A Branch-and-Price Algorithm Enhanced by Decision Diagrams for the Kidney Exchange Problem

Lizeth Carolina Riascos-Álvarez
Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario M5S 3G8, Canada, carolina.riascos@mail.utoronto.ca

Merve Bodur
Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario M5S 3G8, Canada, bodur@mie.utoronto.ca

Dionne M. Aleman
Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario M5S 3G8, Canada
Institute of Health Policy, Management and Evaluation, University of Toronto, Toronto, Ontario M5S 3E3, Canada
Techna Institute at University Health Network, Toronto, Ontario M5G 1P5, Canada, aleman@mie.utoronto.ca

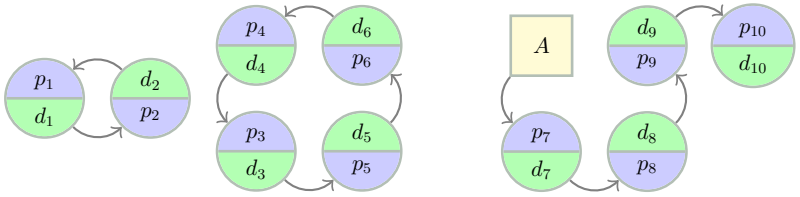
Kidney paired donation programs allow patients registered with an incompatible donor to receive a suitable kidney from another donor, as long as the latter's co-registered patient, if any, also receives a kidney from a different donor. The kidney exchange problem (KEP) aims to find an optimal collection of kidney exchanges taking the form of cycles and chains. Existing exact solution methods for KEP either are designed for the case where only cyclic exchanges are considered, or can handle long chains but are scalable only if cycles are short. We develop the first decomposition method that is able to consider long cycles and long chains for projected large realistic instances. More specifically, we propose a branch-and-price framework in which the pricing problems are solved (for the first time in packing problems in a digraph) through multi-valued decision diagrams. Typically, hierarchical objectives are solved sequentially by adding rationality constraints and re-optimizing afterwards. Our algorithm allows prioritization of the solution composition, e.g., chains over cycles or vice versa in a single run. We present a new upper bound on the optimal value of KEP, stronger than the one proposed in the literature, which is obtained via our master problem. Computational experiments show superior performance of our method over the state of the art by optimally solving almost all instances in the PrefLib library for multiple cycle and chain lengths. Lastly, we find that chains benefit the matching efficiency, similar to previous findings.

Key words: Kidney exchange; integer programming; branch and price; multi-valued decision diagrams

1. Introduction

The preferred treatment for kidney failure is transplantation, and demand for deceased-donor kidneys usually outnumbers supply (CIHI 2019). An alternative, often desirable, is living-donor transplantation. A living donor is typically a close relative, partner or friend who is willing to donate one of their kidneys to grant a life-saving chance to a beloved one. However, biological incompatibilities, such as blood type or antibodies related discrepancies, between the patient in need of a kidney and the potential donor may exist. It is in these cases where Kidney Paired Donation Programs (KPDPs), present in multiple countries, have played a life-saving role in kidney transplantation systems. A KPDP is a centralized registry operated at a local or national level, where each patient registers voluntarily along with his or her incompatible (suboptimal) donor (*paired donor*) as a pair. Patients in these patient-donor pairs (PDPs) are willing to exchange their paired donors under the promise that they will receive a suitable kidney from a different donor. To accomplish this goal, two types of exchanges are allowed: cyclic and chain-like exchanges.

Figure 1a illustrates a pool of six patient-donor pairs (p_1, d_1), (p_2, d_2),...,(p_6, d_6) arranged in two cycles. In the cyclic exchange on the left, donor d_2 donates to patient p_1 and donor d_1 donates to patient p_2, thereby forming a *cycle*. On the right, a cycle involving four PDPs is depicted, where donor d_4 donates a kidney to patient p_3 and patient p_4 receives a kidney from donor d_6, after sequential donations. Due to pair drop-outs, aggravated health condition of a pair member, or last-minute detected incompatibilities, the patient in the first pair may never receive a kidney back if the donor in any subsequent pair in the cycle fails to donate. To avoid such a risk, cycles of kidney transplants are performed simultaneously in practice, imposing limitations on the maximum size of a cycle, $K \in \mathbb{Z}_+$. In the literature, a k-way cycle refers to a cycle involving k transplants, with $k \leq K$. Although



(a) 2-way cycle (left) and 4-way cycle (right) (b) 4-length chain

Figure 1 **Example exchanges. PDPs are represented by circle nodes, while the NDD is the squared one.**

in some countries such as the United States, it is common to find $K = 3$ (Flechner et al. 2018), other countries have reported longer cycles (Malik and Cole 2014, Cantwell et al. 2015), up to $K = 6$ in Canada (CBS 2019). Depending on the country's population and matching frequency, KPDP sizes may vary. For instance, there have been KPDPs with roughly 500 PDPs in The Netherlands (Glorie et al. 2014). However, in the United States, a nationwide KPDP would include 10,000 PDPs (Abraham et al. 2007).

In the presence of singleton donors, chains become an exchange alternative. Some singleton donors are *altruistic donors* because they are willing to donate one of their kidneys without having a paired patient. A *chain* is a path that starts with a singleton donor donating to a patient in a PDP, after which all remaining patients receive a kidney from a paired donor (Figure 1b). The donor in the last pair of a chain can either donate to a patient on the transplant waitlist or become a *bridge donor* (a single donor in a future chain). In general, chain-initiating donors such as altruistic, bridge, or even deceased donors (Wall et al. 2018), are referred to as *non-directed donors* (NDDs). Unlike cycles, no paired donor in a chain risks not receiving a kidney for their intended recipient, making it possible to relax the simultaneity constraint. Very often, KPDPs also set an upper bound on the number of transplants in a chain, $L \in \mathbb{Z}_+$, $L \geq K$ (Malik and Cole 2014, Flechner et al. 2018).

As some transplants may be more suitable or urgent from medical and logistical points of view, a score may be given to every potential transplant. A common objective in kidney exchange, although not the only one, is to match donors (paired and singleton) to patients

such that the sum of the transplants' score is maximized (Abraham et al. 2007, Roth et al. 2007). The matching must satisfy that every PDP belongs to at most one cycle or chain and every NDD to at most one chain. Finding a matching of maximum score in this context is known as the *kidney exchange problem* (KEP).

Motivated by practical settings for which $K \geq 4$ along with long chains (Malik and Cole 2014, Ashlagi and Roth 2021), previous studies showing the value of long chains (Ashlagi et al. 2012, Dickerson et al. 2012b, Ding et al. 2018), the increasing number of participants in KPDPs, and space for improvement in state-of-the-art approaches, we develop a new solution technique for the KEP. Particularly, we make the following contributions:

1. We improve an existing integer programming formulation, upon which we propose a Lagrangian Decomposition. Moreover, we show its relationship with branch-and-price (B&P) algorithms, and its usefulness to provide a tighter upper bound on the optimal value of the KEP compared to an existing upper bound.
2. We devise a B&P algorithm that incorporates cycles and long chains.
3. We solve the pricing problems via multi-valued decision diagrams (MDDs). To the best of our knowledge, this is the first study using MDDs in cycle and path packing problems in a digraph, and one of the two works in the B&P literature Raghunathan et al. (2018).
4. We present an effective three-phase solution method for the pricing problems, shifting between MDDs and linear (worst-case integer) programs.
5. We demonstrate computational improvement over state-of-the-art methods in benchmark realistic instances.
6. Understanding that multiple optima can exist in KEP, and that some of them may be preferred for logistical or fairness reasons, our algorithm allows the prioritization of chains or cycles in a single run. Additionally, we perform an experimental analysis showing the impact of the chain/cycle composition of multiple solutions.

7. We empirically demonstrate the benefits of chains for different graph densities and sizes, taking into account the presence of highly sensitized patients. Our results are in agreement with previous studies (Ashlagi et al. 2012, Dickerson et al. 2012b, Ding et al. 2018).

