more preferred than a solution in which 3-cycles do not have this property. Dickerson et al. (2013, 2019) study a setting with equal failure probabilities and equal value for each transplant and maximize the expected number of transplants. Bidkhori et al. (2020) extend this model to a setting with inhomogeneous arc existence probabilities and give a compact formulation. Since stochastic models require accurate estimations for arc existence probabilities, an alternative model based on robust optimization is proposed in McElfresh et al. (2019). In addition to handling the possibility of failures a priori, an additional decision can be made once the failures are observed, called the recourse decision. In a recourse decision, an initially proposed kidney exchange solution is adapted after the failures have been identified. This adaptation might lead to a significant number of additional transplants that can be realized.
see, e.g., Bray et al. (2015). Different types of recourse for kidney exchange have been studied in the literature. Klimentova et al. (2016) and Alvelos et al. (2019) consider recourse in a stochastic setting. They initially select small disjoint subsets of recipient-donor pairs. After failures are revealed, recourse is limited to transplants between pairs within the same subset. Initial selections are made to maximize the expected number of transplants. Smeluders et al. (2022) studies a setting where initially a set of potential transplants, limited by a budget constraint, are tested. Again, the goal is to select these arcs to maximize the expected number of transplants after failure is revealed. Compared to earlier stochastic approaches, the additional flexibility in determining the transplant plan comes at a significant computational cost. Carvalho et al. (2021) consider recourse in a robust setting, as described in the introduction. Zeynivand et al. (2023) consider a policy that checks if non-failing pairs involved in a failed cycle or chain can be inserted in a successful exchange without violating the maximum cycle or chain length constraints.

Most algorithms related to kidney exchange programs are based on integer programming (IP). An intuitive IP formulation is the cycle-chain formulation (CC) due to Roth et al. (2007), which uses a binary variable for each cycle and each chain, and constraints for each pair and each NDD $j$, stating that only one cycle or chain can be chosen that includes $j$. The number of variables in these formulations grows rapidly as the cycle and chain limits increase, as well as when the size of the pool of participating pairs and NDDs becomes larger. A common technique for solving the cycle-chain model is branch-and-price (Glorie et al., 2014; Plaut et al., 2016; Lam and Mak-Hau, 2020; Riascos-Alvarez et al., 2020; Omer et al., 2022). Another fundamental formulation is the edge formulation due to Abraham et al. (2007). Instead of cycle and chain variables, this formulation uses binary variables $y_{ij}$ for each $(i,j) \in A$, that indicate whether the donor of $i$ donates a kidney to the recipient of $j$, leading to a polynomial number of variables. However, the number of constraints is exponential in the number of pairs and NDDs. Constantino et al. (2013) introduced the first compact formulations, i.e., models with a polynomial number of variables and constraints. The drawback of this formulation is that its LP relaxation is generally weaker than the one of the CC formulation. Dickerson et al. (2016) introduced the position-indexed cycle edge formulation (PICEF), which is a hybrid variant of different models: as the CC formulation, it uses binary variables for each cycle. But instead of using variables for each chain, chains are modeled by variables that express the position of an individual transplant within a chain. The resulting formulation is thus compact only concerning the chains. Dickerson et al. (2016) show that the LP relaxation of PICEF is strictly weaker than that of CC. However, we will use the PICEF formulation and show its advantages in the robust setting we consider. More recently, new formulations have been proposed that model cycles as two half-cycles, i.e., two paths that only intersect in their endpoints (Delorme et al., 2023).

As we study a worst-case setting, following Carvalho et al. (2021), the failures can be modeled as deliberate attacks of an optimizing adversary. Riascos-Alvarez et al. (2022) have studied a generalization of this model based on inhomogeneous failures of both vertices and arcs. This makes the model a specific example of a general class of optimization models known as the class of defender-attacker-defender (DAD) models. Problems involving network interdiction are often intuitively modeled by a DAD model, many of which are surveyed in Smith and Song (2020). The second and third stages of the model we study can separately be seen as an attacker–defender model and specifically an interdigation game with monotonicity as studied, e.g., by Fischetti et al. (2019). We will indeed model these stages as an interdigation game and make use of their general framework.

2. Algorithmic framework

In this section, we reconsider the defender–attacker–defender model for robust kidney exchange proposed by Carvalho et al. (2021) and algorithmic concepts used for solving this model. We start by providing some preliminary knowledge on modeling basic kidney exchange programs. Secondly, we describe the DAD model in a general sense, and concretize it for two recourse policies. Finally, we describe a cutting-plane approach for solving the adversary problem.

2.1. Preliminaries

The basic KEP model considers a compatibility graph $G = (V, A)$. The vertex set $V$ consists of a set $P$ of incompatible donor-recipient pairs and a set $N$ of non-directed donors, i.e., $V = P \cup N$. Each arc $(i,j) \in A$ indicates that the donor of vertex $i \in P \cup N$ is compatible with the recipient of vertex $j \in P$. As each arc corresponds to transplanting a donor kidney to a recipient, we will also refer to arcs as transplants. We denote by $C_k$ the set of (directed) cycles in $G$ with at most $K$ arcs.

Furthermore, a non-directed donor $n \in N$ can initiate a (directed) chain of transplants, further involving a set of pairs, with at most $L$ arcs. The set of all chains in $G$ of length at most $L$ is denoted by $D_L$. We refer to $\mathcal{C}_{K,L} = C_k \cup D_L$ as the set of feasible exchanges, or simply $\mathcal{C}$ if $K$ and $L$ are clear from the context.

Given an exchange $e \in \mathcal{C}$, we denote by $V(e)$ and $A(e)$ the set of vertices and arcs involved in $e$. We denote by $\mathcal{C}_L$ the set of exchanges involving vertex $j$, with similar notions $C'_k$ and $D'_L$ respectively for cycles and chains involving $j$.

2.2. Robust kidney exchange model

The model introduced by Carvalho et al. (2021) considers a variant of kidney exchange programs, in which withdrawals of non-directed donors and pairs are taken into account in a worst-case setting. This problem can be interpreted as a two-player game, in which the KEP owner (defender) plays against an optimizing adversary (attacker). Observe that the attacker is not considered to be an actual decision maker, but instead a concept for modeling the worst-case realization of withdrawals. Since an initial KEP solution can be adapted by the defender once an attack has been observed, our robust KEP can be interpreted as a defender–attacker–defender model. Given the initial solution proposed by the defender and an attack proposed by the attacker, the defender’s goal is to identify a feasible recourse solution in which the largest number of recipients involved in the initial solution again receive a transplant. The high-level aim of the defender is to coordinate their initial decision in such a way that the objective value of the recourse problem is maximized with respect to the worst-case attack. Consequently, solving the underlying robust optimization problem faced by the defender, i.e., with respect to the set of feasible attacks for the attacker, consists of solving a trilevel optimization problem.

We denote by $P$ the set of initial solutions. The set of possible attacks (sets of failing pairs or non-directed donors) is denoted by $U'$. Furthermore, we denote by $P(X, U) \subseteq P$ the set of recourse solutions $Y'U$ given initial solution $X \in F$ and attack $U \in U'$. Notice that $P(X, U)$ depends on the (implicit) recourse policy at hand. Furthermore, we denote by $P(X) \subseteq P$ the set of pairs involved in an exchange of the initial solution $X$. Similarly, we denote by $P(Y') \subseteq P$ the set of pairs involved in an exchange of the recourse solution $Y' \in P(X, U)$ given initial solution $X \in F$ and attack $U \in U'$. The robust optimization problem faced by the defender can then be described as

$$\max \min \max_{X \in F} \max_{U \in U'} \max_{Y' \in P(X, U)} \{|P(X) \cap P(Y')| : X \in F, U \in U', Y' \in P(X, U)|$$

(1)

In the remainder of this section, we specify the exact structure of the set $F$, $U'$, and $P(F, X, U)$ (given $X$ and $U$), as well as the objective function $f$. We now distinguish the three phases of the model more formally:
Stage 1: initial solutions. In the first stage of the model – the initial stage – the defender selects an initial solution. We assume the defender can select any kidney exchange solution, i.e., the set of feasible initial solutions is the set $F$ of feasible kidney exchange solutions on $G$, i.e.,
\[ F := \{ X \subseteq E | V(e) \cap V(e') = \emptyset \text{ for all } e, e' \in X, e \neq e' \}. \] (2)
Given an initial solution $X \in F$, we refer to a cycle $c \in X$ as an initial cycle, and to a chain $d \in X$ as an initial chain.

Stage 2: attacks. For the second stage – the attacker’s stage – the attacker has an attack budget $B \in \mathbb{Z}_{\geq 0}$, indicating the number of pairs and non-directed donors that can be “attacked,” corresponding to donors/recipients withdrawing. In other words, the set $U'$ of feasible attacks is given by
\[ U' := \{ U \subseteq V | |U| \leq B \}. \] (3)
i.e., each attack $U \in U'$ corresponds to a subset of at most $B$ pairs and non-directed donors withdrawing from the program. Given this definition of $U'$, for any vertex $j \in P \cup N$, we consider $j$ to be attacked if and only if $j \in U$.

Stage 3: recourse solutions. In the final stage of the model – the recourse stage – the defender is given the opportunity to adopt the initial solution $X \in F$ based on the attack $U \in U'$. For this purpose, let us define the set $\mathcal{E}[R_U]$, where $R_U := V \setminus U$, as the set of non-attacked (remaining) exchanges $e \in E$, i.e., an exchange $e \in \mathcal{E}[R_U]$ if and only if $V(e) \subseteq R_U$. We use similar notions $C_k[R_U]$ and $D_k[R_U]$ for cycles and chains, respectively. Furthermore, we define the set $F[R_U]$ as the set of kidney exchange solutions $X$ with only non-attacked exchanges, i.e., $X \subseteq \mathcal{E}[R_U]$.

In general, we have that $F(X, U) \subseteq F[R_U]$. The subset sign indicates that there might exist kidney exchange solutions $X \in F[R_U]$ that are not allowed due to further restrictions imposed by the underlying (implicit) recourse policy. Whenever we consider an explicit recourse policy $\Pi$, we write $F_{\Pi}(X, U)$ to indicate the set of recourse solutions respecting policy $\Pi$.

Recourse policies. We now consider two policies for recourse in kidney exchange problems: the Full Recourse (FR) policy proposed by Carvalho et al. (2021) and the novel Fix Successful Exchanges (FSE) policy.

In the Full Recourse (FR) policy, no additional restriction is imposed on the set of recourse solutions, i.e., given initial solution $X$ and attack $U$, we have that
\[ F_{FR}(X, U) := F[R_U] \].
As a result, any feasible kidney exchange solution that does not involve withdrawing donors and recipients is also a feasible recourse solution. One clear advantage of FR is that it provides the highest degree of flexibility in selecting recourse solutions. Nevertheless, the full recourse policy might leave individual recipients dissatisfied. Namely, recipients who are selected for a (successful) exchange in the initial solution are not guaranteed to receive a transplant according to the recourse solution.

With this issue in mind, we consider the Fix Successful Exchanges (FSE) policy, which addresses these drawbacks and aims at minimizing such disadvantages. To our knowledge, all recourse policies previously studied in the literature of kidney exchange programs are based on restricting recourse based on a subset $\mathcal{E}(X, U) \subseteq \mathcal{E}[R_U]$ of allowed exchanges for any recourse solution. We consider similar notions $C_k(X, U)$ and $D_k(X, U)$ for the sets of allowed cycles and chains. In contrast, the FSE policy additionally indicates a set of exchanges that must be enforced given initial solution $X$ and attack $U$, i.e., exchanges that must be included in any feasible recourse solution. We denote the set of enforced exchanges by $\mathcal{E}(X, U) \subseteq \mathcal{E}[R_U]$.

For the FSE policy, we can formalize the definition of $\mathcal{E}(X, U)$ as follows. First, the cycles of the set $\mathcal{E}(X, U)$ are the initial cycles $c \in X \cap C_k$ such that $c \in C_k[R_U]$, i.e., initial cycles containing no attacked (withdrawing) pairs. Second, for each initial chain $d \in X \cap D_k$, we enforce the unique subchain consisting of all vertices up to (but not including) the first withdrawing pair or non-directed donor, if any. More formally, we define this subchain as the enforced partial chain of $d$.

Definition 1 (Enforced Partial Chain). Let us be given an initial solution $X$ and attack $U$, and let $d = (n, p_1, \ldots, p_L) \in X \cap D_k$ be an initial chain, i.e., $n \in N$, and $p_i \in P$ for each $i \in \{1, \ldots, L\}$ for some $1 \leq \ell \leq L$. Then, we define the enforced partial chain of $d$ in any feasible recourse solution $Y_U \in F_{FSE}(X, U)$ as
\[ \text{epc}_U(d) = \begin{cases} \ell, & \text{if } d \in D_k[R_U] \\ \ell, & \text{if } n \not\in R_U \text{ or } p_\ell \not\in R_U \\ \ell, & \text{if } \{ n, p_1, \ldots, p_\ell \} \subseteq R_U, \text{ and } p_{\ell+1} \not\in R_U \\ (k \leq \ell - 1), & \text{if } \{ n, p_1, \ldots, p_k \} \subseteq R_U. \end{cases} \]

An example of enforced partial chains for a chain of length 3 is depicted in Fig. 2.

To summarize, the chains of the set $\mathcal{E}(X, U)$ are the enforced partial chains of the initial chains $d \in X \cap D_k$. Hence, we can write the set of enforced exchanges as
\[ \mathcal{E}(X, U) := \{ X \cap C_k[R_U] \} \cup \{ \text{epc}_U(d) | d \in X \cap D_k[R_U] \}. \]
This definition of $\mathcal{E}(X, U)$ allows us to formalize the set of feasible FSE recourse solutions, i.e.,
\[ F_{FSE}(X, U) := \{ Y \in \mathcal{F}[R_U] | \mathcal{E}(X, U) \subseteq Y \} \]
In other words, we enforce (fix) the successful (parts of) initial exchanges to be included in the recourse solution. Furthermore, any other exchange $e \in \mathcal{E}[R_U]$ involving the remaining non-attacked pairs and non-directed donors could be added to the recourse solution.

On the one hand, this recourse policy has significant advantages from an organizational point of view, as cycles and chains without failures can immediately proceed to transplant. This reduces the waiting time for the recipients involved in a successful exchange. In contrast, under full recourse, the recipients involved in such successful exchanges are not guaranteed to receive a transplant, and pairs might get reassigned to new exchanges. The FSE policy thus reduces uncertainty for recipients and transplant centers alike.

Nevertheless, the drawback of the FSE policy is that we restrict the set of feasible recourse solutions. Consequently, it might occur that in the final recourse solution concerning FSE, the number of transplanted recipients is strictly smaller than it would be the case with FR. We believe, however, that the negative impact of deploying the FSE policy will be limited. Our computational results in Section 4 show that for most instances, the attainable objective value for the FSE setting either coincides or is just one less compared to the objective value in the FR setting. The FSE policy thus provides an intuitive solution concept that both benefits the donor and recipient experience as transplants are guaranteed to occur once none of the involved pairs and non-directed donors withdraw from the kidney exchange program.

We will show next in Sections 2.3 and 2.4 how the trilevel optimization problem can be formalized using mathematical programming formulations.

