A Benders-type Approach for Robust Optimization of Kidney Exchanges under Full Recourse

Danny Blom[1], Christopher Hojny[1], and Bart Smeulders[1]

[1]Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands

May 19, 2021

Abstract

The goal of kidney exchange programs is to match recipients with a willing but incompatible donor with another compatible donor, so as to maximize total (weighted) transplants. There is significant uncertainty in this process, as planned transplants may be cancelled for a variety of reasons. Planning exchanges while considering failures, and options for recourse, is therefore crucial. In this paper, we reconsider a robust optimization model with recourse proposed in Carvalho et al. (2020) that takes into account the event that a number of donors leaves the KEP. After these donors have left the program, a new set of exchanges is identified (the recourse decision). Current algorithmic considerations do not allow to find optimal solutions for the robust optimization model for realistic sized kidney exchanges with a reasonable time frame. Following the iterative framework in Carvalho et al. (2020) for solving a large-scale mixed integer programming model, we propose a new variable and constraint generation method based on Generalized Benders' Decomposition. A characterization of this method is given based on two widely used integer programming models for kidney exchange programs. Furthermore, a lifting technique is proposed to obtain stronger Benders cuts to speed up computation. Computational results show that our algorithm is very competitive, improving on the running time of the state-of-the-art method by one order of magnitude. Furthermore, our methods are able to solve a large number of previously unsolved instances within the same time limit.

Keywords: kidney exchange; multilevel optimization; mixed-integer programming; Benders decomposition;

MSC: 90-08

1 Introduction

In the final stage of chronic kidney disease, patients suffer from end-stage renal disease (ESRD). Treatments for this disease consist of dialysis or the more preferred option of a kidney transplant. However, the supply of healthy kidneys from deceased donors does not adequately meet the demand from ESRD patients. An alternative are living donations, 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, see, e.g., 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 (see Roth et al. (2004), Roth et al. (2005), and Roth et al. (2007)). In a KEP, a set of incompatible patient-donor pairs is given, and every donor is willing to give one of its kidneys to any patient as long as their paired recipient receives a transplant in turn from some other donor. In the simplest case, one is thus looking for tuples (P_1, P_2) of such pairs such that the kidney of P_1's donor is compatible with the recipient of P_2 and vice versa. Once such a match is identified, the patients from P_1 and P_2 can receive the kidney from the other pair's donor. We call such a swap a *2-cycle* as it includes two patient-donor pairs. Of course, this idea can be generalized to *k-cycles*, 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 a single cycle are usually performed simultaneously to remove the risk of donors reneging once their paired patient has received a kidney, long cycles are logistically challenging and thus avoided.

Extensions to the basic KEP model include *non-directed donors*, as studied in Morrissey et al. (2005) and Roth et al. (2006). These donors are willing to donate to any recipient, without requiring a return transplant to a paired recipient. Such non-directed donors can start a chain of transplants, in which the donor of the i-th 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 becomes 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 $v \in V$ correspond to recipient-donor pairs and non-directed donors, and an arc $(i, j) \in A$ corresponds to compatibilities between the donor of i and the recipient of j. One intuitive objective in kidney exchange programs is to maximize the number of transplants that can be realized. Figure 1 illustrates an example of a kidney exchange program.

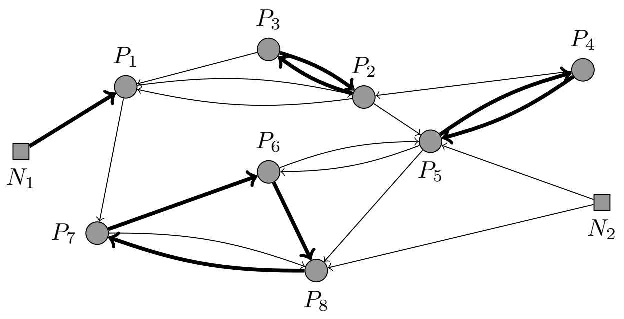


Figure 1: Example of a KEP model. Each vertex corresponds to a recipient-donor pair $(P_1, \ldots, P_8)$ or a non-directed donor (N_1, N_2). Possible transplants are depicted by the bold arcs leading to one chain of length one, two 2-cycles and one 3-cycle, i.e., eight transplants in total.

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) report that 93% of proposed matches fail in the UNOS program. In the NHS Living Kidney Sharing Scheme, 30% of identified matches did not proceed to transplant between 2013–2017 (NHS, 2017). Such failures can occur for a variety of reasons, two examples are the identification of additional medical incompatibilities between matching run and transplant, or recipients and donors may (temporarily) drop out due to health issues. Mathematical models for KEPs incorporating the possibilities of failure of intended transplants have been studied extensively in the literature. Dickerson et al. (2019) first consider arc failures with equal arc existence probabilities and weight for each cycle and chain based on their expected number of transplants rather than only on the maximum achievable number of transplants. Bidkhori et al. (2020) extends this model to a setting with inhomogeneous arc existence probabilities and gives 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, the initially proposed solution can be adapted subject to some constraints, which might allow for a significant percentage of additional transplants to be realized, see e.g. Bray et al. (2015). One possible implementation of recourse is to allow an entirely new set of cycles and chains, only involving pairs and NDDs corresponding to donors that are not affected by failure. Carvalho et al. (2020) combines these concepts in a three-stage robust optimization approach, where the final stage is the recourse decision.

1.1 Our contribution

This paper builds further on the three-stage robust optimization model proposed by Carvalho et al. (2020). In the first stage, an initial set of cycles and chains is proposed for potential transplants. The second stage incorporates the possibility that a limited number of donors will fail pre-transplant such that a part of the intended transplants cannot proceed. The final stage involves a recourse decision. Throughout this paper, we consider the setting of *full recourse*, meaning that all cycles and chains can be considered that are not affected by donors that have failed in the second stage, since it is the least restrictive and most natural implementation of recourse. The aim of the model is to maximize the number of recipients that both (i) are included in the initial set of cycles and chains, and (ii) correspond to a proceeding transplant after recourse. To solve their robust model, Carvalho et al. (2020) proposed an algorithm in which a mixed-integer program (MIP) of exponential size needs to be solved. This problem is tackled by an iterative procedure that solves a restricted version of the full MIP that only takes some possible failure patterns into account. If the obtained solution can be confirmed to be already an optimal solution of the original MIP, the method stops. Otherwise, new patterns are generated, which results in adding further variables and constraints to the restricted MIP which needs to be solved again. They report computational results in which the robust optimization model can be solved to optimality within one hour for realistic instances with up to 50 vertices, i.e., the number of recipient-donor pairs and non-directed donors combined, with $K = 3$ and $L = 2$ and a maximum number $B \in \{1, 2, 3, 4\}$ of donors leaving the program. Nevertheless, the time limit was frequently reached for instances with 100 vertices, where the main bottleneck is the generation of new failure patterns. Since KEPs of these sizes are already operating, and their sizes are still increasing, algorithms need to be devised that are much more efficient in practice. Furthermore, longer chains are of interest since these are believed to lead to a significant benefit in the number of transplants that can be realized. However, setting $L > 2$ will lead to even larger MIP models and thus higher computation times.

For these reasons, we propose an alternative algorithm to generate failure patterns, and thus, to solve the robust optimization problem. Specifically, we suggest to use a Benders-type approach, which is used to solve a MIP for generating new patterns. We discuss this method for two MIP models for KEP, the well-known cycle-chain formulation due to Roth et al. (2007) and the position-indexed cycle edge formulation (PICEF) introduced in Dickerson et al. (2016). Since the time needed to solve the robust model heavily depends on the number of patterns that need to be generated, we also discuss means to generate patterns that are more meaningful for the problem. We show through computational experiments on benchmark instances that our algorithms for the robust optimization problem outperform the current state-of-the-art optimization method of Carvalho et al. (2020) by an order of magnitude. These improvements pave the way for solving larger and more difficult instances.

To provide a basis for our new algorithm, we continue in the following subsection with a short review of the relevant literature. Then, we give a precise mathematical description of the robust optimization model of Carvalho et al. (2020) and review integer programming formulations and algorithms for this model in Section 2. In Section 3, we propose new MIP models for a subproblem of the robust optimization model, which is solved through a new iterative algorithm based on a generalized version of Benders' decomposition. Several enhancements are provided in Section 4 that have the purpose of reducing the number of iterations of our algorithms. We conclude the article with an extensive numerical experiments in Section 5.

1.2 Literature review

The model in Carvalho et al. (2020) that we study in this paper is 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). DAD models often follow a general multilevel structure of a sequential game with two players, the defender and the attacker, and three decisions. First, player one can spend a budget to protect or give additional value to parts of a network, e.g., a subset of vertices or arcs in a graph. Afterwards, the second player acts as an adversary and has a separate budget to disrupt parts of network that are not protected by player one. In the third and final decision, player one aims to maximize some graph parameter on the non-disrupted part of the network. We formalize how the model in Carvalho et al. (2020) fits in this class in Section 2.

Research related to kidney exchange programs is not restricted to KEPs with incompatible recipient-donor pairs and non-directed donors and robust models. There exist kidney exchange programs in practice which also allow *compatible pairs* to participate. Although transplantation within this pair is feasible, the recipient of such a pair might benefit from the existence of donors in the kidney exchange pool forming a better match, see, e.g., Roth et al. (2006) and Gentry et al. (2007). Furthermore, it might happen that also a larger number of recipients from incompatible pairs will now receive a kidney as a result of the presence of compatible pairs. There exist multiple objectives that can indicate the ultimate goal of KEPs. Among those are maximizing the number of transplants, the number of hard-to-match patients, and the number of 2-cycles, but also the quality of transplants. Hierarchical optimization over a variety of objectives is common in fielded exchanges, see Biró et al. (2020) for an overview of established KEPs in Europe and their stated objectives.

In addition, more realistic models for kidney exchange, such as stochastic models and recourse in kidney exchange programs, are extensively studied. Alvelos et al. (2019) propose a stochastic model with *internal recourse*, meaning that once a cycle or chain fails, another cycle or chain can be chosen on the remaining non-failing vertices. Klimentova et al. (2016) extends this notion to *subset recourse*, where a number of small disjoint subsets of vertices are tested for failures, and recourse is allowed within these subsets. An alternative recourse model is proposed in Smeulders et al. (2019), that aims to identify which crossmatch tests should be performed, given a limited testing budget, in order to maximize the expected number of transplants.

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. One readily verifies that the number of variables in this formulation is in the worst-case exponential in $|V|$ and therefore becomes impractical when the number of participating pairs and NDDs increases. A common technique for solving the cycle-chain model is branch-and-price, see Glorie et al. (2014) and Plaut et al. (2016). 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 the LP relaxation is generally weaker than of the CC formulation. Dickerson et al. (2016) introduced the *position-indexed cycle edge formulation (PICEF)*, which is a hybrid variant of different models: equivalent to CC, it uses binary variables for each cycle. But instead of using variables for each individual chain, chains are modeled by variables that express the position of an individual transplant within a chain. If no chains are considered, CC and PICEF thus coincide. Otherwise, the LP relaxation of PICEF is shown to be strictly weaker than that of CC.

2 Problem description

In this article, we consider KEPs consisting of a set P of incompatible recipient-donor pairs and a set N of non-directed donors. Before we formally define the problem we want to solve, we need to introduce some terminology and notation. To represent a KEP, we use a directed graph $G = (V, A)$, the *compatibility graph*, with vertex set $V \coloneqq P \cup N$ and arc set $A \subseteq V \times P$. An arc $(i, j) \in A$ encodes that the donor of vertex $i \in P \cup N$ is compatible with the patient of vertex $j \in P$. Moreover, throughout the article, K and L denote upper bounds on the maximum number of arcs that the KEP allows in cycles and chains, respectively. The cycles and chains that can be used according to the policy of the KEP are denoted by $\mathcal{C}_K$ and $\mathcal{D}_L$, respectively. That is, $\mathcal{C}_K$ denotes the set of directed cycles in G consisting of at most K arcs; $\mathcal{D}_L$ is the set of directed paths in G that start from some NDD and consist of at most L arcs. Furthermore, we denote by $V(c)$ and $A(c)$ the set of vertices and arcs in c, respectively. For a vertex $j \in V$, we now denote by $\mathcal{C}_K^j$ ($\mathcal{D}_L^j$) the sets of cycles (chains) c such that $j \in V(c)$. We further denote by $\mathcal{F}_G$ the set of subsets $S \subseteq \mathcal{C}_K \cup \mathcal{D}_L$ such that the elements of S are pairwise vertex-disjoint, i.e., S forms a cycle and chain packing on G. We will refer to $\mathcal{F}_G$ as the set of *feasible KEP solutions*, since each element of $\mathcal{F}_G$ describes a feasible set of exchanges.

As mentioned earlier, we can describe the model in Carvalho et al. (2020) as a defender-attacker-defender model consisting of three stages. In the first stage, a defender can make a decision x contained in some set $\mathcal{X}$. Then, the attacker makes a decision u contained in some set $\mathcal{U}(x)$ that might depend on the defender's decision x. Finally, the defender is allowed to select another decision y contained in some set $\mathcal{Y}(x, u)$ which might depend on both the attacker's decision u and the defender's initial decision x. Furthermore, in each of the three levels an objective has to be taken into account. We denote these objectives by $f(\cdot, \cdot, \cdot), g(\cdot, \cdot)$, and $h(\cdot)$, respectively. The goal of a defender-attacker-defender problem is to solve the trilevel optimization problem

$$
\begin{array}{lrlrlrl}
\max & f(x,u^*,y^*) & & & & & \\
\text{s.t.} & x \in & \mathcal{X}, & & & & \\
 & u^* \in & \arg\min & g(u,y^*) & & & \\
 & & \text{s.t.} & u \in & \mathcal{U}(x), & & \\
 & & & y^* \in & \arg\max & h(y) & \\
 & & & & \text{s.t.} & y \in & \mathcal{Y}(x,u).
\end{array}
$$

The problem considered by Carvalho et al. (2020), referred to as the *full recourse robust kidney exchange problem*, follows this general defender-attacker-defender scheme. We detail the choices in each of the three stages here.

1. Given the compatibility graph $G = (V, A)$, the defender selects a feasible set of exchanges $x \in \mathcal{X}$, where

$$\mathcal{X} = \mathcal{F}_G \; = \left\{\chi^S \;:\; S \subseteq \mathcal{C}_K \cup \mathcal{D}_L \text{ with } V(c) \cap V(c') = \emptyset \text{ for all distinct } c, c' \in S\right\}.$$

   Here, χ^S denotes the characteristic vector of S. In other words, $\mathcal{X}$ denotes the set of all feasible KEP solutions on G, or the set of *initial KEP solutions*, since $\mathcal{X}$ corresponds to the domain of initial decisions for the defender. For a solution $x \in \mathcal{X}$, we denote by $V(x)$ the set of pairs $j \in P$ for which there exists some cycle or chain c with $j \in V(c)$ such that $x_c = 1$. These pairs correspond to the set of recipients that will receive a transplant according to the initial KEP solution x.