The rest of the paper is organized as follows. In Section 2, we review the relevant literature. In Section 3, we present a formal definition of the KEP. In Section 4, we introduce a Lagrangian decomposition. In Section 5, we detail our B&P algorithm, including the reformulation of pricing problems via MDDs. In Section 6, we present our general solution approach. In Section 7, we show experimental results comparing our algorithm with the state of the art. Lastly, we draw some conclusions and point to future work in Section 8.

2. Literature Review

A very-well studied variant of the KEP is the cycle-only version, i.e., a problem instance in which either there are no NDDs or if present, chains are "turned" into cycles by adding an arc from every PDP to NDDs. As a result, the maximum length is the same for both cycles and chains. Different methods, mostly from mixed integer programming (MIP), have been used to model this variant of the KEP in the literature. Abraham et al. (2007) and Roth et al. (2007) proposed two widely known formulations: the *cycle formulation*, which has an exponential number of decision variables, and the *edge formulation*, which has an exponential number of constraints. Constantino et al. (2013) showed that the edge formulation scales substantially worse than the cycle formulation, reaching more than three million constraints in instances with only 50 PDPs, while the cycle formulation leads to memory issues when $K \geq 4$ in medium and high density instances with 100 PDPs or more.

Constantino et al. (2013) proposed the first two MIP formulations where the number of constraints and variables are polynomial in the size of the input, referred to as *compact*

formulations. It was shown that their *extended edge formulation* outperforms their *assignment edge formulation*. Although the cycle formulation is theoretically stronger than both, the extended edge formulation is able to scale in instances where the cycle formulation requires more than three million variables.

B&P (Barnhart et al. 1998) is commonly used to overcome the exponential number of variables of the cycle formulation, yielding the most successful solution methods for the KEP to date (Abraham et al. 2007, Klimentova et al. 2014, Dickerson et al. 2016, 2019). For cycle-only KEP, the state of the art is the B&P-and-cut developed by Lam and Mak-Hau (2020), where the cycle formulation is used as a master problem, and few cuts were added to the master problem in most runs on problem instances from the PrefLib library (Mattei and Walsh 2013). Most instances with 2048 PDPs and $K = 3$ reported total run-time of 2s and for the majority of instances with up to 1024 PDPs, < 1s. For $K = 4$ only instances with up to 1024 PDPs could be solved, taking up to 22min.

In the general version of the KEP, chains are allowed, and usually $L \geq K$ or unbounded. Anderson et al. (2015) proposed two formulations for unbounded chains: a recursive one and one based on the prize-collecting traveling salesman problem (PC-TSP). Instances with up to 1179 PDPs and 62 NDDs were tested with $K = 3$. The recursive formulation outperformed the PC-TSP formulation on a large historical dataset, although, in "difficult" instances, the PC-TSP formulation was more successful. This formulation can be modified to include bounds on the length of chains. However, Plaut et al. (2016b) showed that the PC-TSP is effective when unbounded chains and $K = 3$ are considered, otherwise, B&P-based algorithms outperform it.

Mak-Hau (2015) introduced two formulations (EE-MTZ and SPLIT-MTZ), inspired by Abraham et al. (2007), Constantino et al. (2013) and the well-known Miller-Tucker-Zemlin

constraints. The largest instance presented includes 250 PDPs and 6 NDDs with arc density of 5%. Overall, EE-MTZ optimally solves more instances than SPLIT-MTZ.

Dickerson et al. (2016) proposed three formulations: a compact formulation for the cycle-only version, PIEF, a compact variation of PIEF allowing long chains, HPIEF, and an exponential-sized formulation also allowing long chains, PICEF. The first two formulations are adapted from the extended edge formulation and the last one is inspired by both the extended edge formulation and the cycle formulation. Instances from real match runs and from a realistic simulator were used to test seven algorithms (Abraham et al. 2007, Klimentova et al. 2014, Anderson et al. 2015, Glorie et al. 2014, Plaut et al. 2016b, Dickerson et al. 2016), with $K = 3$ and a time limit of 1h. On real match runs from the United Kingdom, the algorithm by Klimentova et al. (2014) outperformed the others when no NDDs were included. However, for larger (simulated) instances with up to 700 PDPs and 171 NDDs, HPIEF and PICEF were the most effective solution methods. Glorie et al. (2014) discussed polynomial time algorithms for pricing problems, but some arguments were shown to be incorrect (Plaut et al. 2016a). Dickerson et al. (2016) provides insight on the comparative runs related to Glorie et al. (2014).

More recently, McElfresh et al. (2019) proposed the position-indexed traveling-salesman problem formulation (PI-TSP) as part of a robust optimization model in which cycles are handled as in the cycle formulation, and chains are expressed by combining the indexing scheme presented in PICEF and ideas from the PC-TSP formulation. Experimentally, robust solutions were compared to deterministic solutions obtained by PICEF.

Other variations of the KEP include finding robust solutions (Dickerson 2014, McElfresh et al. 2019, Carvalho et al. 2020), solutions that maximize the expected number of transplants (Klimentova et al. 2016, Alvelos et al. 2019) and a matching that optimizes the best outcome over time (Awasthi and Sandholm 2009, Monteiro et al. 2020).

Most of these approaches determine a matching that ultimately maximizes the number of transplants without distinguishing among several optima. However, real-world applications often require multiple criteria to break ties (Glorie et al. 2014, Biró et al. 2019). These tie-breaking criteria consist of an ordered list of objective functions that are solved sequentially by adding rationality constraints every time (Glorie et al. 2014). While our main goal remains to maximize the number of transplants, we consider the prioritization of an exchange type (cycles or chains) as a second objective. We achieve this goal by strategically including columns in our column generation algorithm, allowing us to optimize both objectives in one run.

Table 1 summarizes the literature review for the deterministic KEP. Note that in other variations of the problem, the deterministic optimization algorithms are frequently used as a base line (Dickerson et al. 2012a, Chisca et al. 2018, Ding et al. 2018, Gao 2019).

Table 1 Summary of exact optimization algorithms for the deterministic KEP

Authors	Max #PDPs	Max #NDDs	Arc Density	Methodology	Exchange type	Hierarchical
Abraham et al. (2007)	10000	NA	Unspecified	B&P	Cycles	No
Roth et al. (2007)	1000	NA	Unspecified	MIP	Cycles	No
Constantino et al. (2013) [1]	1000	NA	Unspecified	MIP	Cycles	No
Glorie et al. (2014) [2]	1000	NA	Unspecified	B&P	Cycles	Yes
Klimentova et al. (2014)	2000	NA	Unspecified	B&P	Cycles	Yes
Anderson et al. (2015)[3]	1179	162	19%	B&Cut	Cycles, U. Chains	No
Dickerson et al. (2016)[4]	700	175	Unspecified	MIP	Cycles, B. Chains	No
Lam and Mak-Hau (2020)	2048	NA	32%	B&P	Cycles	No
This work	2048	307	46%	B&P, MIP	Cycles, B. Chains	Yes

[1] Instances with up to 200 PDPs reported an arc density of 70%. [2] Results with NDDs are not interpretable Plaut et al. (2016b). PDPs and NDDs were aggregated. [3] Authors indicated how to bound chains. [4] The UNOS instances with average 231 PDPs and 7 NDDs have an arc density of 9.4%, NA stands for not applicable. "U." stands for unbounded and "B." for bounded chains.