2.3. Stage 1: column-and-constraint generation (C&CG) algorithm

One approach for solving defender-attacker-defender models is to first reformulate the model as a single-level optimization problem. This can be done by considering all feasible attacks $U \in U'$ for the attacker.

In particular, the robust kidney exchange problem with recourse can be described in general as
\[ z(U) := \max \theta \] (4a)
the C&CG algorithm works as follows:

\[ \text{s.t. } \delta \leq |P(X) \cap P(Y^U)|, \quad U \in U', \quad (4b) \]

\[ \begin{align*}
X & \in F, \quad (4c) \\
Y^U & \in F(X, U), \quad U \in U', \quad (4d) \\
\delta & \in R_{\geq 0}. \quad (4e)
\end{align*} \]

We show now that \( z(U') \) provides a valid formulation for the trilevel optimization problem. Objective (4a) introduces a dummy variable \( \delta \in R_{\geq 0} \) that will be used to capture the worst-case objective value. Clearly, Constraint (4c) requires \( X \) to be a feasible initial solution, and Constraints (4d) require \( Y^U \) to be a feasible recourse solution given attack \( U \in U' \). Furthermore, it follows from Constraints (4b) that the value of \( \delta \) in an optimal solution will be equal to

\[ \delta := \min_{U} \max_{Y^U} |P(X) \cap P(Y^U)|, \quad (5) \]

whenever \( X \in F \) is an optimal initial solution, and \( Y^U \in F(X, U) \) is an optimal recourse solution given initial solution \( X \) and attack \( U \in U' \).

The main advantage of transforming a trilevel optimization model into a single-level optimization model is we can use techniques known from the literature, such as algorithms for solving mixed-integer programs (MIPs). We will show in Section 3 that this model can be written as a mixed-integer linear program for commonly used encodings of kidney exchange solutions. Nevertheless, the linearization of the multi-level optimization model comes at the cost of creating an extremely large MIP. As Problem (4) has both constraints and variables indexed by each attack \( U \in U' \), the number of constraints and variables are exponential in the size \(|V|\) of the kidney exchange pool. Today’s MIP solvers are not able to deal with such large formulations, hence we cannot simply provide this full description to a solver. Nevertheless, this mathematical program contains a special structure, as the constraint matrix contains a block submatrix for each attack \( U \in U' \), with a relatively small number of coupling constraints (4b).

One commonly used approach for solving MIP models of this structure is the column-and-constraint generation algorithm (C&CG), as described below. We refer the reader to Zeng and Zhao (2013) for a description of the column-and-constraint generation algorithm for general two-stage robust optimization models. In our specific setting, the C&CG algorithm works as follows:

1. Let \( \tilde{U} \subseteq U' \) be a small subset of feasible attacks.
2. Solve the restricted problem \( z(\tilde{U}) \), with optimal solution \( (\tilde{X}, \tilde{Y}, \tilde{\delta}) \), where \( \tilde{Y} = (Y^U)_{U \in \tilde{U}} \).
3. Solve the bilevel subproblem given optimal initial solution \( \tilde{X} \in F \):

\[ s(\tilde{X}) := \min_{Y^U} \max_{U} |P(\tilde{X}) \cap P(Y^U)|, \quad (6) \]

Let \( (\tilde{X}, Y^U, \tilde{\delta}) \) be an optimal solution to \( s(\tilde{X}) \).
4. If \( s(\tilde{X}) < \delta \), let \( \tilde{U} = \tilde{U} \cup \tilde{U} \) and go to Step 2. Otherwise, return \( \tilde{\delta} \) as then \( z(\tilde{U}) = z(U') \).

From a high-level perspective, one solves the problem while only accounting for a restricted set \( \tilde{U} \) of feasible attacks. The initial solution \( \tilde{X} \) is then tested against other attacks to see whether there exists a not yet considered attack \( \tilde{U} \in U' \setminus \tilde{U} \) for which an optimal recourse solution \( Y^U \) violates the current solution of the restricted problem, i.e.,

\[ |P(\tilde{X}) \cap P(Y^U)| < \delta. \]

If this is true, it is clear that the current set of attacks \( \tilde{U} \) is not enough to describe worst-case behavior and we will need to add \( \tilde{U} \) to it. Otherwise, an optimal solution has been found. As \( |U'| \) is finite, this algorithm will eventually terminate such that the problem is solved to optimality.

2.4. Stage 2: cutting plane algorithm

Carvalho et al. (2021) proposed a branch-and-bound type algorithm for solving \( s(\tilde{X}) \), where branching is performed by either fixing a vertex to be attacked (\( j \in U \)) or not attacked (\( j \notin U \)), leading to two new subproblems. For each subproblem, a lower bound is computed by considering the initial solution \( \tilde{X} \) and the fixed attacks, and extending the attack \( U \) in a greedy fashion as if no recourse options exist. Aside from that, the best recourse solution \( Y^U \in F(X, U) \) with respect to \( U \) is computed. As \( U \) is feasible, but not necessarily optimal to \( s(\tilde{X}) \), this gives us an upper bound to the optimal value. The column-and-constraint generation method, using this branch-and-bound approach for generating columns and constraints, was tested in the computational experiments of Carvalho et al. (2021). It was shown that for realistic instances with 50 incompatible donor–recipient pairs and attack budget \( B \in \{1, 2, 3, 4\} \), this method was able to solve the entire defender–attacker–defender model within the time limit of one hour but struggled with instances of 100 pairs due to large sizes of the branch-and-bound trees involved.

In this paper, we propose an alternative algorithm based on cutting planes for solving the bilevel subproblem \( s(X) \) to provide a method that can solve larger instances. This is done through linearizing the model by first projecting out the variables of the lower-level problem, namely the variables encoding the recourse solution \( Y^U \in F(X, U) \). Additional constraints are imposed to ensure that the restrictions of the recourse policy are properly addressed.

We propose a reformulation of \( s(X) \) based on value functions (Dempé et al., 2015; Bolusani and Ralphs, 2022). The value function of this formulation can be written as

\[ \Phi(X, U) := \max_{Y^U} |P(X) \cap P(Y^U)|, \quad (7) \]

where \( Y^U \in F(X, U) \) is an optimal recourse solution given \( X \) and \( U \). The issue with this value function is that the feasible set \( F(X, U) \) depends on the initial solution \( X \) and the attack \( U \). To be able to reformulate the problem as a single-level problem, we first need to formulate a value function for which the feasible set does not depend on preceding decisions.

Observe that any feasible recourse solution \( Y^U \in F(X, U) \) can be obtained from some maximal kidney exchange solution \( Y \in F \) by
removing each \( e \in \mathcal{E}(X, U) \) from \( Y \). In other words, if we denote the set of disallowed exchanges given \( X \) and \( U \) by \( I(X, U) = \mathcal{E} \setminus \mathcal{E}(X, U) \), we can write \( Y' = Y \setminus I(X, U) \). For given \( X \) and \( U \), the set of disallowed exchanges \( I(X, U) \) is fixed, hence we can restrict the defender’s decision by choosing a feasible kidney exchange solution \( Y \in F \). The value function can thus be rewritten as

\[
\Phi(X, U) = \max_{Y \in F} \{ P(X) \cap P(Y) \} - \{ P(X) \cap Y \cap I(X, U) \}. \tag{8}
\]

In that way, the bilevel optimization problem \( s(X) \) can be reformulated as a single-level optimization problem, as the feasible set of \( \Phi(X, U) \) does not depend on preceding decisions anymore. The inner maximization problem can then be reformulated by adding constraints for each feasible kidney exchange solution \( \tilde{Y} \in F \). As a result, we get the following single-level reformulation for \( s(X) \).

\[
\begin{align*}
\min \quad & \varphi \\
\text{s.t.} \quad & \varphi \geq |P(X) \cap P(Y)| - |P(X) \cap Y \cap I(X, U)|, \quad \tilde{Y} \in F \\
& U \in U', \\
& \varphi \in \mathbb{R}_{\geq 0},
\end{align*}
\tag{9}
\]

One drawback of this reformulation is it comes at the expense of an exponential number of constraints of type (9b). Following Fischetti et al. (2019), who also studied this type of constraints, we refer to these as interdiction cuts, as the set \( I(X, U) \) interdicts exchanges that are disallowed in any recourse solution, based on the restrictions imposed by initial solution \( X \), attack \( U \), and (implicit) recourse policy \( \Pi \).

As kidney exchange instances consist of rather sparse graphs, we will see in Section 3 that \( s(X, F) \) can be formulated as a mixed-integer linear program, using a relatively small number of variables compared to the number of constraints. This justifies the use of a cutting plane method to solve this problem. We start by considering the restricted problem \( s(\hat{X}, F) \) based on initial solution \( \hat{X} \) and a small subset \( F \subset F \) of feasible kidney exchange solutions. We can describe the algorithm as follows:

1. Solve the problem \( s(\hat{X}, F) \), i.e., with Constraints (9b) restricted to \( F \). Suppose \( (\hat{U}, \hat{\varphi}) \) is an optimal solution to \( s(\hat{X}, F) \).
2. Solve the separation problem

\[
R(\hat{X}, \hat{U}) := \max_{Y \in F} \{ P(\hat{X}) \cap P(Y) \} - \{ P(\hat{X}) \cap Y \cap I(\hat{X}, \hat{U}) \}. \tag{10}
\]

Observe that \( R(\hat{X}, \hat{U}) \) is equivalent to the weighted kidney exchange problem, in which exchange weights \( \omega_e(X, U) \) are defined as follows:

\[
\omega_e(X, U) = \begin{cases} 
0, & \text{ if } e \in I(X, U), \\
|P(e) \cap P(X)|, & \text{ otherwise}.
\end{cases}
\]

Suppose \( \hat{Y} \) is an optimal solution to \( R(\hat{X}, \hat{U}) \). The efficiency of the cutting plane algorithm is based on the empirical observation that kidney exchange problems are easy to solve in practice despite their theoretical hardness.

3. If \( R(\hat{X}, \hat{U}) < \epsilon (\hat{U}) \), then we have that \( \hat{U} \) needs to be added to \( \hat{U} \). In other words, the attack \( \hat{U} \) corresponds to both a set of columns and rows that need to be added to the constraint matrix of the restricted master problem. Otherwise, if \( R(\hat{X}, \hat{U}) > \hat{\varphi} \), we add the interdiction cut related to \( \hat{Y} \) to the model, i.e., we consider \( s(\hat{X}, \hat{F} \cup \{ \hat{Y} \}) \), and go to Step 1. Otherwise, it holds that \( s(\hat{X}, \hat{F}) = R(\hat{X}, \hat{U}) \), i.e., the master problem is solved to optimality.

Regarding the interpretation of the separation problem (10) as a weighted kidney exchange program, notice that the weight of each disallowed exchange equals zero. Therefore, there is no guarantee that \( \hat{Y} \) is also maximal, as there might exist a disallowed exchange \( e \in I(X, U) \setminus \hat{Y} \) that could be added such that \( Y' = \hat{Y} \cup \{ e \} \in F \) as well. Notice that the interdiction cut (9b) corresponding to \( Y' \) is stronger than the one corresponding to \( \hat{Y} \); indeed, note that \( |P(X) \cap P(\hat{Y} \cup \{ e \})| - |P(X) \cap P(\hat{Y})| = |P(X) \cap P(e)| \), while

\[
\begin{align*}
|P(X) \cap P(\hat{Y} \cup \{ e \}) \cap I(X, U)| & - |P(X) \cap P(\hat{Y} \cap I(X, U))| \\
& = \begin{cases} 
|P(X) \cap P(e)|, & \text{ if } e \in I(X, U) \\
0, & \text{ otherwise}.
\end{cases}
\end{align*}
\]

As such, adding exchange \( e \) to the solution cannot weaken the interdiction cut (9b). Moreover, for attacks \( \hat{U} \) with \( e \notin I(X, \hat{U}) \), and \( |P(X) \cap P(e)| \geq 1 \), the interdiction cut (9b) for \( Y' \) is strictly stronger. We will refer to these stronger cuts as lifted interdiction cuts, as such cuts can be obtained from \( \hat{Y} \) by adding – i.e., lifting the coefficients of – disallowed exchanges \( e \in I(X, \hat{U}) \).

To separate lifted interdiction cuts, we consider a modified variant \( R'(X, \hat{U}) \) of the separation problem. Let us denote by \( |Y| \) the number of exchanges that \( Y \) consists of. Based on the approach of Fischetti et al. (2019), the modified separation problem can be formulated as follows:

\[
R'(X, \hat{U}) := \max_{Y \in F} \{ P(X) \cap P(Y) \} - \{ P(X) \cap P(Y \cap I(X, U)) \} + \epsilon \cdot |Y| \tag{11}
\]

This modified separation problem differs from the original separation problem as it considers a hierarchy of two objectives. The first objective coincides with the objective of the original separation problem, whereas the secondary objective (i.e., \( \epsilon \cdot |Y| \) prioritizes kidney exchange solutions consisting of a large number of exchanges. In particular, this ensures that we obtain a maximal kidney exchange solution \( Y \), which often means that there exist disallowed exchanges \( e \in Y \), i.e., \( e \notin \mathcal{E}(X, U) \). Indeed, this is only a secondary objective, or tiebreaker, due to the introduction of the sufficiently small parameter \( \epsilon > 0 \). It would suffice to select \( \epsilon < 1/|V| \) to ensure that it does not interfere with the primary objective of finding an optimal recourse solution.

We now illustrate in Fig. 3 the practical benefit of deploying this modified separation problem, showing how it can lead to significantly stronger cuts. In this example, suppose \( k = 3 \) and \( \lambda = 2 \). There exists an initial solution \( X \in F \) covering all vertices, namely the cycles \( (P_1, P_3, P_5), (P_2, P_4, P_6), (P_5, P_8, P_7) \), and the single chain \( (N_1, P_1, P_2) \). Furthermore, let \( U = \{ P_1, P_5 \} \in U' \) be a feasible attack (suppose \( B = 2 \)). The black arcs indicate an optimal solution \( Y \) to \( R(X, U) \). This solution, however, is not maximal and hence not optimal for \( R'(X, U) \), but the alternative solution \( Y' \) obtained from extending \( Y \) with the cycle \( (P_5, P_8, P_7) \) is optimal. Whenever another attack \( U' \) is considered in such that the cycle between \( P_5 \) and \( P_8 \) is allowed, i.e., contained in \( \mathcal{E}(X, U') \) – the contribution of this cycle to the right-hand side of the interdiction cut (9b) of \( Y' \) is positive, meaning that the interdiction cut of \( Y' \) strictly dominates \( Y \).

We will show that the modified separation problem is not significantly harder to solve than the original version. Namely, we can again...
We start by considering the master problem formulation

\[ w^*(X, U) = \begin{cases} 
  \epsilon, & \text{if } e \in I(X, U), \\
  |P(e) \cap P(X)| + \epsilon, & \text{otherwise}.
\end{cases} \]  

(12)

An additional beneficial consequence of this re-weighting is that it favors solutions with a large number of short exchanges, i.e., 2-cycles and short chains. For instance, if there exists a set of six vertices that could be covered by using either (i) two 3-cycles or (ii) three 2-cycles, the latter option will be selected as it includes more exchanges. Such solutions are also typically stronger cuts, as solutions with many short exchanges are more robust against withdrawals of pairs and non-directed donors, compared to solutions covering the same pairs and non-directed donors with larger exchanges. In Section 4, insights are provided on the impact of deploying the original and modified separation problems on the cutting plane algorithm, and how it impacts the practical computational efforts needed by our solution method.

