New Formulations and Algorithms for the Kidney Exchange Problem[1]

L. Carolina Riascos-Álvarez
The University of Toronto
Graduate Program in Mechanical and Industrial Engineering
5 King's College Road
Toronto, ON, Canada M5S 3G8
carolina.riascos@mail.utoronto.ca

Roger Z. Ríos-Mercado
Universidad Autónoma de Nuevo León (UANL)
Graduate Program in Systems Engineering
Av. Pedro de Alba s/n, Cd. Universitaria
San Nicolás de los Garza, NL 66455, Mexico
roger.rios@uanl.edu.mx

Jonathan F. Bard
The University of Texas at Austin
Graduate Program in Operations Research and Industrial Engineering
204 E. Dean Keeton St. C2200
Austin, TX 78712, USA
jbard@utexas.edu

April 2019

[1]Technical report PISIS-2019-02, Graduate Program in Systems Engineering, UANL, San Nicolás de los Garza, NL, Mexico, April 2019

Abstract

Kidney-paired donation programs assist patients in need of a kidney to swap their incompatible donor with another incompatible patient-donor pair for a suitable kidney in return. The Kidney Exchange Problem (KEP) is a combinatorial optimization problem that consists of finding the maximum set of matches in a directed graph representing the pool of incompatible pairs. Depending on the specific framework, these matches can come in the form of (bounded) directed cycles or directed paths. This gives rise to a family of KEP models that have been studied over the past dozen years. Several require an exponential number of constraints to eliminate cycles and chains that exceed a given length. In this paper, we present enhancements to a subset of existing models that exploit the connectivity properties of the underlying graphs. The first is based on the cycle-only version of the KEP and the second is based on the cycle-and-chain version. An efficient algorithm for detecting violated constraints is developed.

To assess the value of our enhanced models, an extensive computational study was performed in which they were compared with existing formulations. The first observation is that the proposed cyclic-version of the KEP is able to quickly solve instances of realistic size for low to medium density graphs, outperforming the classical edge formulation. It was also observed that our second model proved superior to the existing cycle-and-chain formulations in two out of the three comparisons, and is competitive with the best approaches to date.

Keywords: Kidney exchange problem; integer programming; strongly connected components; healthcare.

1 Introduction

The kidney exchange problem (KEP) is a combinatorial optimization problem aimed at maximizing the number of patient-donor matches as represented by cycles or chains in a directed graph. The KEP has received increased attention over the past few years as several different versions have been implemented through kidney paired donation programs world-wide. The need for such a program in the U.S., for example, was outlined by Ross et al. [26] who cited four motivating factors: (a) the high demand for kidneys among the population, (b) the relatively low number of cadaveric (deceased) donors as compared to the demand, (c) growing wait lists, and (d) the very high cost of kidney-related treatments such as hemodialysis. Several researchers have addressed each of these factors from a clinical perspective [10]. Others have taken a prescriptive approach and developed optimization models to aid decision-makers. The first such models are due to Roth et al. [27, 28, 29]. Subsequently, there has been a spate of research, from developing models to handle different versions of the problem, to dealing with the uncertainty in demand, donor-patient matches, and the timing of surgeries.

Depending on the specific application, matches can come in the form of directed cycles or directed paths. A cyclic exchange is when a living donor, who is incompatible with the intended recipient (this is called an incompatible patient-donor pair, PDP), offers a kidney to another patient as long as the donor's intended recipient receives a compatible kidney from another person (see Figure 1). Such exchanges are known as k-way cyclic exchanges, where k is the number of incompatible PDPs involved in the cycle. In practice, due to obvious legal issues, the surgeries in cyclic exchanges are conducted simultaneously. This requirement greatly increases the need for available operating rooms and surgical teams at a specified time and date: $2k$ in each case for every k-way cyclic exchange. For example, a 3-way cycle involves the simultaneous coordination of six operating rooms and six surgical teams. Given this burden, cyclic exchanges with more than three patient-donor pairs are rarely conducted [3].

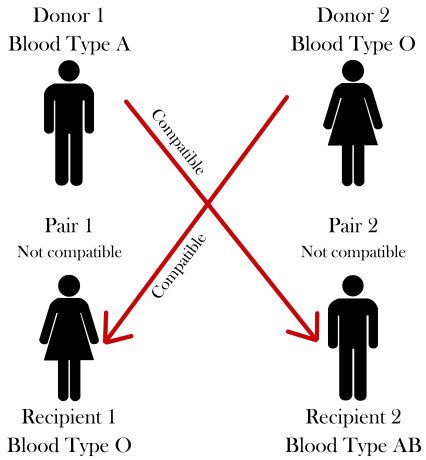

Figure 1: Two-way cycle

A path or chain exchange arises when a non-directed donor (NDD) (i.e., an altruistic person who decides to donate without having an intended recipient) offers a kidney to a patient associated with a particular PDP. The donor in this pair is then matched with a patient of another PDP and so forth, forming a chain with the NDD and l recipients (see Figure 2), where the last PDP in the chain donates to a patient on a waiting list. When a chain is initiated by an NDD, it can be structured so that no donor in a pair has to donate a kidney before the corresponding patient has received one, allowing the simultaneity requirement to be relaxed.

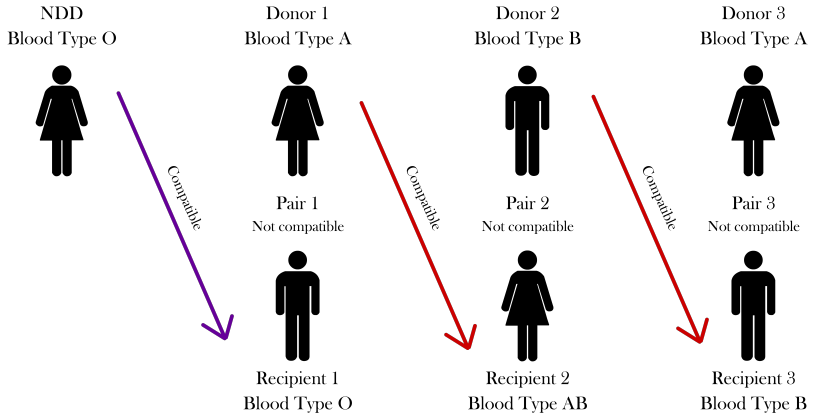

Figure 2: Length-3 chain

The maximum size of chains and cycles is determined by each kidney paired donation program. A debate around chains is whether or not they should be performed simultaneously. Under the *domino-paired donation* scheme [16], an NDD triggers a short simultaneous chain with the donor in the last pair donating to a candidate on the waiting list for a cadaveric organ. The other scheme is the *non-simultaneous extended altruistic donor* chain [4, 6, 11] under which the NDD initiates a theoretically never ending-chain consisting of several short segments, each carried out simultaneously. The last donor in each segment becomes a bridge donor who can initiate some time later the next chain segment, unless he drops out. When the solution can be in the form of unbounded chains or cycles, the KEP turns into the maximum weighted perfect matching problem on a bipartite graph [1], which can be solved in polynomial time. When only 2-cycle exchanges are allowed, the KEP is equivalent to the maximum matching problem, also solvable in polynomial time with Edmonds' maximum cardinality matching algorithm [14]. The general problem with k-way cyclic exchanges is known to be strongly NP-hard for $k \geq 3$ [1, 8].

Current work on solving the KEP is dominated by exact approaches based on integer programming (IP) formulations. Some IP models use a considerably large number of constraints often exponential in number. This limits the size of an instance that can be solved to optimality.

In this paper, we introduce three new models that extend several existing formulations. With the help of special network connectivity properties, we show that many of the current models can be significantly reduced in size without compromising their correctness. The improvements we realize

are based on detecting strong connectivity in the underlying graphs, especially for large instances. In particular, this is the case for the edge formulation proposed by Abraham et al. [1] and Roth et al. [31], which is impractical even for small instances [1, 9, 19] as it requires $\mathcal{O}(m^k)$ constraints for graphs with m edges.

One of the primary contributions of our work is the identification of a special graph structure that allows us to significantly reduce the number of constraints as well as the number of paths needed in the edge formulation of the KEP. In addition, we propose a new set of constraints to strengthen our enhanced formulations. A related contribution is the design and development of an efficient algorithm for applying the proposed reduction and building the corresponding models.

Testing was done over a wide range of instances found in the literature. The performance of our cycle-only and cycle-and-chain versions of the KEP is compared with the performance of the most relevant models previously developed. The results indicate that our version of the cyclic formulations is able to solve practical instances with low to medium density graphs in remarkably less time than the classical edge formulation while finding many more optimal solutions. For the cycle-and-chain formulations, the results show that our approach is superior with respect to two out of the three existing formulations we tested, and competitive with the formulation that provided the shortest runtimes and greatest number of optimal solutions.

The remainder of the paper is organized as follows. In Section 2, the general version of KEP is defined and some special cases are identified. This is followed by a literature review in Section 3. In Section 4, three new formulations are introduced that take advantage of certain network connectivity properties. The discussion includes a comparison of the linear programming relaxations of the corresponding models. In Section 6, we discuss the results from our computational experiments, which compare the performance of the new and existing formulations. Conclusions are drawn on the value of the research in Section 7. Appendix A contains the existing cycle-only and cycle-and-chain versions of the KEP and Appendix B highlights some implementation issues associated with the solution algorithms we developed for the KEP.

2 Problem Description

Given the list of NDDs and PDPs along with their compatibility information, it is possible to build a compatibility graph that depicts the potential matches between donors and patients. PDP nodes represent patient-donor pairs who are generally not biologically compatible, although compatible pairs may also participate with the expectation that the patient will receive a better match. NDD nodes, on the other hand, represent a bridge donor (i.e., a donor whose intended recipient has received a kidney, and therefore binding him as a donor for a future exchange) or an altruist donor. Both can trigger a chain.

In the graph, a directed edge or arc going from one node to another implies that the donor in the first node is compatible with the patient in the next node whether she belongs to an incompatible pair or is a non-directed donor. Thus, each arc represents a potential transplant and has an associated weight that is determined by a medical board to distinguish the priority given to that

candidate. For the chain only version of the KEP, the objective is to maximize the weighted sum of edges [3, 19]. When all weights have unitary value, the objective is to maximize the total number of exchanges [20]. In practice, the maximum cycle length k is established according to the capacity in the transplant center conducting the operations. Chains, however, may or may not be constrained [4, 6, 16]. When they are it is assumed that the maximum length l is also known in advance.

In the developments, let P be the set of patient-donor pairs and N be the set of non-directed donors. We model the KEP on a directed graph $G = (V, E)$ where the set of vertices $V = \{1, \ldots, |V|\}$ is partitioned into $P = \{1, \ldots, |P|\}$ and $N = \{|P|+1, \ldots, |P|+|N|\}$. In the absence of NDDs, as is the case with the cycle-only version, $V = P$. The set of directed edges (interchangeably referred to as *arcs*) E contains edge (i, j) if and only if the donor at node i is compatible with patient in pair j such that $E = \{(i, j) \mid i \in V, j \in P\}$. Note that $\{(i, j) \mid i \in V, j \in N\} = \emptyset$ since NDDs do not have paired patients, and therefore they do not have incoming edges. The digraph has no loops since we assume every PDP is incompatible. Each arc $(i, j) \in E$ has a weight $w_{ij} \in \mathbb{R}^+$, representing the priority given by the transplant center to that transplant. The weights are used to capture various prioritization schemes and other value judgments. There is a maximum cycle length limit given by k due to logistical issues. The largest chain length is constrained to l; however, l may be long or even unbounded.

When only cycles are being considered, the KEP can be modeled as the cycle packing problem on a directed graph [8]. Figure 3 depicts a compatibility graph and an optimal solution when cycles of length at most $k = 3$ are allowed and there are not NDDs in the pool. The optimal assignment is shown by the bold arcs, while the dashed edges represent original compatible arcs that are not part of the optimal solution. Likewise, Figure 4 depicts the KEP instance presented in Figure 3 with NDDs, and its optimal solution when different values of w_{ij} are present.