3. Problem Description

Given a set of PDPs $\mathscr{P}$, a set of NDDs $\mathscr{N}$, and positive integers K and L, the KEP can be defined in a digraph $\mathscr{D} = (\mathscr{V}, \mathscr{A})$. A vertex $v \in \mathscr{V}$ is defined for each PDP and NDD, i.e., $\mathscr{V} = \mathscr{P} \cup \mathscr{N}$. For $v_i, v_j \in \mathscr{V}$, there exists an arc $(v_i, v_j) \in \mathscr{A}$ if the donor in vertex v_i is compatible with the patient in vertex v_j, e.g., see Figure 1. Note that, $\mathscr{A} \subseteq \{(v_i, v_j) \mid v_i \in \mathscr{V}, v_j \in \mathscr{P}\}$, that is, there are no incoming arcs to NDDs, neither from PDPs nor from the other NDDs. Each arc (v_i, v_j) is assigned a score w_{ij} representing the suitability/priority of that transplant. Chains and cycles correspond to simple paths and cycles of $\mathscr{D}$, respectively, formally defined as follows:

Definition 1. A chain $p = (v_1, ..., v_\ell)$ is feasible if and only if: (1) $(v_i, v_{i+1}) \in \mathscr{A}$ $\forall i = 1, ..., \ell - 1$, (2) $v_1 \in \mathscr{N}$ and $v_i \in \mathscr{P}$ $\forall i = 2, ..., \ell$, and (3) $\ell \leq L + 1$. Note that for a chain to include L transplants, $L + 1$ vertices are required.

Definition 2. A cycle $c = (v_1, ..., v_k, v_1)$ is feasible if and only if: (1) $(v_i, v_{i+1}) \in \mathscr{A}$ $\forall i = 1, ..., k - 1$ and $(v_k, v_1) \in \mathscr{A}$, (2) $v_i \in \mathscr{P}$ $\forall i = 1, ..., k$, and (3) $k \leq K$.

A feasible solution to the KEP is a matching of donors to patients. To formally introduce this definition, consider $\mathscr{C}_K$ and $\mathscr{C}_L$ as the set of all feasible cycles and chains in $\mathscr{D}$, respectively. Throughout the paper, notation $\mathscr{V}(\cdot)$ and $\mathscr{A}(\cdot)$ will denote the set of vertices and arcs present in the argument $(\cdot)$, respectively. For instance, $\mathscr{V}(c)$ corresponds to the set of vertices in cycle c.

Definition 3. $M(K, L) \subseteq \mathscr{C}_K \cup \mathscr{C}_L$ is a feasible matching of donors to patients if $\mathscr{V}(m_1) \cap \mathscr{V}(m_2) = \emptyset$ for all $m_1, m_2 \in M(K, L)$ such that $m_1 \neq m_2$.

That is, a matching in the KEP corresponds to a collection of feasible chains and cycles where every patient/donor participates in at most one transplant and type of exchange. Thus, the objective of the KEP is to find a matching, whose set of arcs $\mathscr{A}(M(K, L))$ maximizes the total transplant score, i.e., $\sum_{(i,j) \in \mathscr{A}(M(K,L))} w_{ij}$.

It is known that there are two cases in which the KEP is solved efficiently, although the general setting is NP-hard (Abraham et al. 2007, Biró et al. 2009). In the first case, $\mathscr{V} = \mathscr{P}$ and $K = 2$, i.e., only 2-way cycles are allowed, where the problem can be reduced to a maximum weight matching problem in an undirected graph. The second case occurs when K and L are arbitrarily large, where the problem can be solved as a maximum weight matching problem on a bipartite graph.

4. A New Valid Upper Bound through a Lagrangian Decomposition

We introduce a Lagrangian decomposition of the KEP based on an improved and generalized version of an existing integer programming (IP) formulation. Particularly, we show that the Lagrangian decomposition can be used in B&P to provide a valid upper bound on the optimal value of the KEP, stronger than existing bounds. Moreover, we indicate the use of this new bound in our B&P implementation. Lastly, we show an advantage of using the *disaggregated cycle formulation* (Klimentova et al. 2014) over the so-called *cycle formulation* (Abraham et al. 2007, Roth et al. 2007) in B&P.

4.1. IEEF: An Improved Extended Edge Formulation (EEF)

For the cycle-only version of the KEP, Constantino et al. (2013) cloned the input digraph $\mathscr{D}$, $|\mathscr{P}|$ times, drawn by the fact that $|\mathscr{P}|$ is a natural upper bound on the number of cycles in any feasible solution. The idea is then to find a feasible cycle in each copy of $\mathscr{D}$. If the selected cycles across the copies are vertex-disjoint, they represent a feasible matching. Thus, an IP formulation can be built based on this disaggregation of $\mathscr{D}$ to solve the problem. We extend this formulation by including long chains and reducing the number of copies. To this end, we start by showing that any *feedback vertex set* (FVS) provides a valid upper bound on the number of vertex-disjoint cycles in $\mathscr{D}$, whose proof is omitted due to its simplicity.

DEFINITION 4. Given a directed graph $\mathscr{D} = (\mathscr{V}, \mathscr{A})$, $\mathcal{V}^* \subseteq \mathscr{V}$ is a feedback vertex set of $\mathscr{D}$ if the subgraph induced by $\mathscr{V} \setminus \mathcal{V}^*$ is acyclic.

PROPOSITION 1. *Given a digraph $\mathscr{D} = (\mathscr{V}, \mathscr{A})$, let n be the cardinality of the set with the maximum number of vertex-disjoint cycles in $\mathscr{D}$, and $\mathcal{V}^*$ be an FVS. Then, $n \leq |\mathcal{V}^*|$.*

We create one graph copy per vertex in the FVS. We refer to these copies as *cycle copies* and vertices in the FVS as *feedback vertices*. Thus, the smaller the size of the FVS the smaller the number of copies of $\mathscr{D}$. However, finding a minimum FVS is one of the classical NP-hard problems in Karp's seminal paper (Karp 1972). As an alternative, we introduce a heuristic that allows us to find what we referred to as a *K-limited FVS* of "small" cardinality. That is, instead of finding an FVS, whose elements cover all cycles of any size, we find a *vertex set* that covers cycles whose size does not exceed K. Naturally, when K is sufficiently large, this vertex set corresponds to an FVS of $\mathscr{D}$. It is worth noting that finding the smallest set covering is also a hard problem (Karp 1972).

Algorithm 1 provides a mechanism to obtain a K-limited FVS $\mathscr{V}^* = \{v_1^*, ..., v_{|\mathscr{V}^*|}^*\} \subseteq \mathscr{P}$ as well as to construct corresponding cycle copies $\hat{\mathscr{D}}^i = (\hat{\mathscr{V}}^i, \hat{\mathscr{A}}^i)$, indexed by $\hat{I} = \{1, ..., |\mathscr{V}^*|\}$. The procedure takes as input a digraph $\mathscr{D}$ and an ordered set $\mathcal{S}$ of vertices in $\mathscr{D}$, sorted according to a vertex-ordering rule (e.g., maximum in-degree of a vertex). While there is at least one element in $\mathcal{S}$, the first vertex in that set denoted by s_1, is selected as the next potential feedback vertex v^*. Subsequently, the function *CycleCopy* builds a subgraph where all feasible cycles, if any, include v^*, but no other previously declared feedback vertices. For instance, the *CycleCopy* function can take the form of the pairwise-shortest-path-based rule used by Constantino et al. (2013), or the form of decision diagrams as proposed in Section 5.3.2. The output is a subgraph $\tilde{D}$ that may lack some vertices/arcs in D, if it can be determined that they can be removed without ruling out a feasible cycle.

If the set of arcs $\tilde{A}$ is not empty, then a new cycle copy is created. Vertex v^* is removed from $\mathcal{S}$ afterwards, and a new iteration starts.

To illustrate the advantage of Algorithm 1, consider the digraph provided in Figure 2a, $K = 4$, and sorting the vertices in decreasing order of their out-degree, i.e., $\mathcal{S} = \{4, 5, 3, 2, 1\}$. Suppose the pairwise-shortest-path-based rule is used in line 8 of the algorithm as the procedure to build the cycle copies. Let d_{nm} be the length of the shortest path from vertex n to m in $\mathscr{D}$. Under this procedure, vertices $u \in V \setminus \{v^*\}$, for which $d_{v^*u} + d_{uv^*} > K$ are removed from the graph. Likewise, an arc (u, v) is also removed from $\mathscr{D}$, if $d_{v^*u} + 1 + d_{vv^*} > K$. If a directed path between two vertices does not exist, the shortest path is set to infinity.