3. Formulations

In this section, we provide concrete formulations for the models described in Section 2. We start by providing the formulations based on the cycle-chain (CC) formulation for basic kidney exchange problems for both recourse policies in Section 3.1. Secondly, we do the same for the position-indexed chain-edge formulation (PICEF) in Section 3.2. In particular, we formalize that each of the three decisions to be made can be optimized by solving single-level mixed-integer linear programming (MILP) models.

3.1. Cycle-chain formulation

In this section, we consider the formulations for the three levels of the robust optimization problem as described in general in Section 2.2 based on the cycle-chain formulation for kidney exchange problems. We start by considering the master problem formulation \( z_{P\text{CC}}(U) \) in Section 3.1.1. Formulations for second-level problem – the attacker-defender subproblem – \( z_{P\text{CC}}(x; F) \) are considered in Section 3.1.2. Finally, we consider the recourse problem \( R_{U}(x, u) \) given initial solution \( x \) and attack \( u \) in Section 3.1.3.

3.1.1. CC: Master problem

First, we reconsider the CC-based formulation \( z_{P\text{CC}}(U) \) for the master problem, which was proposed in Carvalho et al. (2021). We consider variables \( x_e \in \{0, 1\} \) encoding whether the exchange \( e \in E \) is contained in the initial solution \( X \in F \). Furthermore, we associate with each attack \( U \in U \) its characteristic vector \( u \in \{0, 1\}^E \), i.e., with entries \( u_j = 1 \) if and only if \( j \in U \). With a slight abuse of notation, in the context of concrete formulations, we will also refer to \( u \in U \) as an attack. Similarly, for each attack \( u \in U \), we consider variables \( y^u_e \in \{0, 1\} \) encoding whether the exchange \( e \in E \) is contained in the recourse solution \( \tilde{Y}^u \in F(X, U) \) given \( X \) and \( U \). Furthermore, we consider variables \( \psi^u_e \in \{0, 1\} \) indicating for each pair \( (j, P) \) whether \( j \in P \times X \cap P(X, U, \tilde{Y}^u) \) or not. The formulation \( z_{P\text{CC}}(U) \) can then be written as follows:

\[
\begin{align*}
\max & \ 8 \ \\
\text{s.t.} & \ \sum_{j \in P} \psi^u_j \leq 1 - u_j, \quad u \in U, \quad j \in P. \\
& \ y^u_j \leq \sum_{e \in P} x_e, \quad u \in U, \quad j \in P. \quad (13a) \\
& \ y^u_j \leq \sum_{e \in P} \psi^u_e, \quad u \in U, \quad j \in P. \quad (13b) \\
& \ \sum_{e \in P} x_e \leq 1, \quad j \in P \cup N. \quad (13c)
\end{align*}
\]

We briefly describe the formulation to argue its correctness. Constraints (13c) and (13d) indeed ensure that \( r^u_j = 1 \) in an optimal solution if and only if \( j \in P \cap P(X, U, \tilde{Y}^u) \). Constraints (13e) and (13f) are packing constraints ensuring that \( X \in F \) and \( \tilde{Y}^u \in \tilde{F}_{P\text{CC}}(X, U) \), Constraints (13b) concretize the constraints (4b) described for the general framework. Notice that the structure of this model allows us to relax the integrality conditions on \( r^u_j \) to \( r^u_j \geq 0 \).

Remark 2. As kidney exchange problems commonly have multiple optimal solutions, it makes sense to implement a tiebreaker to obtain an initial solution \( x \) to (13) that is most preferable in more optimistic scenarios. One example is to consider a solution that would be optimal in case no failures occur, which can be obtained by adapting the objective function (13a) to

\[ \varphi + \epsilon \cdot \sum_{e \in E} |P(e)| x_e, \]

where \( P(e) \) is the set of donor–recipient pairs involved in exchange \( e \). Here, \( \epsilon > 0 \) is a sufficiently small parameter so that this secondary objective does not interfere with the primary objective of getting the most robust solution, i.e.,

\[ \epsilon \leq \frac{1}{\max_{e \in E} |P(e)| x_e}. \]

Modifications for \( z_{F\text{SE}(U)} \)

Formulation (13) needs to be adapted in order to correctly model the FSE recourse policy in the master problem since this policy requires that we enforce (parts of) successful initial exchanges. For that purpose, we adapt the definition of the set of enforced exchanges to this setting with variable encodings \( x \) and \( u \) of the initial solution and attack, respectively.

\[ \text{En}(x, u) = \{e \in C_k(R_u) \mid x_e = 1 \} \cup \{ c_{p\text{se}}(d) \mid d \in D_L, x_d = 1 \}. \]

Here, \( c_{p\text{se}}(d) \) denotes the natural adaptation of the enforced partial chain \( c_{p\text{se}}(d) \) to the characteristic vector representation of attacks, and \( R_u = \{j \in v \mid u_j = 0\} \) the set of non-attacked vertices given attack \( u \). Now, we want to define for each \( j \in P \cup N \) and \( u \in U \), the set of exchanges that, if contained in the initial solution \( x \), would lead to \( j \) being involved in an enforced exchange given \( u \), i.e.,

\[ I'(u) = C_k'(R_u) \cup \{ d \in D_L \mid c_{p\text{se}}(d) \in D'_L \}. \]

We can now obtain the formulation \( z_{F\text{SE}(U)} \) by adapting \( z_{P\text{CC}}(U) \). First, it is important to note that \( r^u_j \) now denotes only the choice of non-enforced exchanges in the recourse solution. The presence of enforced exchanges in recourse solutions is handled through the \( x_e \) variables and sets \( I'(u) \). We then replace Constraints (13d) and (13f) by

\[
\begin{align*}
\sum_{e \in P} x_e + \sum_{e \in P} y^u_e, \quad u \in U, \quad j \in P. \\
\sum_{e \in P} y^u_e \leq 1 - u_j, \quad u \in U, \quad j \in P \cup N. \quad (14b)
\end{align*}
\]

Notice that if for an attack \( u \), there exists some \( e \in I'(u) \) with \( x_e = 1 \), then this implies that the corresponding recourse solution must contain an enforced exchange which includes pair \( j \). As \( y^u_j \) represents the non-enforced exchanges, the right-hand side of Constraint (14a) and left-hand side of Constraint (14b) correctly considers whether there is some exchange containing the pair or NDD \( j \in P \cup N \).
3.1.2. CC: Attacker-defender subproblem

We now consider MILP formulations of the subproblem \( s_{\text{CC}}(x) \) faced by the “attacker” in the second-stage of the trilevel model, given \( x \) optimal to \( s_{\text{FR,CC}}(U) \) for some \( U \subset U' \). Notice that whenever an attacker \( u \in U' \) has been realized, a recourse solution \( y^u \in F_{\text{CC}}(x, u) \) will be selected that does not consider exchanges that have an attacked vertex. For each exchange \( e \in \mathcal{E} \), we define the weight \( w_e(x) \) as

\[
w_e(x) := |P(e) \cap V(e)|,
\]

i.e., the number of pairs \( p \in V(e) \) that are involved in some initial exchange with respect to \( x \). Following the framework described in Section 2.4, we obtain the following single-level reformulation for the bilvel subproblem, where \( z_e \) is a binary variable modeling whether \( e \in \mathcal{E}(x, u) \) is an allowed exchange for the recourse solution given \( u \).

\[
\begin{align*}
\min & \quad \varphi \\
\text{s.t.} & \quad \varphi \geq \sum_{e \in \mathcal{E}} w_e(x) z_e, & X & \in F, \tag{15a} \\
& \quad z_e \geq 1 - \sum_{j \in V(e)} u_j, & e & \in \mathcal{E}, \tag{15c} \\
& \quad \sum_{j \in P(u)} u_j \leq B, \tag{15d} \\
& \quad z_j \in [0, 1], & j & \in \mathcal{E}, \tag{15e} \\
& \quad u_j \in [0, 1], & j & \in V, \tag{15f} \\
& \quad \varphi \geq 0. \tag{15g}
\end{align*}
\]

Here, Constraints (15b) are a concrete description of the interdiction cuts (9b) described for the general framework in Section 2. Moreover, Constraints (15c) ensure that \( e \notin \mathcal{E}(x, u) \) if \( e \) contains attacked vertices.

**Remark 3.** Due to the structure of \( s_{\text{FR,CC}}(x; F) \) (especially, the interdiction cuts (15b)), we can without loss of generality relax the integrality of the \( x \) variables, as there exists some optimal solution for which \( z_e \) is set to its largest lower bound requires by the constraints, which is \( \max\{0, 1 - \sum_{j \in V(e)} u_j\} \in [0, 1] \).

**Modifications for \( s_{\text{SE,CC}}(x; F) \).** To obtain a valid formulation for the attacker-defender subproblem \( s_{\text{SE,CC}}(x; F) \), we again need to handle the enforced exchanges. Indeed, under this recourse policy, exchanges can also be excluded from a recourse solution if they intersect with an enforced exchange. For this purpose, we define the set \( \text{Enf}(x) \) of enforceable exchanges, defined as follows,

\[
\text{Enf}(x) = \bigcup_{u \in U} \text{Enf}(x, u).
\]

This is the set of exchanges that could be enforced for at least one attack \( u \). Observe that it depends on the attack selected by the attacker which of the enforceable exchanges \( e \in \text{Enf}(x) \) are enforced. Finally, we obtain the formulation \( s_{\text{SE,CC}}(x; F) \) by replacing Constraints (15c) with the following linear inequalities:

\[
\begin{align*}
& \quad z_e \geq 1 - \sum_{j \in V(e)} u_j - \sum_{e' \in \text{Enf}(x)} z_{e'}, & e & \in \mathcal{E} \setminus \text{Enf}(x), \tag{16a} \\
& \quad z_e \geq 1 - \sum_{j \in V(e)} u_j, & e & \in \text{Enf}(x), \tag{16b} \\
& \quad z_e \leq 1 - u_j, & e & \in \text{Enf}(x), j \in V(e). \tag{16c}
\end{align*}
\]

Constraints (16a) and (16b) correspond to Constraints (15c) of the FR setting, with the difference that non-enforceable exchanges \( e \in \mathcal{E} \setminus \text{Enf}(x) \) can be disallowed from the recourse solution given \( u \), i.e., \( e \notin \mathcal{E}(x, u) \), if \( e \) intersects with an enforced exchange \( e' \in \text{Enf}(x, u) \). This can even be the case when \( e \in \mathcal{E}(R_u) \). Constraints (16c) are a necessary addition in the FSE setting. Where we could previously rely on \( z_e \) attaining its largest lower bound in some optimal solution, this is no longer the case due to the interaction of \( z \)-variables in Constraints (16a). Without Constraints (16c), it is possible to set \( z_e = 1 \) for an enforceable exchange \( e \in \text{Enf}(x) \), even if \( e \notin \mathcal{E}(R_u) \), i.e., if \( e \) has attacked vertices. This might be beneficial with respect to the objective of the attacker–defender subproblem, as then we can set \( z_e = 0 \) for all non-enforceable exchanges \( e \in \mathcal{E} \setminus \text{Enf}(x) \) intersecting \( e \). For this reason, Constraints (16c) ensure that \( z_e = 0 \) for any attacked enforceable exchange \( e \). Hence, these constraints correctly model the impact of the FSE policy.

3.1.3. CC: Recourse problem

Notice that the formulation of the attacker-defender subproblem \( s_{\text{FR,CC}}(x; F) \) contains an exponential number of Constraints (15b). We will refer to these constraints as interdiction cuts, as these constraints implicitly model for each feasible kidney exchange solution \( X \in F \) which exchanges can be selected in the recourse solution given attack \( u \in U' \). The algorithm proposed in Section 2.4 relies on an efficient method for finding violated constraints. Below we describe the corresponding separation problem. This allows us to write the problem \( R_{\text{FR,CC}}(x, u) \) of finding the optimal recourse solution \( y^u \in F_{\text{FR,CC}}(x, u) \) as a single-level mixed-integer program, namely

\[
\begin{align*}
\min & \quad \varphi \\
\text{s.t.} & \quad \sum_{e \in \mathcal{E}} w_e(x) y^u_e \\ & \quad \sum_{j \in \mathcal{E}} y^u_e \leq 1 - u_j, & j & \in P \cup N, \tag{17b} \\
& \quad y^u_e \in [0, 1], & e & \in \mathcal{E}, \tag{17c}
\end{align*}
\]

Constraints (17b) impose that when vertex \( j \in P \cup N \) is interdicted, then none of the exchanges \( e \in \mathcal{E} \) can be selected for the recourse solution. Therefore, any optimal solution to Problem (17) is also a feasible recourse solution under \( u \). In other words, Problem (17) is equivalent to the problem of finding a weighted kidney exchange solution on the subgraph \( G(R_u) \). We consider a modified version \( R_{\text{FR,CC}}'(x, u) \) of the separation problem that “lifts” optimal solutions for (17) to maximal kidney exchange solutions on the entire compatibility graph \( G \), to separate stronger interdiction cuts (15b). We can obtain this modified separation problem by making two modifications. First, we need to replace objective coefficients \( w_e(x) \) by alternative objective coefficients, consistent with the definition in (12)

\[
\begin{align*}
& \quad w^e(u) := \begin{cases} 1, & \text{if } e \notin \mathcal{E}(x, u) \\ w_e(x)|V| - 1, & \text{otherwise}. \end{cases} \\
\end{align*}
\]

and secondly, we relax the right-hand side of Constraints (17b) to 1 such that also interdicted exchanges \( e \notin \mathcal{E}(x, u) \) could be selected in the solution. Given \( y^u \) feasible to \( R_{\text{FR,CC}}'(x, u) \), the associated set of exchanges \( Y^u = \{ e \in \mathcal{E}(x, u) | y^u_e = 1 \} \) forms an optimal solution to the original problem \( R_{\text{FR,CC}}(x, u) \). This can be seen as follows. The weights \( w^e(u) \) are defined to prioritize non-attacked exchanges with high original weight \( w_e(x) \), and ensure that the set of non-interdicted exchanges in the optimal solution indicate an optimal solution for the original separation problem \( R_{\text{FR,CC}}(x, u) \). Ties are broken by selecting the solution with the largest number of exchanges (due to the +1 term). Our experiments have shown that separating interdiction cuts based on \( R_{\text{FR,CC}}'(x, u) \) leads to strictly stronger cuts, at no significant expense in terms of computation time.