2. The attacker is given a budget $B \in \mathbb{N}$ and can decide to attack up to B vertices in the compatibility graph, corresponding to pairs or non-directed donors leaving the program. The attacker's aim is to disturb the initial KEP solution as much as possible, as well as hinder recourse in the next stage.

   The attacker's domain is defined as

$$\mathcal{U}(x) = \left\{ u \in \{0,1\}^V \;:\; \sum_{j \in V} u_j \leq B \right\}.$$

   We refer to an attacker's decision $u \in \mathcal{U}(x)$ as an *attack*. We encode an attack using a vector of length $|V|$, where $u_j = 1$ means that vertex j is an *attacked vertex* with respect to u, and $u_j = 0$ if j is not attacked. Furthermore, for each attack u, we define

$$V_u \coloneqq \{j \in V \;:\; u_j = 0\},$$

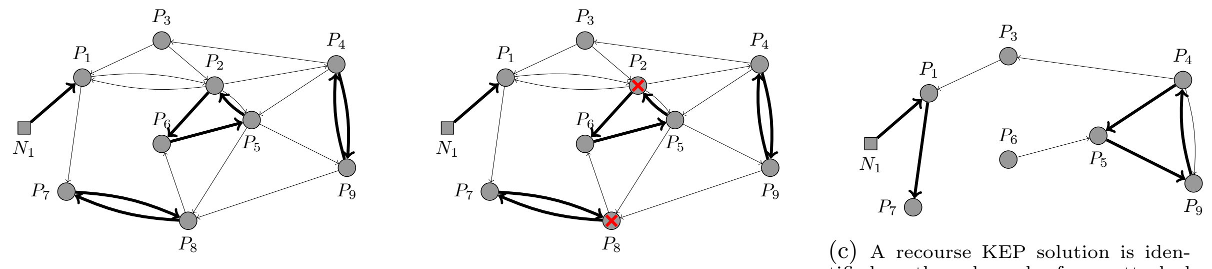
   i.e., the set of vertices that are not attacked by u. Notice that the domain of the attacker does not explicitly depend on the initial KEP solution $x \in \mathcal{X}$ chosen by the defender.

3. The defender finally is given the opportunity to reconsider its initial KEP solution after observing which donors leave the program according to the chosen attack $u \in \mathcal{U}(x)$, in order to (partially) restore the damage inflicted by the attacker upon cycles and chains of the chosen initial KEP solution $x \in \mathcal{X}$. The defender can decide to choose an alternative KEP solution on the vertex-induced subgraph $G[V_u] \coloneqq (V_u, A \; \cap (V_u \times V_u))$.We define the domain of this decision, given an initial KEP solution plan $x \in \mathcal{X}$ and subsequent attack $u \in \mathcal{U}(x)$, by

$$\mathcal{Y}(x,u) \coloneqq \left\{\chi^S \;:\; S \in \mathcal{F}_G \text{ with } V(c) \subseteq V_u \text{ for all } c \in S\right\}, \tag{1}$$

   and we refer to each $y \in \mathcal{Y}(x,u)$ as a *recourse KEP solution*. Using the notation introduced above, $V(y)$ denotes the set of pairs $j \in P$ such that there exists some cycle or chain c with $j \in V(c)$ for which $y_c = 1$. This again corresponds to the set of recipients that will receive a transplant according to the recourse KEP solution $y \in \mathcal{Y}(x,u)$.

The above model still has some degrees of freedom, because its objectives are not fixed. In practice, whenever an initial KEP solution $x \in \mathcal{X}$ is proposed, recipients are notified that they might receive a transplant. It is of course a disappointing experience when recipients will not receive a transplant anymore due to some leaving donor breaking the cycle or chain. Taking this into account, Carvalho et al. (2020) considered the objective of maximizing the worst-case guaranteed number of recipients that receive a transplant according to both the initial KEP solution $x \in \mathcal{X}$ and the recourse solution $y \in \mathcal{Y}(x,u)$ w.r.t.



(a) An initial KEP solution $x \in \mathcal{X}$ is identified (bold arcs).

(b) Budget $B = 2$ is used to attack P_2 and P_8.

(c) A recourse KEP solution is identified on the subgraph of non-attacked vertices (bold arcs).

Figure 2: Three-stage problem applied on a KEP with $|N| = 1$ and $|P| = 9$ with $K = 3, L = 2$, and $B = 2$. The objective value is 5, since five recipients are both in the initial and recourse KEP solution.

all attacks $u \in \mathcal{U}$ that can be realized by the attacker. The three-stage optimization problem described above is illustrated in Figure 2.

Although our definition of $\mathcal{Y}(x, u)$ will generally allow for the best robust solution, the drawback is that for large instances it is computationally challenging to solve this problem to optimality with existing techniques from Carvalho et al. (2020). For this reason, we devise new algorithms for the robust kidney exchange problem with full recourse that are competitive and more scalable for practical applications than the state-of-the-art. To be able to accurately compare our methods with the methods by Carvalho et al. (2020), we provide details on the latter in the next subsection.

2.1 CC-based mixed-integer programming models

In this section, we discuss a mixed-integer programming model of the trilevel problem introduced above, and we provide a summary of the techniques suggested by Carvalho et al. (2020) to solve it. In the next section, we will then develop an alternative method to solve the problem, which turns out to be more efficient computationally. Formally, these method allow us to find an answer to the following decision problem.

Definition 1. *(Full recourse robust kidney exchange problem (FRRKEP))*

Given: *Compatibility graph $G = (V, A)$, defender domains $\mathcal{X}$ and $\mathcal{Y}$, attacker domain $\mathcal{U}$, integer t*
Question: *Does there exist an initial KEP solution $x \in \mathcal{X}$ such that for all attacks $u \in \mathcal{U}(x) \subseteq \mathcal{U}$ there exists a recourse KEP solution $y \in \mathcal{Y}(x, u) \subseteq \mathcal{Y}$ such that $V(x) \cap V(y) \geq t$?*

To be able solve it in practice, Carvalho et al. (2020) exploit that the attacker's domain $\mathcal{U}$ is a finite set, which allows them to model FRRKEP as a MIP. Specifically, they suggest the following MIP model $z_{FR}(\mathcal{U})$ that is based on the cycle-chain formulation, see Roth et al. (2007).

$$
\begin{aligned}
z_{FR}(\mathcal{U}) = \max \quad & Z, && && \text{(2a)}\\
\text{s.t.} \quad & Z - \sum_{j \in P} y_j^u \leq 0, && \forall u \in \mathcal{U}, && \text{(2b)}\\
& y_j^u - \sum_{c \in \mathcal{C}_K^j \cup \mathcal{D}_L^j} x_c \leq 0, && \forall u \in \mathcal{U}, j \in P, && \text{(2c)}\\
& y_j^u - \sum_{c \in \mathcal{C}_K^j \cup \mathcal{D}_L^j} x_c^u \leq 0, && \forall u \in \mathcal{U}, j \in P, && \text{(2d)}\\
& \sum_{c \in \mathcal{C}_K^j \cup \mathcal{D}_L^j} x_c \leq 1, && \forall j \in P \cup N, && \text{(2e)}\\
& \sum_{c \in \mathcal{C}_K^j \cup \mathcal{D}_L^j} x_c^u \leq 1 - u_j, && \forall u \in \mathcal{U}, j \in P \cup N, && \text{(2f)}\\
& y^u \geq 0, && \forall u \in \mathcal{U}, && \text{(2g)}\\
& x, x^u \in \{0,1\}^{\mathcal{C}_K \cup \mathcal{D}_L}, && \forall u \in \mathcal{U}. && \text{(2h)}
\end{aligned}
$$

In this model, variables y_j^u are used for each patient-donor pair $j \in P$ and each attack $u \in \mathcal{U}$, indicating whether or not the recipient of j receives a transplant according to both the initial and the recourse KEP solution under attack u. Furthermore, a binary variable x_c for each cycle and chain $c \in \mathcal{C}_K \cup \mathcal{D}_L$ indicates whether or not c is used in the initial KEP solution, and analogously x_c^u for the recourse KEP solution. Constraints (2b) are used to make sure that the value Z accurately represents the largest guaranteed overlap in the initial and recourse KEP solutions over all attacks. Constraints (2c) and (2d) ensure that $y_j^u = 0$ if the initial or recourse KEP solution under u do not involve some cycle or chain c with $j \in V(c)$. Notice that in each *optimal* solution to $z_{FR}(\mathcal{U})$, we have $y_j^u = 1$ otherwise, since we want to maximize the value of each variable y_j^u while preserving feasibility. Since in any optimal solution, the value of the variables y_j^u is either 0 or 1, it is not necessary to impose integrality constraints on these variables. Furthermore, Constraints (2e) and (2f) impose that both the initial and recourse KEP solution induce vertex-disjoint exchanges, where the latter also guarantees that no cycle or chain containing an attacked node is selected in the recourse solution.

2.1.1 Iterative solution approach

Since this MIP formulation requires many variables and constraints per attack $u \in \mathcal{U}$, it is computationally intractable to solve the full model (2) by a black-box MIP solver. Instead, Carvalho et al. (2020) proposed an iterative solution approach for this MIP. They start with restricting the attacker's domain to a small range $\bar{\mathcal{U}} \subseteq \mathcal{U}$ of attacks. If they can show that the solution of $z_{FR}(\bar{\mathcal{U}})$ is already optimal w.r.t. the set of all attacks $\mathcal{U}$, their method stops. Otherwise, a new attack in $\mathcal{U} \setminus \bar{\mathcal{U}}$ is added to $\mathcal{U}$ and the previous steps are repeated. The main component of their method is thus the mechanism to generate a new attack, $u \in \mathcal{U} \setminus \bar{\mathcal{U}}$ such that, given an initial solution $\bar{x} \in \mathcal{X}$, there exists no recourse solution $x^u \in \mathcal{Y}(x, u)$ with $V(\bar{x}) \cap V(x^u) \geq z_{FR}(\bar{\mathcal{U}})$. This mechanism uses a subroutine based on the problem of finding the best recourse solution x^u given an initial solution $\bar{x} \in \mathcal{X}$ and an attack $u \in \mathcal{U}(\bar{x})$. It can be modeled as the following IP:

$$R(\bar{x}, u) = \max \sum_{c \in \mathcal{C}_K \cup \mathcal{D}_L} w_c(\bar{x}) x_c^u, \tag{3a}$$

$$\sum_{c \in \mathcal{C}_K^j \cup \mathcal{D}_L^j} x_c^u \leq 1 - u_j, \qquad \forall j \in P \cup N, \tag{3b}$$

$$x^u \in \{0,1\}^{\mathcal{C}_K \cup \mathcal{D}_L}, \tag{3c}$$

where the objective coefficients $w_c(\bar{x})$ are defined as

$$w_c(\bar{x}) = \sum_{j \in V(c) \cap P} \sum_{c' \in \mathcal{C}_K^j \cup \mathcal{D}_L^j} \bar{x}_{c'}, \tag{4}$$

i.e., the number of recipient-donor pairs $j \in V(c) \cap P$ that were involved in some cycle or chain c' chosen in the initial KEP solution. In Model (3), the objective is equivalent to maximizing the total sum of weights of cycles and chains chosen in a feasible recourse KEP solution on $G[V_u]$, i.e., the number of transplants to patients that were also planned to receive a transplant in the initial solution $\bar{x}$. Constraints (3b) impose that the recourse KEP solution x^u is vertex-disjoint. This means that $R(\bar{x}, u)$ is equivalent to solving a maximum weight kidney exchange program. In other words, $R(\bar{x}, u)$ computes the optimal recourse decision given an initial KEP solution $\bar{x} \in \mathcal{X}$ and attack $u \in \mathcal{U}(\bar{x})$. A flowchart of this method is given in Figure 3.

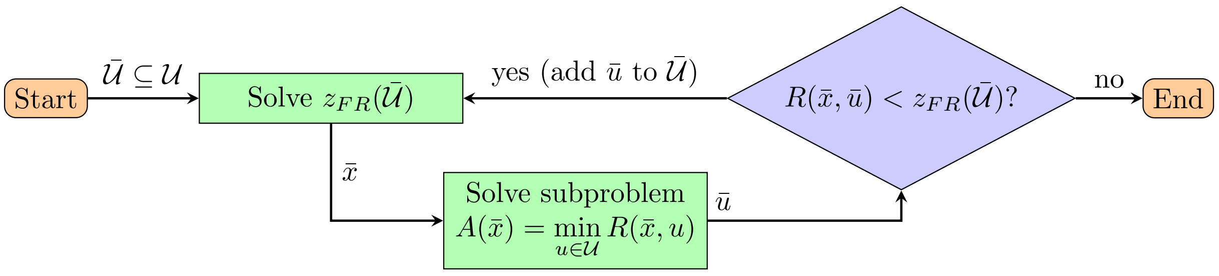

Start
$\bar{\mathcal{U}} \subseteq \mathcal{U}$
Solve $z_{FR}(\bar{\mathcal{U}})$
yes (add $\bar{u}$ to $\bar{\mathcal{U}}$)
$R(\bar{x}, \bar{u}) < z_{FR}(\bar{\mathcal{U}})$?
no
End
$\bar{x}$
Solve subproblem $A(\bar{x}) = \min_{u \in \mathcal{U}} R(\bar{x}, u)$
$\bar{u}$

Figure 3: Flowchart of the iterative method in Carvalho et al. (2020) for the CC-based MIP formulations.

Initially, a small set $\bar{\mathcal{U}} \subseteq \mathcal{U}$ of attacks is selected and $z_{FR}(\bar{\mathcal{U}})$ is computed. This restricted robust model yields an optimal initial solution $\bar{x} \in \mathcal{X}$. In each iteration of the algorithm, we solve the problem

$$A(\bar{x}) = \min_{u \in \mathcal{U}} R(\bar{x}, u). \tag{5}$$

One can then decide based on the value of $A(\bar{x})$ whether there exists an attack $u^* \in \mathcal{U} \setminus \bar{\mathcal{U}}$ such that

$$R(\bar{x}, u^*) < z_{FR}(\bar{\mathcal{U}}). \tag{6}$$

Although it is suggested that we should find the attack $u \in \mathcal{U}(\bar{x})$ with the *smallest* objective for $R(\bar{x}, u)$, it actually suffices to either (i) show there exists no attack satisfying Inequality (6) or (ii) to find an attack satisfying Inequality (6). In the following, we will use that $A(\bar{x})$ is the problem of deciding if case (i) or case (ii) holds and refer to it as the *attack generation problem.* If case (i) holds, we can conclude that $z_{FR}(\bar{\mathcal{U}}) = z_{FR}(\mathcal{U})$, so the problem is solved to optimality. Otherwise, an attack u^* satisfying Inequality (6) is identified and added to $\bar{\mathcal{U}}$. This means that the corresponding variables and constraints are added to $z_{FR}(\bar{\mathcal{U}})$ and the method iterates. Notice that this method terminates in finite time, since only a finite number of distinct attacks $u \in \mathcal{U}$ can be generated.