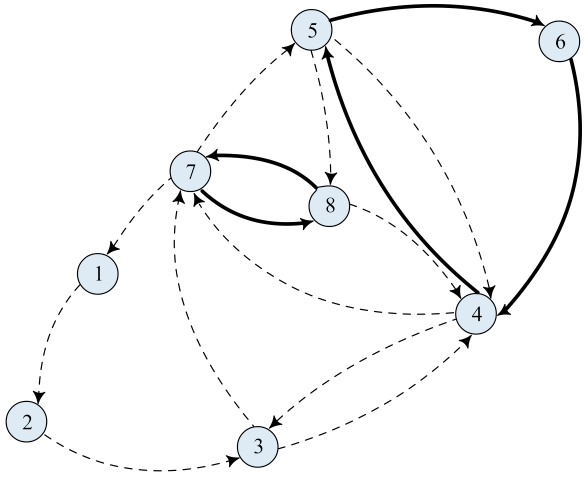

Figure 3: Compatiblity graph of cycle-only KEP with $|P| = 8$, $w_{ij} = 1$ and $k = 3$; optimal solution depicted by bold arcs

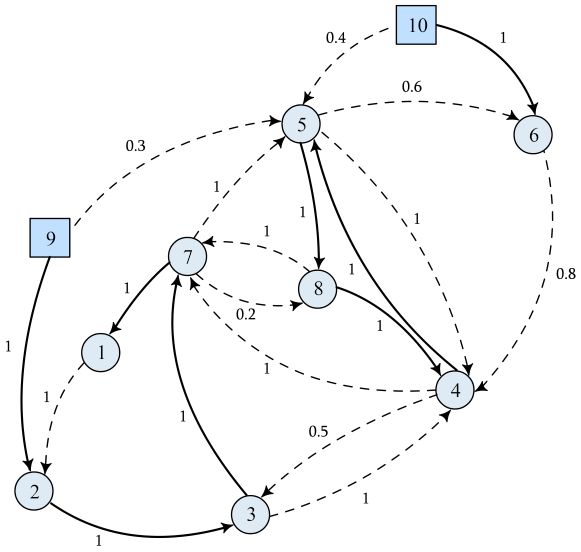

Figure 4: Compatibility graph of cycle-and-chain KEP with $|P| = 8$, $|N| = 2$, $k = 3$ and l unbounded; optimal solution is depicted in bold arcs

3 Literature Review

The concept of kidney exchange was due to Rapaport [23]. Later on, Ross et al. [26] and Ross and Woodle [25] discussed its ethical aspects. Roth et al. [27] first proposed organizing kidney exchanges on a large scale and applying operations research methods to solve the KEP (also see Roth et al. [28, 30]). Abraham et al. [1] and Roth et al. [31] introduced the first two IP models for the cycle-only version, which consisted of the cycle formulation and the edge formulation. The former includes one binary decision variable for each feasible cycle. The latter includes one binary decision variable for each arc in the graph for each compatible pair. In the cycle formulation, the number of constraints is polynomial in the size of the input, but the number of variables is exponential. In the edge formulation, the number of variables is polynomial while the number of constraints is exponential. Abraham et al. [1] reported experimental results with simulated test instances (as proposed by Saidman et al. [32]) with up to 10,000 incompatible pairs for the cycle formulation. A branch-and-price algorithm was used for the computations.

The initial models proposed for kidney exchange programs have exponential numbers of constraints or variables. Constantino et al. [9] presented the first two compact IP formulations for the cycles-only KEP that included the edge-assignment formulation and the extended edge formulation. A compact formulation is polynomially bounded in both the number of variables and number of constraints. For testing purposes, they randomly generated instances with correspondingly low, medium, and high density graphs, the largest with 1000 incompatible patient-donor pairs. The extended edge formulation was empirically effective in finding the optimal solution for high-density graphs and cycles involving more than 3 pairs. It was seen to dominate the edge-assignment formulation but was dominated by the cycle formulation.

Mak-Hau [19] introduced a compact formulation (denoted by EE-MTZ) that integrates chains and cycles by using the extended edge formulation to address cycles and a variant of the Miller-Tucker-Zemlin inequalities for the traveling salesman problem (TSP) to model chains. They also proposed an exponential version of the EE-MTZ that handles cycles the same way the edge formulation does. The largest instance tackled consisted of 256 PDPs and 6 NDDs.

Anderson et al [4] introduced two formulations for the KEP with bounded cycle lengths and unbounded chain lengths. The first is an edge-based model that uses binary variables to represent the selection of edges. This leads to an exponential number of constraints to prevent cycle lengths greater than k (cycle-cardinality constraints). The second uses an exponential number of binary variables to represent cycles but also requires an exponential number of constraints. For the first model, they presented a solution algorithm that relaxes the cycle-cardinality constraints (but not the integrality constraints) and iteratively adds them back as needed, solving an IP at each iteration. Their appoach could be called branch-and-bound-and-cut. For the second model, they implemented a branch-and-cut algorithm. Testing was done using randomly generated instances based on data from the National Kidney Registry and the Alliance for Paired Donation pool. The largest instances they tried to solve contained up to 700 PDPs and 175 NDDs.

Dickerson et al. [12] presented three IP formulations that combined the extended edge formulation with position-indexed variables used to eliminate subtours. They analyzed both real instances obtained from the United Network for Organ Sharing (UNOS) in the U.S. and the U.K. kidney exchange program (NLDKSS), and simulated data. On average, the UNOS instances contained 231 PDPs and 2 NDDs (UNOS runs the algorithm twice a week keeping the number of altruists small), and the NLDKSS instances 201 PDPs and 7 NDDs. The simulated data was based on historical UNOS data, and reflected the size of expected instances in the future, with up to 700 PDPs and 175 NDDs.

To avoid the need to keep the entire model in memory, several branch-and-price and branch-and-cut algorithms have been proposed. The fastest algorithms to date use column generation to find solutions to the cycle formulation of the KEP [1, 2, 12, 13, 17, 22]. The only approach that we are aware of that uses constraint generation is due to Anderson et al. [4] as mentioned. For a version of the edge formulation, they showed it to be effective for solving instances when the cycle-length limit is 3 and chain lengths are unbounded. The algorithm, however, is uncompetitive when compared to branch-and-price-based approaches where the chain size is constrained [22].

Alternative objectives to those of finding the maximum number of exchanges or the maximum weighted sum of all exchanges, include maximizing the expected number of transplants [2, 13, 21] and the lexicographic optimization of a hierarchy of objectives [17, 20]. Other variations of the KEP consider *multiple donors* associated with a single patient where at most one is permitted to be in the final solution [1]. Another version allows *compatible pairs* to participate in kidney paired donation programs as long as the patient in that pair receives a "better" match than his current donor [15]. The purpose of this requirement is to increase the potential number of matches in a final solution that otherwise could not be reached without the inclusion of compatible pairs.

The actual objective depends on the organization administering the exchange program. In Eu-

rope, most countries run these programs at the national level with matches being determined about once every three months. In the U.S., the situation differs sharply. There are multi-transplant-center exchanges, single-transplant-center exchanges, and the deceased donor waiting list. Many hospitals such as the Mass General in Boston, Methodist in San Antonio and Johns Hopkins in Baltimore, run their own programs and compete with each other. Nationwide exchanges include UNOS, the National Kidney Registry, and the Alliance for Paired Donations. Note that cadaver kidneys are viable for up to 24 hours, and so are not suitable for exchange programs.

One aspect of the KEP that has not been considered in the papers mentioned above is the evolution of the pool over time. There are versions, however, that include dynamic pools as well as compatibility-based preferences [35]. Finally, Awasthi and Sandholm [7] address the KEP as an online problem where patient-donor pairs and altruistic donors appear and expire as their circumstances change. For a more extensive survey on kidney exchange models, see the recent work by Dickerson et al. [13].

4 New Edge-Based Formulations

In this section, we introduce three new formulations taking as reference the edge formulation proposed by Abraham et al. [1] and Roth et al. [31]. See Appendix A.

4.1 Strong connectivity

From graph theory, we know that a graph G that is not a strongly connected component (SCC) by itself, can be partitioned into a collection of vertex-disjoint and arc-disjoint strongly connected subgraphs.

Definition 1. *A strongly connected component of a directed graph G is a maximal subset of vertices $S \subseteq V$ such that for every pair of vertices u and v in S, we have both $u \rightsquigarrow v$ and $v \rightsquigarrow u$; that is, there is a directed path from u to v and a directed path from v to u.*

One property of strong connectivity is that it partitions vertices so that each vertex belongs to exactly one SCC, as stated in the following lemma.

Lemma 1. *In a directed graph, each vertex belongs to exactly one SCC.*

Proof. By contradiction. Let us suppose that there is a vertex v belonging to two different strongly connected components in a directed graph $G = (V, E)$, namely, S and S'. Since $v \in S$, for each vertex $u \in S$, we have $u \rightsquigarrow v \rightsquigarrow u$. Similarly, since $v \in S'$, for each vertex $w \in S'$, we have $w \rightsquigarrow v \rightsquigarrow w$. It follows that $u \rightsquigarrow w$ and $w \rightsquigarrow v$ for every vertex $u \in S$ and $w \in S'$, that is, $S = S'$, which is a contradiction. Therefore, vertex v belongs to exactly one SCC. ■

Note that if there exists arcs leaving one SCC S and entering another distinct $S^{'}$, such arcs will never be part of a cycle since G does not contain a path from $S^{'}$ to S that returns to $S^{'}$; otherwise $S = S^{'}$. Figure 5 depicts a graph with 4 SCCs, 2 of which can actually form a cycle.

In fact, the only component that represents an optimization problem is the one containing nodes {1,2,3,4}, since nodes 6 and 7 clearly form part of the optimal solution for $k \geq 2$. Therefore, arcs (3,6), (4,5), (6,7), (7,6), (8,5), and (8,7) can be disregarded from any edge-based model without affecting optimality.

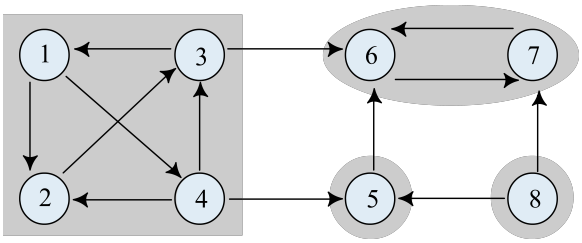

Figure 5: Example of SCCs

There are several classical linear-time algorithms to find strongly connected components of a digraph, for example, Tarjan's algorithm [34] and Kosaraju-Sharir's algorithm [33]. Depending on the graph representation, both run in $\mathcal{O}(|V| + |E|)$ time.

4.2 Partitioned edge formulation (*PE)

This formulation addresses the cycle version of the KEP. Let Q be the set of subgraphs induced by the SCCs of $G = (V, E)$ and let q be the number of non-trivial SCCs (i.e, subgraphs containing nodes with degree greater than zero). Let Π be the set of all length-k paths in a graph. Each of these paths is formed by $k + 1$ adjacent nodes or k edges, with no repetition of nodes. In the context of KEP, a chain is also a path but we reserve the word 'chain' to refer to the final solution delivered by an algorithm.

Accordingly, we have q optimization subproblems, one for each SCC, where $Q = \{Q_1, ..., Q_h, ...Q_q\}$. Also, let $G_h = (V_h, E_h)$ be the h-th subgraph in Q and Λ_h be the full set of length-k paths in the h^{th} SCC. Now, for each arc $(i, j) \in E_h$ we define a variable x_{ij} such that

$$x_{ij} = \begin{cases} 1 & \text{if the donor in pair } i \text{ gives a kidney to the patient in pair } j \\ 0 & \text{otherwise} \end{cases}$$

The partitioned edge formulation (*PE) can be expressed as follows.

$$\text{Maximize} \quad \sum_{(i,j)\in E_h} w_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j:(i,j)\in E_h} x_{ij} = \sum_{j:(j,i)\in E_h} x_{ji} \qquad i \in V_h \tag{2}$$

$$\sum_{j:(i,j)\in E_h} x_{ij} \leq 1 \qquad i \in V_h \tag{3}$$

$$\sum_{1 \leq p \leq k} x_{i_p i_{p+1}} \leq k-1 \qquad (i_1, ..., i_k, i_{k+1}) \in \Lambda_h \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in E_h \tag{5}$$

The objective function (1) maximizes the weighted sum of matches. In case of unit weights, it maximizes the total number of transplants. Constraints (2) guarantee that donor i provides a kidney if and only if patient i receives one in return. Constraints (3) ensure that a person can only donate a single kidney, and constraints (4) enforce the length of any cycle to be less than or equal to k. Any feasible cycle will always contain a path with at most $k-1$ edges (no repeating edges). Therefore, if we preclude all length-k paths from being part of a cycle, cycles of length $k+1$ or greater are excluded from a feasible solution as well.