**Modifications for \( R_{\text{SE,CC}}(x; F) \) and \( R_{\text{SE,CC}}'(x; F) \).** Furthermore, we obtain the formulations for \( R_{\text{SE,CC}}(x, u) \) and \( R_{\text{SE,CC}}'(x, u) \), by additionally requiring that the recourse solution \( y^u \in F_{\text{SE,CC}}(x, u) \) indeed contains the enforced exchanges \( e \in \text{Enf}(x, u) \). This can be realized by taking the formulations of \( R_{\text{FR,CC}}(x, u) \) and \( R_{\text{FR,CC}}'(x, u) \) respectively, and fixing the variables corresponding to enforced exchanges to 1:

\[
y^u_e = 1, & \quad e \in \text{Enf}(x, u).
\]
3.2. Position-indexed chain-edge formulation

We also provide mixed-integer programming formulations based on the position-indexed chain-edge formulation (PICEF) introduced by Dickerson et al. (2016). PICEF can be regarded as a hybrid variant of the cycle-chain formulation and edge formulation. Its main advantage is that it avoids full enumeration of all chains in the compatibility graph, as the number of chains might be exponential in \( L \).

Instead, it considers a formulation of chains in terms of arc variables with an additional index for their position in a chain. For basic kidney exchange problems, this is of particular interest when the maximum chain length \( L \) increases. Similar to \( CC \), we consider initial variables \( x_c \in \{0,1\} \) for each cycle \( c \in C_K \) indicating whether or not \( c \) is contained in the initial solution. Nevertheless, chains are not explicitly enumerated, but instead described by a collection of variables \( \xi_{j,c} \), for each arc \((i,j) \in A \) and position index \( \ell \in \mathcal{L} = \{1, \ldots , L\} \). Here, arcs going out of the non-directed donor have position index 1, the subsequent arc has index 2, etc. Therefore, for each arc \((i,j) \in A \), the actual set \( \mathcal{L}(i,j) \) of possible position indices is often more restrictive. In particular, for each arc \((i,j) \), we have \( \mathcal{L}(i,j) = \{1\} \) if \( j \in N \). Conversely, an arc \((i,j) \) with \( i \in P \) could have any position index greater than 1, i.e., \( \mathcal{L}(i,j) = \mathcal{L} \setminus \{1\} \) if \( i \in P \). We refer to the paper by Dickerson et al. (2016) for techniques to reduce the sizes of these lists \( \mathcal{L}(i,j) \) based on the topology of the compatibility graph \( G \), as this leads to formulations with (significantly) fewer variables.

We denote by \( P_L = \{(i,j,\ell) \in A \times \mathcal{L} \mid \ell \in \mathcal{L}(i,j)\} \) the set of position-indexed arcs, i.e., the building blocks that the chains consist of. The PICEF-based formulation will thus involve variables \( \xi_{j,c} \in \{0,1\} \) for each \((i,j,\ell) \in P_L \), indicating whether or not there exists an initial chain using arc \((i,j) \) in index position \( \ell \). We denote the initial solution encoding by \( x^* = (x, \xi) \). Furthermore, \( P^*_L \) denotes the set of position-indexed arcs involving vertex \( j \in P \cup N \), i.e.,

\[
P^*_L := \{(u, v, \ell) \in P \mid j \in \{u, v\}\}.
\]

### 3.2.1. PICEF: Master problem

Similar to the initial solution encoding \( x^* \), we consider for each attack \( u \in U^r \) the variables \( y^u \in \{0,1\} \) for each cycle \( d \in C_P \), and \( y^u \in \{0,1\} \) for each position-indexed arc \((i,j,\ell) \in P_L \) to encode which cycles and chains are contained in the recourse solution under \( u \). We denote by \((y^u)^* = (y^u, y^u)\) the entire encoding of the recourse solution under \( u \), for all possible attacks \( u \in U^r \). Again, we consider for each pair \( j \in P \) and attack \( u \in U^r \) the variables \( y^u \in \{0,1\} \), which have the exact same interpretation as in the CC-based formulations. The PICEF-based formulation \( \mathcal{F}_P(\text{PICEF}) \) of the master problem – first presented in Carvalho et al. (2021) – can then be described as follows:

\[
\begin{align*}
\text{max} \quad & \theta \\
\text{s.t.} \quad & \sum_{j \in \mathcal{L}(i)} \xi_{j,c} \leq \sum_{(i,j) \in E} \xi_{j,c}, \quad (17a) \\
& u \in U^r. \quad (17b) \\
& \xi_{j,c} \leq \sum_{(i,j) \in E} x_{c}, \quad (17c) \\
& \sum_{(i,j) \in E} x_{c} + \sum_{(i,j) \in E} \xi_{j,c} \leq 1, \quad (17d) \\
& \sum_{(i,j) \in E} y^u \leq 1, \quad (17e) \\
& \sum_{j \in \mathcal{L}(i)} \xi_{j,c} \leq 1 \quad (17f) \\
& \sum_{j \in \mathcal{L}(i)} y^u \leq 1 \quad (17g) \\
& \sum_{j \in \mathcal{L}(i)} u_{j} \leq 1 \quad (17h) \\
& \sum_{j \in \mathcal{L}(i)} y^u \leq 1 \quad (17i) \\
& u \in U^r. \quad (17j)
\end{align*}
\]

Here, notice that Constraints (18c)–(18f) are obtained from Constraints (13c)–(13d) of the CC-based master problem by replacing for each \( j \in P \) and \( (i,j) \in A \) the variables \( x_{c} \) and \( y^u \) for each chain \( d \in P_L \) and attack \( u \in U^r \), respectively with the variables \( \xi_{j,c} \) and \( y^u \), for each position-indexed arc \((i,j,\ell) \in P^*_L \) and attack \( u \in U^r \). Constraints (18g) and (18h) indicate that for each \( N \) and attack \( u \in U^r \), at most one of the arcs going out of \( j \) can be selected for the initial and recourse solution given \( u \). Finally, Constraints (18i) and (18j) are precedence constraints for the initial and recourse solutions respectively, indicating for each \( j \in P \) and \( \ell \in \mathcal{L} \setminus \{1\} \) that there can only select an arc going out of \( j \) with index \( \ell' + 1 \) if an arc going into \( j \) is selected with index \( \ell \).

### Modifications for \( \mathcal{F}_P(\text{PICEF}) \)

Again, we need to apply several modifications to derive a PICEF-based formulation \( \mathcal{F}_P(\text{PICEF}) \) of the master problem in the FSE setting. Let us consider an initial solution \( x^* = (x, \xi) \) and an attack \( u \in U^r \). Recall that for the cycle-chain formulation, we could rely on a set \( \mathcal{I}(u) \) of exchanges that, if selected as an initial exchange, would imply an enforced transplant in the recourse solution given \( u \). This was possible due to the enumerative nature of the formulation. In contrast, PICEF gives us only local information about chains, namely single transplants and their position in the chain. For that reason, we can only say if a position-indexed arc (at index \( \ell \)) is enforced in the recourse solution if we know which set of \( \ell - 1 \) arcs precede it. In other words, with regard to PICEF, there exists an additional dependence on \( \xi \). We define the set \( A(\xi, u) \) of enforced arcs in the recourse solution given \( u \) in a recursive fashion as follows:

\[
\begin{align*}
\xi_{i,j} & = 1; \quad u_{i} = 0; \quad \text{and} \quad u_{j} = 0 \quad \Rightarrow (i,j) \in A(\xi, u) \quad (19a) \\
\xi_{i,j} & = 1; \quad 3k : (k,i) \in A(\xi, u); \quad \text{and} \quad u_{j} = 0 \quad \Rightarrow (i,j) \in A(\xi, u) \quad (19b)
\end{align*}
\]

Observe that we can omit position indices from this definition, as these can be derived from \( \xi \). To model for each arc \((i,j) \in A(\xi, u) \) whether it is enforced in the recourse solution given \( \xi \) and \( u \), we consider additional variables \( \beta_{i,j} \in \{0,1\} \). This can be modeled by adding the following constraints, which we discuss below:

\[
\begin{align*}
\sum_{(i,j) \in A} \beta_{i,j} & \leq \sum_{(i,j) \in A} \xi_{i,j}, \quad (20a) \\
\sum_{j \in A} \beta_{i,j} & \leq \sum_{k:(k,i) \in A} \beta_{i,j} \quad u \in U^r, (i,j) \in A. \quad (20b) \\
\beta_{i,j} & \leq u_{i} - u_{j} \quad u \in U^r, (i,j) \in A. \quad (20c) \\
\beta_{i,j} & \leq u_{j} - u_{i} \quad u \in U^r, (i,j) \in A. \quad (20d) \\
\beta_{i,j} & \geq \xi_{j,i} - u_{i} - u_{j} \quad u \in U^r, (i,j) \in A. \quad (20e)
\end{align*}
\]
We define the cycle weight \( w_c(x') \) for each \( c \in C_k \) as
\[
w_c(x') = |P(x') \cap V(c)|,
\]
and the arc weight \( w_{ij}(x') \) for each \( (i, j) \in A \) as
\[
w_{ij}(x') = |P(x') \cap \{j\}|
\]
Furthermore, for any \( j \in P \cap V(d) \), we denote by \( d^{-j} \) the subchain of \( d \) up to and including pair \( j \in P \). The attacker–defender subproblem \( s_{FR,PICEF}(x'; F) \) can then be formulated as a MILP as
\[
\begin{align*}
\text{min } & \varphi \\
\text{s.t. } & \sum_{c \in C_k} w_c(x') z_c \\
& + \sum_{d \in D_L : (j, i) \in A(d)} u_j(x') \sum_{z_{ij} \in \{0, 1\}} w_{ij}(x') z_{ij}^d, \quad X \in F, \quad (22a) \\
& z_c \geq 1 - \sum_{j \in P(c)} u_j, \quad c \in C_k, \quad (23a) \\
& c' \sum_{j \in P(d^{-j})} u_j \geq 1, \quad d \in D_L : (i, j) \in A(d), \quad (23b) \\
& \sum_{j \in P(d^{-j})} u_j \leq B, \quad (23c) \\
& z_c \in [0, 1], \quad c \in C_k, \quad (23d) \\
& \forall c' \sum_{j \in P(d^{-j})} u_j \geq 1, \quad d \in D_L : (i, j) \in A(d), \quad (23e) \\
& \forall u_j \geq 0, \quad j \in V, \quad (23f) \\
& \varphi \geq 0. \quad (23g)
\end{align*}
\]

Here, Constraints (22a) again take the role of implementing the interdiction constraints (9b). Constraints (23a) and (23b) model that \( c \in C_k(x, u) \), and \( (i, j) \in A(k, u) \) for \( (i, j) \in A(d) \). Then, for each feasible kidney exchange solution \( X \in F \), the right-hand side of Constraints (22a) equals the weight of the cycles (and partial) chains containing no interdicted vertices with respect to the variables \( u \).

**Modifications for \( s_{PICEF}(x', F) \).** Below, we also provide the formulation \( s_{PICEF}(x', F) \) tailored to the FSE recourse policy. We again need some modifications to the formulation \( s_{FR,PICEF}(x'; F) \) to correctly model the attacker–defender subproblem in the FSE setting.

To model \( s_{PICEF}(x', F) \) via PICEF, we again need to distinguish between the set of enforceable exchanges – encoded by \( \text{Enf}(x') \) – and the non-enforceable exchanges. Furthermore, we denote by \( \text{Enf}(x) \) and \( \text{Enf}(\xi) \) the sets of enforceable cycles and enforceable chains respectively, i.e.,
\[
\text{Enf}(x) = \{ c \in C_k | x_c = 1 \},
\]
\[
\text{Enf}(\xi) = \{ d \in D_L | \xi_{ij} = 1 \text{ for all } (i, j, c) \in P_c \text{ such that } (i, j) \text{ is the } \ell \text{th arc of } d \}.
\]

For each enforceable cycle \( c \in \text{Enf}(x) \), we require that the variable \( z_c \geq 1 \) if and only if \( c \in C_k(x, u) \) given \( u \), i.e., we require \( z_c \geq 1 \) if \( c \) is an enforced cycle given \( x \) and \( u \). Similarly, we require for each enforceable chain \( d \in \text{Enf}(\xi) \) that the variable \( z_{ij}^d \geq 1 \) if and only if \( (i, j) \in A(\xi, u) \) given \( \xi \) and \( u \). In other words, \( z_{ij}^d \geq 1 \) if the arc \( (i, j) \) is enforced given \( \xi \) and \( u \).

Furthermore, we introduce variables \( x_{ij} \in \{0, 1\} \) for each pair or NDD \( j \in P \cup N \) indicating whether or not \( j \) is involved in an enforced cycle \( c \in C_k(x, u) \) or an enforced arc \( (i, j) \in A(\xi, u) \), depending on the attack \( u \).

We provide the entire formulation \( s_{PICEF}(x'; F) \), as it is quite different from the formulation \( s_{FR,PICEF}(x'; F) \) for full recourse:
\[
\begin{align*}
\text{min } & \varphi \\
\text{s.t. } & \sum_{c \in C_k} w_c(x') z_c \\
& + \sum_{d \in D_L : (j, i) \in A(d)} u_j(x') \sum_{z_{ij} \in \{0, 1\}} w_{ij}(x') z_{ij}^d, \quad X \in F, \quad (24a) \\
& z_c \geq 1 - \sum_{j \in P(c)} u_j, \quad c \in C_k, \quad (24b) \\
& \forall c' \sum_{j \in P(d^{-j})} u_j \geq 1, \quad d \in D_L : (i, j) \in A(d), \quad (24c) \\
& \forall u_j \geq 0, \quad j \in V, \quad (24d)
\end{align*}
\]
max for each arc \( c \in w \) weights involved in an enforced arc \( t \). Finally, we provide the PICEF-based formulation

\[
\sum_{(i,j) \in A} \psi^w_{i,j} \leq \sum_{(i,j) \in P_L} \psi^v_{i,j}, \quad j \in P, \epsilon \in L \setminus \{L\}. \tag{26d}
\]

Constraints (26b) and (26c) ensure that the solution \((\psi, \psi^w)\) encodes a packing of cycles and position-indexed arcs, as pairs and NDDs respectively are limited to be involved in at most one exchange. Constraints (26d) are the precedence constraints for position-indexed arcs in the recourse solution, as an arc going out of pair \( j \) with position index \( \epsilon + 1 \) can only exist in the recourse solution if there exists an arc going into pair \( j \) with position index \( \epsilon \). The separation problem modeled by Formulation (26) is equivalent to a weighted kidney exchange problem on the subgraph induced by \( R_u \).

To obtain the modified version \( R'_u \) of the PICEF-based separation problem, we also need to take into account which cycles and (partial) chains of solutions are feasible. In the case of cycles, this can be realized again through a modification of the cycle weights, i.e.,

\[
w'_i(x', u) := \begin{cases} 1, & \text{if } c \notin C_k(x, u), \\ w_i(x')V + 1, & \text{otherwise}. \end{cases}
\]

Again, as position-indexed arcs only hold local information of a chain, a similar modification does not work for position-indexed arc variables, as it might allow for infeasible chains of large weight. To overcome the issue, we introduce variables \( \psi^w_{i,j} \) for each arc \((i, j) \in P\) to encode the feasible partial chains of the chains encoded by \( \psi^w \) in the recourse solution under \( u \) (i.e., the part of the chains encoded by \( \psi^w \), still feasible after attacks \( u \)).