2.1.2 Solving the attack generation problem

We now focus on the subroutine of the iterative method that solves the attack generation problem $A(\bar{x})$. Carvalho et al. (2020) propose a branch-and-bound type algorithm to solve $A(\bar{x})$. A branch-and-bound tree is created and at every non-leaf node, a vertex $j \in V$ is chosen to be attacked or not attacked. This decision is incorporated in the tree by creating two child nodes that fix u_j to 1 and 0, respectively. Consequently, at each node t of the tree, a subset of the binary u-variables has already been fixed, which is used to derive bounds on $z_{FR}(\mathcal{U})$. Lower bounds are obtained through extending the u-variables fixed to 1 to an attack u^t of B vertices and counting the number of vertices in the cycles/chains not affected by the attack. This is indeed a lower bound on $z_{FR}(\mathcal{U})$, because the attack removes cycles/chains from $\bar{x}$ and we do not compute an alternative recourse solution. Upper bounds are computed through solving $R(\bar{x}, u^t)$, which is valid since u^t is a feasible attack, i.e., its objective is always at least as high as the value of $A(x)$. Notice that the model can terminate early whenever we compute an upper bound that is smaller than $z_{FR}(\mathcal{U})$, since the corresponding attack u^t then satisfies Inequality (6).

2.2 PICEF-based formulations

The solution approach described above can also be adapted to fit the definition for other integer programming formulations for the kidney exchange problem. Carvalho et al. (2020) proposed an alternative formulation based on the position indexed cycle edge formulation (PICEF) introduced by Dickerson et al. (2016). Instead of enumerating all chains $d \in \mathcal{D}_L$, it considers binary variables $\xi_{ij\ell} \in \{0, 1\}$ for each $(i, j) \in A$ and each position index $\ell \in \mathcal{L}(i, j) \subseteq \mathcal{L} = \{1, \ldots, L\}$ that describes the position of the arc in a chain. Note that for each arc $(i, j) \in A$, a position index $\ell \in \mathcal{L}(i, j)$ exists if and only if there exists a path of length $\ell - 1$ from an NDD to i that does not traverse j. Since chains $d \in \mathcal{D}_L$ start from a non-directed donor $i \in N$, this means that arcs $(i, j) \in A$ with $i \in N$ have $\mathcal{L}(i, j) = \{1\}$. Furthermore, we define

$$A_\ell = \{(i, j) \in A \ : \ \ell \in \mathcal{L}(i, j)\},$$

i.e., the set of arcs that can have position index ℓ. PICEF is in fact a hybrid formulation for the basic KEP model in the sense that there still exist binary variables x_c for each cycle $c \in \mathcal{C}_K$, but the chain variables are replaced by position indexed arc variables $\xi_{ij\ell}$ for each arc (i, j) and position $\ell \in \mathcal{L}(i, j)$. The full recourse robust kidney exchange problem $z_{FR}(\mathcal{U})$ can be formulated with PICEF as

$$
\begin{aligned}
z_{FR}(\mathcal{U}) = \max \quad & Z, && && \text{(7a)}\\
\text{s.t.} \quad & Z - \sum_{j \in P} y_j^u \leq 0, && \forall u \in \mathcal{U}, && \text{(7b)}\\
& y_j^u - \sum_{c \in \mathcal{C}_K^j} x_c - \sum_{\ell \in \mathcal{L}} \sum_{(i,j) \in A_\ell} \xi_{ij\ell} \leq 0, && \forall u \in \mathcal{U}, j \in P, && \text{(7c)}\\
& y_j^u - \sum_{c \in \mathcal{C}_K^j} x_c^u - \sum_{\ell \in \mathcal{L}} \sum_{(i,j) \in A_\ell} \xi_{ij\ell}^u \leq 0, && \forall u \in \mathcal{U}, j \in P, && \text{(7d)}\\
& \sum_{c \in \mathcal{C}_K^j} x_c + \sum_{\ell \in \mathcal{L}} \sum_{(i,j) \in A_\ell} \xi_{ij\ell} \leq 1, && \forall j \in P, && \text{(7e)}\\
& \sum_{i:(j,i) \in A} \xi_{ji1} \leq 1, && \forall j \in N, && \text{(7f)}\\
& \sum_{c \in \mathcal{C}_K^j} x_c^u + \sum_{\ell \in \mathcal{L}} \sum_{(i,j) \in A_\ell} \xi_{ij\ell}^u \leq 1 - u_j, && \forall u \in \mathcal{U}, j \in P, && \text{(7g)}\\
& \sum_{i:(j,i) \in A} \xi_{ji1}^u \leq 1 - u_j, && \forall u \in \mathcal{U}, j \in N, && \text{(7h)}\\
& \sum_{i:(j,i) \in A_\ell} \xi_{ij\ell} - \sum_{i:(i,j) \in A_{(\ell-1)}} \xi_{ji(\ell-1)} \leq 0, && j \in P, \ell \in \mathcal{L} \setminus \{1\}, && \text{(7i)}\\
& \sum_{i:(j,i) \in A_\ell} \xi_{ij\ell}^u - \sum_{i:(i,j) \in A_{(\ell-1)}} \xi_{ji(\ell-1)}^u \leq 0, && \forall u \in \mathcal{U}, j \in P, \ell \in \mathcal{L} \setminus \{1\}, && \text{(7j)}\\
& y^u \geq 0, && \forall u \in \mathcal{U}, && \text{(7k)}\\
& x_c, x_c^u \in \{0,1\}, && \forall c \in \mathcal{C}_K, u \in \mathcal{U}, && \text{(7l)}\\
& \xi_{ij\ell}, \xi_{ij\ell}^u \in \{0,1\}, && \forall (i,j) \in A, \ell \in \mathcal{L}(i,j), \forall u \in \mathcal{U}. && \text{(7m)}
\end{aligned}
$$

The formulation (7) can be explained as follows. Again, Constraints (7b) are used to model the minimum guaranteed number of patients that receive a transplant according to both the initial and recourse KEP solution. Constraints (7c) and (7d) again ensure that $y_j^u = 0$ whenever a recipient of pair $j \in P$ either does not receive a transplant in the initial or the recourse KEP solution with respect to an attack $u \in \mathcal{U}$. The structure of the MIP again allows us to pick the value of y_j^u as large as possible while preserving feasibility, so $y_j^u = 1$ whenever both the initial solution and recourse solution under u involve a transplant for recipient j. The PICE formulation makes a distinction between patient-donor pairs and non-directed donors when it comes to imposing vertex-disjointedness of KEP solutions. For each patient-donor pair $j \in P$, Constraints (7e) and (7g) imply that we can either choose a cycle containing j or a position indexed arc going into j, for the initial and recourse KEP solutions respectively. For each non-directed donor $j \in N$, Constraints (7f) and (7h) ensure that there is at most one position indexed arc going out of j selected. Constraints (7g) and (7h) also ensure that recourse KEP solutions under u do not involve attacked vertices. Finally, Constraints (7i) and (7j) ensure that an arc (i,j) at position $\ell \geq 2$ can only be chosen whenever a preceding arc going into i is picked in position $\ell - 1$.

Furthermore, given an initial KEP solution $(\bar{x}, \bar{\xi})$, the weight of an arc $(i,j) \in A$ is given by

$$
w_{ij}(\bar{x}, \bar{\xi}) := \sum_{c \in \mathcal{C}_K^j} \bar{x}_c + \sum_{\ell \in \mathcal{L}} \sum_{i:(i,j) \in A_\ell} \bar{\xi}_{ij\ell}, \tag{8}
$$

i.e., $w_{ij}(\bar{x}, \bar{\xi}) = 1$ if patient j receives a transplant according to the initial KEP solution $(\bar{x}, \bar{\xi})$, and 0 otherwise. Similarly, the weight of a cycle is defined as

$$
w_c(\bar{x}, \bar{\xi}) := \sum_{j \in V(c)} \left(\sum_{c' \in \mathcal{C}_K^j} \bar{x}_{c'} + \sum_{\ell \in \mathcal{L}} \sum_{i:(i,j) \in A_\ell} \bar{\xi}_{ij\ell} \right), \tag{9}
$$

i.e., the number of patients of cycle $c \in \mathcal{C}_K$ involved in the initial KEP solution $(\bar{x}, \bar{\xi})$.

The problem $R(\bar{x}, \bar{\xi}, u)$ of finding the best recourse KEP solution given an initial solution $(\bar{x}, \bar{\xi})$ and attack u can then be formulated as

$$
\begin{aligned}
R(\bar{x}, \bar{\xi}, u) = \max \ & \sum_{c \in \mathcal{C}_K} w_c(\bar{x}, \bar{\xi}) x_c^u + \sum_{\ell \in \mathcal{L}} \sum_{(i,j) \in A_\ell} w_{ij}(\bar{x}, \bar{\xi}) \xi_{ij\ell}^u, && && \text{(10a)} \\
\text{s.t.} \quad & \sum_{c \in \mathcal{C}_K^j} x_c^u + \sum_{\substack{i:(i,j) \in A \\ \ell \in \mathcal{L}(i,j)}} \xi_{ij\ell}^u \leq 1 - u_j, && \forall j \in P, && \text{(10b)} \\
& \sum_{i:(j,i) \in A} \xi_{ji1}^u \leq 1 - u_j, && \forall j \in N, && \text{(10c)} \\
& \sum_{i:(j,i) \in A_\ell} \xi_{ij\ell}^u - \sum_{i:(i,j) \in A_{(\ell-1)}} \xi_{ji(\ell-1)}^u \leq 0, && \forall j \in P, \ell \in \mathcal{L} \setminus \{1\}, && \text{(10d)} \\
& x_c^u \in \{0,1\}, && \forall c \in \mathcal{C}_K, && \text{(10e)} \\
& \xi_{ij\ell}^u \in \{0,1\}, && \forall \ell \in L, (i,j) \in A_\ell. && \text{(10f)}
\end{aligned}
$$

It is then straightforward to adapt the iterative approach due to Carvalho et al. (2020) to these PICEF-based models, as can be seen in Figure 4.

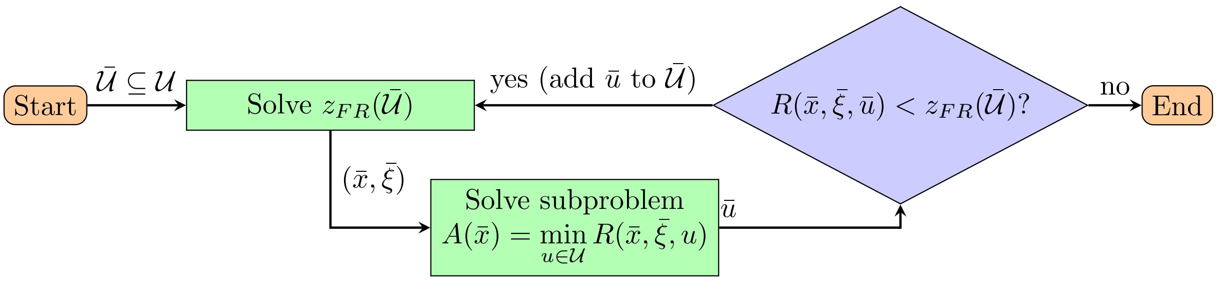

Start
$\bar{\mathcal{U}} \subseteq \mathcal{U}$
Solve $z_{FR}(\bar{\mathcal{U}})$
yes (add $\bar{u}$ to $\bar{\mathcal{U}}$)
$R(\bar{x}, \bar{\xi}, \bar{u}) < z_{FR}(\bar{\mathcal{U}})$?
no
End
$(\bar{x}, \bar{\xi})$
Solve subproblem $A(\bar{x}) = \min_{u \in \mathcal{U}} R(\bar{x}, \bar{\xi}, u)$
$\bar{u}$

Figure 4: Flowchart of the iterative method in Carvalho et al. (2020) for the PICEF-based MIP formulations.

3 Benders-type approaches to the full recourse robust kidney exchange problem

In this section, we describe a new algorithmic approach to solve the full recourse robust kidney exchange problem. For our new approach, we still make use of the basic framework depicted in Figure 3. Our main contribution is an alternative algorithm for generating new attacks. It is based on a reformulation of the attack generation problem $A(\bar{x})$ given an initial KEP solution $\bar{x} \in \mathcal{X}$. We investigate two variants of these IPs, based on the cycle-chain formulation (CC) and the position-indexed cycle edge formulation (PICEF) respectively.

3.1 Generalized Benders decomposition algorithm

We first provide a single-level MIP reformulation of the attack generation problem $A(\bar{x})$ using the cycle-chain formulation for kidney exchange programs. This new formulation is based on the set of all feasible KEP solutions $\mathcal{F}_G$. We define the *remainder* of a feasible KEP solution $S \subseteq \mathcal{F}_G$ with respect to attack $u \in \mathcal{U}$ as

$$\rho(S, u) := \{c \in S \ : \ V(c) \subseteq V_u\},$$

i.e., the set of cycles and chains in S not affected by the attack u. Note that the problem $R(x, u)$ of finding the best recourse solution x^u given an initial KEP solution $x \in \mathcal{X}$ and attack $u \in \mathcal{U}(x)$ is equivalent to solving

$$\max_{S \in \mathcal{F}_G} \sum_{c \in \rho(S,u)} w_c(\bar{x}),$$

i.e., to maximize the total weight of the remainder over all feasible KEP solutions $S \in \mathcal{F}_G$. Using this observation, we can reformulate $A(\bar{x})$ as a single MIP $z(\mathcal{F}_G, \bar{x})$ based on the set of feasible KEP solutions $\mathcal{F}_G$:

$$
\begin{aligned}
z(\mathcal{F}_G, \bar{x}) = \min \quad & Z && && \text{(11a)}\\
\text{s.t.} \quad & Z \geq \sum_{c \in S} w_c(\bar{x}) x_c, && \forall S \in \mathcal{F}_G, && \text{(11b)}\\
& x_c \geq 1 - \sum_{j \in V(c)} u_j, && \forall c \in \mathcal{C}_K \cup \mathcal{D}_L, && \text{(11c)}\\
& x \geq 0, && && \text{(11d)}\\
& u \in \mathcal{U}. && && \text{(11e)}
\end{aligned}
$$

We now show that this is indeed a valid reformulation for $A(\bar{x})$.

Theorem 2. *The integer program (11) is a formulation for the attack generation problem $A(\bar{x})$ given by (5).*

Proof. For a given feasible attack $u \in \mathcal{U}$, it is imposed through Constraints (11c) that $x_c \geq 1$ for a cycle or chain $c \in \mathcal{C}_K \cup \mathcal{D}_L$ whenever none of the vertices $j \in V(c)$ is attacked by u, and $x_c \geq 0$ otherwise. Hence, there exists an optimal solution that is binary: given an optimal solution x, we can reduce x_c to 1 (or 0) if none of c's nodes is attacked (there is an attacked node in c) without affecting the objective value of x since the weights $w_c(\bar{x})$ are non-negative. Moreover, each constraint from (11b) corresponds to a feasible KEP solution $S \in \mathcal{F}_G$, and the right-hand side consists of the variables and weights of cycles and chains in S. Therefore, for every attack $u \in \mathcal{U}$, the objective value is equal to the maximum of the total weight of the remainders $\rho(S, u)$ over all feasible KEP solutions $S \in \mathcal{F}_G$, or in other words, the value of $R(\bar{x}, u)$. Since we consider a minimization problem, we search for the attack $u \in \mathcal{U}$ such that this maximum weight is as small as possible. This shows the equivalence of this formulation with the attack generation problem $A(\bar{x})$. $\square$

Note that because of Constraints (11b), the number of constraints in Model (3) might be exponential in $|V|$. To reduce the number of constraints, observe that Constraint (11b) for a KEP solution $S \in \mathcal{F}_G$ is redundant if there exists another KEP solution $S' \in \mathcal{F}_G$ with $S \subset S'$. For this reason, it is sufficient to only include (11b) for KEP solutions $S \in \mathcal{F}_G$ that are (inclusionwise) *maximal*. Since the number of maximal KEP solutions can still be exponential in $|V|$, we use a row generation technique that allows us to solve (11) iteratively.