This model looks like the edge formulation (20)-(24), referred to as formulation E in Appendix A, yet has significant differences. It is applied independently to each SCC in Q so optimizing over G is reached by optimizing over each subgraph G_h. Note that Λ_h here represents a subset of the k-length paths in Π in the edge formulation, and that the upper bound provided by the linear relaxation of (1) - (5) is at least as good as that obtained by the E formulation (as shown in Section 4.5) This follows because the only non-redundant length-k paths are contained in Λ_h. In the edge formulation set Π can be replaced by $\bigcup_{h=1}^{q} \Lambda_h$ without altering the correctness of the model.

4.3 Partitioned and reduced edge formulation (*PRE)

This formulation represents the cycle-only version of the KEP. The partitioned edge formulation significantly reduces the number of constraints, but when the graph G is itself a SCC, it turns into the edge formulation. In this section, we propose a new formulation for general graphs.

SCC-based search for length-k paths

The first important observation is that the number of paths of length k present under the edge or reduced edge formulation is larger than the number of infeasible cycles to be ruled out. For instance, assuming $k = 3$, for an infeasible cycle of length 4 formed by nodes $C = (v_1, v_2, v_3, v_4)$, there are four different directed paths of length 3 in Π, each starting at a different node in cycle C. Evidently, three of them are redundant and therefore not needed in the formulation. Under this reasoning, each infeasible cycle accounts for as many path constraints as the number of nodes in it.

Given the size this example one can easily identify the redundant paths, but this is not the case in a large graph. The essence of our approach is to provide a clever way of identifying redundant paths and keeping only those that are needed for ruling out infeasible cycles.

Algorithm 1 formally defines the process just described. The algorithm takes as input the directed graph G, decompose it into each of its SCCs and finds Ω_h, the set of length-k paths in SCC h, where $h = 1, \ldots, q$. Note that Λ_h is the full set of paths of length k for component h, whereas Ω_h is a subset of Λ_h such that the cardinality constraints over the cycles are satisfied.

Let $StrCC(G)$ be a procedure that finds the non-trivial SCCs in a directed graph G. We used

the Kosaraju-Sharir algorithm [33]. Let *Pool* be the set of connected components from which the next strongest connected component T is chosen for decomposition. Every time the SCCs are recomputed for a given component T, a new set of SCCs T' is obtained. Let A^u be the set of edges that are incident to u.

Additionally, let $choose(T, g)$ be a procedure for finding a node in T to be removed depending on the node selection strategy given by "g", where $g \in$ {in-degree, out-degree, total degree}; e.g., $choose(T, out-degree)$ returns the node in T with highest out-degree. Ties are broken arbitrarily. Let $DepthFirstSearch(k, u, T)$ be the well-known algorithm for traversing graph data structures that starts at node u and traverses T in the search for paths of length k. Thus, it returns the set of length-k paths starting at node u. Also, let $Pool\_max(Pool)$ be a function that returns the subgraph $T = (\bar{V}, \bar{E})$ in *Pool* with the largest cardinality node set.

Algorithm 1 SCC-based search for finding length-k paths in a KEP graph.

Inupt: $G = (V, E), k \in \mathbb{N} \geq 2$

Output: $\Omega_h \equiv$ set of length-k paths found in each SCC, Q_h

$Q = \{Q_1, \ldots, Q_q\} \leftarrow StrCC(G)$
$\Omega_h \leftarrow \emptyset$
for $h = 1$ to q **do**
    $Pool \leftarrow \{Q_h\}$
    $T = (\bar{V}, \bar{E}) \leftarrow Pool\_max(Pool)$
    while $(|\bar{V}| \geq k + 1)$ **do**
        $Pool \leftarrow Pool \setminus T$
        $u \leftarrow choose(T, g)$
        $\Omega_h \leftarrow \Omega_h \cup DepthFirstSearch(k, u, T)$
        $\bar{V} \leftarrow V \setminus \{u\}$
        $\bar{E} \leftarrow E \setminus A^u$
        $T' \leftarrow StrCC(T)$
        $Pool \leftarrow Pool \cup T'$
        $T = (\bar{V}, \bar{E}) \leftarrow Pool\_max(Pool)$
    end while
    return Ω_h
end for

To illustrate the algorithm and its underlying concepts consider the graph in Figure 3 with $k = 3$, that is, with only cycles of length k or less allowed. In edge-based models one way of eliminating cycles of length $k + 1$ or more is to use path-based cycle-elimination constraints as in (4). There, the authors establish that if a k-length path can only use $k-1$ edges, then there cannot be cycles of length $k+1$ or more because adding an edge to the path to form a cycle would increase its length from $k - 1$ to k. Such path-based cycle elimination constraints have been used before [1, 31].

In our example, there are nine infeasible cycles (of length 4 or more) given by $\{c_1 = (1, 2, 3, 7, 1), c_2 = (1, 2, 3, 4, 7, 1), c_3 = (1, 2, 3, 4, 5, 8, 7, 1, 4), c_4 = (3, 7, 5, 4, 3), c_5 = (3, 7, 8, 4, 3), c_6 = (3, 7, 5, 6, 4, 3), c_7 = (3, 7, 5, 8, 4, 3), c_8 = (4, 7, 5, 6, 4), c_9 = (4, 7, 5, 8, 4)\}$. In terms of using k-length paths for cycle elimination, Table 1 displays the full set of length-3 paths. We now define two different types of k-length paths: (i) A *critical* k-length path is a path that could lead to a cycle of length $k + 1$ if an arc is added, and (ii) a *non-critical* or *redundant* k-length path is the opposite, that is, one where no cycle of length $k + 1$ or more can be formed by adding an arc to it. For instance, path $4 \to 3 \to 7 \to 5$ from Table 1 is a critical path because if arc $(5, 4)$ is added a cycle of length 4 is formed. In contrast, path $4 \to 3 \to 7 \to 1$ is non-critical because no cycle of length 4 can be formed from it by adding any arc.

The importance of distinguishing these type of paths is that to prevent cycles of length $k + 1$ or more, path-based cycle elimination constraints should be imposed over critical paths only. In other words, path-based cycle elimination constraints imposed on non-critical paths are redundant. In Table 1, the critical paths are shown with a star (*) at the end. Naturally, we do not know in advance what the critical and redundant paths are, but the idea is to have or generate as few path-based cycle elimination constraints as possible by keeping all the constraints associated to critical paths and using as few constraints associated to redundant paths as possible.

Now, if we choose an arbitrary node, say, node 4, and identify all length-k constraints starting at this node (this would eliminate all infeasible cycles associated with node 4), then any infeasible cycle will not contain node 4. Therefore, we could remove node 4 from the graph and apply the same reasoning to the remaining graph. By pursuing this strategy in an iterative way, one node at a time, we end up generating constraints that prevent all infeasible cycles. This is precisely the main idea of the algorithm.

Table 1: Full set of length-3 paths for Figure 3

$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$	$i = 8$
1→2→3→4	2→3→4→5	3→4→5→6	4→3→7→1	5→4→3→7*	6→4→3→7	7→1→2→3*	8→4→3→7*
1→2→3→7*	2→3→4→7	3→4→5→8	4→3→7→5*	5→4→7→1	6→4→5→8	7→5→4→3*	8→4→5→6
	2→3→7→1*	3→4→7→1	4→3→7→8	5→4→7→8	6→4→7→1	7→5→6→4*	8→4→7→1
	2→3→7→5	3→4→7→5	4→5→8→7	5→6→4→3	6→4→7→5*	7→5→8→4*	8→4→7→5*
	2→3→7→8	3→4→7→8	4→7→1→2	5→6→4→7*	6→4→7→8	7→8→4→3*	8→7→1→2
		3→7→1→2*	4→7→5→6*	5→8→4→3		7→8→4→5	8→7→5→4*
		3→7→5→4	4→7→5→8*	5→8→4→7*			8→7→5→6
		3→7→5→6		5→8→7→1			
		3→7→5→8					
		3→7→8→4					

Figure 6 shows the steps of the algorithm for this particular example. In quadrant I, node 4 is arbitrarily chosen giving rise to 7 length-3 paths (identified by different markers on the arcs). In quadrant II, after removing node 4, two new SCCs remain (shaded areas). One of them is a trivial single-node SCC and the other is s new subgraph. Quadrant III shows the next iteration of the algorithm where an arbitrary node is chosen, node 7 in this case, and its length-3 paths are generated (only one path in this case). In quadrant IV, after removing node 7, SCCs are

recomputed, but this time we obtain 6 trivial components (one node each) because the nodes are not mutually reachable from each other as stated in Definition 1.

At this point, the process ends because no more strongly connected components with more than k nodes remain in the graph. Thus, we have found 8 length-3 paths, as identified in Table 2. Observe that the selected nodes along with the ones contained in the remaining non-trivial strong components (with at most k nodes), are included in any collection of feasible cycles in the graph. As such, the process not only obtains a significant reduction in the number of length-k paths but also finds an upper bound on the number of cycles in a solution as well as the nodes that are necessarily included in them.

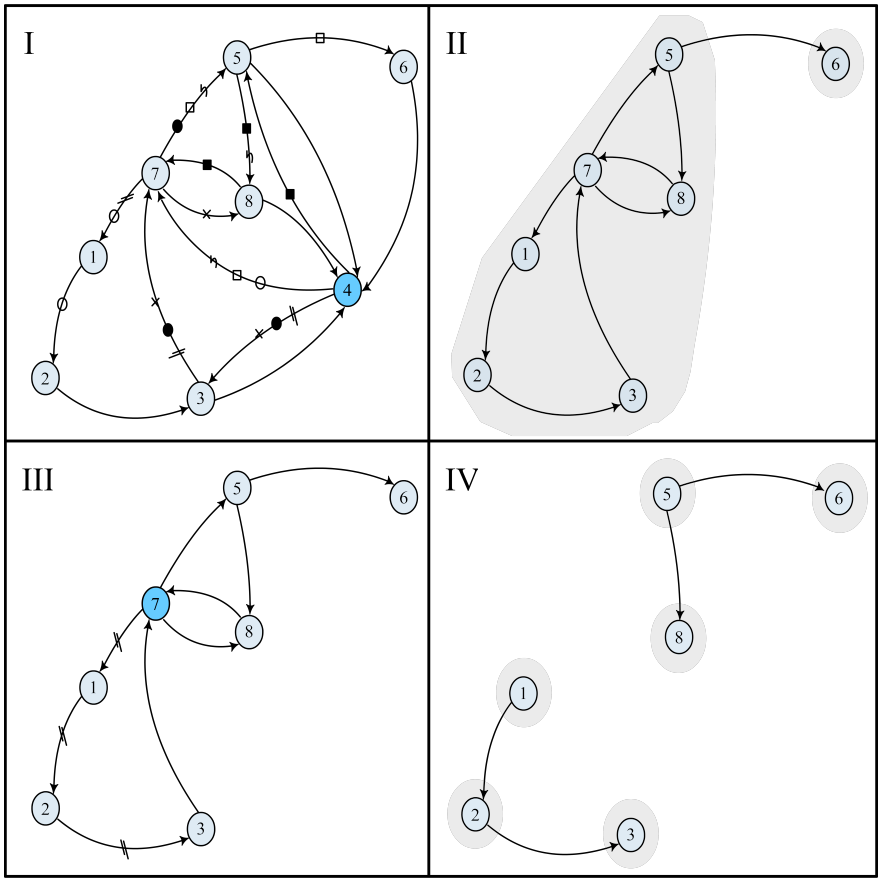

Figure 6: Sequentially removing a vertex from a strongly connected component

As shown in this example, one important aspect of the iterative process is to choose the node u to be removed. In this regard, three different node-selection rules were tested; namely, choose the node with (a) maximum in-degree, (b) maximum out-degree, and (c) maximum total degree. It was found that, in general, the maximum in-degree rule produced the best results; that is, a larger number of paths were eliminated under this rule when compared to the others. This can be explained in part by noting that as the number of outgoing edges from the selected node u increases so does the number of paths, at least empirically. When picking the node with highest in-degree

Table 2: Set of length-3 paths using the SCC-based search for Figure 3

$i = 4$	$i = 7$
4→3→7→1	7→1→2→3
4→3→7→5	
4→3→7→8	
4→7→1→2	
4→7→5→6	
4→7→5→8	
4→5→8→7	

instead, the number of outgoing edges is generally smaller, and therefore fewer paths are obtained. Thus, a large number of cycles is discarded by using a smaller number of paths.