The arc weights \( w'_i(x', u) \) for all arcs \((i, j) \in A\) are then defined as

\[
w'_i(x', u) := \begin{cases} w_i(x')V + 1, & \text{if } i \in N, \\ w_i(x')V, & \text{otherwise}. \end{cases}
\]

Here, the arcs going out of a non-directed donor \( i \in N \) are given a slightly larger weight to prioritize solutions with the maximum number of chains. Such solutions intuitively lead to stronger interdiction cuts, as it will be more difficult for an attacker with a limited attack budget to disrupt all exchanges.

Although the feasible (sub)chains of the optimal recourse solution are given by the \( \psi \)-variables, the terms for the chains in the cuts we generate are induced by the \( \psi^w \)-variables. The idea here is to find a maximal feasible kidney exchange solution \((\psi^w, \psi^w)\)—with respect to the entire graph \( G \)—such that its non-attack cycles and (sub)chains form an optimal recourse solution under \( u \). To ensure maximality, we assign an objective coefficient of 1 to the \( \psi \)-variables. This value ensures any optimal solution to the modified recourse problem still induces an optimal solution to the original recourse problem if we restrict ourselves to the set of non-interdicted exchanges. The modified recourse problem \( R'_u \) can then be written as

\[
\max \sum_{c \in C_k} w^u_c \leq \sum_{(i,j) \in P_L} \psi^v_{i,j}, \quad j \in P, \epsilon \in L \setminus \{L\}. \tag{27a}
\]

s.t. \[
\sum_{c \in C_k} \psi^w_{i,j} \leq 1 - u_i, \quad i \in N. \tag{27b}
\]

\[
\sum_{(i,j) \in P_L} \psi^v_{i,j} \leq 1 - u_i, \quad i \in N. \tag{27c}
\]

\[
\sum_{(i,j) \in P_L} \psi^w_{i,j} \leq 1 - u_i, \quad i \in N. \tag{27d}
\]
In this model, Constraints (27b)–(27d) require that the solution \( x^n \), \( y^n \) induces a feasible kidney exchange solution on \( G \). The remaining constraints are used to restrict the values of the variables \( x^n \) for each arc \((i,j) \in A\), i.e., the variables that correspond to feasible subchains with respect to \( u \). Constraints (27e) impose that \( y^n \) can be set to 1 only if \( y^n \) is 1 for some position index \( \ell \). Furthermore, Constraints (27f)–(27g) require that \( y^n \) is 0 if either \( i \) or \( j \) is attacked. Constraints (27h) are precedence constraints, i.e., the arcs \((i,j) \in A\) with \( y^n_{ij} = 1 \) indeed induce chains.

**Modifications for \( \text{FSSE,CC}(x; F) \).** We can obtain the formulations for \( R_{\text{FSSE,CC}}(x; u) \) and \( R_{\text{FR,FSSE,CC}}(x; u) \) from Formulation (26) and Formulation (27), respectively, by fixing variables as follows:

\[
\begin{align*}
\gamma^u &= 1, \quad c \in C_k(x; u) \\
\nu^u_{ij} &= 1, \quad (i,j) \in A(\xi, u).
\end{align*}
\]

**Remark 4.** All of the formulations provided in this section could be extended to the case of both vertex and arc failures. For this purpose, one needs to redefine the set of attacks as

\[
U' = \{(U, A') \mid U \subseteq V, A' \subseteq A, |U| \leq B', |A'| \leq B'\}.
\]

Furthermore, we need additional variables \( u_{ij} \in \{0,1\} \) for each \((i,j) \in A\), indicating whether or not the arc \((i,j) \) is attacked, and additional constraints need to be added to the formulations that disallow the use of an exchange if one of its arcs is attacked.

### 3.3. Strength of formulations

Typically, the strength of different MIP formulations is compared by the value of their LP relaxations. In the context of the basic (single-stage) kidney exchange problem, this has been done by Dickerson et al. (2016) for the cycle-chain formulation and position-indexed formulations respectively. This comparison, however, is only meaningful in case the compared MIP models have the same optimal objective value. In the following, we will show that the optimal MIP value for PICEF can strictly dominate the one of CC, with regard to the second-level problems \( s_{\text{CC,CC}}(x; F) \) and \( s_{\text{FR,CC}}(x; F) \) respectively.

Recall that we solve these through a cut generation procedure. Most importantly, we solve the relaxations \( s_{\text{FR,CC}}(x; F) \) and \( s_{\text{FR,CC}}(x; F) \), where a limited set \( F \subset F \) of kidney exchange solutions is considered. We show that the relaxation of the PICEF-based formulation is stronger than the relaxation of the CC-based formulation. We denote by \( s_{\text{FR,CC}}(x; F) \) the value of the optimal solution to \( (24) \), and by \( s_{\text{FR,CC}}(x; F) \) the optimal solution value of \((15)\).

**Theorem 5.** Given a graph \( G \), set of exchanges \( \mathcal{E} \), an initial solution, encoded by CC and PICEF by \( x \) and \( x' \) respectively, and a set of feasible kidney exchange solutions \( F \subset F \), we have that \( s_{\text{FR,CC}}(x; F) \geq s_{\text{FR,CC}}(x; F) \). Furthermore, there exist instances for which \( F \), \( s_{\text{FR,CC}}(x; F) \geq s_{\text{FR,CC}}(x; F) \).

**Proof.** To prove \( s_{\text{FR,CC}}(x'; F) \geq s_{\text{FR,CC}}(x; F) \), we compare our approach to the recent work of Dickerson et al. (2016) that the LP relaxation of the semi-chain for-
the implementation based on the cycle-chain and the position-indexed chain-edge formulations, respectively. These cutting plane approaches are compared with the branch-and-bound approach by Carvalho et al. (2021) on practical instances for the Full Recourse setting. In this section, we investigate the following questions:

**Full Recourse:**

(Q1) Are cut-CC and/or cut-PICEF computationally more efficient than the branch-and-bound approach, both with respect to computation times and the number of solved instances?

(Q2) Does cut-PICEF outperform cut-CC or vice versa?

**FSE:**

(Q3) Are cut-CC and/or cut-PICEF computationally tractable methods for solving the robust kidney exchange problem whenever recourse is allowed according to the FSE policy?

(Q4) Does cut-PICEF outperform cut-CC or vice versa?

(Q5) Are the robust objective values under the FSE setting comparable to the values for Full Recourse, i.e., how large is the loss in the guaranteed number of transplants by imposing FSE constraints on recourse solutions?

In the following, we first discuss our computational setup and used test sets in Section 4.1. Afterward, we present a comparison of the cutting plane algorithms against the branch-and-bound scheme in Section 4.2. Moreover, we explore the tractability of instances under the FSE policy by cut-CC and cut-PICEF in Section 4.3. We finish by discussing the impact of our cut-lifting techniques in Section 4.4.

### 4.1. Test sets and computational setup

In our experiments, we have used the same test set as Carvalho et al. (2021) which is publicly available. This test set consists of ninety graphs of which thirty contain 20, 50, and 100 vertices, respectively. More characteristics of these graphs can be found in Carvalho et al. (2021), they are based on the instance generator described by Saidman et al. (2006). We note that recently Delorme et al. (2022) introduced new instance generators based on UK data. The resulting instances are sparser, and therefore generally easier computationally. From the graphs in the Carvalho et al. set, we generate different instances of the full recourse robust kidney exchange problem by bounding the length $K$ and $L$ of the considered cycles and chains, respectively.

To allow for a fair comparison, we have implemented both our new methods and the reference method in C/C++ using the mixed-integer programming framework SCIP 8.0.2 with SoPlex 6.0.2 as LP solver, see Bestuzheva et al. (2023). All computations were run on a Linux cluster with Intel Xeon Platinum 8260 2.4 GHz processors. The code was executed using a single thread. The time limit of all computations is 3600s per instance.

For cut-CC and the branch-and-bound method, we compute for each instance all cycles of length at most $K$ and chains of length at most $L$ present in its compatibility graph. For cut-PICEF, we again compute the cycles and generate all position-indexed arcs that could potentially be part of a feasible chain, i.e., each position-indexed arc $(i, j, f) \in \mathcal{P}$ for which there exists a path from an NDD to $i$ of length $f \leq 1$ not containing $j$.

In all methods, all cycle variables (and, if applicable, all chain or all position-indexed arc variables) are added to the initial model. Column generation techniques exist for these models (see Glorie et al., 2014; Plaut et al., 2016 among others), but these are not efficient for the smaller chain and cycle sizes we study (Dickerson et al., 2016). In the branch-and-bound method, there are some degrees of freedom in selecting the branching strategy. To be able to produce results as consistent as possible with the ones described in Carvalho et al. (2021), we have implemented the same strategies as discussed there. A summary of these strategies can be found in Appendix B. When comparing with the branch-and-bound scheme of Carvalho et al. (2021), we consider the most basic implementation of cut-CC and cut-PICEF, without lifting. We also conducted experiments with the cut lifting methods described in Sections 3.1.3 and 3.2.3 enabled, which we will discuss in Section 4.4 in more detail.

### 4.2. Comparison of solution methods for full recourse

To empirically find answers to Questions 1, we have conducted experiments with varying maximum cycle and chain lengths $K$ and $L$ for the compatibility graphs described in Section 4.1. The maximum cycle length in our experiments takes value $K \in \{3, 4\}$; the maximum chain length is $L \in \{2, 3, 4\}$. We allow to attack between 1 and 4 vertices per instance in our experiments, i.e., the attack budget $B \in \{1, 2, 3, 4\}$; attacks of arcs are not taken into account, which is consistent with the experiments by Carvalho et al. (2021).

In comparison to the experiments described in Carvalho et al. (2021), we consider the impact of varying cycle lengths, while they only considered $K = 3$. We do include $K = 4$ since multiple European countries allow cycles of this size, e.g., the Czech Republic and The Netherlands, see Biró et al. (2020). To be concise, we only report the tables for $K = 3$ in the main text, as the computational results for $K = 4$ depict a similar comparison between methods. The results for $K = 4$ are presented in Appendix C. The different chain lengths in our experiments are the same as in Carvalho et al. (2021) (note, however, that they measure the length of a chain by its number of vertices instead of arcs, i.e., the numerical values are shifted by 1 compared to their article). Table 1 reports on our experiments for $K = 3$ and different $L$ and $B$ with respect to the Full Recourse policy.

Before we proceed with a discussion of our experiments, we describe the common structure of the presented tables. All mean values are given in arithmetic mean with the only exception being the average solving time, which is measured in shifted geometric mean

$$\frac{1}{n} \prod_{i=1}^{n} (y_i + s)^{1/n} - s,$$

where we use a shift of $s = 10$. The reason for the latter is to reduce the impact of outliers and instances with a very small running time.

Column “#vertices” reports the number of vertices in the tested graphs, whereas column “#opt” provides the number of instances solved within the time limit. The mean solution time per instance (in seconds) is given in “time total”. For cut-CC and cut-PICEF, the columns “stage 2” and “stage 3” present the average proportion of running time spent in solving attacker-defender subproblems and recourse problems, respectively. For the branch-and-bound method, “stage 2” represents the problem of solving the attacker-defender subproblem with the branch-and-bound algorithm. Note that all reported times are in seconds. Column “#att.” reports on the mean number of generated attacks per solved instance; if no instance from a test set could be solved within the time limit, the corresponding entry is “—”. Finally, column “#BB-nodes” gives the mean number of branch-and-bound nodes per solved instance for the branch-and-bound method. For cut-CC and cut-PICEF, the column “#sub” shows the mean number of recourse problems that needed to be solved per instance solved within the time limit.

In our experiments for maximum cycle length $K = 3$, we observe that all methods can solve all instances based on graphs with 20 vertices within a matter of seconds. For graphs with 50 vertices, however, there exist instances that the branch-and-bound method fails to solve within the time limit, for $L = 2$ and attack budget $B = 4$; for larger maximum chain lengths, i.e., $L \in \{3, 4\}$, there already exist instances
that could not be solved for which the attack budget \( B \) equals 3 or 2, respectively. In contrast to this, cut-CC could solve all instances with up to 50 vertices within the time limit whenever \( L \in \{2,3\} \), except for one instance. For \( L = 4 \), cut-CC can still solve over 96% of these instances within one hour. For cut-PICEF, there are slightly more (4) instances with at most 50 vertices and \( L \in \{2,3\} \) that could not be solved within the time limit. However, for \( L = 4 \), almost 99% of these instances could still be solved within the time limit. Furthermore, the instances that could not be solved were based on \( B = 4 \) exclusively.

For instances with 100 vertices in the compatibility graph, the branch-and-bound method is hardly able to solve any instance if \( B \geq 3 \) and \( L \geq 2 \); for increasing maximum chain lengths \( L \), even instances with \( B \in \{1,2\} \) become challenging for the branch-and-bound approach. Whenever \( L = 2 \), the cut-CC approach is superior as it allows solving more instances than cut-PICEF for all possible attack budgets \( B \). The opposite holds whenever \( L \geq 3 \). The benefit of cut-PICEF over cut-CC and branch-and-bound is most apparent for \( L = 4 \). In this case, cut-PICEF succeeds in solving over two-thirds of the number of instances with 100 vertices, whereas both cut-CC and branch-and-bound could only solve 4 and 3 out of the 120 instances, respectively.

Based on these experiments, we can answer Question (Q1) affirmatively as the cutting plane approaches can consistently solve more instances than the branch-and-bound method within the time limit. Specifically, we have accomplished a running time improvement of one order of magnitude for most instance sets. Furthermore, regarding Question (Q2), comparing the two cutting plane methods with each other, there is a transition related to the choice of \( L \) in which cut-PICEF starts outperforming cut-CC. Whenever the maximum chain length \( L \) is small, there exists an overhead in the formulation size for cut-PICEF causing it to be slower than cut-CC, whereas this comparison flips in case \( L \) increases. These conclusions are also supported by our experiments with maximum cycle length \( K = 4 \), as can be seen in Table C1. This behavior is expected, due to the advantages in model size of PICEF with longer chain lengths. Furthermore, the advantage of PICEF's stronger cuts (Theorems 5 and 6) for the attacker–defender subproblem will grow as the chain length increases.

Observe that this is also reflected in our computational results. For most parameter settings, the average number of subproblems solved per master iteration is higher for cut-CC compared to cut-PICEF, even when \( L = 2 \). However, we also observe substantial variability; on an instance level, there are many examples of cut-CC requiring fewer iterations than cut-PICEF. This is possible as the recourse problems have multiple optima and the solvers for cut-CC and cut-PICEF might determine different recourse solutions, i.e., leading to different interdiction cuts.

**Remark 7.** When comparing the number of generated attacks for the branch-and-bound method with the numbers reported by Carvalho et al. (2021), we observe that our numbers are higher in general.
Although we tried to implement the method as close as possible to the implementation of Carvalho et al. (2021), cf. Appendix B, there are still factors that might negatively impact the number of generated attacks in our implementation. Among others, the order of cycles/chains might be different in the two implementations, which might cause to find different optimal solutions in each iteration, and the MIP solver used in our experiments is different from the one used by Carvalho et al. (2021). Thus, since these factors impact both the branch-and-bound and cutting plane algorithms equally, our implementation does not give an advantage to either of these methods.