In the first iteration vertex $v^* = 4$ is selected as a *potential* feedback vertex. After following the pairwise-shortest-path-based rule in line 8, the resulting graph is as shown in Figure 2b. Since the subgraph is not empty, v^* is confirmed as a feedback vertex and a new cycle copy is created. Then, vertex 4 is removed from $\mathcal{S}$ and a new potential feedback vertex is selected. This time, $v^* = 5$ is selected, giving rise to the cycle copy in Figure 2c. Overall, two cycle copies and 15 arcs are obtained. If the vertices in $\mathcal{S}$ were sorted in non-decreasing order of their index as in Constantino et al. (2013), four copies and 21 arcs would be obtained instead.

The pairwise-shortest-path-based rule is not infallible since some unnecessary vertices/arcs can pass the test even though they do not belong to any feasible cycle, e.g., arc (5,1) in Figure 2b. Also, notice that cycle $c = (5, 3, 5)$ is present in both cycle copies. By constructing such copies as decision diagrams (Section 5.3.2), we make sure that every cycle is present only in the copy of its corresponding feedback vertex.

By considering the cycle copies given by Algorithm 1 as digraphs, we can now introduce our extension of the extended edge formulation, which will provide us with a mechanism to compute a valid upper bound on the maximum number of matches.

Algorithm 1: K-limited FVS heuristic

Result: Feedback vertex set $\mathscr{V}^* \subseteq \mathscr{V}$ and graph copies $\hat{\mathscr{D}}^i = (\hat{\mathscr{V}}^i, \hat{\mathscr{A}}^i)\ \forall i \in \hat{I}$

Input: Digraph $\mathscr{D} = (\mathscr{V}, \mathscr{A})$, vertex ordering $\mathcal{S}$, maximum length K

$\mathscr{V}^* = \emptyset$

$\hat{I} = \emptyset$

$i = 0$

while $|\,\mathcal{S}\,| \geq 1$ **do**

  $v^* \leftarrow s_1 \in \mathcal{S}$

  $V \leftarrow \mathscr{V} \setminus \mathscr{V}^*$

  $A \leftarrow \{(u,v) \in \mathscr{A} : u, v \in V\}$

  $\tilde{D} = (\tilde{V}, \tilde{A}) \leftarrow CycleCopy(v^*, D = (V, A))$

  if $\tilde{A} \neq \emptyset$ **then**

    $i \leftarrow i + 1$

    $\hat{I} \leftarrow \hat{I} \cup \{i\}$

    $\hat{\mathscr{D}}^i \leftarrow \tilde{D}$

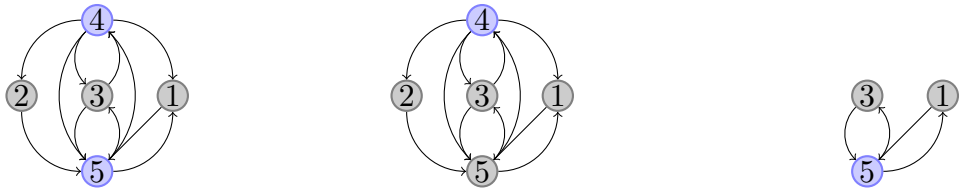
    $\mathscr{V}^* \leftarrow \mathscr{V}^* \cup \{v^*\}$

  end

  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{v^*\}$

end

Because our extension includes chains, we also create copies of the input graph for chains, one for every NDD. Let $\bar{I}$ denote the index set of chain copies, with $|\bar{I}| \leq |\mathscr{N}|$. The i-th chain copy of $\mathscr{D}$, $i \in \bar{I}$, is represented by graph $\bar{\mathscr{D}}^i = (\bar{\mathscr{V}}^i, \bar{\mathscr{A}}^i)$, whose vertex set $\bar{\mathscr{V}}^i = \mathscr{P} \cup \{u_i, \tau\}$ is formed by all PDPs, the i-th NDD, u_i, and a dummy vertex τ, representing a dummy patient receiving a fictitious donation from the paired donor in the last pair of a chain.



(a) $\mathscr{D} = (\mathscr{V}, \mathscr{A})$ (b) $\hat{\mathscr{D}}^1 = (\hat{\mathscr{V}}^1, \hat{\mathscr{A}}^1)$ (c) $\hat{\mathscr{D}}^2 = (\hat{\mathscr{V}}^2, \hat{\mathscr{A}}^2)$

Figure 2 Proposed graph copies. Blue vertices correspond to feedback vertices.

The set of arcs $\bar{\mathscr{A}}^i = (\mathscr{A} \setminus \{(u,v) \in \mathscr{A} \mid u \in \mathscr{N} \setminus u_i, v \in \mathscr{P}\}) \cup \{(v,\tau) \mid v \in \mathscr{P}\}$ removes arcs emanating from NDDs other than u_i and adds one dummy arc from every PDP to τ. Thus, a chain can only be started by the NDD u_i on the ith chain copy. Additionally, let $\bar{\mathscr{C}}^i$ and $\bar{\mathscr{C}}^i_K \subseteq \bar{\mathscr{C}}^i$ be the set of all simple cycles and the set of all feasible cycles in $\bar{\mathscr{D}}^i$, respectively. Lastly, let $\hat{x}^i_{uv}$ and $\bar{x}^i_{uv}$ be decision variables taking the value 1 if arc $(u,v) \in \hat{\mathscr{A}}^i$ and arc $(u,v) \in \bar{\mathscr{A}}^i$ is selected in a cycle and chain, respectively, and 0 otherwise. Then, the improved extended edge formulation is formulated as follows:

$$\max \quad \sum_{i \in \hat{I}} \sum_{(u,v) \in \hat{\mathscr{A}}^i} w_{uv} \hat{x}^i_{uv} + \sum_{i \in \bar{I}} \sum_{(u,v) \in \bar{\mathscr{A}}^i} w_{uv} \bar{x}^i_{uv} \tag{IEEF}$$

$$\text{s.t.} \quad \sum_{i \in \hat{I}} \sum_{(u,v) \in \hat{\mathscr{A}}^i} \hat{x}^i_{uv} + \sum_{i \in \bar{I}} \sum_{(u,v) \in \bar{\mathscr{A}}^i} \bar{x}^i_{uv} \leq 1 \qquad u \in \mathscr{V} \tag{1a}$$

$$\sum_{v:(u,v) \in \hat{\mathscr{A}}^i} \hat{x}^i_{uv} = \sum_{v:(v,u) \in \hat{\mathscr{A}}^i} \hat{x}^i_{vu} \qquad i \in \hat{I},\ u \in \hat{\mathscr{V}}^i \tag{1b}$$

$$\sum_{v:(u,v) \in \bar{\mathscr{A}}^i} \bar{x}^i_{uv} = \sum_{v:(v,u) \in \bar{\mathscr{A}}^i} \bar{x}^i_{vu} \qquad i \in \bar{I},\ u \in \bar{\mathscr{V}}^i \setminus \{u_i, \tau\} \tag{1c}$$

$$\sum_{(u,v) \in \hat{\mathscr{A}}^i} \hat{x}^i_{uv} \leq K \qquad i \in \hat{I} \tag{1d}$$

$$\sum_{(u,v) \in \bar{\mathscr{A}}^i} \bar{x}^i_{uv} \leq L \qquad i \in \bar{I} \tag{1e}$$

$$\sum_{v:(u,v) \in \hat{\mathscr{A}}^i} \hat{x}^i_{uv} \leq \sum_{v:(v^*_i,v) \in \hat{\mathscr{A}}^i} \hat{x}^i_{v^*_i v} \qquad i \in \hat{I};\ u \neq v^*_i \tag{1f}$$

$$\sum_{v:(u,v) \in \bar{\mathscr{A}}^i} \bar{x}^i_{uv} \leq \sum_{v:(u_i,v) \in \bar{\mathscr{A}}^i} \bar{x}^i_{u_i v} \qquad i \in \bar{I};\ u \in \bar{\mathscr{V}}^i \setminus \{u_i, \tau\} \tag{1g}$$

$$\sum_{(u,v)\in\mathscr{A}(c)} \bar{x}^i_{uv} \leq |\mathscr{V}(c)| - 1 \qquad i \in \bar{I},\ c \in \bar{\mathscr{C}}^i \setminus \bar{\mathscr{C}}^i_K \tag{1h}$$

$$\hat{x}^i_{uv} \in \{0,1\} \qquad i \in \hat{I},\ (u,v) \in \hat{\mathscr{A}}^i \tag{1i}$$

$$\bar{x}^i_{uv} \in \{0,1\} \qquad i \in \bar{I},\ (u,v) \in \bar{\mathscr{A}}^i \tag{1j}$$