New set of constraints

The length-k paths represent the left-hand side of the cardinality constraints for both the edge formulation (23) in Appendix A and the partitioned edge formulation (4). In this section, we strengthen these constraints giving our next contribution. Consider the following length-k path and its associated constraint.

$$x_{i_1 i_2} + x_{i_2 i_3} + ... + x_{i_k i_{k+1}} \leq k - 1 \tag{6}$$

The well-known idea behind constraint (6) is to break a path with $k + 1$ nodes (k edges) such that at most $k - 1$ of its edges are used to form a valid cycle. Otherwise, the path with k edges would lead to an infeasible cycle. By using one or more edges not included in (6), alongside these $k - 1$ edges, cycles with cardinality less or equal to k can be obtained. Note that a subset of the nodes in path $x_{i_1 i_2}, x_{i_2 i_3}, ..., x_{i_k i_{k+1}}$ can form several sub-paths if the corresponding edges exist in the original graph, i.e., paths like $x_{i_2 i_5}, x_{i_5 i_6}, ..., x_{i_k i_{k+1}}$ or $x_{i_1 i_{k+1}}$ may also exist. Also observe that the flow-balance constraints (9) or (21) will allow only one such path or set of sub-paths. Because all of them will collectively contain at most $k - 1$ edges, the following set of constraints is valid for the KEP.

$$\sum_{1 \leq p \leq k} x_{i_p i_{p+1}} + \sum_{1 \leq p \leq k-1} \sum_{e \leq b \leq k+1} x_{i_p i_b} \leq k - 1 \qquad (i_1, ..., i_k, i_{k+1}) \in \Lambda_h \tag{7}$$

where $e = p + 2$ and $k \geq 3$.

Only sub-paths flowing in one direction, e.g., from lower to higher node indices need be taken into account to guarantee that the left-hand side of (7) only yields paths. We now replace constraints (4) by the tighter set given in (7). To understand the value of these inequalities, consider again the graph in Figure 3. In path $4 \rightarrow 7 \rightarrow 5 \rightarrow 6$, there exists a sub-path $4 \rightarrow 5 \rightarrow 6$. If $k = 3$, the

new associated constraint is $x_{47} + x_{75} + x_{56} + x_{45} \leq 2$. Now, consider the fractional solution $x_{47} = x_{45} = x_{75} = 0.5$ and $x_{56} = x_{64} = 1$, which is feasible to constraints (4), namely, $x_{47} + x_{75} + x_{56} \leq 2$, but not to constraints (7), which cut off this fractional solution and rule out its infeasible cycle 4 $\rightarrow$ 7 $\rightarrow$5 $\rightarrow$ 6 $\rightarrow$ 4.

The partitioned and reduced edge (*PRE) formulation is then given as follows.

$$\text{Maximize} \quad \sum_{(i,j)\in E_h} w_{ij}x_{ij} \tag{8}$$

$$\text{subject to} \quad \sum_{j:(i,j)\in E_h} x_{ij} = \sum_{j:(j,i)\in E_h} x_{ji} \qquad i \in V_h \tag{9}$$

$$\sum_{j:(i,j)\in E_h} x_{ij} \leq 1 \qquad i \in V_h \tag{10}$$

$$\sum_{1\leq p\leq k} x_{i_p i_{p+1}} + \sum_{1\leq p\leq k-1} \sum_{e\leq b\leq k+1} x_{i_p i_b} \leq k-1 \qquad (i_1, ..., i_k, i_{k+1}) \in \Lambda_h \tag{11}$$

$$x_{ij} \in \{0, 1\} \qquad (i,j) \in E_h \tag{12}$$

4.4 The reduced exponential-sized SPLIT formulation (*ReSPLIT)

This formulation addresses the cycle-and-chain version of the KEP. A natural extension of the exponential-size SPLIT formulation given in Section A.2.4 is to replace constraints (61) with constraints (11). By using Algorithm (1), we aim to find a small subset of k-paths instead of the full set, while maintaining model correctness. When the graph is partitioned into its SCCs some edges are removed, losing feasible paths and perhaps the optimal solution. Therefore, we cannot solve each SCC separately because we now have to find a collection of node and arc disjoint paths.

The reduced exponential-sized SPLIT formulation (*ReSPLIT) is as follows:

$$\text{Maximize} \quad \sum_{(i,j)\in E'} w_{ij}x_{ij} + \sum_{(i,j)\in E} w_{ij}z_{ij} \tag{13}$$

$$\text{subject to} \quad (49) - (50), \quad (53) - (56), \quad (62) - (63) \tag{14}$$

$$\text{replace } x_{ij} \text{ with } z_{ij} \text{ in (11)} \tag{15}$$

$$x_{ij}, z_{ij} \in \{0, 1\} \qquad (i,j) \in E \tag{16}$$

4.5 Model Properties

Given two formulations F_1 and F_2 with the same objective function for an optimization problem, we say that F_1 dominates F_2 if the feasible region of F_1 is a subset of the feasible region of F_2. The strength of the linear programming (LP) relaxation is used as a performance measure in LP-based solution methods such as branch and bound. In this section, we compare the upper bound provided by the LP relaxations of the cycle-version models given in Sections A.1, 4.2, and 4.3.

Lemma 2. *The cycle formulation (17)-(19) dominates both the partitioned edge formulation (1)-(5) and the partitioned and reduced edge formulation (8)-(12).*

Lemma 3. *The cycle formulation (17)-(19) dominates the extended edge formulation (25)-(31)*

Lemma 4. *No domination relationships exist among the extended edge formulation (25) - (31), the edge formulation (20)-(24), the partitioned edge formulation (1)-(5) and the partitioned and reduced edge formulation (8)-(12).*

The same proof given with regard to the cycle and edge formulation in [1] can be used to prove Lemma 2. The proofs for Lemmas 3 and 4 are given by Constantino et al. [9]. Although neither the partitioned edge formulation nor the partitioned and reduced edge formulation are compared explicitly in [9], the same reasoning applies as used in the other comparisons.

5 Solution Algorithms

All the existing and proposed formulations presented in Appendix A and the previous sections were solved with CPLEX. However, there are some variations in the solution algorithms due to the exponential number of cardinality constraints in some models. In this section, we provide details about the solution algorithms for the new formulations. For the existing formulations, we refer the reader to Appendix B.

5.1 The *PE formulation

The solution algorithm for the *PE formulation consists of two steps. First, the strongly connected components with associated k-lenght paths are found, that is, subsets Λ_h for each $h = 1, \ldots, q$ are identified. Then, the corresponding path-elimination constraints (4) are generated. The resulting *PE model (1)-(5) is fed to the solver to obtain the optimal solution. There are several algorithms in the literature for finding strong components and simple paths, several of which were mentioned in Section 4.3. In this study, whenever it was necessary to find the strongly connected components and the full set of length-k paths, we used the Kosaraju-Sharir algorithm [33] and a depth-first search procedure, respectively.

5.2 The *PRE and *ReSPLIT formulations

After finding the cardinality constraints with Algorithm 1, the next step is to solve both the *PRE and the *ReSPLIT models. For the former, there are as many subproblems as the number of strongly connected components of $G = (V, E)$, just as in the case of the *PE formulation. For the latter, the entire instance is optimized using the length-k paths contained in Ω. Since the *ReSPLIT formulation finds cycles and chains, it is not possible to solve each subproblem separately.

Before optimizing either model, we preprocess the graphs removing all length-k paths not leading to a feasible cycle. In Section 4.3, we showed that some paths in a strongly connected component may be part of a circuit and so there is no need to constraint them. Moreover, even if a path

forms a simple cycle, it may be infeasible. If an arc belonging to a path in Ω_h does not lead to a feasible cycle, i.e., it is only part of a circuit or infeasible simple cycle, we remove the arc (this is done for the cycle version only) and the path from Q_h and Ω_h, respectively. Finally, we recompute the strongly connected components of Q_h. If after doing so, we obtain new separate components, we will have as many subproblems as the number of the new strong components instead of only one represented by Q_h. Recall from Section 4.2, that Q_h, $h = 1, \ldots, q$, are the original strongly connected components of graph G. This increase in the number of subproblems resulting from the decomposition is especially useful for hard-to-solve instances.

Now, we formally introduce the procedure outlined above. Let $G_h = (V_h, E_h)$ be the subgraph represented by Q_h and let ω^h be a path in Ω_h. In addition, let $Q^{'}$ represent the set of new components after removing redundant edges from Q_h and let $\Omega^{'}$ represent the updated set of length-k paths after removing non-necessary paths. The procedure is shown in Algorithm 2 and has complexity $O(|V|^{k+1})$ [24]. After updating the number of subproblems and paths, we proceed to solve both formulations.

Algorithm 2 Removing unnecessary edges and paths in subgraph G_h

Inupt: $G_h = (V_h, E_h), \Omega_h, k \geq 2$

Output: $Q^{'}$ = set of new strong components obtained after decomposing Q_h, $\Omega^{'}$ = set of length-k paths in $Q^{'}$

```
for ω^h ∈ Ω_h do
    for e ∈ ω^h do
        if e ∉ C_h then
            Ω' ← Ω' \ ω^h
            E_h ← E_h \ e
        end if
    end for
end for
Q' ← StrCC(G_h = (V_h, E_h))
return Ω', Q'
```

6 Computational Experiments

The instances used to test our new formulations were taken from Anderson et al. [4] who simulated the National Kidney Registry (NKR) pool over a two year period from May 24, 2010 to May 24, 2012. The initial pool contained 63 patient-donor pairs while an additional 410 pairs registered over the course of their study. The dataset also contained 75 altruistic donors. Compatibility between donors and patients was determined primarily by blood type and human leukocyte antigen compatibility rules, although some patient preferences were also taken into account.

To create representative snapshots of actual instances encountered by a kidney-paired donation program they considered the fact that easy-to-match patients tend to wait only a short time before

being selected, thus leaving the hard-to-match patients in the pool after each match run. Overall, the graph density of these instances is low, being less than or equal to 20%. For full details, see Anderson et al. [4] and Anderson [5].

In the remainder of this section, we discuss our computational experiments that were designed to compare existing formulations with our new formulations with respect to runtime and solution quality (when an optimal solution was not found). All models were either solved directly with CPLEX 12.7 using Concert Technology or by a specialized method (as described in Appendix B) on a PC with an Intel Core i7 processor running at 3.40 GHz, and with 8 Gb of RAM. In general, all models were coded using C++ and a time limit of 1800 seconds was placed on all runs.

The computational analysis was performed as follows. First, 10 instances were grouped into each trial set, sorted in non-decreasing order of the average number of PDPs for the cycle-variant formulations and the cycle-and-chain formulations. The maximum cycle lengths considered were $k = \{3, 4\}$, and no bound was placed on the length of chains. The original instances include NDDs so to test the cycle-variant formulations we dropped nodes and edges associated with them. All cycle-variant formulations were tested for all instances. For the cycle-and-chain formulations, we first tested all instances with $k = 3$ and $l = \infty$ and then took the two best performers and compared them for $k = 4$ and $l = \infty$. The objective function for all models was the weighted sum of edges.

Notation

Table 3 lists each formulation, its abbreviation, the reference, and the KEP variant. The IP formulations in the literature that include chains and cycles, in general, do not include upper bounds on chain lengths. Similarly, we did not consider unbounded chains. In the table the argument "i" in *PRE(i) and *ReSPLIT(i) refers to the maximum in-degree node-selection strategy used by the SCC-based search algorithm to find the length-k paths for each formulation. Since the maximum out-degree and maximum degree strategies yielded a larger number of paths, we only present results for the in-degree node-selection strategy discussed in Section 4.3.

Table 3: Notation used to reference KEP formulations

Formulation name	Notation	Authors	KEP variant
Cycle formulation	C	Abraham et al. [1] and Roth et al. [31]	C
Edge formulation	E	Abraham et al. [1] and Roth et al. [31]	C
Reduced extended edge formulation	rEE	Constantino et al. [9]	C
Partitioned edge formulation	*PE	This paper	C
Partitioned and reduced edge formulation	*PRE(i)	This paper	C
Anderson arc-based formulation	AA	Anderson et al. [4]	C&C
PC-TSP-based formulation	PC-TSP	Anderson et al. [4]	C&C
Polynomial-sized SPLIT formulation	pSPLIT	Mak-Hau [19]	C&C
Exponential-sized SPLIT formulation	eSPLIT	Mak-Hau [19]	C&C
Reduced Exponential-sized SPLIT formulation	*ReSPLIT(i)	This paper	C&C
KEP variant: (C) Cycles, (C&C) Cycles and chains			

When presenting the output statistics, we make use of the following notation.