The idea of this algorithm is to initially select a restricted set $\mathcal{F}'_G \subseteq \mathcal{F}_G$ of feasible KEP solutions and to solve $z(\mathcal{F}'_G, \bar{x})$. This restricted model gives us an optimal attack $u' \in \mathcal{U}$ and objective value Z'. To check whether an inequality of type (11b) is violated, we solve the problem $R(\bar{x}, u')$ of finding a best recourse solution given an initial solution $\bar{x}$ and attack u'. Suppose that x' is such an optimal recourse solution. If $R(\bar{x}, u')$ is at most Z', then we have found an optimal solution to (11). Otherwise, we add the constraint associated with x' to $\mathcal{F}'_G$ and iterate until no violated constraint can be found anymore. Although $R(\bar{x}, u)$ is equivalent to solving a weighted kidney exchange problem and thus NP-hard in general due to Abraham et al. (2007), it can still be solved efficiently in practice since many MIP formulations for KEPs, such as the cycle-chain formulation and PICEF, have a rather tight LP relaxation.

We call this a Benders-type approach since we can split the problem into a master and subproblem. Here, the problem $z(\mathcal{F}'_G, \bar{x})$ serves as the master problem. Furthermore, given an optimal attack $u' \in \mathcal{U}$ with respect to this master problem, we solve the subproblem $R(\bar{x}, u')$ to identify the best recourse solution given initial solution $\bar{x}$ and attack u'. The optimal solutions of the subproblems in turn each provide a constraint from (11b), which are the Benders cuts based on feasible KEP solutions for the master problem. Algorithm 1 provides the exact description of our method.

In Algorithm 1, we initialize the restricted set of feasible KEP solutions $\mathcal{F}'_G = \{\bar{x}\}$. Next, we solve $z(\mathcal{F}'_G, \bar{x})$ and check two conditions. On the one hand, if $z(\mathcal{F}'_G, \bar{x}) \geq z_{FR}(\bar{\mathcal{U}})$, then for any attack u there exists a recourse KEP solution on $G[V_u]$ with weight at least $z_{FR}(\bar{\mathcal{U}})$. Hence there cannot exist an attack $u \in \mathcal{U}$ such that

$$R(\bar{x}, u) \geq \min_{\bar{u} \in \bar{\mathcal{U}}} R(\bar{x}, \bar{u}) = z_{FR}(\bar{\mathcal{U}}), \quad \forall u \in \mathcal{U}. \tag{12}$$

We conclude that $z_{FR}(\bar{\mathcal{U}}) = z_{FR}(\mathcal{U})$, i.e., the full recourse robust kidney exchange problem is solved to optimality. This condition is checked in Line 4, and we break out of the loop whenever this occurs. On the other hand, there might exist an attack $u^* \in \mathcal{U} \setminus \bar{\mathcal{U}}$ such that (12) is violated. Then, whenever we solve the KEP on $G[V_{u^*}]$, the if-statement in Line 6 is triggered, as $z_{FR}(\bar{\mathcal{U}}) = \min_{\bar{u} \in \bar{\mathcal{U}}} R(\bar{x}, \bar{u})$. We return this attack as output of the iterative algorithm and add it to $\bar{\mathcal{U}}$. Furthermore, we add both the

Algorithm 1: Benders-type algorithm for solving the attack generation problem based on the cycle-chain formulation

```
Data: graph $G = (V, A)$, set $\bar{\mathcal{U}} \subseteq \mathcal{U}$ of attacks, optimal initial KEP solution $\bar{x} \in \mathcal{X}$ for $z_{FR}(\bar{\mathcal{U}})$
Result: attack $u^* \in \mathcal{U} \setminus \bar{\mathcal{U}}$ with $R(\bar{x}, u^*) < R(\bar{x}, \bar{u})$ for all $\bar{u} \in \bar{\mathcal{U}}$, or show nonexistence
1 $\mathcal{F}'_G = \{\bar{x}\}$;
2 while true do
3     $u^* \in \arg\min_{u \in \mathcal{U}} z(\mathcal{F}'_G, \bar{x})$;
4     if $z(\mathcal{F}'_G, \bar{x}) \geq z_{FR}(\bar{\mathcal{U}})$ then
5         return null;
6     if $R(\bar{x}, u^*) < z_{FR}(\bar{\mathcal{U}})$ then
7         return $u^*$;
8     $x(u^*) \in \arg\max_{x^{u^*}} R(\bar{x}, u^*)$;
9     $\mathcal{F}'_G \leftarrow \mathcal{F}'_G \cup \{x(u^*)\}$;
```

sets of Variables (2g)–(2h) and Constraints (2c)–(2e) corresponding to u^* to obtain $z_{FR}(\bar{\mathcal{U}} \cup \{u^*\})$. If both conditions do not hold, we consider a recourse KEP solution $x(u^*)$ that is optimal on $G[V_{u^*}]$, add it to the restricted set of feasible maximal KEP solutions $\mathcal{F}'_G$, and iterate. We will now show that this algorithm terminates in finite time and gives a correct decision about whether or not an attack exists that violates the optimal solution with respect to the restricted robust problem $z_{FR}(\bar{\mathcal{U}})$.

Theorem 3. *For any input $\bar{\mathcal{U}}$ and $\bar{x}$, Algorithm 1 terminates after finitely many iterations. At termination, it either provides an attack $u^* \in \mathcal{U} \setminus \bar{\mathcal{U}}$ with $R(\bar{x}, u^*) < R(\bar{x}, \bar{u})$ for all $\bar{u} \in \bar{\mathcal{U}}$ or returns "null" if no such u^* exists.*

Proof. We start by observing that the domain of attacks $\mathcal{U}$ is finite as it is a subset of finite-dimensional binary vectors. Furthermore, one can check that—whenever we iteratively solve $z(\mathcal{F}'_G, \bar{x})$ and add attacks to $\bar{\mathcal{U}}$—each attack $u \in \mathcal{U}$ minimizes $z(\mathcal{F}'_G, \bar{x})$ at most twice. To see this, suppose there is an attack $u^* \in \mathcal{U}$ minimizing $z(\mathcal{F}'_G, \bar{x})$ for the second time during the execution of the algorithm. Then, during the first iteration in which this occurs, we get both $z(\mathcal{F}'_G, \bar{x}) < z_{FR}(\bar{\mathcal{U}})$ and $R(\bar{x}, u^*) \geq z_{FR}(\bar{\mathcal{U}})$. Some optimal recourse KEP solution $x(u^*)$ corresponding to u^* is then added to $\mathcal{F}'_G$. Notice now that if u^* becomes the minimizer of $z(\mathcal{F}'_G, \bar{x})$ for a second time, we get that $z(\mathcal{F}'_G, \bar{x}) \geq R(\bar{x}, u^*) \geq z_{FR}(\bar{\mathcal{U}})$ as some best recourse KEP solution $x(u^*)$ with respect to u^* has already been added to $\mathcal{F}'_G$. Therefore, the algorithm terminates since we satisfy the condition in Line 4 after a finite number of iterations. Since $z(\mathcal{F}_G, \bar{x})$ is a valid formulation for the attack generation problem, we will return no attack if and only if there really exists no attack $u^* \in \mathcal{U}$ for which the best recourse solution has smaller weight than for all $\bar{u} \in \bar{\mathcal{U}}$, hence the algorithm decides correctly on the attack generation problem. □

3.2 PICEF-based implementation

We consider also a reformulation for the attack generation problem based on PICEF due to Dickerson et al. (2016). The advantage of using PICEF compared to the cycle-chain formulation is that instead of an exponential number of chain variables, PICEF considers position indexed arcs, of which there exist polynomially many. PICEF is preferred especially for kidney exchange programs involving many non-directed donors and in which the maximum chain length is large (or even unbounded). In our application, using a PICEF-based formulation introduces some complications. In the cycle-chain formulation of this problem, we require only a single variable for each possible cycle or chain. Indeed, if a vertex in a cycle or chain is attacked, this cycle or chain is unavailable for every KEP solution. In a PICEF based formulation, arcs may not only become unavailable by attacks on their origin or destination, but also by attacks on vertices earlier in the chain. Whether an arc may be used thus depends on the value of some other arc variables, which might vary depending on the KEP solution. Figure 5 demonstrates that choosing the value of a position indexed arc variable is not unambiguous.

Example 4. Consider the compatibility graph depicted in Figure 5 consisting of four vertices: the two NDDs N_1 and N_2, and the two recipient-donor pairs P_1 and P_2. Furthermore, two feasible KEP solutions are indicated with red arcs and dashed blue arcs respectively. Notice that both solutions use the arc $(P_1, P_2) \in A$ as the second arc of a chain, i.e. at position $\ell = 2$. Whenever the attacker performs an attack on vertex N_2, it completely breaks the blue chain, meaning $\xi_{P_1,P_2,2}$ should have value 0. However, the red chain completely remains, meaning $\xi_{P_1,P_2,2}$ should have value 1. Therefore, there is an ambiguity in how to choose the value for this position indexed arc variable.

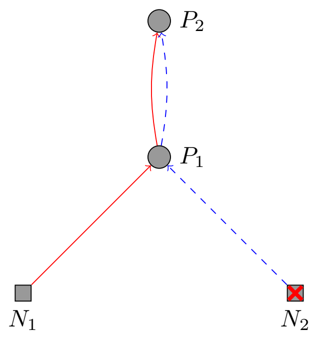


Figure 5: Difficulties arise when using PICEF for the reformulation for the Benders approach.

When we want to derive a PICEF implementation for the attack generation problem $A(\bar{x}, \bar{\xi})$ in a similar fashion as in Model (11), we need to address this issue. We can overcome this problem by adding an extra index to the arc variables indicating the feasible KEP solution $S \in \mathcal{F}_G$ it corresponds to. This way, there cannot exist any confusion on the value of an arc variable given an attack $u \in \mathcal{U}$. Let us now denote by A^S_ℓ the set of arcs with position index $\ell \in \mathcal{L}$ used in a feasible KEP solution $S \in \mathcal{F}_G$.

The corresponding reformulation $z(\mathcal{F}_G, \bar{x}, \bar{\xi})$ of the attack generation problem $A(\bar{x}, \bar{\xi})$ can then be given by

$$
\begin{aligned}
z(\mathcal{F}_G, \bar{x}, \bar{\xi}) = \min \quad & Z && && \text{(13a)}\\
\text{s.t.} \quad & Z \geq \sum_{c \in \mathcal{C}_K(S)} w_c(\bar{x}, \bar{\xi}) x_c + \sum_{\ell \in \mathcal{L}} \sum_{(i,j) \in A^S_\ell} w_{ij}(\bar{x}, \bar{\xi}) \xi^S_{ij\ell} && S \in \mathcal{F}_G, && \text{(13b)}\\
& x_c \geq 1 - \sum_{j \in V(c)} u_j && c \in \mathcal{C}_K, && \text{(13c)}\\
& \xi^S_{ij1} \geq 1 - u_i - u_j && S \in \mathcal{F}_G, (i,j) \in A^S_1, && \text{(13d)}\\
& \xi^S_{ij\ell} \geq \xi^S_{ki(\ell-1)} - u_j && (S, \ell, k) \in \mathfrak{T}, && \text{(13e)}\\
& x \geq 0, && && \text{(13f)}\\
& \xi^S \geq 0, && S \in \mathcal{F}_G, && \text{(13g)}\\
& u \in \mathcal{U}, && && \text{(13h)}
\end{aligned}
$$

where $\mathfrak{T} := \{(S, \ell, k) \in \mathcal{F} \times (\mathcal{L} \setminus \{1\}) \times V \ : \ (i,j) \in A^S_\ell, \ (k,i) \in A^S_{\ell-1}\}$.

We briefly go over this model to explain the differences. Again, consider some attack $u \in \mathcal{U}$. Then, Constraints (13b) consider all feasible KEP solutions and the value on the right-hand side again corresponds to the weight of all cycles and chains that are not attacked by u. Constraints (13c) impose as before that we set $x_c = 1$ if none of the vertices of cycle $c \in \mathcal{C}_K$ is attacked by u. We then make a distinction between arcs going out of a non-directed donor and arcs going out of patient-donor pairs. In the first case, Constraints (13d) ensure that an arc (i,j) at position 1 of a chain remains if neither of i and j is attacked by u, since the first arc in a chain is only affected by attacks on these two vertices. In the latter case, Constraints (13e) impose that an arc (i,j) occurring at position $\ell \geq 2$ in a chain of KEP solution S remains in a solution if both its preceding arc (k,i) at position $\ell - 1$ remains and j is not attacked by u.

Using these adaptations for the MIP formulations of the three levels of our trilevel problem, we can run Algorithm 2 to solve the attack generation problem $A(\bar{x}, \bar{\xi})$. The only difference here compared to Algorithm 1 is the notation for both the initial and recourse KEP solutions.

Remark 5. Whenever we consider a feasible KEP solution $S \in \mathcal{F}_G$, the inequalities corresponding to feasible KEP solutions are stronger for PICEF in general. In the cycle-chain formulation (11), whenever a chain $d \in \mathcal{D}_L$ has at least one attacked vertex, the entire chain breaks down due to Constraint (11c). In that case, this chain d has zero contribution to the right-hand side of Constraint (11b) associated to S. However, for PICEF, with the same attack, the chain up to the first attacked vertex remains. In that case, chain d has a positive contribution to the right-hand side of Constraint (13b). This means that whenever the same feasible solutions $S \in \mathcal{F}_G$ are added to the model, the PICEF-based model will have a value at least as large as the CC-based model for the attack generation problem. As a result, it will often occur that Line 4 is triggered after a smaller number of iterations of the Benders-type algorithm.