### 4.3. FSE: exploring tractability and efficiency

As the benefit of the cutting plane methods over branch-and-bound for the Full Recourse setting is evident, we have implemented the Fix Successful Exchanges (FSE) policy only for cut-CC and cut-PICEF. The results of our computational experiments for FSE, again based on the most basic implementation without cut lifting, can be found in Table 2.

Again, we see that all of the instances based on graphs with 20 vertices can be solved within a matter of seconds, both for cut-CC and cut-PICEF.
Regarding the comparison of cut-CC and cut-PICEF, we again observe a transition as in the FR setting; for the most challenging instances with \( L \in \{2, 3\} \), cut-CC is superior to cut-PICEF: for 100 vertices and all different attack budgets \( B \in \{1, 2, 3, 4\} \), cut-CC can solve more instances within the time limit compared to cut-PICEF, whereas the opposite is true whenever \( L \geq 4 \). A similar comparison can be made on the average computation times over all solved instances. The only difference here is that the transition occurs for a larger maximum chain length; cut-PICEF now starts to outperform cut-CC only whenever \( L = 4 \), whereas, in the Full Recourse setting, this was the case for \( L = 3 \) already.

To answer Questions, we can conclude that for small instances, both cut-CC and cut-PICEF can solve the problem within seconds. Nevertheless, when the kidney exchange pools and/or attack budgets grow in size, both cut-CC and cut-PICEF struggle to solve a significant fraction of the instance sets within the time limit. The impact of increasing \( L \) is more gradual for cut-PICEF, whereas, for cut-CC, a clear transition is visible between \( L = 2 \) and \( L = 3 \), leading to a similar performance comparison between cut-CC and cut-PICEF as we observed for Full Recourse. More sophisticated algorithmic techniques will be necessary to solve instances of real-life mid-size to large kidney exchange programs.

Furthermore, we have investigated the impact of the constraints of the FSE policy on the final master objective value of the instances, which is depicted in Table 3. For this comparison, we can only include the instances that we were able to solve within the time limit for both the Full Recourse and the FSE setting. We have aggregated our results for each kidney exchange pool size (20, 50, 100), as we believe it is one of the main instance features impacting the objective value in general.

Observe that for most instances, the objective value, i.e., a guaranteed number of patients from the initial solution receiving a transplant, is rather similar for both recourse policies. In particular, one can see that for more than 85% of the instances with 20 vertices, the objective values of the Full Recourse and FSE settings coincide, whereas the objective values differ by more than one for less than 0.5% of the instances. Moreover, for any of the instances solved to optimality for both FR and FSE, the maximum difference in the objective values observed is 2. We conclude that imposing the constraints on the recourse options with respect to fixing successful initial exchanges does not drastically decrease the guaranteed number of initial patients receiving a transplant given all possible attacks in a limited attack budget. Consequently, imposing the FSE policy might be a feasible option in practice, if the advantages of reducing the logistical challenges outweigh the small, albeit non-negligible, loss of transplants in the worst case.

### 4.4. Evaluation of the impact of lifting

In the experiments discussed in Section 4, we have deliberately not considered cut lifting procedures to allow for a fair comparison of cut-CC and cut-PICEF with the branch-and-bound scheme. As mentioned above, we have also conducted experiments to measure the impact on the cutting plane approaches of the lifting schemes proposed in Sections 3.1.3 and 3.2.3. Table 4 shows the computational results for both cut-CC and cut-PICEF, and for Full Recourse and FSE, respectively. We have restricted our results in this table to instances with 100 vertices, to focus on the hardest instances where algorithmic improvements are most necessary. In the columns below “with lifting”, we present the results where the modified recourse problems are used to separate interdiction cuts. In the columns below “without lifting”, we review the results of cut-CC and cut-PICEF from Tables 1 and 2 for easier comparison. Again, the attack budgets \( B \) from 1 to 4 are used.

For Full Recourse, both cut-CC and cut-PICEF benefit from cut lifting, as it has a positive impact on both the number of instances solved and the total computation time. While the number of generated attacks per optimally solved instance is roughly the same, the number of subproblems that need to be solved is significantly smaller whenever cut lifting is enabled. As a result, proving the optimality of solutions to the attacker–defender subproblem takes fewer iterations, which suggests cut lifting is of benefit for reducing computation times.

Interestingly, this only leads to a reduction in running time and an increased number of solved instances for Full Recourse. Especially when cut-PICEF is employed, and for instances with larger maximum chain lengths, the number of solved instances increases substantially when cut lifting is enabled compared to the basic variant of cut-PICEF, as well as a computation time reduction of over 50% for the instances with attack budget \( B = 4 \).

For FSE, no significant benefit is visible when cut lifting is enabled or vice versa. For both cut-CC and cut-PICEF, enabling cut lifting slightly improves the average computation times for most parameter settings, but it cannot be regarded as an improvement under all circumstances. For chain length \( L = 2 \) and \( B = 1 \), enabling cut lifting decreases the number of instances solved within the time limit, although for larger \( L \) and \( B \), the opposite holds. Moreover, Table 5 shows an aggregated overview of the results of cut lifting. As the
<table>
<thead>
<tr>
<th>cut-CC: FR</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>30</td>
<td>33.27</td>
<td>17.13</td>
<td>20.83</td>
<td>3.4</td>
<td>20.83</td>
<td>30</td>
<td>37.47</td>
</tr>
<tr>
<td>B = 2</td>
<td>30</td>
<td>74.27</td>
<td>9.13</td>
<td>19.33</td>
<td>5.3</td>
<td>57.27</td>
<td>30</td>
<td>102.18</td>
</tr>
<tr>
<td>B = 3</td>
<td>30</td>
<td>163.57</td>
<td>20.28</td>
<td>23.41</td>
<td>6.1</td>
<td>124.57</td>
<td>26</td>
<td>396.43</td>
</tr>
<tr>
<td>B = 4</td>
<td>30</td>
<td>258.16</td>
<td>28.08</td>
<td>21.96</td>
<td>7.5</td>
<td>215.33</td>
<td>21</td>
<td>763.96</td>
</tr>
<tr>
<td>Chain length: 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>28</td>
<td>540.66</td>
<td>5.82</td>
<td>21.28</td>
<td>3.7</td>
<td>26.28</td>
<td>25</td>
<td>597.68</td>
</tr>
<tr>
<td>B = 2</td>
<td>23</td>
<td>1175.08</td>
<td>5.42</td>
<td>21.29</td>
<td>5.4</td>
<td>55.42</td>
<td>18</td>
<td>1415.26</td>
</tr>
<tr>
<td>B = 3</td>
<td>20</td>
<td>1660.15</td>
<td>7.33</td>
<td>28.94</td>
<td>6.2</td>
<td>130.22</td>
<td>11</td>
<td>2893.21</td>
</tr>
<tr>
<td>B = 4</td>
<td>17</td>
<td>2155.48</td>
<td>6.32</td>
<td>20.18</td>
<td>8.1</td>
<td>181.96</td>
<td>6</td>
<td>2956.98</td>
</tr>
<tr>
<td>Chain length: 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>3</td>
<td>3159.06</td>
<td>10.15</td>
<td>4.91</td>
<td>3.7</td>
<td>16.03</td>
<td>3</td>
<td>3283.32</td>
</tr>
<tr>
<td>B = 2</td>
<td>2</td>
<td>3401.13</td>
<td>8.40</td>
<td>7.42</td>
<td>4.5</td>
<td>74.51</td>
<td>1</td>
<td>3477.83</td>
</tr>
<tr>
<td>B = 3</td>
<td>2</td>
<td>3400.58</td>
<td>5.16</td>
<td>5.91</td>
<td>4.0</td>
<td>86.00</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>B = 4</td>
<td>1</td>
<td>3400.00</td>
<td>5.16</td>
<td>5.91</td>
<td>4.0</td>
<td>86.00</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>cut-CC: FSE</td>
<td>#opt</td>
<td>Time</td>
<td>#att</td>
<td>#sub</td>
<td>#opt</td>
<td>Time</td>
<td>#att</td>
<td>#sub</td>
</tr>
<tr>
<td>Chain length: 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>26</td>
<td>189.69</td>
<td>30.38</td>
<td>0.60</td>
<td>6.5</td>
<td>22.77</td>
<td>26</td>
<td>187.68</td>
</tr>
<tr>
<td>B = 2</td>
<td>12</td>
<td>1490.95</td>
<td>3.97</td>
<td>0.52</td>
<td>11.1</td>
<td>50.22</td>
<td>11</td>
<td>1733.28</td>
</tr>
<tr>
<td>B = 3</td>
<td>6</td>
<td>2901.00</td>
<td>5.41</td>
<td>0.42</td>
<td>16.2</td>
<td>227.88</td>
<td>4</td>
<td>2808.18</td>
</tr>
<tr>
<td>B = 4</td>
<td>1</td>
<td>3510.64</td>
<td>2.65</td>
<td>0.23</td>
<td>13.0</td>
<td>107.00</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>Chain length: 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>24</td>
<td>2132.62</td>
<td>3.62</td>
<td>0.70</td>
<td>5.6</td>
<td>20.13</td>
<td>24</td>
<td>2198.62</td>
</tr>
<tr>
<td>B = 2</td>
<td>2</td>
<td>3546.73</td>
<td>11.28</td>
<td>0.54</td>
<td>17.5</td>
<td>84.00</td>
<td>2</td>
<td>3547.28</td>
</tr>
<tr>
<td>B = 3</td>
<td>0</td>
<td>3600.00</td>
<td>12.27</td>
<td>0.52</td>
<td>–</td>
<td>–</td>
<td>2</td>
<td>3547.28</td>
</tr>
<tr>
<td>B = 4</td>
<td>0</td>
<td>3600.00</td>
<td>9.51</td>
<td>0.19</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td>3518.64</td>
</tr>
<tr>
<td>Chain length: 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>1</td>
<td>3561.68</td>
<td>19.96</td>
<td>0.17</td>
<td>6.0</td>
<td>20.00</td>
<td>1</td>
<td>3566.68</td>
</tr>
<tr>
<td>B = 2</td>
<td>0</td>
<td>3600.00</td>
<td>12.39</td>
<td>0.05</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>B = 3</td>
<td>0</td>
<td>3600.00</td>
<td>15.62</td>
<td>0.03</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>B = 4</td>
<td>0</td>
<td>3600.00</td>
<td>17.94</td>
<td>0.04</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>cut-PICEF: FR</td>
<td>#opt</td>
<td>Time</td>
<td>#att</td>
<td>#sub</td>
<td>#opt</td>
<td>Time</td>
<td>#att</td>
<td>#sub</td>
</tr>
<tr>
<td>Chain length: 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>30</td>
<td>62.88</td>
<td>11.65</td>
<td>45.83</td>
<td>3.3</td>
<td>21.77</td>
<td>30</td>
<td>43.32</td>
</tr>
<tr>
<td>B = 2</td>
<td>30</td>
<td>152.87</td>
<td>11.69</td>
<td>48.87</td>
<td>5.3</td>
<td>57.13</td>
<td>30</td>
<td>137.89</td>
</tr>
<tr>
<td>B = 3</td>
<td>30</td>
<td>319.64</td>
<td>12.80</td>
<td>53.06</td>
<td>6.4</td>
<td>94.51</td>
<td>25</td>
<td>518.40</td>
</tr>
<tr>
<td>B = 4</td>
<td>29</td>
<td>542.21</td>
<td>17.86</td>
<td>55.70</td>
<td>7.8</td>
<td>166.00</td>
<td>18</td>
<td>1254.98</td>
</tr>
<tr>
<td>Chain length: 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>30</td>
<td>119.61</td>
<td>13.72</td>
<td>40.38</td>
<td>3.9</td>
<td>22.89</td>
<td>30</td>
<td>114.01</td>
</tr>
<tr>
<td>B = 2</td>
<td>30</td>
<td>244.16</td>
<td>9.25</td>
<td>55.42</td>
<td>4.6</td>
<td>56.78</td>
<td>29</td>
<td>303.46</td>
</tr>
<tr>
<td>B = 3</td>
<td>27</td>
<td>752.24</td>
<td>9.85</td>
<td>60.42</td>
<td>6.4</td>
<td>95.89</td>
<td>21</td>
<td>1262.74</td>
</tr>
<tr>
<td>B = 4</td>
<td>27</td>
<td>911.99</td>
<td>10.91</td>
<td>59.01</td>
<td>7.5</td>
<td>141.63</td>
<td>18</td>
<td>2140.89</td>
</tr>
<tr>
<td>Chain length: 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>20</td>
<td>181.29</td>
<td>10.58</td>
<td>49.64</td>
<td>3.3</td>
<td>24.00</td>
<td>30</td>
<td>195.42</td>
</tr>
<tr>
<td>B = 2</td>
<td>30</td>
<td>409.29</td>
<td>8.86</td>
<td>46.99</td>
<td>5.3</td>
<td>46.77</td>
<td>27</td>
<td>674.06</td>
</tr>
<tr>
<td>B = 3</td>
<td>25</td>
<td>1165.20</td>
<td>7.56</td>
<td>54.31</td>
<td>6.6</td>
<td>83.97</td>
<td>18</td>
<td>1908.69</td>
</tr>
<tr>
<td>B = 4</td>
<td>21</td>
<td>1525.97</td>
<td>10.41</td>
<td>63.18</td>
<td>7.3</td>
<td>115.13</td>
<td>6</td>
<td>3083.76</td>
</tr>
<tr>
<td>cut-PICEF: FSE</td>
<td>#opt</td>
<td>Time</td>
<td>#att</td>
<td>#sub</td>
<td>#opt</td>
<td>Time</td>
<td>#att</td>
<td>#sub</td>
</tr>
<tr>
<td>Chain length: 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>24</td>
<td>654.49</td>
<td>0.36</td>
<td>0.11</td>
<td>4.7</td>
<td>11.33</td>
<td>25</td>
<td>661.56</td>
</tr>
<tr>
<td>B = 2</td>
<td>10</td>
<td>2199.59</td>
<td>0.17</td>
<td>0.06</td>
<td>6.8</td>
<td>23.11</td>
<td>9</td>
<td>2259.02</td>
</tr>
<tr>
<td>B = 3</td>
<td>5</td>
<td>2765.90</td>
<td>1.20</td>
<td>0.21</td>
<td>8.4</td>
<td>176.76</td>
<td>2</td>
<td>3445.56</td>
</tr>
<tr>
<td>B = 4</td>
<td>1</td>
<td>3586.64</td>
<td>1.16</td>
<td>0.15</td>
<td>7.0</td>
<td>68.00</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>Chain length: 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>15</td>
<td>1091.93</td>
<td>0.40</td>
<td>0.06</td>
<td>3.9</td>
<td>8.73</td>
<td>14</td>
<td>1169.76</td>
</tr>
<tr>
<td>B = 2</td>
<td>1</td>
<td>3435.58</td>
<td>0.08</td>
<td>0.06</td>
<td>4.0</td>
<td>18.00</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>B = 3</td>
<td>0</td>
<td>3600.00</td>
<td>0.07</td>
<td>0.06</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>B = 4</td>
<td>0</td>
<td>3600.00</td>
<td>0.09</td>
<td>0.10</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>Chain length: 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B = 1</td>
<td>6</td>
<td>2415.62</td>
<td>1.89</td>
<td>0.04</td>
<td>4.0</td>
<td>9.00</td>
<td>6</td>
<td>2563.12</td>
</tr>
<tr>
<td>B = 2</td>
<td>0</td>
<td>3600.00</td>
<td>0.07</td>
<td>0.04</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>B = 3</td>
<td>0</td>
<td>3600.00</td>
<td>0.09</td>
<td>0.06</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>B = 4</td>
<td>0</td>
<td>3600.00</td>
<td>0.12</td>
<td>0.06</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
</tbody>
</table>
positive impact of cut lifting was most pronounced for Full Recourse, we restrict ourselves to the results of this recourse policy.