- PDPsR: Range for the number of PDPs after grouping 10 instances in non-decreasing order

of PDPs. This value is represented by an interval [b, u], written as b-u in the output tables, where b (u) represents the smallest (largest) number of PDPs found in the specific subset.

- NDDsR: Interval for the number of NDDs after grouping the instances into sets of 10 in non-decreasing order of PDPs. Again, this value is represented by an interval b-u, where b (u) represents the smallest (largest) number of NDDs found in the specific subset.
- *PRE(i): Partitioned and reduced edge formulation when the node selection strategy for the SCC-based search algorithm is the maximum in-degree.
- nf: Number of subproblems that failed to obtain a feasible solution, either because CPLEX was unable to solve the initial LP relaxation after the time limit or because CPLEX displayed an out-of-memory status before starting branching. Notation (s_1/s_2) indicates that there were s_1 and s_2 instances in which no feasible solutions were found because of the former and latter cases, respectively.
- aVars: Average number of variables in a formulation for a subset of instances.
- saVars: Relative decrease in the number of variables passing from the edge formulation to the *PE or *PRE(i) formulations. The former is given by

$$\frac{1}{n}\sum_{j=1}^{n}\frac{nVarsE(j) - nVarsX(j)}{nVarsE(j)} \times 100,$$

  where n stands for the total number of instances, e.g., $n = 10$, and $nVarsE(j)$ ($nVarsX(j)$) is the number of variables for instance j obtained by using the E (*PE or *PRE(i)) formulation.
- aCons: Average number of constraints for a set of instances. In the case of the AA and the PC-TSP formulations the violated lazy constraints (see Appendix B) added throughout the solution process are included.
- saCons: Relative decrease in the number of constraints passing from the edge formulation to the *PE or *PRE(i) formulations. The former is given by

$$\frac{1}{n}\sum_{j=1}^{n}\frac{nConsE(j) - nConsX(j)}{nConsE(j)} \times 100,$$

  where $nConsE(j)$ ($nConsX(j)$) is the number of constraints for instance j obtained by using the E (*PE or *PRE(i)) formulation.
- nSCC: Average number of strongly connected components (subproblems) in which the graphs of each set of instances are split into their separate components
- SCCp(i): Average number of constraints for a set of instances of the *PRE(i) formulation before preprocessing (discussed in Section 5.2), and using the maximum indegree strategy in Algorithm 1.

- sSCCp(i): Relative decrease in the number of constraints passing from the edge formulation to the *PRE(i) formulation before preprocessing given by the expression

$$\frac{1}{n}\sum_{j=1}^{n}\frac{nConsE(j) - nConsPREb(j)}{nConsE(j)} \times 100,$$

  where $nConsPREb(j)$ is the number of constraints for instance j obtained by using the *PRE(i) formulation before preprocessing.

- tc and tp: Time to find feasible cycles in order to determine the variables needed in the cycle formulation, and the time to find length-k paths in each formulation, respectively. To search for feasible cycles, we implemented an adaptation of Johnson's algorithm [18].

- tsep: Average time needed to find and add all the violated lazy constraints to the AA and PC-TSP formulations.

- tr: Average time needed to perform the preprocessing step (discussed in Section 5.2).

- time: Average runtime in seconds to reach the best feasible solution for a set of instances. The runtime limit was set to 1800 seconds for all formulations. Note that the *PE and PRE formulations split the original problem into independent subproblems. Thus, the times reported for those formulations are the sum of times for all subproblems.

- opt: Number of instances solved to optimality. In all cases, the objective function is the weighted edge sum. Whenever this column does not appear in the output tables, it is because all instances were solved to optimality.

- gap: Average relative optimality gap associated with a formulation, defined by $\frac{(UB - LB)}{LB} \times 100$, where UB is the upper bound provided by the linear relaxation of the formulation and LB is either the best lower bound found or the optimal value when known. This column does not appear when all instances were solved to optimality.

The solution algorithms for the proposed formulations are discussed in Section 5. Discussion on how existing formulations were solved is given in Appendix B.

6.1 Assessment of cycle-variant formulations

In this subsection, results for cycle-only formulations are presented. Table 4 highlights the comparisons for 3-way cycles, that is, for $k = 3$. Each row represents average findings for a subset of instances (as indicated in the first column). Although most instances turned out to be "easy" for all the formulations tested, the C, rEE and *PRE(i) formulations show dominance over the E and *PE formulations, since they were able to optimally solve all instances in less time on average. Summing the time for preprocessing, searching for paths, and then finding the solution, the *PRE(i) formulation is comparable to the performance of the rEE formulation for most sets of

instances. The increase in runtimes for the first and last set of instances is related to the number of length-3 paths in some subproblems. In contrast, in any set of instances there are graphs that can be partitioned into several subgraphs, giving an extra advantage to the *PE formulation over the E formulation. Also note that the preprocessing procedure for the *PRE(i) formulation is not computationally expensive. The E formulation performed fairly well, but memory ran out in two instances in the last data set.

In Table 5 we show the results for 4-way cyclic exchanges. The dominance of the C, rEE and *PRE(i) formulations is clear in terms of runtime and number of optimal solutions found in comparison to the E and PE formulations. In the last set of instances, however, the *PRE(i) formulation did not perform as well as the C and rEE formulations with respect to runtimes, although they were still reasonable.

In Tables 6 and 7 the input size of instances is analyzed for all formulations, as well as the percentage reduction in the number of variables and constraints for the *PE and *PRE(i) formulations before and after the reduction procedure given in Section 5. The first observation is that the input size of the C formulation is the smallest for all sets of instances, which explains its outstanding performance. The second is the huge reduction in the number of variables and constraints for the *PE and *PRE(i) formulations. This was especially true for the latter, reaching savings of more than 85% and 99% in the number of variables and paths, respectively. These results demonstrate the benefits of using our approach for reducing the number of length-k paths for an edge-based formulation when k = 3 and 4. For the instances tested, the number of paths of length k seem to increase as k increases, and in some sets of instances as the number of PDPs increase as well.

It is worth noting, however, that the SSC-based search algorithm was particularly successful because the compatibility graphs of the kidney exchange pool were sparse. If the graph is complete and has n nodes, the time complexity of our algorithm is $\mathcal{O}(n^{k+1})$. For the instances tested, such bound was significantly loose, since the number of paths of length k seem to increase linearly with n for $k = 3$ (column SCCp in Table 6) and with n^2 for $k = 4$ (column SCCp in Table 7).

Table 4: Comparison of formulations for 3-way cyclic exchanges

	C		E					rEE		PE						PRE(i)				
PDPsR	tc	time	tp	time	opt	gap	nf	tr	time	nSCC	tp	time	opt	gap	nf	nSCC	tp	tr	time	
114-216	0.02	0.03	1.57	180.86	9	0.57	0/0	0.06	0.24	1.20	0.52	180.21	9	0.01	0/0	1.30	0.05	0.04	32.38	
221-236	0.03	0.03	2.59	21.34	10	0.00	0/0	0.12	0.07	1.50	0.13	1.60	10	0.00	0/0	1.50	0.04	0.02	1.01	
241-285	0.03	0.03	3.23	248.35	9	0.09	0/0	0.16	0.07	1.60	0.12	5.85	10	0.00	0/0	1.60	0.04	0.02	0.12	
289-317	0.05	0.03	3.95	6.87	10	0.00	0/0	0.24	0.06	1.50	0.16	0.99	10	0.00	0/0	1.90	0.06	0.02	0.13	
324-343	0.05	0.03	4.85	360.37	8	0.68	0/0	0.30	0.11	1.10	1.28	183.14	9	0.00	0/0	1.20	0.07	0.02	1.25	
345-348	0.04	0.02	0.56	0.35	10	0.00	0/0	0.28	0.07	1.00	0.13	0.13	10	0.00	0/0	1.00	0.05	0.02	0.07	
349-368	0.08	0.02	4.15	1.81	10	0.00	0/0	0.36	0.07	1.20	0.23	0.42	10	0.00	0/0	1.20	0.06	0.02	0.07	
371-474	0.29	0.05	26.41	12.66	8	0.00	0/2	0.90	0.90	1.30	5.58	200.57	9	0.03	1/0	1.30	0.20	0.07	82.29	

Table 5: Comparison of formulations for 4-way cyclic exchanges

	C		E					rEE		PE						PRE(i)						
PDPsR	tc	time	tp	time	opt	gap	nf	tr	time	nSCC	tp	time	opt	gap	nf	nSCC	tp	tr	time	opt	gap	nf
114-216	0.03	0.08	17.40	203.08	8	0.03	0/1	0.06	2.51	1.20	7.47	19.23	9	0.00	0/1	1.30	0.32	0.54	145.43	10	0.00	0/0
221-236	0.03	0.05	20.88	1.05	5	0.00	0/5	0.12	0.29	1.50	0.67	227.31	10	0.00	0/0	1.50	0.08	0.08	2.87	10	0.00	0/0
241-285	0.03	0.05	26.74	0.76	5	0.00	0/5	0.16	0.36	1.60	1.14	108.08	10	0.00	0/0	1.60	0.10	0.16	13.35	10	0.00	0/0
289-317	0.04	0.05	59.21	18.88	8	0.00	0/2	0.23	0.18	1.50	0.67	32.15	10	0.00	0/0	1.90	0.10	0.05	0.29	10	0.00	0/0
324-343	0.05	0.04	54.79	2.66	8	0.00	0/2	0.29	0.73	1.10	15.16	201.15	8	0.22	0/1	1.20	0.17	0.11	14.48	10	0.00	0/0
345-348	0.03	0.02	2.86	3.28	10	0.00	0/0	0.27	0.22	1.00	0.46	1.83	10	0.00	0/0	1.00	0.10	0.06	0.12	10	0.00	0/0
349-368	0.06	0.03	23.02	1.91	7	0.00	0/3	0.34	1.29	1.20	0.68	1.79	10	0.00	0/0	1.20	0.12	0.04	0.12	10	0.00	0/0
371-474	0.31	0.17	486.48	6.21	6	0.00	0/4	0.85	60.07	1.30	100.13	66.03	8	0.00	0/2	1.30	1.15	0.78	361.57	8	3.16	1/0

Table 6: Size of formulations for 3-way cyclic exchanges.

	C		E		rEE		PE				PRE(i)					
PDPsR	aVars	aCons	aVars	aCons	aVars	aCons	aVars	aCons	saVars	saCons	aVars	aCons	SCCp	saVars	saCons	sSCCp
114-216	274.4	183.0	3,959.1	270,128.3	1,259.9	505.2	661.5	91,659.7	81.36	78.16	329.7	3,190.8	4,527.4	89.21	99.49	99.04
221-236	162.7	258.1	6,853.1	436,337.7	472.0	478.3	544.5	20,308.9	92.03	90.56	322.9	675.9	1,098.1	95.38	99.82	99.66
241-285	146.0	284.3	9,100.1	542,559.6	505.7	494.3	446.5	16,201.8	95.13	92.36	287.8	663.9	1,022.3	96.87	99.82	99.57
289-317	133.4	337.3	11,907.3	655,659.0	388.3	505.3	592.7	21,579.9	94.57	92.11	308.0	470.8	754.7	97.20	99.89	99.71
324-343	201.8	366.5	13,718.7	797,498.8	749.6	604.6	1,376.9	220,375.0	91.55	83.52	1,032.3	1,073.0	1,606.6	93.82	99.79	99.65
345-348	99.4	393.0	13,232.7	72,522.7	314.8	506.5	513.3	14,706.0	96.12	79.82	317.6	179.3	302.0	97.60	99.75	99.58
349-368	177.3	407.8	14,438.8	680,322.1	541.6	654.8	717.8	31,220.2	95.13	85.99	455.9	336.7	551.5	96.96	99.78	99.57
371-474	583.3	454.3	20,699.9	4'278,204.8	2,915.8	949.4	2,823.1	970,304.3	89.72	83.26	1,713.6	6,755.2	11,674.6	94.09	99.85	99.58

Table 7: Size of formulations for 4-way cyclic exchanges.