Algorithm 2: Benders-type algorithm for solving the attack generation problem based on PICEF

```
    Data: graph $G = (V, A)$, set $\bar{\mathcal{U}} \subseteq \mathcal{U}$ of attacks, optimal initial KEP solution $\bar{x} \in \mathcal{X}$ for $z_{FR}(\bar{\mathcal{U}})$
    Result: attack $u^* \in \mathcal{U} \setminus \bar{\mathcal{U}}$ with $R(\bar{x}, \bar{\xi}, u^*) < R(\bar{x}, \bar{\xi}, \bar{u})$ for all $\bar{u} \in \bar{\mathcal{U}}$, or show nonexistence
1   $\mathcal{F}'_G = \{(\bar{x}, \bar{\xi})\}$;
2   while true do
3       $u^* \in \arg\min_{u \in \mathcal{U}} z(\mathcal{F}'_G, \bar{x}, \bar{\xi})$;
4       if $z(\mathcal{F}'_G, \bar{x}, \bar{\xi}) \geq z_{FR}(\bar{\mathcal{U}})$ then
5           return null;
6       if $R(\bar{x}, \bar{\xi}, u^*) < z_{FR}(\bar{\mathcal{U}})$ then
7           return $u^*$;
8       $(x(u^*), \xi(u^*)) \in \arg\max_{(x^{u^*}, \xi^{u^*})} R(\bar{x}, \bar{\xi}, u^*)$;
9       $\mathcal{F}'_G \leftarrow \mathcal{F}'_G \cup \{x(u^*), \xi(u^*)\}$;
```

4 Lifting Benders cuts

In this section, we discuss a method to strengthen the constraints of type (11b) and (13b), which we refer to as the Benders constraints. Let us first consider two KEP solutions $S, S' \in \mathcal{F}_G$ with $S \subset S'$. Then, the Benders constraint related to S is strictly implied by the Benders constraint for S', i.e., if the constraint corresponding to S is violated, so is the constraint corresponding to S'. We say the constraint for S' *lifts* the constraint for S. We consider the term *lifting* here since the cut for S' is obtained from the cut for S through lifting the cycle and chain variables x_c for $c \in S' \setminus S$. We refer to the book of Conforti et al. (2014) for an extensive survey of different lifting techniques for general mixed-integer programs. The algorithm laid out in the previous section can produce sets S that can still be lifted. Indeed, in the current implementation of the subproblem, $R(\bar{x}, u')$, is a maximum weight kidney exchange problem on a subgraph $G[V_{u'}]$, i.e, the graph after removing attacked vertices with respect to u'. An optimal solution S to this subproblem thus cannot contain attacked cycles and chains. The solution may be expanded by adding such cycles, and thus lifting the resulting constraint. We explore for both the cycle-chain formulation and PICEF a method for finding lifted cuts through re-weighting the cycles, chains and arcs. The main idea is to identify KEP solutions that maximize the recourse value, given the attack u. Ties are broken by choosing the largest number of cycles, chains, and arcs in the solution. This means that attacked cycles or chains are included if possible, because even though they do not add recourse value, they do increase the number of cycles and chains. We can achieve this goal solving a single KEP on the entire compatibility graph by appropriate re-weightings of all cycles and chains.

4.1 Cycle-chain formulation

Let us be given a subset $\mathcal{F}'_G \subset \mathcal{F}_G$ of feasible KEP solutions. In order to obtain stronger cuts for $z(\mathcal{F}'_G, \bar{x})$, we need to make two modifications to $R(\bar{x}, u)$ defined in (3). First of all, given an attack $u \in \mathcal{U}$, let us consider different weights $\hat{w}_c(\bar{x}, u)$ for each cycle or chain $c \in \mathcal{C}_K \cup \mathcal{D}_L$ given by

$$\hat{w}_c(\bar{x}, u) = \begin{cases} w_c(\bar{x})|V| + 1, & \text{if } V(c) \subseteq V_u, \\ 1, & \text{otherwise.} \end{cases} \tag{14}$$

Furthermore, we relax the right hand sides $1 - u_j$ of Constraints (3b) to 1, meaning that we are not restricting a solution to non-attacked cycles and chains anymore. We denote the resulting problem by $R_{\text{lift}}(\bar{x}, u)$.

$$R_{\text{lift}}(\bar{x}, u) = \max \sum_{c \in \mathcal{C}_K \cup \mathcal{D}_L} \hat{w}_c(\bar{x}, u) x^u_c, \tag{15a}$$

$$\sum_{c \in \mathcal{C}^j_K \cup \mathcal{D}^j_L} x^u_c \leq 1, \qquad \forall j \in P \cup N, \tag{15b}$$

$$x^u \in \{0, 1\}^{\mathcal{C}_K \cup \mathcal{D}_L}. \tag{15c}$$

We can strengthen Benders cuts through solving $R_{\text{lift}}(\bar{x}, u)$, which is proven in the following lemma.

Lemma 6. *Let $G = (V, A)$ be a digraph with $\mathcal{F}_G \neq \emptyset$. Let $\bar{x} \in \mathcal{X}$ be an arbitrary initial KEP solution, and $u \in \mathcal{U}$ an arbitrary attack. Then, for any $S' \in \mathcal{F}_G$ that is an optimal recourse KEP solution to $R_{\text{lift}}(\bar{x}, u)$, we have that $S = \rho(S', u)$ is an optimal recourse KEP solution to $R(\bar{x}, u)$.*

Proof. Let z^*_{lift} and z^* be the optimal objective values to Problems (15) and (3) respectively. Furthermore, let $S' \in \mathcal{F}_G$ be any optimal solution to (15). Notice that by the new weights $\hat{w}_c(\bar{x}, u)$ defined in (14), we have that $z^*_{\text{lift}} > z^*|V|$, since any feasible solution to (3) is also feasible to (15). We claim now that the remainder $S = \rho(S', u)$ of S' under attack u is optimal to (3). Suppose on the contrary S is not optimal for (3), then the objective value of the feasible solution S to (3) is at most $z^* - 1$. Since a feasible KEP solution consists of at most $\frac{|V|}{2}$ cycles, we get that

$$\sum_{c \in S'} \hat{w}_c(\bar{x}) = \sum_{c \in S} \hat{w}_c(\bar{x}) + \sum_{c \in S' \setminus S} \hat{w}_c(\bar{x}) \leq \left((z^* - 1)|V| + \frac{|V|}{2}\right) + \frac{|V|}{2} = z^*|V| < z^*_{\text{lift}},$$

which contradicts the fact that S' is optimal for (15). Therefore, $S \in \mathcal{F}_G$ must be optimal for (3). □

Consequently, Lemma 6 shows that the Benders cuts derived from solving $R_{\text{lift}}(\bar{x}, u)$ are at least as strong as those obtained from $R(\bar{x}, u)$. The difference, however, is that we now allow to find feasible solutions $S' \in \mathcal{F}_G$ including attacked cycles and chains $c \in \mathcal{C}_K \cup \mathcal{D}_L$ with respect to u, while the solution $S = \rho(S', u)$ is still an optimal solution to the original best recourse problem $R(\bar{x}, u)$. Since Problems (5) and (15) are both weighted kidney exchange problems, we expect that both can be solved equally efficient, while the latter will in general derive the strongest cuts. Notice that we identify, among all solutions S' with remainder $S = \rho(S', u)$ being optimal to $R(\bar{x}, u)$, the solution containing the largest number of cycles and chains because of the new weights $\hat{w}_c(\bar{x}, u)$. Such solutions often consist of relatively small cycles and chains, meaning that the remainder of the solution is often larger after an attack $u \in \mathcal{U}$ is observed, indicating once again the strength of these lifted cuts.

4.2 Position-indexed cycle edge formulation (PICEF)

A similar adjustment can be performed on the maximum weight kidney exchange problem based on PICEF to derive stronger cuts (13b). In order to obtain solutions with cycles that are not contained in $G[V_u]$ given an attack $u \in \mathcal{U}$, we again need to relax the right-hand sides of constraints, namely constraints (10b) and (10c). The re-weighting of the cycle weights is also similar to that in the Cycle-Chain formulation, (14), i.e.,

$$\hat{w}_c(\bar{x}, \bar{\xi}, u) = \begin{cases} w_c(\bar{x}, \bar{\xi})|V| + 1, & \text{if } V(c) \subseteq V_u, \\ 1, & \text{otherwise} \end{cases} \tag{16}$$

The main additional complication here is to assign proper weights to the arc variables. An arc which is not attacked directly may still not contribute to the recourse value, if an arc earlier in the chain is already attacked. An arc weight based simply on the existence in V_u thus does not work. To solve this problem, we use two types of arc-variables, ξ^u_{ijl}, the traditional binary PICEF variables indicating whether arc (i, j) is used in position l of a chain, and y_{ij} a binary variable indicating whether that arc indeed contributes to the recourse value, i.e. there has not been an attack earlier in the chain that uses this arc (i, j). For the y_{ij}-variables, we introduce weights

$$\hat{w}_{ij}(\bar{x}, \bar{\xi}, u) = \begin{cases} w_{ij}(\bar{x}, \bar{\xi})|V| + 1, & \text{if } i \in N, \\ w_{ij}(\bar{x}, \bar{\xi})|V|, & \text{if } i \in P \end{cases} \tag{17}$$

with $w_{ij}(\bar{x}, \bar{\xi})$ as defined in (8). Lastly, we also assign weights $\hat{w}_{ij\ell}(\bar{x}, \bar{\xi}) = \frac{1}{|V|}$ to the arc variables $\xi_{ij\ell}$. The modified model $R_{\text{lift}}(\bar{x}, \bar{\xi}, u)$ is then given by

$$
\begin{aligned}
R_{\text{lift}}(\bar{x},\bar{\xi},u) = \max \; & \sum_{c\in\mathcal{C}_K} \hat{w}_c(\bar{x},\bar{\xi},u)x^u_c + \sum_{(i,j)\in A} \hat{w}_{ij}(\bar{x},\bar{\xi},u)y_{ij} \\
& + \sum_{\ell\in\mathcal{L}} \sum_{(i,j)\in A_\ell} \hat{w}_{ij\ell}(\bar{x},\bar{\xi})\xi^u_{ij\ell}, && (18a) \\
\text{s.t.} \; & \sum_{c\in\mathcal{C}^j_K} x^u_c + \sum_{\substack{i:(i,j)\in A \\ \ell\in\mathcal{L}(i,j)}} \xi^u_{ij\ell} \le 1, && \forall j \in P \quad (18b) \\
& \sum_{i:(j,i)\in A} \xi^u_{ji1} \le 1, && \forall j \in N, \quad (18c) \\
& \sum_{i:(j,i)\in A} y_{ji} - \sum_{i:(i,j)\in A} y_{ij} \le 0, && \forall j \in P, \quad (18d) \\
& y_{ij} - \min\{1-u_i, 1-u_j\} \le 0, && \forall (i,j) \in A, \quad (18e) \\
& y_{ij} - \sum_{\ell\in\mathcal{L}(i,j)} \xi^u_{ij\ell} \le 0, && \forall (i,j) \in A, \quad (18f) \\
& \sum_{i:(j,i)\in A_\ell} \xi^u_{ji\ell} - \sum_{i:(i,j)\in A_{(\ell-1)}} \xi^u_{ij(\ell-1)} \le 0, && \forall j \in P, \ell \in \mathcal{L}\setminus\{1\}, \quad (18g) \\
& x^u_c \in \{0,1\}, && \forall c \in \mathcal{C}_K, \quad (18h) \\
& y_{ij} \in \{0,1\}, && \forall (i,j) \in A, \quad (18i) \\
& \xi^u_{ij\ell} \in \{0,1\}, && \forall \ell \in \mathcal{L}, (i,j) \in A_\ell. \quad (18j)
\end{aligned}
$$

This model can be explained as follows. In Constraints (18b) and (18c), we again make sure that the recourse KEP solution (x^u, ξ^u) is vertex-disjoint. Constraints (18g) tell us that a position indexed arc variable $\xi_{ij\ell}$ with $\ell \geq 2$ can only be set to 1 whenever there exists a preceding arc (k,i) going into i at position $\ell - 1$ such that $\xi_{ki(\ell-1)} = 1$, since (i,j) needs to be part of a chain starting from an NDD. The remaining constraints deal with the arc variables y_{ij} for each arc $(i,j) \in A$. Constraints (18d) have a similar meaning as Constraints (18g), to make sure the arcs with $y_{ij} = 1$ indeed induce vertex-disjoint chains. Furthermore, Constraints (18e) make sure that $y_{ij} = 0$ if either i or j is attacked by u, since then (i,j) is not an arc in $G[V_u]$. Constraints (18f) imposes that arc variables y_{ij} can only be set to 1 if one of the corresponding position indexed arc variables $\xi_{ij\ell} = 1$ for $\ell \in \mathcal{L}(i,j)$.

We can once again prove that stronger Benders cuts can be obtained to solving $R_{\text{lift}}(\bar{x},\bar{\xi},u)$ instead of the model in (10).

Lemma 7. *Let $G = (V,A)$ be a digraph with $\mathcal{F}_G \neq \emptyset$. Let $(\bar{x},\bar{\xi}) \in \mathcal{X}$ and $u \in \mathcal{U}$. Then, for any $S' \in \mathcal{F}_G$ that is an optimal recourse KEP solution to $R_{\text{lift}}(\bar{x},\bar{\xi},u)$, $S = \rho(S',u)$ is an optimal recourse KEP solution to $R(\bar{x},\bar{\xi},u)$.*

Proof. Let z^*_{lift} and z^* be the optimal objective values for (18) and (10) respectively. Furthermore, let (x^u, y, ξ^u) be any optimal solution to (18) inducing a feasible KEP solution $S' \in \mathcal{F}_G$. Once again, since any feasible solution to (10) can be mapped to a feasible solution to (18), we get $z^*_{\text{lift}} > z^*|V|$ by considering the new weights. Again, we claim that the subsolution $S = \rho(S',u) \subseteq S'$ is optimal. By contradiction, suppose it is not optimal, then the objective value corresponding to (x^u, y, ξ^u) in (18) is

$$
\begin{aligned}
& \sum_{c\in\mathcal{C}_K} \hat{w}_c(\bar{x},\bar{\xi})x^u_c + \sum_{(i,j)\in A} \hat{w}_{ij}(\bar{x},\bar{\xi},u)y_{ij} + \sum_{\ell\in\mathcal{L}} \sum_{(i,j)\in A_\ell} \hat{w}_{ij\ell}(\bar{x},\bar{\xi})\xi^u_{ij\ell} \\
& \le (z^*-1)|V| + \frac{|V|}{2} + \sum_{\ell\in\mathcal{L}} \sum_{(i,j)\in A_\ell} \hat{w}_{ij\ell}(\bar{x},\bar{\xi})\xi^u_{ij\ell} \\
& \le (z^*-1)|V| + \frac{|V|}{2} + 1 < z^*|V|.
\end{aligned}
$$

Once again, we have obtained a contradiction, since we assumed S' is optimal to (18) with value $z^*_{\text{lift}} > z^*|V|$. Therefore, $S' \in \mathcal{F}_G$ must be optimal to (10). □

This proves that also with the PICEF reformulation of the attack generation problem $A(\bar{x},\bar{\xi})$, we can deploy a new formulation for the Benders submodel to obtain Benders cuts that are at least equally strong. In the following section we provide computational experience where we compare our Benders-type approach, including these lifting schemes, to the reference method due to Carvalho et al. (2020).

5 Computational experience

The aim of this section is to evaluate and compare the performance of our new Benders-type approaches with the branch-and-bound-based approach by Carvalho et al. (2020) on practical instances. To this end, we investigate the following two questions:

Q1 Is our new Benders-type approach computationally more efficient than the branch-and-bound-based approach?

Q2 Does the PICEF-based Benders approach dominate the approach based on the cycle-chain formulation or vice versa?