We conclude that under almost all parameter settings, cut lifting is a valuable feature for both cut-CC and cut-PICEF to obtain a performant algorithm for the Full Recourse policy, especially for less restrictive maximum lengths of cycles and chains in a kidney exchange solution. This is consistent with similar experiments conducted in literature, such as the modifications of interdiction cuts as reported by Fischetti et al. (2019).

### 5. Conclusion

In this paper, we investigate a robust optimization variant of the kidney exchange problem. In this setting, donors and/or recipients might decide to drop out of the program after the proposal of exchanges and before the actual transplants take place, due to a variety of reasons. This is a practical problem since donors and pairs leaving a kidney exchange program lead to broken cycles and chains of initially proposed transplants.

We propose a cutting plane algorithm for solving the attacker–defender subproblem based on the cycle-chain formulation and the position-indexed chain-edge formulation. The algorithms are based on a reformulation of the attacker–defender subproblem with constraints for any feasible kidney exchange solution. Cuts are separated by solving the problem of finding the optimal recourse solution given an initial kidney exchange solution and attack. These algorithms are implemented for two recourse policies, the Full Recourse policy introduced in Carvalho et al. (2021) and the Fix Successful Exchanges policy. The latter aims to give additional guarantees to participating recipients in case none of the involved people of an exchange drop out of the kidney exchange program. Furthermore, a lifting technique is deployed to obtain solutions on the entire compatibility graph that still have maximum recourse value on the non-attacked cycles and chains.

Our computational results show that both the CC- and PICEF-based implementations (cut-CC and cut-PICEF) of our cutting plane algorithm outperform the state-of-the-art in the Full Recourse setting, which is a branch-and-bound type algorithm proposed in Carvalho et al. (2021) when we consider benchmark instances with $|V| \geq 50$. We observe a transition phase between cut-CC and cut-PICEF. Whenever we allow only small chains, cut-CC outperforms cut-PICEF. However, when larger chains are allowed, cut-PICEF becomes the stronger method and the benefit of having a polynomial-size formulation in terms of chains becomes strongly apparent. In the FSE setting, cut-CC and cut-PICEF can solve most instances with up to 50 instances and small attack budgets $B$, rendering the problem tractable for small kidney exchanges but challenging for mid-size to large pools in practice.

Enabling the cut lifting technique leads, under most choices for parameter settings, to a significant reduction in the running time under most choices for parameter settings. The most apparent reason for this is that fewer iterations of the separation problem are needed to solve the attacker–defender subproblem, since they give better feedback on the recourse value that can be realized given a set of attacked vertices. As a result, the generated attacks are more effective, and typically, we need fewer attacks to solve the entire robust optimization model to optimality.

The success of cut lifting highlights an interesting aspect of cutting plane approaches to multi-level problems. Whenever multiple valid cuts are violated in an iteration, the decision of which cut to add can have a significant impact on runtime. The variability in the number of iterations required between cut-CC and cut-PICEF, where different parameter settings, to a significant reduction in the running time under most choices for parameter settings. The most apparent reason for this is that fewer iterations of the separation problem are needed to solve the attacker–defender subproblem, since they give better feedback on the recourse value that can be realized given a set of attacked vertices. As a result, the generated attacks are more effective, and typically, we need fewer attacks to solve the entire robust optimization model to optimality.

### Table 5

Comparison of the cut-CC method with and without lifting for maximum cycle length 3 and different chain lengths, for the Full Recourse policy.

<table>
<thead>
<tr>
<th>Cut-CC</th>
<th>With lifting</th>
<th>Without lifting</th>
</tr>
</thead>
<tbody>
<tr>
<td>#vertices</td>
<td>#opt</td>
<td>Time</td>
</tr>
<tr>
<td>Chain length: 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 120 0.1</td>
<td>2.8</td>
<td>8.0</td>
</tr>
<tr>
<td>50 120 7.3</td>
<td>5.1</td>
<td>41.7</td>
</tr>
<tr>
<td>100 120 132.3</td>
<td>5.6</td>
<td>104.5</td>
</tr>
<tr>
<td>Chain length: 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 120 0.2</td>
<td>2.8</td>
<td>8.4</td>
</tr>
<tr>
<td>50 120 29.6</td>
<td>5.4</td>
<td>45.4</td>
</tr>
<tr>
<td>100 88 1382.8</td>
<td>5.9</td>
<td>98.6</td>
</tr>
<tr>
<td>Chain length: 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 120 0.8</td>
<td>2.9</td>
<td>8.8</td>
</tr>
<tr>
<td>50 118 247.9</td>
<td>5.4</td>
<td>50.7</td>
</tr>
<tr>
<td>100 8 3364.8</td>
<td>5.5</td>
<td>66.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cut-PICEF</th>
<th>With lifting</th>
<th>Without lifting</th>
</tr>
</thead>
<tbody>
<tr>
<td>#vertices</td>
<td>#opt</td>
<td>Time</td>
</tr>
<tr>
<td>Chain length: 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 120 0.2</td>
<td>2.7</td>
<td>7.9</td>
</tr>
<tr>
<td>50 120 11.6</td>
<td>5.3</td>
<td>35.4</td>
</tr>
<tr>
<td>100 119 269.4</td>
<td>5.7</td>
<td>84.8</td>
</tr>
<tr>
<td>Chain length: 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 120 0.3</td>
<td>2.8</td>
<td>7.8</td>
</tr>
<tr>
<td>50 120 27.4</td>
<td>4.9</td>
<td>40.8</td>
</tr>
<tr>
<td>100 114 507.0</td>
<td>5.6</td>
<td>79.2</td>
</tr>
<tr>
<td>Chain length: 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 120 0.5</td>
<td>2.7</td>
<td>7.7</td>
</tr>
<tr>
<td>50 120 48.7</td>
<td>4.9</td>
<td>43.0</td>
</tr>
<tr>
<td>100 106 820.4</td>
<td>5.6</td>
<td>67.4</td>
</tr>
</tbody>
</table>
Proof. To prove \( s^*_{\text{FSE,PICEF}}(x'; F) \geq s^*_{\text{FSE,CC}}(x; F) \), we compare the right-hand sides of Constraints (15b) and (25b). We claim that for a given recourse solution \( x \in F \) and any fixed attack \( u \), the right-hand side of the PICEF variant (25b) is at least as large as the right-hand side of the CC variant (15b), i.e.,

\[
\sum_{c \in X \cap c_k} w_c(x)z_c + \sum_{d \in X \cap D_2} \sum_{(i,j) \in A(d)} w_{ij}(x)c_{ij}^d \\
\geq \sum_{c \in X \cap c_k} w_c(x)z_c + \sum_{d \in X \cap D_2} w_d(x)z_d.
\]

Since the cycle variables are handled identically in both formulations, we must only prove that

\[
\sum_{d \in X \cap D_2} \sum_{(i,j) \in A(d)} w_{ij}(x)c_{ij}^d \geq \sum_{d \in X \cap D_2} w_d(x)z_d.
\]

We can distinguish two cases for the chains \( d \in X \cap D_1 \), and show that

\[
\sum_{(i,j) \in A(d)} w_{ij}(x)c_{ij}^d \geq w_d(x)z_d
\]

in all cases.

Case 1: \( d \) is enforceable. In this case, if no vertex \( v \in V(d) \) is attacked, we have \( z_d = 1 \) (due to Constraints (25c) and (25e)), as well as \( c_{ij}^d = 1 \) for all \((i,j) \in A(d) \) (due to Constraints (25f) and (25h)). By definition, \( \sum_{(i,j) \in A(d)} w_{ij}(x') = w_d(x) \), thus an enforceable chain that is not attacked contributes the same value to the right-hand side in both formulations. If the chain is attacked at any point, then \( z_d = 0 \), and the chain does not contribute to the right-hand side of Constraints (15b). As \( w_{ij}(x') \geq 0 \) and \( c_{ij}^d \geq 0 \) for all \((i,j) \in A(d) \), the inequality in (30) follows.

Case 2: \( d \) is not enforceable. In this case, if no vertex \( v \in V(d) \) is attacked and no intersecting enforceable exchange is enforced, we have \( z_d = 1 \) (due to Constraints (16a)) and \( c_{ij}^d = 1 \) for all \((i,j) \in A(d) \) (due to Constraints (25g)). By definition, \( \sum_{(i,j) \in A(d)} w_{ij}(x') = w_d(x) \), thus a non-enforceable chain that is not attacked contributes the same value to the right-hand sides in both formulations. If the chain is attacked at any point or has an enforced intersecting exchange, then \( z_d = 0 \), and the chain does not contribute to the right-hand side of Constraints (15b). Again, as \( w_{ij}(x') \geq 0 \) and \( c_{ij}^d \geq 0 \) for all \((i,j) \in A(d) \), the inequality in (30) follows.

We again consider the instance depicted in Fig. 5, and show that \( s^*_{\text{FSE,PICEF}}(x'; F) > s^*_{\text{FSE,CC}}(x; F) \) for a suitable subset \( F \subset F \). Again, we consider attack budget \( B = 1 \). The initial solution consists only of the chain \((1,2,3,4)\), which is encoded by \( x \) for CC and \( x' \) for PICEF. As a result, the value of every exchange in the recourse solutions is equal to the number of pairs it contains, i.e., \( w_{ij}(x) = w_{ij}(x') = |V(c)| \) for each cycle \( c \). \( w_{ij}(x) = |V(d)| \) for each chain \( d \) and \( w_{ij}(x') = 1 \) for each arc \((i,j) \in A \). Let \( F = \{X_1, X_2\} \) where \( X_1 = \{(1,2,3,4)\} \) and \( X_2 = \{(3,4)\} \).

For this instance, \( s^*_{\text{FSE,CC}}(x; F) = 0 \), which can be achieved by attacking pair 3 or pair 4. However, in the PICEF formulation, attacking pair 3 or 4 forces \( c_{1,2}^d = 1 \), and \( s^*_{\text{FSE,PICEF}}(x'; F) > 1 \) due to recourse solution \( X_1 \). Not attacking either of these pairs forces \( z_d = 1 \) for the cycle \((3,4)\), and \( s^*_{\text{FSE,PICEF}}(x'; F) \geq 2 \). Consequently, for this instance \( s^*_{\text{FSE,PICEF}}(x'; F) > s^*_{\text{FSE,CC}}(x; F) \). \( \square \)

Appendix B: Implementation of the Carvalho et al. (2021) algorithm

In this appendix, we briefly describe our implementation of the branch-and-bound algorithm by Carvalho et al. (2021) used to solve the combination of stage 2 and 3 in their iterative solution framework. In particular, in this appendix, we lay out the choices we made where decisions are not fully specified in Algorithm 2 in their original paper. We matched these decisions as closely as possible to add details on the Carvalho et al. (2021) implementation received through personal communications. References to lines are as they appear in Algorithm 2 suggested by Carvalho et al. (2021).

Line 8: Node selection. We deploy a depth-first search strategy to select the next subproblem with priority to the attack branch. In other words, if we generated two child nodes \( t_0 \) and \( t_1 \) of branch-and-bound node \( t \), which extend the branching decisions at \( t \) by selecting a new vertex to be attacked and not to be attacked, respectively, we first explore the subtree rooted at \( t_0 \) before proceeding with the subtree rooted at \( t_1 \).

Line 9: Fill for maximal attack. When choosing the maximal attack, we add vertices to be attacked sequentially. For each new node, we first identify the highest weight cycle or chain for which no vertices are attacked yet, and for which there is a vertex that is still unfixed. Ties between cycles and chains are broken arbitrarily. For this cycle or chain, we add the lowest index vertex to the attack. If no such cycles or chains exist, i.e., at least one vertex is already attacked, or all vertices are fixed to be attacked or not attacked, we add the lowest index vertex that is not yet fixed to the attack.

Line 25: Branching. We choose the first vertex we added to the attack

Appendix C: Computational results for \( K = 4 \)

This appendix provides tables summarizing our numerical experiments for instances with cycle length \( K = 4 \). See Tables C.1–C.4.
### Table C.1
Comparison of the different methods for maximum cycle length 4 and different chain lengths, for the Full Recourse policy.

| #vertices | Branch-and-Bound | cut-CC | cut-PGCP | #opt | Time | #att. | #B&B-nodes | #opt | Time | #att. | #sub | #opt | Time | #att. | #sub |
|-----------|-----------------|--------|----------|-------|------|------|------------|-------|------|------|------|-------|------|------|------|------|
|           | Total           | Stage 2| Stage 3  |       |      |      |            | Total | Stage 2| Stage 3|       | Total | Stage 2| Stage 3|       |      |
| Chain length: 2 |
| 1 | 20 | 0.19 | 27.38 | 1.9 | 15.0 | 30 | 0.15 | 24.96 | 17.75 | 1.8 | 7.7 | 30 | 0.24 | 18.77 | 24.50 | 2.0 | 7.3 |
| 30 | 0.18 | 34.78 | 3.0 | 35.5 | 30 | 18.03 | 8.32 | 26.97 | 2.9 | 22.7 | 30 | 18.15 | 7.37 | 24.84 | 3.0 | 21.2 |
| 100 | 23 | 1078.88 | 50.34 | 3.3 | 115.0 | 26 | 893.61 | 5.14 | 37.25 | 3.5 | 45.5 | 26 | 918.91 | 5.62 | 43.40 | 3.5 | 42.1 |
| Chain length: 3 |
| 1 | 20 | 0.61 | 48.54 | 3.2 | 97.2 | 30 | 0.40 | 26.39 | 20.56 | 2.9 | 15.2 | 30 | 0.44 | 23.22 | 16.24 | 3.0 | 14.5 |
| 30 | 0.59 | 71.76 | 5.1 | 813.7 | 30 | 50.99 | 25.07 | 27.72 | 5.0 | 10.5 | 30 | 51.56 | 22.87 | 25.41 | 5.0 | 7.9 |
| 100 | 4 | 3345.15 | 64.84 | 7.4 | 2139.2 | 15 | 1984.66 | 7.78 | 41.12 | 4.9 | 147.7 | 15 | 2384.89 | 8.82 | 44.55 | 5.3 | 142.7 |
| Chain length: 4 |
| 1 | 20 | 1.33 | 63.32 | 3.8 | 324.0 | 30 | 0.55 | 31.08 | 13.95 | 3.6 | 16.1 | 30 | 0.45 | 34.28 | 19.87 | 3.2 | 14.8 |
| 30 | 29 | 727.97 | 90.48 | 7.4 | 1284.86 | 30 | 82.93 | 38.76 | 22.90 | 6.1 | 207.3 | 30 | 139.06 | 43.64 | 18.97 | 6.4 | 240.4 |
| 100 | 0 | 3600.00 | 60.64 | – | – | 8 | 2972.21 | 13.30 | 30.27 | 5.9 | 333.6 | 6 | 3284.15 | 11.60 | 38.67 | 7.8 | 312.8 |
| Chain length: 5 |
| 1 | 20 | 2.82 | 62.26 | 4.8 | 964.1 | 30 | 0.38 | 33.03 | 14.29 | 3.3 | 12.9 | 30 | 0.38 | 36.45 | 15.05 | 3.3 | 13.3 |
| 30 | 12 | 2422.80 | 95.11 | 12.7 | 6349.2 | 28 | 164.12 | 90.80 | 15.87 | 7.8 | 273.6 | 27 | 187.59 | 51.30 | 13.60 | 7.6 | 227.5 |
| 100 | 0 | 3600.00 | 60.86 | – | – | 3 | 3410.61 | 19.56 | 24.22 | 9.0 | 5347.1 | 1 | 3572.15 | 16.26 | 25.13 | 17.0 | 872.0 |