	C		E		rEE		PE				PRE(i)					
PDPsR	aVars	aCons	aVars	aCons	aVars	aCons	aVars	aCons	saVars	saCons	aVars	aCons	SCCp	saVars	saCons	sSCCp
114-216	2,343.8	183.0	3,959.1	2'943,510.1	6,832.3	1,476.8	661.5	1'288,998.0	81.36	80.08	544.3	45,395.8	46.392.5	85.17	99.50	99.39
221-236	688.4	258.1	6,853.1	3'493,639.8	1,614.9	793.1	544.5	11,400.5	92.03	94.68	433.4	4,169.0	4,426.6	93.74	99.92	99.90
241-285	665.6	284.3	9,100.1	4'446,489.6	1,856.1	793.5	446.5	97,454.9	95.13	93.65	370.4	4,182.1	4,515.1	95.96	99.84	99.76
289-317	557.3	337.3	11,907.3	4'531,846.7	1,604.3	820.4	592.7	105,970.4	94.57	92.62	401.6	2,210.6	2,506.5	96.41	99.91	99.87
324-343	1,055.0	366.5	13,718.7	8'457,243.3	3,753.4	1,171.5	1,376.9	2'562,561.2	91.55	85.37	987.9	8,360.5	9,375.2	93.88	99.88	99.83
345-348	279.5	393.0	13,232.7	367,287.3	1,058.6	710.9	513.3	65,936.0	96.12	82.20	409.4	465.3	698.0	96.91	99.87	99.81
349-368	467.9	407.8	14,438.8	3'650,121.7	6,815.2	3,583.8	717.8	101,708.4	95.13	89.09	520.0	626.3	999.5	96.46	99.92	99.88
371-474	6,018.8	454.3	20,699.9	71'420,196.5	18,276.9	2,192.9	2,823.1	13'400,718.5	89.72	85.95	1,750.4	121,542.4	140,506.0	93.77	99.91	99.82

6.2 Assessment of cycle-and-chain variant formulations

In this subsection, we evaluate the performance of the variants of the cycle-and-chain formulations. The one exception is the eSPLIT formulation which is omitted because its cycle variant, the edge formulation (E), exhibited the worst performance in the first set of experiments discussed in the previous section. That is, it evidenced the largest runtimes and greatest number of failures due to memory overload.

Table 8: Comparison of formulations for 3-way cyclic exchanges and unbounded chains

		AA			PC-TSP			pSPLIT			ReSPLIT(i)		
PDPsR	NDDsR	tsep	time	opt	tsep	time	opt	time	opt	gap	time	opt	gap
114-216	3-22	1.13	550.49	7	1.18	720.21	6	369.95	8	11.73	373.4	8	11.73
221-236	22-34	3.92	900.10	5	3.01	900.26	5	900.09	5	13.47	900.11	5	13.31
241-285	6-39	4.63	964.65	5	3.57	1,210.16	4	906.06	5	11.04	906.14	5	11.38
289-317	6-39	4.46	1,081.13	4	4.57	1,084.57	4	631.63	7	21.53	656.05	7	21.53
324-343	6-46	3.16	778.65	7	3.56	1,350.3	2	219.82	9	1.30	222.06	9	1.30
345-348	46-46	5.36	1,303.48	3	3.23	1,675.25	1	110.52	10	0.00	105.31	10	0.00
349-368	39-49	2.75	547.13	7	1.99	713.16	7	219.89	9	5.60	214.65	9	5.60
371-474	47-50	1.73	362.02	8	2.34	362.61	8	368.53	8	6.60	546.23	7	7.60

Table 8 reports the results for 3-way cyclic exchanges and unbounded chains for the remaining formulations. As can be seen, the KEP now becomes harder to solve. With respect to our performance measures, the pSPLIT and the *ReSPLIT(i) formulations dominate the AA and the PC-TSP formulations in terms of runtime and number of optimal solutions found. Except for the last set of instances, the statistics for the pSPLIT and the *ReSPLIT(i) formulations are practically the same.

Table 9: Comparison of instances in Anderson et al. [4] with 3-way cyclic exchanges and unbounded chains

			AA	PC-TSP	pSPLIT		ReSPLIT(i)	
PDPs	NDDs	Edges	time	time	time	gap	time	gap
198	7	4,882	>1,800	>1,800	83.70		90.47	
202	3	4,706	0.14	0.19	0.25		0.22	
215	6	6,145	104.45	>1,800	12.05		12.27	
261	6	8,915	547.98	>1,800	37.61		36.70	
263	6	8,939	40.52	410.83	7.22		7.27	
284	5	10,126	46.92	837.70	12.08		13.86	
312	6	13,045	>1,800	>1,800	>1,800	25.06	>1,800	25.12
324	6	13,175	271.02	>1,800	22.74		39.47	
328	6	13,711	264.59	>1,800	12.28		13.25	
330	6	13,399	1,450.31	>1,800	79.02		79.94	

The solution algorithm is fully described in Appendix B). Essentially, the exponential constraints in Eq. (37) are initially dropped from the AA and PC-TSP models and only added back if they are violated when B&B finds an integer solution. The model is then reoptimized in an iterative fashion. Note that if after reaching the time limit, the integer solution found still violates some

cardinality constraints, i.e., additional constraints are still needed, that solution is deemed to be infeasible. Under this algorithm, the solutions for the AA and PC-TSP formulations are either optimal or infeasible so the gap columns are omitted in Table 8.

Anderson et al. [4] compared the AA and the PC-TSP formulations using a lazy-constraint and a branch-and-cut approach, respectively, on a set of "difficult" instances. Ten of those instances are included in our eight data sets. In Table 9 we compare the results obtained in [4] with our two cycle-and-chain variant formulations. As can be seen, the pSPLIT and the *ReSPLIT formulations, which also use a lazy constraint approach, outperform the AA and the PC-TSP formulations. In addition, a comparison of the results obtained from the pSPLIT and the *ReSPLIT formulations show little difference with respect to runtimes, gaps, and number of optimal solutions found. The AA formulation, on the other hand, is superior to the PC-TSP formulation for all instances tested, identifying a greater number of optimal solutions in less time.

Table 10: Comparison of formulations for 4-way cyclic exchanges and unbounded chains.

PDPsR	NDDsR	pSPLIT			*ReSPLIT(i)			
		time	opt	gap	time	opt	gap	nf
114-216	3-22	365.24	8	4.82	365.68	8	5.20	
221-236	22-34	900.07	5	10.12	900.12	5	10.10	
241-285	6-39	722.61	7	7.93	18.24	7	7.90	
289-317	6-39	550.87	7	8.74	562.88	7	8.74	
324-343	6-46	205.47	9	0.60	231.01	9	1.10	
345-348	46-46	91.51	10	0.00	81.64	10	0.00	
349-368	39-49	209.51	9	2.50	205.46	9	2.50	
371-474	47-50	421.53	8	5.80	720.67	6	7.30	2/0

In Table 10 we show the runtimes, gaps and number of optimal solutions reached by the pSPLIT and *ReSPLIT formulations when $k = 4$ and $l = \infty$. The performance of both formulations is again very similar, except for the last set of instances where CPLEX could not find an integer solution within the 1800-second time limit for two subproblems associated with one instance each under the *ReSPLIT(i) formulation. This behavior can be explained, in part, by analyzing the statistics in Tables 11 and 12. For both formulations, the size of the instances is roughly the same except for the last data set where there is a sharp increase in the number of constraints for the *PRE(i) formulation, especially when $k = 4$.

7 Summary and Conclusions

This paper presents three new formulations for the Kidney Exchange Problem, namely, (1) the partitioned edge formulation (*PE), (2) the partitioned and reduced edge formulation (*PRE), and (3) the reduced exponential-sized SPLIT formulation (*ReSPLIT), which are empirically shown to be superior to the well-known edge formulation. An algorithm to reduce the number of paths of length k is proposed and represents the first reported attempt to do so for an edge-based formulation. Dominance proofs are provided for the cycle-variant formulations studied, and extensive

Table 11: Size of formulations for 3-way cyclic exchanges and unbounded chains.

PDPsR	NDDsR	AA		PC-TSP		pSPLIT		*ReSPLIT(i)	
		aVars	aCons	aVars	aCons	aVars	aCons	aVars	aCons
114-216	3-22	3,980.0	11,129.3	4,254.4	2,786.1	5,606.9	4,851.2	4,739.8	7,643.8
221-236	22-34	6,884.4	8,761.3	7,047.1	2,177.4	7,873.6	7,878.9	7,725.6	8,174.9
241-285	6-39	9,136.5	11,949.5	9,282.5	4,545.2	10,211.8	10,199.4	9,995.8	10,486.9
289-317	6-39	11,975.9	11,906.8	12,109.3	3,936.6	13,039.8	13,155.8	12,962.0	13,191.2
324-343	6-46	13,746.8	17,452.9	14,220.4	5,799.4	15,230.4	15,084.4	15,515.0	15,772.5
345-348	46-46	13,232.8	25,149.1	13,332.2	7,702.4	14,334.6	14,525.3	14,337.4	14,497.6
349-368	39-49	14,564.2	6,622.0	14,741.5	3,638.7	15,922.4	16,034.6	15,846.6	15,970.2
371-474	47-50	20,747.9	3,346.6	21,331.2	1,678.9	24,573.3	22,605.9	23,374.9	28,559.5

Table 12: Size of formulations for 4-way cyclic exchanges and unbounded chains.

PDPsR	NDDsR	pSPLIT		*ReSPLIT(i)	
		aVars	aCons	aVars	aCons
114-216	3-22	11,179.3	5,822.8	4,891.3	49,842.9
221-236	22-34	9,016.5	8,193.7	7,835.1	11,544.7
241-285	6-39	11,562.2	10,498.6	10,076.7	13,735.8
289-317	6-39	14,255.8	13,470.9	13,053.4	14,199.2
324-343	6-46	18,234.2	15,651.3	15,468.8	22,880.8
345-348	46-46	15,078.4	14,729.7	14,429.2	14,781.9
349-368	39-49	22,196.0	18,963.6	15,902.1	16,208.7
371-474	47-50	39,934.4	23,849.4	23,408.0	142,726.6

computational tests are performed to compare the solution quality, runtimes, and related statistics obtained for the existing and new formulations.

Regarding the cycle-variant formulations, computational results confirm that the cycle formulation (C) is very effective for low-density graphs, outperforming the other cycle-variants we tested. In contrast, the standard edge formulation is shown to be relatively ineffective for solving any instances and gets substantially worse as k increases. The partitioned edge formulation outperforms the edge formulation, but only when the graphs are not a unique strongly connected component.

The *PRE(i) formulation uses a new set of constraints and a small subset of the full set of length-k paths that are included in the edge formulation. The reduction in the number of such paths is greater than 98% for all instances tested. The performance of the *PRE(i) is very similar to that of the extended edge formulation considering the reduction presented in Appendix B, although the latter does better on the largest size instances as well as when $k = 4$.

With respect to the cycle-and-chain formulations, the extensions of the exponential-sized SPLIT formulation (*ReSPLIT) outperformed Anderson's arc-based formulation and the PC-TSP formulation on all instances tested. In particular, when tested on the "difficult" instance set from Anderson et al. [4], the proposed model (*ReSPLIT) provided the best results when compared to the AA and PC-models solved by a lazy-constraint algorithm.

The most successful computational work for the KEP has been performed within a branch-and-price framework since the edge formulations with an exponential number of constraints have proven

unsolvable with commercial software even for small instances. As indicated in the literature review section, there are two types of KEP models: those based on an exponential number of variables and those based on an exponential number of constraints. For the former, column generation and branch-and-pric approaches have been developed with a certain degree of success. For the latter, existing models have again proven unsolvable with commercial software even for relatively small instances.

The only constraint-generation approach [4] known to date remains competitive only when the cycle length is 3 and the chain length is unbounded. Therefore, an interesting avenue of research would involve the use of polyhedral theory to develop branch-and-cut algorithms for the partitioned and reduced edge formulation. In this study, we proposed a heuristic to reduce the exponential number of path elimination constraints while ensuring optimality. However, only "static" strategies were considered when selecting the node. A deeper examination of the problem structure might lead to more tailored strategies and strong valid inequalities that would improve the upper bound provided by the LP relaxation.

The complexity and size of real instances in many countries are still manageable by the most efficient methods reported in the literature. In the long run, though, if a trade-off exists between losing optimality and achieving a greater number of transplants, heuristics may become the norm for the KEP.

Acknowledgments: We would like to thank the Mexican Council for Science and Technology (CONACyT) for providing the financial resources to conduct this research. Additionally, we would like to thank Ross Anderson, Itai Ashlagi, and Alvin Roth for kindly sharing their test instances with us and for the helpful comments. We also acknowledge enlightening discussions with John Dickerson about kidney exchanges.