For the sake of brevity, we refer to our approaches using the cycle-chain formulation and the PICE formulation as Benders and Benders PICEF method, respectively. The reference method by Carvalho et al. (2020) is referred to as branch-and-bound method. In the following, we first discuss our computational setup and used test sets in Section 5.1. Afterwards, we present our numerical results in Section 5.2 and discuss the impact of lifting in Section 5.3.

5.1 Test sets and computational setup

In our experiments, we have used the same test set as Carvalho et al. (2020), which is publicly available[1]. This test set consists of ninety graphs of which thirty contain 20, 50, and 100 vertices, respectively. More characteristics of these instances can be found in Carvalho et al. (2020). From these graphs, 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 7.0.2 with SoPlex 5.0.2 as LP solver[2], see Gamrath et al. (2020). All computations were run on a Linux cluster with Intel Xeon E5 3.5 GHz quad core processors and 32 GB memory. The code was executed using a single thread. The time limit of all computations is 1 h per instance.

For both the Benders and branch-and-bound method, we compute all cycles of length at most K and chains of length at most L present in the compatibility graph of an instance, whereas for the Benders PICEF method we just compute all cycles. 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) and 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 Benders methods, we use the lifting methods as described in Section 4 to separate stronger Benders cuts. We also conducted experiments without lifting, which we will discuss briefly in Section 5.3. Since the Benders approaches with lifting dominate the ones without, we only present numbers for the approaches with lifting in our comparison with the branch-and-bound method for the ease of presentation.

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. (2020), we have implemented the same strategies as discussed there. A summary of these strategies is given in Appendix A.

5.2 Numerical results

To empirically find answers to Questions Q1 and Q2, we have conducted experiments with varying maximum cycle and chain lengths K and L for the compatibility graphs described in Section 5.1. 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; attacks of edges are not taken into account, which is consistent with the experiments by Carvalho et al. (2020). In comparison to the experiments described in Carvalho et al. (2020), we consider the impact of varying cycle lengths, while they only considered length $K = 3$, since multiple European countries allow longer cycles, e.g. Czech Republic and The Netherlands, see Biró et al. (2020). The different chain lengths in our experiments are the same as in Carvalho et al. (2020) (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).

[1] `https://rdm.inesctec.pt/dataset/ii-2020-001`
[2] Our implementation is publicly available at `https://github.com/DannyBlom/BendersRobustKEP/commit/5b44c0c2`

Table 1 reports on our experiments for $K = 3$ and different values of L as well as varying numbers of attacks; Table 4 summarizes our experiments for $K = 4$.

Before we proceed with a discussion of our experiments, we describe the common structure of the presented tables. Column "#vertices" reports on the number of vertices in the tested graphs, whereas column "#opt" provides the number of instances solved within the time limit. The average solution time per instance is given in "time total", whereas "stage 2" and "stage 3" present the average proportion of running time spent in stage 2 and 3, respectively. For the branch-and-bound method, "stage 2" represents the problem of solving the attack generation problem with the branch-and-bound algorithm. For the Benders and Benders PICEF method, "stage 2" and "stage 3" correspond to solving the Benders master problems and the Benders subproblems, respectively. Note that all reported times are in seconds. Column "#att." reports on the average 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 "#B&B-nodes" gives the average number of branch-and-bound nodes per optimally solved instance for the branch-and-bound method, and "#sub" shows the average number of subproblems in stage 3 that needed to be solved by the Benders methods for instances that could be solved within the time limit.

All average values are given in arithmetic mean with the only exception being the average solving time, which is measured in shifted geometric mean

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

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.

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 seconds. For graphs with 50 vertices, however, the branch-and-bound method fails to solve all instances for $L = 2$ and $B = 4$; for $L = 3$ and $L = 4$, it starts encountering problems in solving all instances already if the attack budget B is 3 or 2, respectively. In contrast to this, the Benders approach can solve all instances with up to 50 vertices within the time limit of one hour except for one instance with $L = 4$ and attack budget $B \geq 3$. Benders PICEF is able to solve all instances within the time limit. This is also reflected in the running time of the different methods: the Benders approaches are on average one order of magnitude faster than the branch-and-bound method.

For instances with 100 vertices in the compatibility graph, the branch-and-bound method is hardly able to solve any instance if the attack budget B is at least 3 for $L = 2$; for increasing chain length, even instances with $B \in \{1, 2\}$ become challenging for the branch-and-bound approach. The Benders approach is much more competitive on these instances as it allows to solve all instances with $L = 2$ within the time limit, and also for $L = 3$ it can solve at least half of the instances. If the maximum chain length exceeds 3, however, also the Benders approach fails to solve more than 3 instances per test set. Using the Benders PICEF method, we can solve almost all instances and there is only a small degradation of the number of solved instances if the maximum chain length L is increased.

Based on these experiments, we can answer Question Q1 affirmatively as the Benders approaches are able to consistently solve more instances than the branch-and-bound method within the time limit of one hour, which results in a running time improvement of one order of magnitude for most instance sets. Regarding the comparison of the Benders and the Benders PICEF method, we can observe that for instances with $L = 2$, the Benders method is superior to the Benders PICEF approach: for 100 vertices and all different attack budgets $B \in \{1, 2, 3, 4\}$, the Benders method is at least 31.5 % faster than the Benders PICEF approach; for $B = 1$, it is even 46.9 % faster. For $L \geq 3$, however, the picture changes and Benders PICEF is the more competitive method. As already explained above, it is able to solve almost all instances independent from the number of attacks, whereas Benders can hardly solve any instance with 100 vertices. But even if we consider test sets in which both Benders and Benders PICEF can solve almost all instances, Benders PICEF is slightly faster if $L = 3$. For $L = 4$ and graphs with 50 vertices, Benders PICEF is between 71.5 % and 84.0 % faster than the Benders method.

Two explanations for this behavior are, first that the number of variables in the cycle-chain formulation depends exponentially on the maximum chain and cycle length, leading to already millions of chains for the graphs in our test set with 100 vertices and $L = 4$. The impact of the maximum chain length L on the number of variables in PICEF, however, is only linear, which leads to a much more compact model than the cycle-chain formulation. Our experiments show that we can benefit from the smaller formulation already if the chain length $L \geq 2$, because the corresponding integer programs can be solved more efficiently. A second reason is the slightly stronger formulation of the attack generation problem using PICEF. In Remark 5, we note that theoretically, the cycle-chain formulation may require more

Table 1: Comparison of the different methods for maximum cycle length 3 and different chain lengths.

	Branch-and-Bound					Benders						Benders PICEF					
		time					time						time				
#vertices	#opt	total	stage 2	#att.	#B&B-nodes	#opt	total	stage 2	stage 3	#att.	#sub	#opt	total	stage 2	stage 3	#att.	#sub
chain length: 2																	
$B = 1$																	
20	30	0.06	46.21 %	1.8	12.2	30	0.10	25.02 %	25.50 %	2.2	5.9	30	0.11	29.73 %	26.85 %	2.1	5.7
50	30	3.17	32.05 %	2.9	39.6	30	2.92	5.76 %	20.41 %	3.0	14.5	30	4.69	13.68 %	33.48 %	2.9	14.1
100	30	50.25	39.83 %	3.4	75.0	30	37.42	2.84 %	26.52 %	3.0	24.4	30	70.47	10.11 %	38.28 %	3.7	27.3
$B = 2$																	
20	30	0.19	57.25 %	3.1	57.1	30	0.13	28.61 %	26.40 %	2.8	8.1	30	0.17	33.30 %	28.02 %	3.0	9.2
50	30	18.74	65.00 %	5.7	645.3	30	6.12	13.59 %	16.95 %	4.8	33.6	30	9.04	15.68 %	36.76 %	4.5	31.6
100	27	515.54	83.01 %	5.4	2039.9	30	75.16	9.88 %	28.62 %	4.7	56.4	30	110.57	10.27 %	45.83 %	4.7	46.8
$B = 3$																	
20	30	0.57	70.05 %	3.4	230.3	30	0.18	29.96 %	27.23 %	3.2	8.5	30	0.20	33.69 %	26.49 %	3.3	9.0
50	30	100.87	85.90 %	8.0	7857.3	30	9.70	19.88 %	14.03 %	5.7	49.4	30	15.47	25.12 %	31.67 %	6.0	57.0
100	8	2722.20	94.32 %	8.2	14 878.6	30	185.30	20.48 %	27.84 %	6.2	169.8	30	270.58	17.48 %	45.04 %	6.1	132.4
$B = 4$																	
20	30	0.81	70.17 %	3.8	437.4	30	0.21	29.78 %	28.05 %	3.4	7.9	30	0.10	34.57 %	26.03 %	2.9	6.5
50	27	448.70	93.95 %	10.4	49 483.1	30	14.76	31.29 %	13.04 %	6.9	85.9	30	16.79	27.04 %	30.05 %	6.6	64.4
100	1	3556.71	91.86 %	14.0	129 956.0	30	228.17	31.04 %	27.39 %	7.0	220.6	29	338.15	19.70 %	46.04 %	7.4	169.9
chain length: 3																	
$B = 1$																	
20	30	0.20	40.58 %	1.8	12.1	30	0.13	24.15 %	26.32 %	2.0	6.1	30	0.21	28.74 %	27.16 %	1.9	5.6
50	30	13.67	26.60 %	3.1	44.7	30	12.00	4.20 %	20.09 %	3.0	16.8	30	8.05	13.07 %	47.50 %	2.6	14.1
100	27	713.16	41.23 %	3.3	93.3	28	544.28	1.96 %	25.92 %	3.4	28.5	30	109.74	9.60 %	44.09 %	3.4	26.9
$B = 2$																	
20	30	0.63	51.66 %	3.4	72.3	30	0.26	26.90 %	24.89 %	2.8	8.8	30	0.30	30.32 %	28.19 %	2.8	8.9
50	30	64.15	58.30 %	6.0	682.3	30	21.42	9.28 %	19.27 %	4.6	38.7	30	19.90	15.26 %	42.76 %	4.7	36.0
100	8	2723.01	54.99 %	5.5	1472.0	24	1053.88	3.47 %	31.69 %	4.5	96.1	30	236.19	8.94 %	47.95 %	4.7	63.9
$B = 3$																	
20	30	0.85	67.87 %	3.5	225.1	30	0.24	28.08 %	26.11 %	2.9	7.8	30	0.40	31.83 %	27.58 %	3.2	9.5
50	28	347.72	83.01 %	8.9	8235.3	30	30.49	15.18 %	18.68 %	5.6	71.3	30	29.78	17.19 %	43.75 %	6.1	57.3
100	1	3431.08	65.23 %	6.0	8969.0	19	1733.56	4.67 %	29.93 %	6.5	177.9	29	439.32	12.57 %	49.23 %	6.1	118.5
$B = 4$																	
20	30	1.68	69.54 %	4.3	766.7	30	0.17	28.78 %	26.93 %	2.9	7.3	30	0.27	31.44 %	25.13 %	3.0	7.1
50	17	1200.94	92.45 %	10.7	38 786.4	30	34.63	18.21 %	17.65 %	7.0	98.6	30	29.80	22.01 %	44.39 %	6.0	76.0
100	0	3600.00	66.09 %	—	—	16	2023.72	5.03 %	20.78 %	8.0	182.8	28	645.27	16.04 %	52.03 %	7.5	177.2
chain length: 4																	
$B = 1$																	
20	30	0.44	39.89 %	1.9	14.5	30	0.36	22.34 %	23.75 %	2.0	6.3	30	0.29	26.45 %	29.20 %	1.9	5.8
50	30	102.38	33.50 %	3.0	57.3	30	91.22	3.09 %	22.19 %	3.0	20.0	30	14.60	10.15 %	43.09 %	2.7	16.4
100	3	3173.35	24.55 %	3.7	72.0	3	3126.14	0.47 %	18.14 %	4.0	24.0	30	179.18	7.82 %	44.80 %	3.0	26.3
$B = 2$																	
20	30	1.10	53.79 %	3.3	97.7	30	0.74	24.97 %	23.78 %	2.9	9.4	30	0.39	28.53 %	28.78 %	2.7	9.2
50	25	384.13	70.31 %	5.6	822.5	30	138.06	4.27 %	19.55 %	4.8	37.1	30	26.92	11.57 %	43.47 %	4.1	34.4
100	0	3600.00	18.63 %	—	—	3	3545.57	0.38 %	9.79 %	7.0	43.3	30	415.05	7.35 %	47.30 %	4.8	70.1
$B = 3$																	
20	30	1.82	66.07 %	3.4	314.3	30	0.71	24.63 %	23.71 %	3.2	8.7	30	0.36	29.52 %	26.78 %	2.9	7.7
50	17	1115.28	85.14 %	8.4	7408.2	29	164.60	8.33 %	23.45 %	5.6	66.4	30	35.70	14.91 %	44.33 %	5.1	52.7
100	0	3600.00	15.72 %	—	—	3	3413.66	0.80 %	11.71 %	6.3	118.7	28	783.46	10.07 %	46.87 %	6.0	114.9
$B = 4$																	
20	30	2.83	68.78 %	3.9	784.9	30	0.58	25.63 %	24.27 %	3.2	7.9	30	0.41	30.34 %	24.27 %	3.0	7.2
50	8	2266.21	90.94 %	12.1	33 891.5	29	162.00	10.35 %	21.18 %	6.7	79.5	30	46.22	15.95 %	44.79 %	6.3	75.2
100	0	3600.00	12.99 %	—	—	2	3503.18	0.57 %	8.17 %	9.5	99.5	26	1105.38	13.03 %	51.88 %	7.6	178.8

Table 2: Comparison of the Benders method with and without lifting for maximum cycle length 3 and different chain lengths.

	with lifting				without lifting			
#vertices	#opt	time	#att.	#sub	#opt	time	#att.	#sub
chain length: 2								
20	120	0.2	2.9	7.6	120	0.2	2.9	8.9
50	120	8.4	5.1	45.9	120	14.8	5.0	107.4
100	120	131.5	5.2	117.8	110	341.0	5.5	271.8
chain length: 3								
20	120	0.2	2.6	7.5	120	0.3	2.7	9.8
50	120	24.6	5.1	56.3	120	32.4	5.0	109.5
100	87	1338.9	5.6	121.3	71	1635.8	5.8	243.1
chain length: 4								
20	120	0.6	2.8	8.1	120	0.7	2.9	10.9
50	118	139.0	5.0	50.7	115	186.0	5.2	96.8
100	11	3397.1	6.7	71.4	8	3449.8	6.8	196.6

KEP solutions and thus more iterations of the Benders algorithm to obtain the same result, and this is reflected in our computational results. For almost every setting, more KEP solutions are generated using the cycle-chain formulation. Regarding Question Q2, we thus conclude that for short chains the Benders method dominates the Benders PICEF approach, whereas for $L \geq 3$, Benders PICEF is more efficient. These conclusions are also supported by our experiments with maximum cycle length $K = 4$. For the sake of brevity, we do not report on these numbers in detail and refer the reader to Appendix B for detailed figures.