The objective function maximizes the weighted sum of transplant scores. Constraints (1a) enforce that a donor (paired or single) donates at most one kidney. Constraints (1b) guarantee that in a selected cycle copy, if a PDP receives a kidney, then it donates one to another pair in the same copy. Constraints (1c) ensure that in a chain if a patient in a PDP receives a kidney, his or her paired donor donates either to a PDP in the same chain or to a dummy vertex, as it is the case of the donor in the last PDP of the chain. Constraints (1d) and (1e) limit the number of arcs in a cycle to K and the number of arcs in a chain to L in a copy, respectively. Constraints (1f) forbid the use of the i-th cycle copy unless the i-th vertex in the FVS, v^*_i, is selected in that copy. Likewise, constraints (1g) allow the use of the i-th chain copy only when the singleton donor, u_i, triggers a chain. The presence of cycles (subtours) in chain copies, particularly of those with size higher than K, jeopardize the correctness of the formulation. Therefore, constraints (1h) ensure the elimination of all infeasible cycles from chain copies. These constraints resemble those used in the recursive formulation of Anderson et al. (2015), except that their arc variables are based on the input graph in a model allowing arbitrarily long chains. We note that Constantino et al. (2013) and Klimentova et al. (2014) did not consider infeasible-cycle-breaking constraints in their discussion on how to include NDDs in the EEF, and thus can be deemed incomplete. Infeasible-cycle-breaking constraints, e.g., Constraints (1h), are *necessary* to preserve the correctness of the EEF, and thus that of IEEF. A counter example showing the need of such constraints in EEF to allow chains and a formal proof for Proposition 2 are provided in Appendix A and B, respectively.

PROPOSITION 2. *IEEF is a correct formulation of the matching problem in the KEP.*

4.2. Lagrangian relaxation

Consider introducing the following valid (redundant) constraints to (IEEF):

$$\sum_{(u,v)\in\hat{\mathscr{A}}^i} \hat{x}^i_{uv} \leq 1 \qquad i \in \hat{I}, u \in \hat{\mathscr{V}}^i \tag{2a}$$

$$\sum_{(u,v)\in\bar{\mathscr{A}}^i} \bar{x}^i_{uv} \leq 1 \qquad i \in \bar{I}, u \in \bar{\mathscr{V}}^i \tag{2b}$$

Before justifying these new constraints, let us first approximate the optimal objective value of (IEEF) by relaxing constraints guaranteeing at most one donation from every donor Constraints (1a) and then penalizing their violation by imposing Lagrange multipliers ($\boldsymbol{\lambda}$) in the objective function. This relaxation may allow a vertex to be selected in more than one graph copy, thus, in more than one exchange. However, if a copy is selected, such a vertex can be selected at most once within that copy, due to Constraints (2). Given $\lambda \in \mathbb{R}_+^{|\mathscr{V}|}$, a Lagrangian relaxation to the KEP can be formulated as follows:

$$\mathcal{Z}(\lambda) := \tag{LR1}$$

$$\max \sum_{i\in\hat{I}} \sum_{(u,v)\in\hat{\mathscr{A}}^i} w_{uv}\hat{x}^i_{uv} + \sum_{i\in\bar{I}} \sum_{(u,v)\in\bar{\mathscr{A}}^i} w_{uv}\bar{x}^i_{uv} + \sum_{v\in\mathscr{V}} \lambda_v \left(1 - \sum_{i\in\hat{I}} \sum_{(u,v)\in\hat{\mathscr{A}}^i} \hat{x}^i_{uv} + \sum_{i\in\bar{I}} \sum_{(u,v)\in\bar{\mathscr{A}}^i} \bar{x}^i_{uv}\right)$$

$$\text{s.t. } (1b) - (1j), (2a) - (2b) \tag{3a}$$

As the only constraints linking decision variables associated with different copies are now relaxed, (LR1) can be decomposed by graph copies as follows:

$$\mathcal{Z}(\lambda) = \sum_{i\in\bar{I}} \hat{\mathcal{Z}}^i(\lambda) + \sum_{i\in\hat{I}} \bar{\mathcal{Z}}^i(\lambda) + \sum_{u\in\mathscr{V}} \lambda_u \tag{LR2}$$

where, $\forall i \in \hat{I}$ and $\forall i \in \bar{I}$, we have the following subproblems, respectively:

$$\hat{\mathcal{Z}}^i(\lambda) := \max \left\{ \sum_{(u,v)\in\hat{\mathscr{A}}^i} (w_{uv} - \lambda_u)\hat{x}^i_{uv} \mid (1b), (1d)(1f), (1i), (2a) \right\}, \tag{CC}$$

$$\bar{\mathcal{Z}}^i(\lambda) := \max \left\{ \sum_{(u,v)\in\bar{\mathscr{A}}^i} (w_{uv} - \lambda_u)\bar{x}^i_{uv} \mid (1c), (1e), (1h), (1j), (2b) \right\}. \tag{CH}$$

Given a set of Lagrange multipliers, λ, each subproblem finds either a feasible cycle, (CC), or a feasible chain, (CH), whose vertex assignment has minimum penalty, thus, maximum weight. Observe that the inclusion of the dummy vertex τ is useful to capture the Lagrange multiplier of the last pair in a chain in the objective function of (CH). The Lagrangian dual problem can be defined as $\sigma^{LD} := \min\{\mathcal{Z}(\lambda) : \lambda \in \mathbb{R}_+^{|\mathscr{V}|}\}$. That is, σ^{LD} is the smallest upper bound that can be obtained when the set of Lagrange multipliers favor an assignment of vertices to cycle and chain copies with minimum intersection. If we define $\mathcal{Z}^{LP}$ as the optimal objective value of the linear programming relaxation of (IEEF), it is possible that $\sigma^{LD} < \mathcal{Z}^{LP}$. Moreover, we show that the quality of the bound provided by σ^{LD} is as tight as the one provided by the linear programming relaxation of the disaggregated cycle formulation (Klimentova et al. 2014), one of the formulations in the literature providing the tightest linear relaxation.

PROPOSITION 3. *If $\mathcal{Z}_c^{LP}$ is the optimal objective value of the linear programming relaxation of the disaggregated cycle formulation, then $\sigma^{LD} = \mathcal{Z}_c^{LP}$.*