### Notes
- The table compares the performance of Branch-and-Bound, cut-CC, and cut-PGCP for different chain lengths with maximum cycle length 4.
- The columns represent the number of vertices, branch-and-bound time, cut-CC time, cut-PGCP time, and the number of att. B&B-nodes and suboptimal solutions.
- The table includes results for chain lengths 2, 3, 4, and 5.
- Each row represents a different number of vertices, showing the time and number of vertices for each method.
- The table highlights improvements in efficiency and solution quality as the chain length increases.
<table>
<thead>
<tr>
<th>#vertices</th>
<th>#opt</th>
<th>Time</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>#opt</th>
<th>Time</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H = 1</td>
<td>20</td>
<td>30</td>
<td>0.99</td>
<td>31.88 %</td>
<td>8.01 %</td>
<td>3.0</td>
<td>7.9</td>
<td>30</td>
</tr>
<tr>
<td>25</td>
<td>342.60</td>
<td>11.85 %</td>
<td>0.51 %</td>
<td>7.7</td>
<td>35.0</td>
<td>23</td>
<td>425.01</td>
<td>0.50 %</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3481.47</td>
<td>21.97 %</td>
<td>0.36 %</td>
<td>6.5</td>
<td>27.5</td>
<td>3</td>
<td>3494.44</td>
</tr>
<tr>
<td>H = 2</td>
<td>20</td>
<td>30</td>
<td>2.83</td>
<td>35.18 %</td>
<td>6.30 %</td>
<td>4.3</td>
<td>21.4</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>16</td>
<td>978.54</td>
<td>14.67 %</td>
<td>0.83 %</td>
<td>10.1</td>
<td>104.1</td>
<td>14</td>
<td>1235.50</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>15.30 %</td>
<td>0.31 %</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>H = 3</td>
<td>20</td>
<td>30</td>
<td>1.89</td>
<td>48.14 %</td>
<td>6.34 %</td>
<td>4.1</td>
<td>20.9</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>9</td>
<td>1983.83</td>
<td>24.70 %</td>
<td>0.84 %</td>
<td>10.9</td>
<td>226.3</td>
<td>11</td>
<td>1669.92</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>15.24 %</td>
<td>0.21 %</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>H = 4</td>
<td>20</td>
<td>30</td>
<td>1.46</td>
<td>42.06 %</td>
<td>5.83 %</td>
<td>4.1</td>
<td>22.2</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>2271.96</td>
<td>40.37 %</td>
<td>0.91 %</td>
<td>14.1</td>
<td>304.0</td>
<td>8</td>
<td>2039.06</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>15.20 %</td>
<td>0.12 %</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>Chain length: 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H = 1</td>
<td>20</td>
<td>30</td>
<td>2.24</td>
<td>28.40 %</td>
<td>8.36 %</td>
<td>3.3</td>
<td>9.9</td>
<td>30</td>
</tr>
<tr>
<td>25</td>
<td>525.96</td>
<td>14.69 %</td>
<td>0.48 %</td>
<td>7.1</td>
<td>30.0</td>
<td>20</td>
<td>604.26</td>
<td>0.34 %</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>32.01 %</td>
<td>0.38 %</td>
<td>–</td>
<td>–</td>
<td>2</td>
<td>3444.94</td>
</tr>
<tr>
<td>H = 2</td>
<td>20</td>
<td>30</td>
<td>4.10</td>
<td>33.52 %</td>
<td>9.66 %</td>
<td>4.9</td>
<td>21.5</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>12</td>
<td>1738.12</td>
<td>16.03 %</td>
<td>0.69 %</td>
<td>10.2</td>
<td>103.0</td>
<td>11</td>
<td>2003.66</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>19.59 %</td>
<td>0.20 %</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>H = 3</td>
<td>20</td>
<td>30</td>
<td>3.42</td>
<td>43.68 %</td>
<td>15.84 %</td>
<td>4.1</td>
<td>24.5</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>2464.55</td>
<td>26.10 %</td>
<td>0.71 %</td>
<td>11.1</td>
<td>183.3</td>
<td>7</td>
<td>2326.23</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>21.03 %</td>
<td>0.16 %</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>H = 4</td>
<td>20</td>
<td>30</td>
<td>2.27</td>
<td>41.98 %</td>
<td>10.27 %</td>
<td>4.1</td>
<td>21.7</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>2507.32</td>
<td>35.23 %</td>
<td>0.67 %</td>
<td>10.8</td>
<td>156.8</td>
<td>6</td>
<td>2278.47</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>20.95 %</td>
<td>0.14 %</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>Chain length: 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H = 1</td>
<td>20</td>
<td>30</td>
<td>4.87</td>
<td>31.53 %</td>
<td>4.27 %</td>
<td>3.2</td>
<td>9.1</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>13</td>
<td>1749.18</td>
<td>19.61 %</td>
<td>0.62 %</td>
<td>6.9</td>
<td>23.6</td>
<td>18</td>
<td>1105.75</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>25.56 %</td>
<td>0.18 %</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td>3554.88</td>
</tr>
<tr>
<td>H = 2</td>
<td>20</td>
<td>30</td>
<td>8.64</td>
<td>35.55 %</td>
<td>9.41 %</td>
<td>5.2</td>
<td>25.0</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>3134.54</td>
<td>17.63 %</td>
<td>0.55 %</td>
<td>9.0</td>
<td>49.7</td>
<td>7</td>
<td>2561.70</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>21.42 %</td>
<td>0.06 %</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>H = 3</td>
<td>20</td>
<td>30</td>
<td>6.97</td>
<td>43.25 %</td>
<td>9.94 %</td>
<td>4.4</td>
<td>30.4</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>3431.51</td>
<td>22.22 %</td>
<td>0.39 %</td>
<td>11.5</td>
<td>113.5</td>
<td>6</td>
<td>2337.49</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>22.95 %</td>
<td>0.06 %</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
<tr>
<td>H = 4</td>
<td>20</td>
<td>30</td>
<td>3.77</td>
<td>44.70 %</td>
<td>5.09 %</td>
<td>4.0</td>
<td>21.3</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>3900.53</td>
<td>32.45 %</td>
<td>0.40 %</td>
<td>11.0</td>
<td>142.0</td>
<td>5</td>
<td>2530.14</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>3600.00</td>
<td>24.38 %</td>
<td>0.06 %</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
</tr>
</tbody>
</table>
Table C.3
Impact of cut lifting restricted to instances with 100 pairs/NDDs, for cut-CCE and cut-PICEF, given maximum cycle length $k = 4$ and different chain lengths, for Full Recourse (FR) and PICEF.

<table>
<thead>
<tr>
<th>cut-CCE: FR</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
<th>#opt</th>
<th>Time</th>
<th>Total</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 2</td>
<td>B = 1</td>
<td>28</td>
<td>591.86</td>
<td>5.19%</td>
<td>24.36%</td>
<td>3.4</td>
<td>25.0</td>
<td>26</td>
<td>893.64</td>
</tr>
<tr>
<td>B = 2</td>
<td>23</td>
<td>1052.83</td>
<td>5.88%</td>
<td>25.35%</td>
<td>4.6</td>
<td>64.8</td>
<td>15</td>
<td>1904.66</td>
<td>7.79%</td>
</tr>
<tr>
<td>B = 3</td>
<td>15</td>
<td>1873.45</td>
<td>5.75%</td>
<td>20.20%</td>
<td>5.3</td>
<td>93.6</td>
<td>8</td>
<td>2972.71</td>
<td>13.30%</td>
</tr>
<tr>
<td>B = 4</td>
<td>17</td>
<td>1833.98</td>
<td>7.87%</td>
<td>21.47%</td>
<td>6.4</td>
<td>127.2</td>
<td>3</td>
<td>3450.61</td>
<td>19.56%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cut-PICEF: FR</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
<th>#opt</th>
<th>Time</th>
<th>Total</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 2</td>
<td>B = 1</td>
<td>3</td>
<td>3394.74</td>
<td>21.92%</td>
<td>0.37%</td>
<td>5.7</td>
<td>24.0</td>
<td>2</td>
<td>3481.47</td>
</tr>
<tr>
<td>B = 2</td>
<td>0</td>
<td>3600.00</td>
<td>13.46%</td>
<td>0.27%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>15.30%</td>
</tr>
<tr>
<td>B = 3</td>
<td>0</td>
<td>3600.00</td>
<td>13.77%</td>
<td>0.18%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>15.24%</td>
</tr>
<tr>
<td>B = 4</td>
<td>0</td>
<td>3600.00</td>
<td>13.49%</td>
<td>0.11%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>15.20%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cut-CCE: PICEF</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
<th>#opt</th>
<th>Time</th>
<th>Total</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 2</td>
<td>B = 1</td>
<td>12</td>
<td>2539.39</td>
<td>6.93%</td>
<td>33.10%</td>
<td>6.1</td>
<td>82.3</td>
<td>6</td>
<td>3254.35</td>
</tr>
<tr>
<td>B = 2</td>
<td>13</td>
<td>2464.68</td>
<td>7.09%</td>
<td>31.69%</td>
<td>6.5</td>
<td>81.8</td>
<td>1</td>
<td>3572.31</td>
<td>16.26%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cut-PICEF: PICEF</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
<th>#opt</th>
<th>Time</th>
<th>Total</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 2</td>
<td>B = 1</td>
<td>26</td>
<td>771.57</td>
<td>7.97%</td>
<td>35.34%</td>
<td>3.3</td>
<td>22.6</td>
<td>26</td>
<td>918.91</td>
</tr>
<tr>
<td>B = 2</td>
<td>22</td>
<td>1284.22</td>
<td>6.97%</td>
<td>43.76%</td>
<td>4.5</td>
<td>50.4</td>
<td>15</td>
<td>2154.89</td>
<td>8.62%</td>
</tr>
<tr>
<td>B = 3</td>
<td>12</td>
<td>2539.39</td>
<td>6.93%</td>
<td>33.10%</td>
<td>6.1</td>
<td>82.3</td>
<td>6</td>
<td>3254.35</td>
<td>11.80%</td>
</tr>
<tr>
<td>B = 4</td>
<td>13</td>
<td>2464.68</td>
<td>7.09%</td>
<td>31.69%</td>
<td>6.5</td>
<td>81.8</td>
<td>1</td>
<td>3572.31</td>
<td>16.26%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cut-CCE: FSE</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
<th>#opt</th>
<th>Time</th>
<th>Total</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 2</td>
<td>B = 1</td>
<td>26</td>
<td>951.26</td>
<td>7.48%</td>
<td>42.90%</td>
<td>2.6</td>
<td>23.2</td>
<td>24</td>
<td>1399.71</td>
</tr>
<tr>
<td>B = 2</td>
<td>22</td>
<td>1770.23</td>
<td>6.38%</td>
<td>44.15%</td>
<td>4.4</td>
<td>50.1</td>
<td>7</td>
<td>2970.19</td>
<td>6.55%</td>
</tr>
<tr>
<td>B = 3</td>
<td>8</td>
<td>2954.78</td>
<td>6.64%</td>
<td>37.47%</td>
<td>6.1</td>
<td>60.9</td>
<td>1</td>
<td>3575.11</td>
<td>6.89%</td>
</tr>
<tr>
<td>B = 4</td>
<td>10</td>
<td>2997.36</td>
<td>6.43%</td>
<td>32.87%</td>
<td>6.8</td>
<td>93.5</td>
<td>0</td>
<td>3600.00</td>
<td>10.26%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cut-PICEF: FSE</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
<th>#opt</th>
<th>Time</th>
<th>Total</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 2</td>
<td>B = 1</td>
<td>2</td>
<td>3202.94</td>
<td>0.35%</td>
<td>0.11%</td>
<td>4.0</td>
<td>14.5</td>
<td>3</td>
<td>3494.44</td>
</tr>
<tr>
<td>B = 2</td>
<td>0</td>
<td>3600.00</td>
<td>0.36%</td>
<td>0.13%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>0.34%</td>
</tr>
<tr>
<td>B = 3</td>
<td>0</td>
<td>3600.00</td>
<td>0.27%</td>
<td>0.07%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>0.29%</td>
</tr>
<tr>
<td>B = 4</td>
<td>0</td>
<td>3600.00</td>
<td>0.29%</td>
<td>0.07%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>0.33%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cut-CCE: FSE</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
<th>#opt</th>
<th>Time</th>
<th>Total</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 3</td>
<td>B = 1</td>
<td>1</td>
<td>3477.97</td>
<td>0.30%</td>
<td>0.09%</td>
<td>4.0</td>
<td>22.0</td>
<td>2</td>
<td>3444.94</td>
</tr>
<tr>
<td>B = 2</td>
<td>0</td>
<td>3600.00</td>
<td>0.21%</td>
<td>0.03%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>0.23%</td>
</tr>
<tr>
<td>B = 3</td>
<td>0</td>
<td>3600.00</td>
<td>0.23%</td>
<td>0.05%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>0.22%</td>
</tr>
<tr>
<td>B = 4</td>
<td>0</td>
<td>3600.00</td>
<td>0.23%</td>
<td>0.05%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>0.24%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cut-PICEF: FSE</th>
<th>#opt</th>
<th>Time</th>
<th>#att</th>
<th>#sub</th>
<th>#opt</th>
<th>Time</th>
<th>Total</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain length: 4</td>
<td>B = 1</td>
<td>1</td>
<td>3570.22</td>
<td>0.26%</td>
<td>0.07%</td>
<td>5.0</td>
<td>32.0</td>
<td>1</td>
<td>3554.88</td>
</tr>
<tr>
<td>B = 2</td>
<td>0</td>
<td>3600.00</td>
<td>0.29%</td>
<td>0.15%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>0.28%</td>
</tr>
<tr>
<td>B = 3</td>
<td>0</td>
<td>3600.00</td>
<td>0.21%</td>
<td>0.04%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>0.18%</td>
</tr>
<tr>
<td>B = 4</td>
<td>0</td>
<td>3600.00</td>
<td>0.28%</td>
<td>0.07%</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>3600.00</td>
<td>0.19%</td>
</tr>
</tbody>
</table>
Biró, Péter, Haase-Kromwijk, Bernadette, Andersson, Tommy, Ásgeirsson, Eyjól-