References

[1] D. J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 295–304, San Diego, June 2007.

[2] F. Alvelos, X. Klimentova, and A. 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.

[3] R. Anderson, I. Ashlagi, D. Gamarnik, M. Rees, A. E. Roth, T. Sönmez, and M. U. Ünver. Kidney exchange and the alliance for paired donation: Operations research changes the way kidneys are transplanted. *Interfaces*, 45(1):26–42, 2015.

[4] R. Anderson, I. Ashlagi, D. Gamarnik, and A. E. Roth. Finding long chains in kidney exchange using the traveling salesman problem. *Proceedings of the National Academy of Sciences*, 112 (3):663–668, 2015.

[5] R. M. Anderson. *Stochastic Models and Data Driven Simulations for Healthcare Operations*. PhD thesis, Massachusetts Institute of Technology, Cambridge, 2014.

[6] I. Ashlagi, D. S. Gilchrist, A. E. Roth, and M. A. Rees. Nonsimultaneous chains and dominos in kidney-paired donation – Revisited. *American Journal of Transplantation*, 11(5):984–994, 2011.

[7] P. Awasthi and T. Sandholm. Online stochastic optimization in the large: Application to kidney exchange. In C. Boutilier, editor, *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 405–411, AAAI Press, Menlo Park, July 2009.

[8] P. Biró, D. F. Manlove, and R. Rizzi. Maximum weight cycle packing in directed graphs, with application to kidney exchange programs. *Discrete Mathematics, Algorithms and Applications*, 1(4):499–517, 2009.

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

[10] F. L. Delmonico. Exchanging kidneys – advances in living-donor transplantation. *The New England Journal of Medicine*, 350(18):1812–1814, 2004.

[11] J. P. Dickerson, A. D. Procaccia, and T. Sandholm. Optimizing kidney exchange with transplant chains: Theory and reality. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12), Volume 2*, pages 711–718, International Foundation for Autonomous Agents and Multiagent System, Richland, 2012.

[12] J. P. Dickerson, D. F. Manlove, B. Plaut, T. Sandholm, and J. Trimble. Position-indexed formulations for kidney exchange. In *Proceedings of the 2016 ACM Conference on Economics and Computation (EC'16)*, pages 25–42, ACM, New York, 2016.

[13] J. P. Dickerson, A. D. Procaccia, and T. Sandholm. Failure-aware kidney exchange. *Management Science*, 65(4):1768–1791, 2019.

[14] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[15] S. E. Gentry, D. L. Segev, M. Simmerling, and R. A. Montgomery. Expanding kidney paired donation through participation by compatible pairs. *American Journal of Transplantation*, 7 (10):2361—2370, 2007.

[16] S. E. Gentry, R. A. Montgomery, B. J. Swihart, and D. L. Segev. The roles of dominos and nonsimultaneous chains in kidney paired donation. *American Journal of Transplantation*, 9 (6):1330–1336, 2009.

[17] K. M. Glorie, J. J. van de Klundert, and A. P. M. Wagelmans. Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manufacturing and Service Operations Management*, 16(4):498–512, 2014.

[18] D. B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.

[19] V. Mak-Hau. On the kidney exchange problem: Cardinality constrained cycle and chain problems on directed graphs: A survey of integer programming approaches. *Journal of Combinatorial Optimization*, 33(1):35–59, 2017.

[20] D. F. Manlove and G. O'Malley. Paired and altruistic kidney donation in the UK: Algorithms and experimentation. *ACM Journal of Experimental Algorithmics*, 19:2.6:1–2.6:21, 2014.

[21] J. P. Pedroso. Maximizing expectation on vertex-disjoint cycle packing. In B. Murgante, S. Misra, A. M. A. C. Rocha, C. Torre, J. G. Rocha, M. I. Falcão, D. Taniar, B. O. Apduhan, and O. Gervasi, editors, *Computational Science and Its Applications – ICCSA 2014, Part II*, volume 8580 of *Lecture Notes in Computer Science*, pages 32–46, Cham, Switzerland, 2014. Springer.

[22] B. Plaut, J. P. Dickerson, and T. Sandholm. Fast optimal clearing of capped-chain barter exchanges. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 601–607, AAAI Press, Phoenix, 2016.

[23] F. T. Rapaport. The case for a living emotionally related international kidney donor exchange registry. *Transplantation Proceedings*, 18(3):5–9, 1986.

[24] L. C. Riascos Alvarez. *Formulations and Algorithms for the Kidney Exchange Problem*. Master thesis, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, NL, Mexico, April 2017.

[25] L. F. Ross and E. S. Woodle. Ethical issues in increasing living kidney donations by expanding kidney paired exchange programs. *Transplantation*, 69(8):1539–1543, 2000.

[26] L. F. Ross, D. T. Rubin, M. Siegler, M. A. Josephson, J. R. Thistlethwaite, and E. S. Woodle. Ethics of a paired-kidney exchange program. *The New England Journal of Medicine*, 336(24): 1752–1755, 1997.

[27] A. E. Roth, T. Sönmez, and M. U. Ünver. Kidney exchange. *The Quarterly Journal of Economics*, 119(2):457–488, 2004.

[28] A. E. Roth, T. Sönmez, and M. U. Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151–188, 2005.

[29] A. E. Roth, T. Sönmez, and M. U. Ünver. A kidney exchange clearinghouse in New England. *American Economic Review*, 95(2):376–380, 2005.

[30] A. E. Roth, T. Sönmez, M. U. Ünver, F. L. Delmonico, and S. L. Saidman. Utilizing list exchange and nondirected donation through 'chain' paired kidney donations. *American Journal of Transplantation*, 6(11):2694–2705, 2006.

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

[32] S. L. Saidman, A. E. Roth, T. Sönmez, M. U. Ünver, and F. L. Delmonico. Increasing the opportunity of live kidney donation by matching for two- and three-way exchanges. *Transplantation*, 81(5):773–782, 2006.

[33] M. Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers and Mathematics with Applications*, 7(1):67–72, 1981.

[34] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1 (2):146–160, 1972.

[35] M. U. Ünver. Dynamic kidney exchange. *The Review of Economic Studies*, 77(1):372–414, 2010.

Appendix A

In this appendix, we present the seven most relevant models in the literature that were used in this work for comparison purposes. The formulations given in Sections A.1.2 and A.2.4) provided the basis for the new models that were introduced in Section 4.

A.1 Cycle-variant formulations

A.1.1 The Cycle Formulation [1, 31]

Let C_k be the set of all cycles in G with length at most k and let $V(c)$ be the set of vertices that belong to cycle $c \in C_k$. Define a variable z_c for each cycle $c \in C_k$.

$$z_c = \begin{cases} 1 & \text{if cycle } c \text{ is selected for an exchange} \\ 0 & \text{otherwise} \end{cases}$$

Define $w_c = \sum_{(i,j)\in c} w_{ij}$. The cycle formulation (denoted by C) can be written as follows.

$$\text{Maximize} \quad \sum_{c \in C_k} w_c z_c \tag{17}$$

$$\text{subject to} \quad \sum_{c: i \in V(c)} z_c \leq 1 \qquad i \in V \tag{18}$$

$$z_c \in \{0,1\} \qquad c \in C_k \tag{19}$$

The objective function (17) maximizes the weighted number of transplants. In the case of unitary weights, w_c equals the number of edges in c, i.e., the number of transplants associated with cycle c. Constraints (18) ensure that every vertex is in at most one of the selected cycles since each donor may donate and each patient may receive only one kidney.

A.1.2 Edge Formulation [1, 31]

Let Π be the set of all length-k paths in a graph formed by $k+1$ nodes or k edges. In the edge formulation (E), a variable x_{ij} is associated with each directed edge $(i,j) \in E$ in the graph (V,E) and is defined as follows.

$$x_{ij} = \begin{cases} 1 & \text{if donor in pair } i \text{ is compatible with patient in pair } j \\ 0 & \text{otherwise} \end{cases}$$

The corresponding model takes the following form.

$$\text{Maximize} \quad \sum_{(i,j) \in E} w_{ij} x_{ij} \tag{20}$$

$$\text{subject to} \quad \sum_{j:(i,j) \in E} x_{ij} = \sum_{j:(j,i) \in E} x_{ji} \qquad i \in V \tag{21}$$

$$\sum_{j:(i,j) \in E} x_{ij} \leq 1 \qquad i \in V \tag{22}$$

$$\sum_{1 \leq p \leq k} x_{i_p i_{p+1}} \leq k-1 \qquad (i_1, ..., i_k, i_{k+1}) \in \Pi \tag{23}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in E \tag{24}$$

The objective function (20) maximizes the weighted sum of matches. Constraints (21) guarantee that donor i provides a kidney if and only if patient i receives one in return. Constraints (22) ensure that a person can only donate a single kidney, and constraints (23) enforce the length of any cycle to be less than or equal to k. Any feasible cycle will always contain a path with at most $k-1$ edges (no repeating edges). Therefore, if we preclude all length-k paths from being part of a cycle, cycles of length $k+1$ or greater are excluded from a feasible solution.

Prior to our work, no constraint reductions had been proposed so in previous implementations all paths of length k are explicitly included in the model (see [1, 9, 19, 31]).

A.1.3 Extended Edge Formulation [9]

Let $G = (V, E)$ be cloned into $|V|$ copies, and let $L = \{1, \ldots, |V|\}$. Note that L is an upper bound on the number of cycles in a solution. In each copy ℓ at most k edges produce a cycle and each node $i \in V$ can belong to at most one cycle. This is enforced by adding cardinality constraints for every copy. The model uses the following variables

$$x_{ij}^{\ell} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is used in copy } \ell \text{ of the graph} \\ 0 & \text{otherwise} \end{cases}$$

for the extended edge formulation (EE) given below.

$$\text{Maximize} \quad \sum_{\ell \in L} \sum_{(i,j) \in E} w_{ij} x_{ij}^{\ell} \tag{25}$$

$$\text{subject to} \quad \sum_{j \in P:(i,j) \in E} x_{ij}^{\ell} = \sum_{j \in P:(j,i) \in E} x_{ji}^{\ell} \qquad i \in V, \ell \in L \tag{26}$$

$$\sum_{\ell \in L} \sum_{j:(i,j) \in E} x_{ij}^{\ell} \leq 1 \qquad i \in V \tag{27}$$

$$\sum_{(i,j) \in E} x_{ij}^{\ell} \leq k \qquad \ell \in L \tag{28}$$

$$\sum_{j:(i,j) \in E} x_{ij}^{\ell} \leq \sum_{j:(i,j) \in E} x_{\ell j}^{\ell} \qquad i > \ell, \ell \in L \tag{29}$$

$$\sum_{j:(i,j) \in E} x_{ij}^{\ell} = 0 \qquad i < \ell, \ell \in L \tag{30}$$

$$x_{ij}^{\ell} \in \{0, 1\} \qquad (i,j) \in E, \ \ell \in L \tag{31}$$

The objective function (25) similarly maximizes the weighted number of transplants. Constraints (26) ensure flow balance at each vertex, i.e., a paired donor will give a kidney to another patient in the pool if and only if the patient in the first pair receives one in return. Constraints (27) ensure that no more than one kidney transplant is involved for each PDP. Constraints (28) guarantee that the cardinality of each cycle and copy is less than or equal to k. The extended edge formulation has symmetry. Constraints (29) and (30) preclude multiplicity of solutions in the IP model induced by a permutation of cycle indices. Symmetry elimination is achieved by restricting the index of a cycle to be exactly the smallest numbered node, i.e., the smallest index among all nodes included in the cycle.

A reduction of model (25) – (31) was also proposed by Constantino et al. [9]. For each copy, the reduction was based on finding both the shortest path from the smallest index node to each of

the other nodes and the shortest path in the reverse direction. A second step involved removing nodes for which the sum of edges in both paths was greater than k. In such cases, no feasible cycles exist. The reduced formulation is called the reduced extended edge formulation (rEE), for which we show results in Section 6.

A.2 Cycle-and-chain Variant Formulations

In this subsection, we present some existing formulations of the KEP with non-directed donors.