Proof. Let $\hat{\mathscr{C}}_K^i$ and $\bar{\mathscr{C}}_L^i$ be the set of feasible cycles (including vertex v_i^*) and chains on the i-th graph copy, $i \in \hat{I}, i \in \bar{I}$, respectively. For a cycle $c \in \hat{\mathscr{C}}_K^i$ and chain $p \in \bar{\mathscr{C}}_L^i$, let $w_c = \sum_{(u,v)\in\mathscr{A}(c)} w_{uv}$ and $w_p = \sum_{(u,v)\in\mathscr{A}(p)} w_{uv}$ be the total weight of a cycle and chain, respectively. Then, (LR2) can be reformulated as follows:

$$\min \quad \sum_{i\in\bar{I}} \hat{\mathcal{Z}}^i + \sum_{i\in\hat{I}} \bar{\mathcal{Z}}^i + \sum_{v\in\mathscr{V}} \lambda_v \tag{LR3}$$

$$\hat{\mathcal{Z}}^i \geq w_c - \sum_{v\in\mathscr{V}(c)} \lambda_v \qquad i \in \hat{I}, c \in \hat{\mathscr{C}}_K^i \qquad (z_c^i) \tag{5a}$$

$$\bar{\mathcal{Z}}^i \geq w_p - \sum_{v \in \mathscr{V}(p)} \lambda_v \qquad i \in \bar{I}, p \in \bar{\mathscr{C}}_L^i \qquad (z_p^i) \qquad (5b)$$

$$\hat{\mathcal{Z}}^i \geq 0 \qquad i \in \hat{I} \qquad (5c)$$

$$\bar{\mathcal{Z}}^i \geq 0 \qquad i \in \bar{I} \qquad (5d)$$

$$\lambda_v \geq 0 \qquad v \in \mathscr{V} \qquad (5e)$$

(LR3) finds the optimal value of decision variables $\hat{\mathcal{Z}}^i$, $\bar{\mathcal{Z}}^i$ and λ. The validity of (LR3) relies on the fact that the maximum weight cycle and chain is selected for every graph copy through Constraints (5a) and (5b), met in the equality by the minimization objective. Moreover, since the objective value of (CC) and (CH) can at least be zero (by selecting $\hat{x} = 0$ and $\bar{x} = 0$), Constraints (5c) and (5d) represent a valid lower bound on the objective value of each sub-problem. To finalize the proof, letting z_c^i and z_p^i be the dual variables of constraints (5a) and (5b), respectively, the dual problem of (LR3) is

$$\max \quad \sum_{i \in \hat{I}} \sum_{c \in \hat{\mathscr{C}}_K^i} w_c z_c^i + \sum_{i \in \bar{I}} \sum_{p \in \bar{\mathscr{C}}_L^i} w_p z_p^i \qquad \text{(IDCF)}$$

$$\sum_{i \in \hat{I}} \sum_{c \in \hat{\mathscr{C}}_K^i : v \in \mathscr{V}(c)} z_c^i + \sum_{i \in \bar{I}} \sum_{p \in \bar{\mathscr{C}}_L^i : v \in \mathscr{V}(p)} z_p^i \leq 1 \qquad v \in \mathscr{V} \qquad (6a)$$

$$z_c^i \geq 0 \qquad i \in \hat{I}, c \in \hat{\mathscr{C}}_K^i \qquad (6b)$$

$$z_p^i \geq 0 \qquad i \in \bar{I}, p \in \bar{\mathscr{C}}_L^i \qquad (6c)$$

Note that Constraints $\sum_{c \in \hat{\mathscr{C}}_K^i} z_c^i \leq 1 \;\; \forall i \in \hat{I}$ are omitted from (IDCF) along with their chain counterpart since they are implied by Constraints (6a). Formulation (IDCF) corresponds to the integer linear programming relaxation of the *disaggregated cycle formulation* in Klimentova et al. (2014), noting that in (IDCF) chain variables are included and cycle copies are found though Algorithm 1. Hence, by strong duality it follows that $\sigma^{LD} = \mathcal{Z}_c^{LP}$.

□

Notice that by enforcing integrality of the decision variables, (IDCF) becomes a valid formulation for the matching problem in kidney exchange, similar to the so-called cycle formulation (Abraham et al. 2007, Roth et al. 2007), except that cycles and chains are not found in specific graph copies. Thus, the i index is dropped. It had not been shown before whether there is an advantage to using one formulation over another, particularly for B&P. Note that the dual variables of Constraints (6a) correspond to the Lagrange multipliers λ in (LR2). Therefore, this result shows that when cycles and chains are disaggregated into graph copies, it is possible to obtain a valid upper bound on the objective value by solving (CC) and (CH), and simply plugging their result into (LR2) afterwards. Even if the set of Lagrange multipliers is not optimal, (LR2) provides a valid upper bound, which can be as good as that of the disaggregated cycle formulation or the cycle formulation itself. Thus, even without proving optimality of (IDCF), its dual variables can still be used to obtain a valid upper bound. To the best of our knowledge, the only previous method in the literature obtaining a valid upper bound consists of solving a relaxed problem with $K = L = \infty$ (Abraham et al. 2007), which as mentioned in Section 3 can be solved in polynomial time. It is easy to see that the bound provided by this special case is weaker than that of the presented Lagrangian dual problem. Since (IDCF) can be used as a master problem in column generation, the goal is to use the bound provided by (LR2) when it is not possible to prove optimality of the master problem. In Section 5.4, we show how this new upper bound can be used not just at every node of a branch-and-bound tree.

5. B&P Algorithm

Our approach to KEP is, to the best of our knowledge, the only B&P for KEP valid for long, yet bounded chains. For completeness, we first discuss the general motivation behind B&P. We then detail our B&P implementation. Particularly, we focus on solving the pricing

problems via multi-valued decision diagrams, a solution method novel to cycle and path packing problems in digraphs, and only used before in a transportation scheduling problem (Raghunathan et al. 2018), to the best of our knowledge.

5.1. Background

For large instances, the cardinality of $\hat{\mathscr{C}}_K^i$ and $\bar{\mathscr{C}}_L^i$ becomes prohibitive up to the point that we cannot exhaustively state all decision variables in (IDCF) or constraints in (LR3). Instead of considering the full set of variables in a linear program, *column generation* works with a small subset of variables (*columns*), forming the well-known *restricted master problem* (RMP). If by duality theory a column is missing in the RMP, a cycle or chain with positive reduced cost must be found and added to the RMP to improve its current objective value. To find such a column(s) one can solve tailored *pricing problems*, which return either a "positive-price" cycle (chain) or a certificate that none exists. RMP and pricing problems are solved iteratively, typically, until strong duality conditions are satisfied. As the solution of the RMP may not be integer, column generation is embedded into a branch-and-bound algorithm to obtain an optimal solution, yielding a B&P algorithm.

5.2. Restricted master problem

(IDCF) or its dual counterpart (LR3) can serve as a master problem in our decomposition. To estimate the performance of both formulations, we solved (IDCF) and (LR3) via a column generation algorithm and a cutting plane method, respectively, for each of the 10 PrefLib instances with 256 PDPs with no NDDs and $K = 3$. In each case, subproblems were solved as shown in Section 6. Both algorithms solved their corresponding master problem below a second, thus, there is no significant difference among the two, although in 7 out of 10 cases (IDCF) was faster. Therefore, we select (IDCF) as our RMP.

5.3. MDDs for Pricing Problems

Finding an assignment of vertices to a graph copy while minimizing penalization is equivalent to finding a positive-price cycle or chain. Therefore, whether we take the primal or dual problem of our RMP, subproblems can take the form of (CC) and (CH). Notice that subproblems can be formulated differently (see, e.g., Anderson et al. (2015), McElfresh et al. (2019)), or not being solved through MIP at all, provided that there is a faster algorithm. Solution methods to pricing problems in the literature for the cycle-only version include heuristics (Abraham et al. 2007), MIP (Roth et al. 2007), or a combination of exact and heuristic algorithms (Klimentova et al. 2014, Lam and Mak-Hau 2020). Also, Glorie et al. (2014) and Plaut et al. (2016b) tackled this version by using a modified Bellman-Ford algorithm in a reduced graph, which applies to the cycle-only version (Plaut et al. 2016a). Pricing algorithms including long chains have been less studied. Dickerson et al. (2019) improved the current solver of a transplant community in the United States, UNOS, based on the previous work of Abraham et al. (2007). The authors studied heuristics for generating positive-price short cycles and arbitrarily large chains when their expected utility is maximized. Tested instances had at most 1000 PDPs and 100 NDDs. Their results show that 38 out of 128 instances with 900 PDPs and 90 NDDs could be solved within an hour. Next, we show how to solve the pricing problems by means of MDDs for the general version of the KEP.