Since the use of a PICEF-based formulation has a positive impact on the running time of the Benders method for long chains, one might wonder whether the same holds true for the branch-and-bound method. Due to our experiments, this seems to be reasonable, but we expect a branch-and-bound method to still be significantly slower than the PICEF Benders method, even when its implementation uses PICEF. First, we note that the branch-and-bound method needs more time to generate a single attack using the same underlying formulation. This effect is the most apparent for the instances with 100 vertices. For example, for $L = 3$ and $B = 2$ on graphs with 50 vertices, the average time per attack is 10.7 s for the branch-and-bound method and 4.7 s for the Benders method. Moreover, the branch-and-bound approach generates more attacks than the Benders-type approach for the cycle-chain formulation. While, as we have shown, the PICEF reduced the numbers of attacks required for the Benders-type approach, there is no reason to expect this in the branch-and-bound framework.

Remark 8. When comparing the number of generated attacks for the branch-and-bound method with the numbers reported by Carvalho et al. (2020), 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. (2020), cf. Appendix A, 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 to the one used by Carvalho et al. (2020). Thus, since these factors impact both the branch-and-bound and Benders models equally, our implementation does not give an advantage to either of these methods.

5.3 Evaluation of the impact of lifting

As mentioned above, we have also conducted experiments to measure the impact on the Benders approaches of the lifting schemes proposed in Section 4. Table 2 and Table 3 show aggregated information for graphs of different sizes using the Benders and Benders PICEF approach, respectively. Again, the attack budgets B from 1 to 4 are used. The aggregated numbers are average values of the numbers provided in the detailed tables of these experiments, which can be found in Appendix C.

For both approaches, we can see that lifting has a positive impact on the performances of the Benders approaches. 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 in the approaches making use of the lifting technique. This is reflected in the running time, which is much smaller if lifting is enabled. Thus, also more instances can be solved. Since a fair comparison of these numbers is only possible if the number of solved instanced is almost the same for the methods to be compared, we report in the following only on more detailed numbers for graphs with at most 50 vertices. Before presenting these

Table 3: Comparison of the Benders PICEF method with and without lifting for maximum cycle length 3 and different chain lengths.

	with lifting				without lifting			
#vertices	#opt	time	#att.	#sub	#opt	time	#att.	#sub
chain length: 2								
20	120	0.1	2.8	7.6	120	0.2	2.8	8.7
50	120	11.5	5.0	41.8	120	19.2	4.9	112.7
100	119	197.4	5.5	94.1	101	659.6	6.2	247.1
chain length: 3								
20	120	0.3	2.7	7.8	120	0.3	2.6	9.4
50	120	21.9	4.8	45.8	119	41.8	4.9	140.8
100	117	357.6	5.4	96.6	82	1242.4	5.8	224.8
chain length: 4								
20	120	0.4	2.6	7.5	120	0.4	2.6	9.4
50	120	30.9	4.6	44.7	117	61.3	4.7	119.5
100	114	620.8	5.3	97.5	66	2043.3	5.6	278.0

numbers, we want to stress that, for instances with 100 vertices, Benders PICEF can solve 42.7 % more instances for $L = 3$ (72.7 % for $L = 4$) if lifting is enabled.

For instances with 20 vertices, there is almost no difference for the running time if lifting is enabled or not. For graphs with 50 vertices, however, the number of solved subproblems for the Benders method using lifting is roughly 50 % smaller than without lifting. This leads to a running time reduction of about 25 % for $L \in \{3, 4\}$ and 43.2 % for $L = 2$. For the Benders PICEF method, we can observe a similar trend, which is even slightly stronger. Here, lifting leads to a running time improvement between 50.4 % and 59.9 % for graphs with 50 vertices, and the number of generated attacks reduces by 62.6 %–67.5 %.

We conclude that lifting is an important component for obtaining a performant Benders method.

6 Conclusion

In this paper, we investigate a robust optimization variant of the kidney exchange problem, based on the fact that donors might decide to opt out of the program after exchanges are proposed but before transplants take place. This is a practical issue, since donors and pairs leaving a kidney exchange program leads to broken cycles and chains of initially proposed transplants.

We propose a Benders decomposition algorithm for solving the attack generation problem based on the cycle-chain formulation and the position-indexed chain-edge formulation. The algorithms are based on a reformulation of the attack generation algorithm with constraints for any feasible KEP solution. Initially, the Benders master problem takes into account a small number of feasible KEP solutions, and new KEP solutions are added by the Benders submodel through solving a weighted kidney exchange problem on an induced subgraph after observing which donors have left the program. The idea of using Benders decomposition is intuitive, since the variables used in the reformulation are decoupled whenever a set of leaving donors is fixed. 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 the Benders decomposition algorithm outperforms the state-of-the-art, which is a branch-and-bound type algorithm proposed in Carvalho et al. (2020), when we consider benchmark instances with $|V| \geq 50$. We observe a transition phase between the Benders methods based on CC and PICEF. Whenever we allow only small chains, i.e., $L = 2$, the CC-based implementation outperforms the PICEF-based implementation, since the number of chain variables is then often smaller than the number of position-indexed arc variables. However, when larger chains are allowed, i.e., $L \geq 3$, the PICEF-based implementation becomes the significantly more competitive method and the benefit of having a polynomial-size formulation in terms of chains becomes really apparent. Enabling the lifting technique leads, under all chosen sets of parameters, to a significant reduction in the running time since fewer cuts need to be generated in the last iteration of the iterative algorithm for solving the large-scale mixed integer programming model of the robust optimization problem. Since the stronger Benders cuts give better feedback on the recourse value that can be realized given an attack, we need fewer attacks to solve the robust optimization model to optimality. This also reflects in an increased number of instances that can be solved within the time limit.

Extending this work to a setting with a less restricted uncertainty set will give more insight in how to incorporate the possibility of donor or transplant failure in a more realistic setting, since one cannot predict in advance the number of donors that leave the KEP. A more sophisticated data analysis of the physiological properties of donors and recipients may be necessary to find an uncertainty set that more accurately approximates the real set of attacks that may occur.

References

David J Abraham, Avrim Blum, and Tuomas Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 295–304, 2007.

Filipe Alvelos, Xenia Klimentova, and Ana Viana. Maximizing the expected number of transplants in kidney exchange programs with branch-and-price. *Annals of Operations Research*, 272(1-2):429–444, 2019.

Hoda Bidkhori, John Dickerson, Duncan McElfresh, and Ke Ren. Kidney exchange with inhomogeneous edge existence uncertainty. In Jonas Peters and David Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 161–170. PMLR, 03–06 Aug 2020. URL http://proceedings.mlr.press/v124/bidkhori20a.html.

Péter Biró, Bernadette Haase-Kromwijk, Tommy Andersson, Eyjólfur Ingi Ásgeirsson, Tatiana Baltesová, Ioannis Boletis, Catarina Bolotinha, Gregor Bond, Georg Böhmig, Lisa Burnapp, et al. Building kidney exchange programmes in Europe—an overview of exchange practice and activities. *Transplantation*, 103(7):1514, 2019.

Péter Biró, Joris van de Klundert, David Manlove, William Pettersson, Tommy Andersson, Lisa Burnapp, Pavel Chromy, Pablo Delgado, Piotr Dworczak, Bernadette Haase, et al. Modelling and optimisation in European kidney exchange programmes. *European Journal of Operational Research*, 291(2):447–456, 2020.

M Bray, W Wang, PX-K Song, AB Leichtman, MA Rees, VB Ashby, R Eikstadt, A Goulding, and JD Kalbfleisch. Planning for uncertainty and fallbacks can increase the number of transplants in a kidney-paired donation program. *American Journal of Transplantation*, 15(10):2636–2645, 2015.

Margarida Carvalho, Xenia Klimentova, Kristiaan Glorie, Ana Viana, and Miguel Constantino. Robust models for the kidney exchange problem. *INFORMS Journal on Computing*, 2020.

Michele Conforti, Gerard Cornuejols, and Giacomo Zambelli. *Integer Programming*. Springer Publishing Company, Incorporated, 2014. ISBN 3319110071.

Miguel Constantino, Xenia Klimentova, Ana Viana, and Abdur Rais. New insights on integer-programming models for the kidney exchange problem. *European Journal of Operational Research*, 231(1):57–68, 2013.

Connie L. Davis and Francis L. Delmonico. Living-donor kidney transplantation: A review of the current practices for the live donor. *Journal of the American Society of Nephrology*, 16(7):2098–2110, 2005. ISSN 1046-6673. doi: 10.1681/ASN.2004100824. URL https://jasn.asnjournals.org/content/16/7/2098.

John P Dickerson, David F Manlove, Benjamin Plaut, Tuomas Sandholm, and James Trimble. Position-indexed formulations for kidney exchange. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 25–42, 2016.

John P Dickerson, Ariel D Procaccia, and Tuomas Sandholm. Failure-aware kidney exchange. *Management Science*, 65(4):1768–1791, 2019.

Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. Technical report, Optimization Online, March 2020. URL http://www.optimization-online.org/DB_HTML/2020/03/7705.html.

Sommer E Gentry, Dorry L Segev, Mary Simmerling, and Robert A Montgomery. Expanding kidney paired donation through participation by compatible pairs. *American Journal of Transplantation*, 7 (10):2361–2370, 2007.

Kristiaan M Glorie, J Joris van de Klundert, and Albert PM Wagelmans. Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manufacturing & Service Operations Management*, 16(4):498–512, 2014.

Xenia Klimentova, João Pedro Pedroso, and Ana Viana. Maximising expectation of the number of transplants in kidney exchange programmes. *Computers & Operations Research*, 73:1–11, 2016. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2016.03.004. URL https://www.sciencedirect.com/science/article/pii/S0305054816300533.

Duncan C McElfresh, Hoda Bidkhori, and John P Dickerson. Scalable robust kidney exchange. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1077–1084, Jul. 2019. doi: 10.1609/aaai.v33i01.33011077. URL https://ojs.aaai.org/index.php/AAAI/article/view/3899.

Paul E Morrissey, Catherine Dube, Reginald Gohh, Angelito Yango, Amitabh Gautam, and Anthony P Monaco. Good samaritan kidney donation. *Transplantation*, 80(10):1369–1373, 2005.

NHS. Annual report on living donor kidney transplantation. Technical report, NHS, 2017.

Benjamin Plaut, John Dickerson, and Tuomas Sandholm. Fast optimal clearing of capped-chain barter exchanges. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

FT Rapaport. The case for a living emotionally related international kidney donor exchange registry. *Transplantation proceedings*, 18(3) Suppl. 2):5–9, June 1986. ISSN 0041-1345. URL http://europepmc.org/abstract/MED/11649919.

Alvin E Roth, Tayfun Sönmez, and M Utku Ünver. Kidney exchange. *The Quarterly journal of economics*, 119(2):457–488, 2004.

Alvin E Roth, Tayfun Sönmez, and M Utku Ünver. Pairwise kidney exchange. *Journal of Economic theory*, 125(2):151–188, 2005.

Alvin E Roth, Tayfun Sönmez, M Utku Ünver, Francis L Delmonico, and Susan L Saidman. Utilizing list exchange and nondirected donation through 'chain'paired kidney donations. *American Journal of transplantation*, 6(11):2694–2705, 2006.

Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review*, 97(3):828–851, June 2007. doi: 10.1257/aer.97.3.828. URL https://www.aeaweb.org/articles?id=10.1257/aer.97.3.828.

Bart Smeulders, Valentin Bartier, Yves Crama, and Frits CR Spieksma. Recourse in kidney exchange programs. 2019.

J Cole Smith and Yongjia Song. A survey of network interdiction models and algorithms. *European Journal of Operational Research*, 283(3):797–811, 2020.

A Implementation of the Carvalho et al. (2020) algorithm

In this appendix, we briefly describe our implementation of the branch-and-bound algorithm by Carvalho et al. (2020) 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 close as possible to additional details on the Carvalho et al. (2020) implementation received through personal communications. References to lines are as they appear in Algorithm 2 suggested by Carvalho et al. (2020).

Line 8: Node selection We deploy a depth-first search strategy to select the next subproblem with priority to the attack branch. That is, 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 to branch on.

B Numerical results for cycle length 4

In the main part of this article, we have discussed the results of our experiments for instances with cycles of length at most 3. This appendix provides a summary of our experiments for the same compatibility graphs and maximum cycle length 4 in Table 4.

C Detailed results for comparing the effect of lifting

In Section 5.3, we have provided aggregated numbers to compare the impact of lifting on the Benders-based approaches. Tables 5 and 6 provide detailed statistics for the Benders and Benders PICEF, respectively.

Table 4: Comparison of the different methods for maximum cycle length 4 and different chain lengths.