A.2.1 The Arc-Based Formulation [4]

Continuing with the notation used in Section A.1.1, let C be the set of all cycles in G and let C_k be the set of all cycles in G with length at most k. In addition, define f_i^e and f_i^o to be the flow entering node i and the flow leaving node i, respectively. The flow variable x_{ij} is associated with directed edge $(i,j) \in E$ and is defined as follows.

$$x_{ij} = \begin{cases} 1 & \text{if the patient in pair } j \text{ receives a kidney from the donor in pair } i \\ 0 & \text{otherwise} \end{cases}$$

In the model, this variable is used in the representation of both chains and cycles. The arc-based formulation (AA), developed by Anderson et al. [4], is given by

$$\text{Maximize} \quad \sum_{(i,j)\in E} w_{ij} x_{ij} \tag{32}$$

$$\text{subject to} \quad \sum_{(j,i)\in E} x_{ji} = f_i^e \qquad i \in V \tag{33}$$

$$\sum_{(i,j)\in E} x_{ij} = f_i^o \qquad i \in V \tag{34}$$

$$f_i^o \le f_i^e \le 1 \qquad i \in P \tag{35}$$

$$f_i^o \le 1 \qquad i \in N \tag{36}$$

$$\sum_{(i,j)\in c} x_{ij} \le |c| - 1 \qquad c \in C \setminus C_k \tag{37}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in E \tag{38}$$

Constraints (33) and (34) define the net flow entering and leaving node i, respectively. Constraints (35) say that PDP nodes may contribute a kidney as long as they have received one. Constraints (36) limit NDD nodes to donating at most one kidney. Constraints (37) rule out cycles of length greater than k. Because chains are considered unbounded, no related constraints are needed.

A.2.2 The PC-TSP-Based Formulation [4]

This model is motivated by an IP formulation of the so-called prize-collecting traveling salesman problem (PC-TSP). Let S be a set of nodes, $S \subset V$, and let $\bar{S} = V \setminus S$. For each $i \in V$, let $C_k(i)$ be the set of cycles in C_k containing an edge incident to i. For this formulation, Anderson et al. [4] split chain and cycle variables such that z_c is a variable for every feasible cycle c and x_{ij} is a variable for edges in unbounded chains; that is:

$$z_c = \begin{cases} 1 & \text{if the donor-patient pairs in cycle } c \text{ are selected for an exchange} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is used in a chain} \\ 0 & \text{otherwise} \end{cases}$$

The PC-TSP-based formulation is given below.

$$\text{Maximize} \quad \sum_{(i,j)\in E} w_{ij}x_{ij} + \sum_{c\in C_k} w_c z_c \tag{39}$$

$$\text{subject to} \quad \sum_{(j,i)\in E} x_{ji} = f_i^e \qquad i \in V \tag{40}$$

$$\sum_{(i,j)\in E} x_{ij} = f_i^o \qquad i \in V \tag{41}$$

$$f_v^o + \sum_{c\in C_k(i)} z_c \le f_i^e + \sum_{c\in C_k(i)} z_c \le 1 \qquad i \in P \tag{42}$$

$$f_i^o \le 1 \qquad i \in N \tag{43}$$

$$\sum_{(j,m):j\in\bar{S},m\in S} x_{jm} \ge f_i^e \qquad S \subseteq P, i \in S \tag{44}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in E \tag{45}$$

$$z_c \in \{0,1\} \qquad c \in C_k \tag{46}$$

The objective function (39) maximizes the weighted sum of cycles and chains. The weights w_{ij} and w_c are input parameters. Constraints (40) and (41) keep track of flow into and out of each node $i \in V$, respectively. Constraints (42) ensure that every patient-donor pair, if part of the solution, must belong to either a cycle or a chain, and that the flow leaving any node i is at most equal to the flow entering that node. Constraints (44) require that if a node i is to be included in any chain, there must exist a flow entering S triggered by a NDD node. Note that if node i is part of some solution, then $f_i^e = 1$. If, additionally, we cut the graph such that only PDPs are in S, in order for that solution to be a chain, there must exist flow going from $\bar{S}$ to S for any cut set.

A.2.3 The Polynomial-Sized SPLIT Formulation [19]

This model augments the extended edge formulation to allow for chains. To this end, an auxiliary sink node τ is introduced to facilitate the implementation of subtour elimination constraints. In particular the authors use the MTZ-constraints, which are often included in asymmetric TSP formulations. The auxiliary node introduces some changes in our notation. Let $E^{'} = \{(i,j) \mid i \in V, j \in P^{\tau}\}$ where $P^{\tau} = P \cup \{\tau\}$ and let $G^{'}$ as $G^{'} = (V \cup \{\tau\}, E^{'})$.

The model has three types of variables: one set of continuous variables t_i to set a time stamp for node i should it be part of a chain, and two sets of decision variables defined as follows.

$$z_{ij}^{\ell} = \begin{cases} 1 & \text{if arc } (i,j) \text{ forms part of the } \ell\text{-th cycle} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ forms part of a chain} \\ 0 & \text{otherwise} \end{cases}$$

The polynomial-sized SPLIT formulation (pSPLIT) takes the form of a mixed-integer linear program.

$$\text{Maximize} \quad \sum_{(i,j)\in E'} w_{ij}x_{ij} + \sum_{l\in L}\sum_{(i,j)\in E} z_{ij}^{\ell} w_{ij} \tag{47}$$

$$\text{subject to} \quad \sum_{j\in P:(i,j)\in E} z_{ij}^{\ell} = \sum_{j\in P:(j,i)\in E} z_{ji}^{\ell} \qquad i \in P, \ell \in L \tag{48}$$

$$\sum_{j\in P^{\tau}:(i,j)\in E'} x_{ij} = \sum_{j\in V:(j,i)\in E'} x_{ji} \qquad i \in P \tag{49}$$

$$\sum_{j\in P^{\tau}:(i,j)\in E'} x_{ij} \leq 1 \qquad i \in N \tag{50}$$

$$\sum_{j\in P^{\tau}:(i,j)\in E'} x_{ij} \leq 1 - \sum_{l\in L}\sum_{j\in P:(i,j)\in E} z_{ij}^{\ell} \qquad i \in P \tag{51}$$

$$(28) - (30) \quad \text{in which } x_{ij}^{\ell} \text{ is replaced with } z_{ij}^{\ell} \tag{52}$$

$$|P| + 1 \geq t_i - t_j + |P|x_{ji} + (|P| + 2)x_{ij} \qquad i \in V, j \in P^{\tau} \tag{53}$$

$$t_i = 0 \qquad i \in N \tag{54}$$

$$t_i \geq 0 \qquad i \in P^{\tau} \tag{55}$$

$$t_{\tau} \leq |P| + 1 \qquad i \in P^{\tau} \tag{56}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in E^{'} \tag{57}$$

$$z_{ij}^{\ell} \in \{0,1\} \qquad (i,j) \in E,\ \ell \in L \tag{58}$$

Constraints (48) and (49) ensure flow balance. The flow into node i must equal the flow out for edges that belong to either a cycle or a chain; the flow out is at most one as assured by constraints (50) and (51). Recall that τ is a sink node that can be reached by every node, thus guaranteeing

that these constraints can be satisfied by chain variables as well. Again, constraints (50) say that an NDD can donate at most a single kidney. Constraints (51) assure patient-donor pairs to be part of either a chain or a cycle. Constraints (52) are borrowed from the extended edge formulation to limit cycle lengths and to avoid symmetry issues. The MTZ constraints (53) prevent chain-edge variables from creating cycles. Constraints (54) – (56) put bounds on the continuous variables.

A.2.4 The Exponential-Sized SPLIT Formulation [19]

Unlike the previous model, this one simply uses a binary variable z_{ij} for each $(i,j) \in E$ to indicate whether or not (i,j) is included in a cycle. In addition, the MTZ-constraints are replaced by a stronger version of cardinality-infeasible-cycle elimination constraints; in particular, the length-k path constraints as in (23).

The exponential-sized SPLIT formulation (eSPLIT) is given by

$$\text{Maximize} \quad \sum_{(i,j)\in E'} w_{ij}x_{ij} + \sum_{(i,j)\in E} w_{ij}z_{ij} \tag{59}$$

$$\text{subject to} \quad (49)-(50), \quad (53)-(56) \tag{60}$$

$$(23) \text{ replacing } x_{ij} \text{ by } z_{ij} \tag{61}$$

$$\sum_{j\in P^\tau:(ij)\in E'} x_{ij} \le 1 - \sum_{j\in P:(i,j)\in E} z_{ij} \qquad i \in P \tag{62}$$

$$\sum_{j\in P:(i,j)\in E} z_{ij} = \sum_{j\in P:(j,i)\in E} z_{ji} \qquad i \in P \tag{63}$$

$$x_{ij}, z_{ij} \in \{0,1\} \qquad (i,j) \in E \tag{64}$$

Constraints (60) and (61) transfer from the polynomial-sized SPLIT formulation and the edge formulation, respectively. Constraints (62) limit each node to belong to either a cycle or a chain while ensuring that the flow out of any node is at most one. Constraints (63) say that for each PDP node i, flow in must equal the flow out.

Appendix B

In this appendix, we present a reduced EE formulation and a lazy-constrained procedure for the AA and PC-TSP formulations.

Reduced EE formulation

In Constantino et al. [9], a reduced model was proposed and tested for the extended edge formulation. Note that in the ℓ^{th} copy of the graph, if there is no cycle of size at most k containing both node ℓ and node i, with $i \ge \ell$, then node i and its adjacent edges can be removed from the model. Note that the node with index ℓ is included in the ℓ-th graph as explained above. Let $G^\ell = (V^\ell, E^\ell)$

be the ℓ^{th} copy of the graph and let $d_{i,j}$ be the distance of the shortest path on G^ℓ from node i to node j. The stay-in nodes and edges are:

$$V^\ell = \{i \in V \mid i \geq \ell, d^\ell_{\ell,i} + d^\ell_{i,\ell} \leq k\}$$
$$E^\ell = \{(i,j) \in E \mid i,j \in V^\ell, d^\ell_{\ell,i} + 1 + d^\ell_{j,\ell} \leq k\}$$

After performing this reduction, we called CPLEX to find solutions.

Lazy-constraint procedure for the AA and the PC-TSP formulations

In many combinatorial optimization problems there are models with an exponential set of constraints that may yield a stronger LP relaxation when compared to the LP relaxation of the original IP formulation that does not include them. For realistic size instances, such "difficult" constraints cannot be explicitly written out and fed to an MILP solver, as is the case for Anderson's arc-based and PC-TSP-based formulations. As an alternative, it is possible to start with a relaxation obtained by dropping the difficult constraints and then adding them back when they are violated.

To implement this idea, Anderson et al. [4] proposed an iterative procedure that uses branch and bound and a lazy constraint generation scheme for the arc-based formulation and, a branch-and-cut approach for the PC-TSP-based formulation. In this work we implemented our version of that lazy constraint generation scheme within CPLEX to solve the above mentioned formulations; for the remaining formulations, CPLEX was called directly.

In particular, by relaxing constraints (37) and (44), we obtain problem instances that can be solved with a standard commercial optimizer. Integer solutions to this relaxation are checked for cycles of length greater greater than k. We implemented a lazy-constraint callback in CPLEX 12.7 that solves the separation problem only for integer solutions. This involved recursively traversing the solution and checking whether all the cycles were feasible. We then added as many violated constraints as the number of infeasible cycles found and repeated the process.

When no more violations can be found the algorithm terminates and returns the optimal solution. This is true because the separation problem is solved exactly.

As an illustration, let $k = 3$ and consider the graph in Figure 7 which was obtained by solving Anderson's arc-based formulation without constraints 37. Applying the separation algorithm we discover cycle $3 \to 4 \to 5 \to 6 \to 3$, which is then eliminated by adding the constraint below to the IP model and re-solving.

$$x_{34} + x_{45} + x_{56} + x_{63} \leq 3$$

To prevent chain variables from forming cycles, we add the following constraints to the PC-TSP formulation and re-solve.

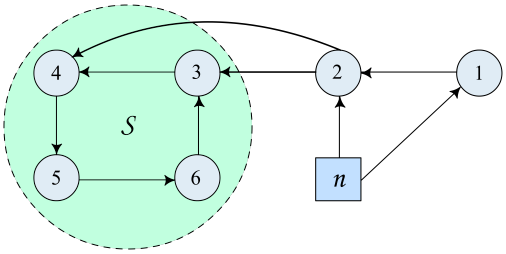

Figure 7: Example of cut set constraints for the PC-TSP model

$$y_{23} + y_{24} \geq y_{63}$$

$$y_{23} + y_{24} \geq y_{34}$$

$$y_{23} + y_{24} \geq y_{45}$$

$$y_{23} + y_{24} \geq y_{56}$$