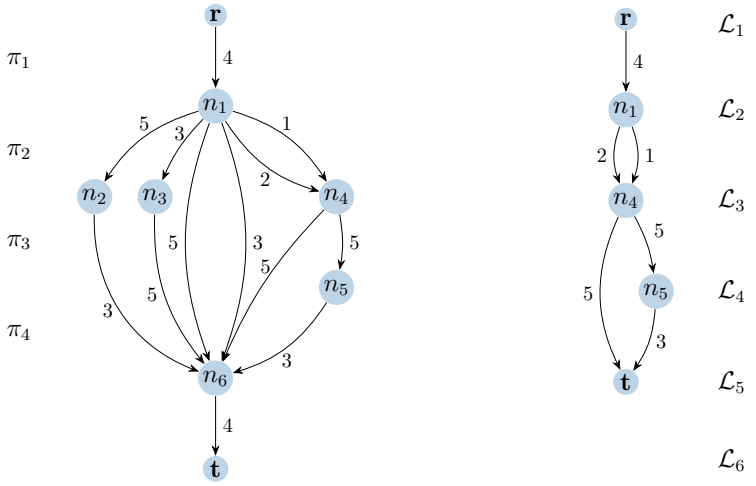
5.3.1. Decision Diagrams A decision diagram is a graphical data structure, used in optimization to represent the solution set of a given problem. Particularly, a decision diagram is a rooted, directed and acyclic layered graph, where (if *exact*), every chain from the root node $\mathbf{r}$ to a terminal node $\mathbf{t}$ has a one-to-one correspondence with a solution in the feasible space of the optimization problem. If the objective function is arc separable, then

a shortest-path-style algorithm can be used to find the optimal objective value. When the decision variables represented in the diagram are binary, the resulting one is a *binary decision diagram* (BDD). Some applications of BDDs in the literature include finding improved optimization bounds (Bergman et al. 2014), solving two-stage stochastic programs (Lozano and Smith 2018) and solving pricing problems in graph coloring (Morrison et al. 2016). On the other hand, when the domain of decision variables represented in the diagram includes three or more values, the decision diagram is an MDD. Cire and van Hoeve (2013) proposed solving sequencing problems using MDDs and showed primary applications in scheduling and routing, (see also Kinable et al. (2017), Castro et al. (2020)). The only work we are aware of, in which MDDs are used to solve pricing problems, is by Raghunathan et al. (2018). They tackle a last-mile transportation problem in which passengers reach their final destination by using a last-mile service system linked to the terminal station of a mass transit system. It is assumed that a desired arrival time of each passenger is known in advance. For each destination, the authors build MDDs for the subset of passengers who share the same destination. They distinguish between one-arcs and zero-arcs, to refer to a passenger being aboard a vehicle or not, respectively. A path in these MDDs represents a partition of passengers who belong to the same trip and the time at which they depart to their final destination. Instead of a partition of elements, we find feasible cycles or chains in each graph copy, as described next.

5.3.2. Construction of MDDs for the KEP Decision diagrams can be constructed by finding the *state transition graph* of a dynamic programming (DP) formulation and reducing it afterwards (see Cire and van Hoeve (2013), Hooker (2013) for more details). We model the pricing problems through DP by formulating two models; one for cycles and the other for chains. Particularly, a DP model is formulated for each cycle copy in (CC),

an for each chain copy in (CH). A *state* in these models represents the vertices $v \in \mathscr{V}$ visited at previous *stages*, where a stage corresponds to the position of a vertex in a cycle or chain. *Controls* $\hat{h}^{|K|}$ and $\bar{h}^{|L|}$ take the index value of a vertex $v \in \mathscr{P}$ and vertex $v \in \mathscr{V}$, indicating that it is assigned to a cycle or chain, respectively, in the position given by the control index, i.e., in position $k \leq K$ of a cycle or position $\ell \leq L$ of a chain. As the domain of $\hat{h}$ and $\hat{h}$ variables contains three or more values, the resulting diagram is an MDD.

It is in this context where the FVS (Section 4) becomes particularly relevant. The goal is to create as many *cycle MDDs* as cycle copies, where each MDD includes feasible cycles with their corresponding vertex $v_i^* \in \mathscr{V}^*$. To this end, line 8 in Algorithm 1 is replaced by the construction of a decision diagram associated to v^* and thus, $\hat{\mathscr{D}}^i$ becomes an MDD. As for chains, a *chain MDD* is created for every chain copy so as to find positive-price chains from every NDD. The MDD $\hat{\mathcal{G}}^i = (\hat{\mathcal{N}}^i, \hat{\mathcal{A}}^i)$ for the i-th cycle copy of $\mathscr{D}$ has its node set $\hat{\mathcal{N}}^i$ partitioned into K layers, $\mathcal{L}_1, ..., \mathcal{L}_K$, corresponding to the decision of which PDP belongs to the k-th position in a cycle, denoted by π_k, plus two terminal layers $\mathcal{L}_{K+1}$ and $\mathcal{L}_{K+2}$ representing the completion of a feasible cycle. Layers $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_{K+1}$ and $\mathcal{L}_{K+2}$ have a single node each. Every arc $a \in \hat{\mathcal{A}}^i$ has an associated label $val(a) \in \hat{\mathscr{V}}^i$ such that $\pi_k = val(a)$ corresponds to assigning vertex $val(a)$ to the k-th position of a cycle. Since a cycle in the i-th cycle copy starts with the i-th feedback vertex, then $\pi_1 = v_i^*$. Thus, an arc-specified path $(a_1, ..., a_k)$ from nodes $\mathbf{r}$ to $\mathbf{t}$ defines the PDP sequence $(\pi_1, , ..., \pi_k) = (val(a_1), ..., val(a_k))$, equivalent to a feasible cycle in $\mathscr{D}$. On the other hand, the set of nodes $\bar{\mathcal{N}}^i$ for the i-th MDD of a chain copy, $\bar{\mathcal{G}}^i = (\bar{\mathcal{N}}^i, \bar{\mathcal{A}}^i)$, is partitioned into $L+2$ layers, with a single node in the first and last layer, representing the start and end of a chain, respectively. Recall that for a chain to involve L transplants, $L+1$ vertices $v \in \mathscr{V}$ are required. Likewise, an arc $a \in \bar{\mathcal{A}}^i$ has a label $\pi_\ell = val(a)$ indicating the vertex $v \in \bar{\mathscr{V}}^i$ at the ℓ-th position in a chain,



(a) Vertex 4 is a PDP. Exact decision diagram, $K = 4$

(b) Vertex 4 is an NDD. Restricted decision diagram, $L = 3$

Figure 3 MDDs for the example of Figure 2b.

noticing that $\pi_1 = u_i$. A path starting at the root node **r** and ending at a node on the third layer or higher represents a feasible chain, since its length is at least one.

Figure 3a depicts all possible sequences of PDPs $\pi_1, \pi_2, ..., \pi_k$ encoding feasible cycles on the graph copy of Figure 2b. A value π_k placed on an arc corresponds to the k-th vertex $v \in \mathscr{P}$ in a cycle covered by vertex 4. For instance, the path $(\mathbf{r}, n_1, n_3, n_6, \mathbf{t})$ in Figure 3a encodes the cycle $c = \{4, 3, 5, 4\}$ in Figure 2b. See Cire and van Hoeve (2013) for details on MDD construction.

Since exact MDDs can grow exponentially large, it might be necessary to limit their size, turning them into *restricted* decision diagrams. A decision diagram is called restricted if the set of solutions corresponding to its **r**-**t** paths is a subset of the entire feasible set of solutions. Figure 3b shows a restricted MDD as if vertex 4 in Figure 2b were an NDD. In this example, the MDD is restricted to have chains including only two out of the four vertices receiving an arc from vertex 4; namely, vertices 1 and 2.