	Branch-and-Bound					Benders						Benders PICEF					
		time					time						time				
#vertices	#opt	total	stage 2	#att.	#B&B-nodes	#opt	total	stage 2	stage 3	#att.	#sub	#opt	total	stage 2	stage 3	#att.	#sub
chain length: 2																	
$B = 1$																	
20	30	0.30	35.87 %	2.0	15.1	30	0.38	20.71 %	21.72 %	2.1	7.2	30	0.29	25.16 %	23.53 %	2.0	6.8
50	30	15.92	42.13 %	2.7	52.2	30	12.09	2.72 %	21.37 %	2.7	17.1	30	17.96	7.44 %	26.31 %	3.0	17.0
100	23	1282.06	41.56 %	3.7	133.0	23	966.47	0.86 %	20.01 %	3.2	25.7	26	955.56	5.36 %	31.17 %	3.5	30.4
$B = 2$																	
20	30	0.66	56.66 %	3.0	96.5	30	0.39	26.16 %	22.66 %	2.7	9.6	30	0.52	27.17 %	22.89 %	2.8	9.5
50	30	93.32	76.00 %	4.8	957.6	30	29.28	6.87 %	18.18 %	4.8	41.1	30	31.39	10.86 %	27.11 %	4.5	35.9
100	2	3486.72	49.68 %	7.0	2279.0	21	1452.59	1.99 %	23.59 %	4.2	71.4	18	1628.44	4.40 %	26.77 %	4.6	53.5
$B = 3$																	
20	30	1.48	68.06 %	3.7	349.3	30	0.63	25.07 %	23.13 %	3.2	9.7	30	0.62	27.72 %	20.79 %	3.5	9.7
50	25	752.99	91.61 %	7.4	13 719.8	30	32.85	13.32 %	18.26 %	5.4	67.3	30	53.31	15.87 %	24.28 %	6.3	73.9
100	0	3600.00	48.77 %	—	—	14	2051.70	3.40 %	18.93 %	5.8	95.4	12	2258.87	4.77 %	26.81 %	5.1	68.0
$B = 4$																	
20	30	2.10	69.38 %	4.1	853.8	30	0.53	25.10 %	24.11 %	3.1	8.5	30	0.60	28.34 %	21.06 %	3.8	9.2
50	11	2279.28	96.46 %	9.8	63 852.3	30	40.12	16.74 %	15.05 %	6.9	82.5	30	51.87	18.54 %	23.06 %	6.6	80.8
100	0	3600.00	47.85 %	—	—	14	2237.11	5.08 %	16.00 %	7.0	136.2	11	2401.45	5.13 %	23.86 %	6.7	115.6
chain length: 3																	
$B = 1$																	
20	30	0.39	33.90 %	2.0	17.4	30	0.55	19.27 %	21.09 %	2.4	8.4	30	0.43	24.94 %	23.10 %	1.9	6.0
50	30	33.67	38.26 %	3.0	57.5	30	22.40	2.71 %	24.20 %	2.7	18.4	30	21.49	9.05 %	33.98 %	2.7	13.9
100	14	2528.30	31.88 %	2.9	111.4	18	2158.04	0.70 %	22.18 %	3.4	33.8	26	947.88	5.82 %	35.51 %	3.2	28.6
$B = 2$																	
20	30	0.96	55.61 %	2.9	109.6	30	0.54	23.87 %	22.39 %	2.8	10.3	30	0.71	26.71 %	23.62 %	2.6	9.2
50	30	207.17	69.96 %	5.5	1000.0	30	49.24	5.14 %	19.85 %	4.7	42.2	30	46.30	9.90 %	37.08 %	4.4	36.6
100	0	3600.00	38.40 %	—	—	12	2483.07	1.55 %	28.53 %	3.8	75.1	17	1935.20	4.18 %	29.47 %	4.3	52.3
$B = 3$																	
20	30	2.02	66.02 %	3.8	370.5	30	0.53	24.06 %	23.49 %	3.1	10.0	30	0.76	27.54 %	22.82 %	3.2	9.4
50	22	1117.58	90.45 %	7.6	11 892.8	30	58.02	10.01 %	19.71 %	5.8	66.5	30	55.51	12.75 %	33.12 %	5.3	52.7
100	0	3600.00	43.51 %	—	—	8	2881.26	1.14 %	12.91 %	5.0	95.5	11	2314.60	4.25 %	23.34 %	4.6	96.5
$B = 4$																	
20	30	2.57	67.25 %	4.2	938.1	30	0.47	24.42 %	23.88 %	3.1	9.0	30	0.63	27.39 %	22.00 %	3.4	8.6
50	11	2568.27	95.59 %	10.5	56 438.5	30	67.80	13.11 %	17.60 %	7.1	82.9	30	79.92	14.39 %	32.75 %	7.1	81.7
100	0	3600.00	38.24 %	—	—	3	3266.03	2.02 %	12.00 %	7.3	125.0	6	2758.51	4.09 %	13.48 %	6.2	82.2
chain length: 4																	
$B = 1$																	
20	30	0.62	34.20 %	2.0	18.6	30	0.80	19.05 %	20.44 %	2.3	8.3	30	0.50	23.20 %	25.60 %	2.0	6.7
50	30	155.57	30.89 %	3.2	61.5	30	110.86	2.62 %	20.56 %	3.0	19.2	30	33.97	8.21 %	36.46 %	2.8	17.2
100	2	3430.96	23.15 %	3.5	119.5	3	3265.61	0.44 %	15.12 %	3.0	28.7	17	1339.69	4.91 %	24.57 %	2.9	24.4
$B = 2$																	
20	30	1.33	57.46 %	2.8	116.9	30	1.20	22.47 %	21.76 %	3.1	12.2	30	0.78	24.69 %	24.81 %	2.5	9.2
50	22	635.75	66.90 %	4.9	1005.0	30	204.67	4.17 %	19.02 %	4.6	44.5	30	56.62	8.87 %	35.76 %	4.3	33.5
100	0	3600.00	20.54 %	—	—	0	3600.00	0.37 %	10.32 %	—	—	13	2140.35	4.49 %	24.94 %	4.4	48.5
$B = 3$																	
20	30	2.33	65.95 %	3.5	362.3	30	1.15	22.19 %	22.11 %	3.4	10.7	30	0.97	26.81 %	22.81 %	3.1	10.7
50	11	1962.35	80.20 %	7.9	9803.4	29	202.17	7.99 %	26.21 %	5.2	75.3	30	85.52	11.04 %	37.69 %	5.5	76.4
100	0	3600.00	18.16 %	—	—	1	3535.35	0.44 %	9.77 %	8.0	97.0	8	2879.06	4.03 %	20.30 %	6.4	79.1
$B = 4$																	
20	30	3.98	67.12 %	4.2	1161.4	30	0.82	22.55 %	22.04 %	3.3	9.3	30	0.65	25.39 %	22.24 %	3.1	8.4
50	4	3118.14	85.02 %	11.8	59 264.2	30	244.33	9.09 %	21.03 %	6.9	93.0	30	101.61	13.27 %	34.09 %	6.7	89.4
100	0	3600.00	18.14 %	—	—	0	3600.00	0.40 %	7.22 %	—	—	9	2777.78	4.52 %	23.85 %	5.8	86.8

Table 5: Comparison of the Benders method with and without lifting for maximum cycle length 3 and different chain lengths.

	with lifting						without lifting					
		time						time				
#vertices	#opt	total	stage 2	stage 3	#att.	#sub	#opt	total	stage 2	stage 3	#att.	#sub
chain length: 2												
$B = 1$												
20	30	0.10	25.02 %	25.50 %	2.2	5.9	30	0.13	24.91 %	24.43 %	2.2	5.7
50	30	2.92	5.76 %	20.41 %	3.0	14.5	30	3.10	5.32 %	18.49 %	3.1	13.8
100	30	37.42	2.84 %	26.52 %	3.0	24.4	30	40.35	3.46 %	33.31 %	3.0	32.6
$B = 2$												
20	30	0.13	28.61 %	26.40 %	2.8	8.1	30	0.19	28.98 %	25.25 %	2.8	9.1
50	30	6.12	13.59 %	16.95 %	4.8	33.6	30	6.21	19.15 %	19.71 %	4.4	47.3
100	30	75.16	9.88 %	28.62 %	4.7	56.4	30	160.74	18.34 %	36.98 %	5.0	186.1
$B = 3$												
20	30	0.18	29.96 %	27.23 %	3.2	8.5	30	0.23	30.68 %	26.55 %	3.3	10.6
50	30	9.70	19.88 %	14.03 %	5.7	49.4	30	18.89	32.26 %	14.55 %	6.1	128.7
100	30	185.30	20.48 %	27.84 %	6.2	169.8	28	338.10	33.95 %	30.10 %	6.4	340.4
$B = 4$												
20	30	0.21	29.78 %	28.05 %	3.4	7.9	30	0.26	31.95 %	26.79 %	3.4	10.2
50	30	14.76	31.29 %	13.04 %	6.9	85.9	30	30.90	51.06 %	12.13 %	6.6	239.9
100	30	228.17	31.04 %	27.39 %	7.0	220.6	22	824.97	52.05 %	23.10 %	7.6	527.9
chain length: 3												
$B = 1$												
20	30	0.13	24.15 %	26.32 %	2.0	6.1	30	0.15	24.18 %	25.45 %	1.9	6.0
50	30	12.00	4.20 %	20.09 %	3.0	16.8	30	13.32	4.16 %	18.94 %	3.3	19.0
100	28	544.28	1.96 %	25.92 %	3.4	28.5	27	559.14	2.30 %	30.77 %	3.2	37.9
$B = 2$												
20	30	0.26	26.90 %	24.89 %	2.8	8.8	30	0.33	27.55 %	24.35 %	2.6	10.5
50	30	21.42	9.28 %	19.27 %	4.6	38.7	30	25.90	13.07 %	23.29 %	4.8	62.2
100	24	1053.88	3.47 %	31.69 %	4.5	96.1	21	1299.82	4.81 %	41.58 %	4.8	125.7
$B = 3$												
20	30	0.24	28.08 %	26.11 %	2.9	7.8	30	0.29	30.40 %	26.26 %	3.0	11.9
50	30	30.49	15.18 %	18.68 %	5.6	71.3	30	38.81	25.56 %	24.80 %	5.4	145.6
100	19	1733.56	4.67 %	29.93 %	6.5	177.9	15	2012.25	7.72 %	29.93 %	6.7	312.8
$B = 4$												
20	30	0.17	28.78 %	26.93 %	2.9	7.3	30	0.26	29.55 %	25.99 %	3.1	10.7
50	30	34.63	18.21 %	17.65 %	7.0	98.6	30	51.66	35.88 %	19.60 %	6.4	211.3
100	16	2023.72	5.03 %	20.78 %	8.0	182.8	8	2672.09	10.47 %	29.29 %	8.6	496.1
chain length: 4												
$B = 1$												
20	30	0.36	22.34 %	23.75 %	2.0	6.3	30	0.39	22.54 %	23.11 %	2.0	6.7
50	30	91.22	3.09 %	22.19 %	3.0	20.0	30	93.03	3.14 %	22.23 %	3.2	21.6
100	3	3126.14	0.47 %	18.14 %	4.0	24.0	3	3264.12	0.54 %	18.61 %	4.3	46.3
$B = 2$												
20	30	0.74	24.97 %	23.78 %	2.9	9.4	30	0.82	25.59 %	23.29 %	2.8	11.3
50	30	138.06	4.27 %	19.55 %	4.8	37.1	29	205.04	6.59 %	23.94 %	5.1	72.5
100	3	3545.57	0.38 %	9.79 %	7.0	43.3	3	3411.47	0.49 %	10.59 %	5.7	73.0
$B = 3$												
20	30	0.71	24.63 %	23.71 %	3.2	8.7	30	0.86	26.33 %	23.65 %	3.5	14.2
50	29	164.60	8.33 %	23.45 %	5.6	66.4	29	206.66	12.57 %	30.55 %	5.7	135.5
100	3	3413.66	0.80 %	11.71 %	6.3	118.7	1	3549.30	0.88 %	13.92 %	6.0	381.0
$B = 4$												
20	30	0.58	25.63 %	24.27 %	3.2	7.9	30	0.75	26.45 %	23.68 %	3.4	11.6
50	29	162.00	10.35 %	21.18 %	6.7	79.5	27	239.34	17.27 %	26.37 %	7.0	157.6
100	2	3503.18	0.57 %	8.17 %	9.5	99.5	1	3574.40	0.71 %	8.75 %	11.0	286.0

Table 6: Comparison of the Benders PICEF method with and without lifting for maximum cycle length 3 and different chain lengths.

	with lifting						without lifting					
		time						time				
#vertices	#opt	total	stage 2	stage 3	#att.	#sub	#opt	total	stage 2	stage 3	#att.	#sub
chain length: 2												
$B = 1$												
20	30	0.11	29.73 %	26.85 %	2.1	5.7	30	0.12	28.30 %	22.99 %	2.1	5.7
50	30	4.69	13.68 %	33.48 %	2.9	14.1	30	3.66	9.61 %	24.80 %	2.9	16.9
100	30	70.47	10.11 %	38.28 %	3.7	27.3	30	62.75	7.73 %	37.75 %	3.6	33.0
$B = 2$												
20	30	0.17	33.30 %	28.02 %	3.0	9.2	30	0.23	34.88 %	23.53 %	2.8	10.6
50	30	9.04	15.68 %	36.76 %	4.5	31.6	30	9.33	26.07 %	22.03 %	4.4	54.2
100	30	110.57	10.27 %	45.83 %	4.7	46.8	30	171.64	21.27 %	40.34 %	5.0	142.3
$B = 3$												
20	30	0.20	33.69 %	26.49 %	3.3	9.0	30	0.19	36.72 %	24.34 %	3.2	10.8
50	30	15.47	25.12 %	31.67 %	6.0	57.0	30	21.24	44.11 %	17.65 %	5.5	132.0
100	30	270.58	17.48 %	45.04 %	6.1	132.4	24	648.54	43.59 %	31.19 %	7.1	346.5
$B = 4$												
20	30	0.10	34.57 %	26.03 %	2.9	6.5	30	0.11	34.01 %	24.10 %	3.1	7.7
50	30	16.79	27.04 %	30.05 %	6.6	64.4	30	42.54	59.62 %	12.17 %	6.6	247.8
100	29	338.15	19.70 %	46.04 %	7.4	169.9	17	1755.43	63.15 %	20.65 %	8.9	466.5
chain length: 3												
$B = 1$												
20	30	0.21	28.74 %	27.16 %	1.9	5.6	30	0.20	26.61 %	23.09 %	2.0	5.9
50	30	8.05	13.07 %	47.50 %	2.6	14.1	30	6.27	9.16 %	37.28 %	2.5	15.5
100	30	109.74	9.60 %	44.09 %	3.4	26.9	30	182.56	8.15 %	55.01 %	3.3	35.1
$B = 2$												
20	30	0.30	30.32 %	28.19 %	2.8	8.9	30	0.30	31.74 %	24.06 %	2.7	10.9
50	30	19.90	15.26 %	42.76 %	4.7	36.0	30	21.42	19.90 %	35.14 %	4.7	70.8
100	30	236.19	8.94 %	47.95 %	4.7	63.9	27	665.93	13.41 %	66.62 %	4.7	169.3
$B = 3$												
20	30	0.40	31.83 %	27.58 %	3.2	9.5	30	0.31	34.72 %	24.15 %	2.9	10.7
50	30	29.78	17.19 %	43.75 %	6.1	57.3	30	48.52	39.57 %	29.60 %	5.8	181.4
100	29	439.32	12.57 %	49.23 %	6.1	118.5	18	1484.93	23.76 %	60.46 %	6.8	309.2
$B = 4$												
20	30	0.27	31.44 %	25.13 %	3.0	7.1	30	0.27	32.26 %	23.80 %	2.9	10.2
50	30	29.80	22.01 %	44.39 %	6.0	76.0	29	90.90	51.98 %	23.48 %	6.6	295.6
100	28	645.27	16.04 %	52.03 %	7.5	177.2	7	2636.32	40.49 %	48.77 %	8.3	385.9
chain length: 4												
$B = 1$												
20	30	0.29	26.45 %	29.20 %	1.9	5.8	30	0.28	25.20 %	24.26 %	1.9	6.0
50	30	14.60	10.15 %	43.09 %	2.7	16.4	30	14.30	7.90 %	40.43 %	2.9	20.5
100	30	179.18	7.82 %	44.80 %	3.0	26.3	30	502.77	7.35 %	71.28 %	3.4	41.8
$B = 2$												
20	30	0.39	28.53 %	28.78 %	2.7	9.2	30	0.37	29.15 %	24.37 %	2.5	10.0
50	30	26.92	11.57 %	43.47 %	4.1	34.4	30	35.19	18.26 %	44.41 %	4.0	77.4
100	30	415.05	7.35 %	47.30 %	4.8	70.1	20	1585.70	9.23 %	75.22 %	5.0	132.9
$B = 3$												
20	30	0.36	29.52 %	26.78 %	2.9	7.7	30	0.43	31.82 %	23.35 %	3.1	12.0
50	30	35.70	14.91 %	44.33 %	5.1	52.7	30	68.47	31.56 %	36.05 %	5.2	170.4
100	28	783.46	10.07 %	46.87 %	6.0	114.9	12	2751.37	14.23 %	77.24 %	6.3	353.2
$B = 4$												
20	30	0.41	30.34 %	24.27 %	3.0	7.2	30	0.39	29.69 %	22.12 %	3.1	9.6
50	30	46.22	15.95 %	44.79 %	6.3	75.2	27	127.39	45.89 %	28.82 %	6.6	209.6
100	26	1105.38	13.03 %	51.88 %	7.6	178.8	4	3333.44	22.98 %	63.90 %	7.5	584.0