5.3.3. Finding a positive-price column Let $\delta^i_-(n_s) \subset \hat{\mathcal{A}}^i$ be the set of incoming arcs to a node n_s in $\hat{\mathcal{N}}^i$ and $\ell(a)$ the layer index of the source node of arc a, e.g, in Figure 3a

$\delta^i_-(n_4) = \{(n_1,n_4)^1,(n_1,n_4)^2\}$ and $\ell((n_1,n_4)^1) = 2$, where the superscripts distinguish the two arcs coming into n_4, one with $\pi_2 = 1$ and the other with $\pi_2 = 2$. We define the recursive function values of an arc $a = (n_s, n_{s'})$ for the i-th MDD for cycles and chains, $\hat{\eta}^i(a)$ and $\bar{\eta}^i(a)$, respectively, as the maximum reduced cost of all paths ending at a:

$$\hat{\eta}^i(a) = \bar{\eta}^i(a) = \begin{cases} 0 & n_s = \mathbf{r} \quad (7a) \\ \max\limits_{a' \in \delta^i_-(n_s)} \left\{ \hat{\eta}^i(a') + w_{val(a'),val(a)} - \lambda_{val(a')} \right\} & \text{otherwise} \quad (7b) \end{cases}$$

The recursive function (7) is valid by Bellman's principle of optimality since the reduced cost of a cycle (or chain) in the i-th MDD, $\hat{r}^i_c$ (or $\bar{r}^i_p$), is arc separable (Glorie et al. 2014) and the portion taken by every arc only depends on the previous PDP or NDD in the sequence. Thus, the maximum reduced cost of a cycle and chain, $\hat{\eta}^i$ and $\bar{\eta}^i$, respectively, is given by

$$\hat{\eta}^i = \max\left\{0, \hat{\eta}^i((n,\mathbf{t}))\right\} \qquad i \in \hat{I} \qquad (8a)$$

$$\bar{\eta}^i = \max\left\{0, \max_{a \in \bar{A}^i : \ell(a) \geq 3} \bar{\eta}^i(a) - \lambda_{val(a)}\right\} \qquad i \in \bar{I} \qquad (8b)$$

Recursion (8a) computes the maximum reduced cost of a cycle at the terminal node $\mathbf{t}$, since all paths need to reach $\mathbf{t}$ to close it up. In this case, n is the only node on the $K+1$ layer with an arc pointing to $\mathbf{t}$ (e.g., node n_6 in Figure 3a). For chains, on the other hand, any portion of a path in $\bar{\mathcal{G}}^i$ is a feasible path, for which the longest path can be found at any layer of the MDD where the length of a chain (in terms or arcs in $\bar{\mathscr{A}}^i$) is at least one. The term subtracted in Equation (8b) captures the Lagrange multiplier of the last pair in a chain. We know from (LR3) that $\lambda_v \in \mathbb{R}_+$ for all $v \in \mathscr{V}$, thus, if the Lagrange multiplier of a given vertex $v \in \mathscr{V}$ is large enough to lead to a negative-price path at node $\mathbf{t}$, we may need to cut that path short at some previous vertex in the sequence to obtain a positive-price chain. For instance, in Figure 3b consider $\lambda_2 = \lambda_3 = 5$, all the other Lagrange multipliers

set to zero and $w_{uv} = 1$ for all $(u, v) \in \mathscr{A}$. Sequence $(4, 1, 5)$ representing a 2-length chain in $\bar{\mathscr{D}}^i$ is contained in $(4, 1, 5, 3)$. Clearly, the former yields the highest reduced cost of a chain in Figure 3b. Thus, $\bar{\eta}^i = \bar{\eta}^i((n_4, n_5)) = 2$.

Next, we show a series of results on the complexity of computing a positive-price column via MDDs. The corresponding proofs are presented in Appendix B.

PROPOSITION 4. Given the reduced costs $\hat{r}^i_c$ and $\bar{r}^i_p$ expressed as an arc-separable function for all $(n_s, n_{s'}) \in \hat{\mathcal{A}}^i$ and $(n_s, n_{s'}) \in \bar{\mathcal{A}}^i$, a positive-price cycle, if one exists, can be found in time $\mathcal{O}(\sum_{i \in \hat{I}} \sum_{(n_s, n_{s'}) \in \hat{\mathcal{A}}^i} |\delta^i_-(n_s)|)$. Similarly, a positive-price chain can be found in $\mathcal{O}(\sum_{i \in \bar{I}} \sum_{(n_s, n_{s'}) \in \bar{\mathcal{A}}^i} |\delta^i_-(n_s)|)$.

PROPOSITION 5. The size of the input $\sum_{i \in \hat{I}} \sum_{(n_s, n_{s'}) \in \hat{\mathcal{A}}^i} |\delta^i_-(n_s)|$ grows as $|\hat{\mathscr{V}}^i|^{K+1}$ does.

PROPOSITION 6. The size of the input $\sum_{i \in \bar{I}} \sum_{(n_s, n_{s'}) \in \bar{\mathcal{A}}^i} |\delta^i_-(n_s)|$ grows as $|\bar{\mathscr{V}}^i|^{L+2}$ does for bounded chains and as $|\bar{\mathscr{V}}^i|!$ when $L \to \infty$.

Despite potentially very large diagram sizes in general, there are three reasons for which finding a positive-price column can still be done efficiently in practice. First, even though $|\mathscr{V}|$ can be large, arc density of $\mathscr{D}$ for real KEP instances is below 50% (Saidman et al. 2006, Anderson et al. 2015, Dickerson et al. 2016). Second, for small values of K and L, it is possible to reduce considerably the size of the input by selecting an FVS, $\mathscr{V}^*$, with small cardinality. Lastly, MDDs are reduced significantly after the state transition graph is obtained (Cire and van Hoeve 2013), e.g., see Figure 3a.

5.4. Branching scheme

The search tree may have exponential depth when branching is done on possibly every cycle, thus, branching on arcs is usually preferred. If $\mathscr{D}$ is a complete graph, there can be up to $|\mathscr{P}||\mathscr{P} - 1| + |\mathscr{P}||\mathscr{N}|$ arcs in $\mathscr{A}$. On the other hand, branching on arcs in $\hat{\mathscr{A}}^i$ and $\bar{\mathscr{A}}^i$ implies that there are up to $(|\bar{I}| + |\hat{I}|)|\mathscr{P}||\mathscr{P} - 1| + |\bar{I}||\mathscr{P}||\mathscr{N}|$ arcs across all graph

copies. Among the two arc-based schemes, branching on arcs in $\mathscr{A}$ results in a lower depth branching tree. We therefore choose this option as our branching scheme.

Particularly, on every fractional node of the search tree we branch on an arc $(u,v) \in \mathscr{A}$ whose fractional value is closest to 0.5. That is, we generate two children, one in which the arc is prohibited and another in which the arc is selected. When banning an arc from the RMP, we modify Equation (7b) by replacing w_{uv} with a sufficiently large negative number M. By doing so, the length of any path in any copy traversing that arc approaches negative infinity, thereby ruling it out due to the definition of $\hat{\eta}^i$ and $\bar{\eta}^i$. On the other hand, enforcing an arc $(u,v) \in \mathscr{A}$ requires the inclusion of the following constraint in the RMP:

$$\sum_{i \in \hat{I}} \sum_{c \in \hat{\mathscr{C}}_K^i : (u,v) \in \mathscr{A}(c)} z_c^i + \sum_{i \in \bar{I}} \sum_{p \in \bar{\mathscr{C}}_L^i : (u,v) \in \mathscr{A}(p)} z_p^i = 1 \qquad (\mu_{(u,v)}) \tag{9}$$

The addition of constraint (9) changes the reduced cost of a chain and cycle. If we let $\mathscr{A}^* \subseteq \mathscr{A}$ be the set of selected arcs in a branch-and-bound node, the reduced cost of a column in the i-th MDD, is now given by

$$\hat{r}_c^i = w_c - \sum_{v \in \mathscr{V}(c)} \lambda_v - \sum_{(u,v) \in \mathscr{A}^* \cap \mathscr{A}(c)} \mu_{(u,v)} \qquad c \in \hat{\mathscr{C}}_K^i \tag{10a}$$

$$\bar{r}_p^i = w_p - \sum_{v \in \mathscr{V}(p)} \lambda_v - \sum_{(u,v) \in \mathscr{A}^* \cap \mathscr{A}(p)} \mu_{(u,v)} \qquad p \in \bar{\mathscr{C}}_L^i \tag{10b}$$

where $\mu_{(u,v)}$ is the dual variable of Constraint (9). Thus, if $(val(a'), val(a)) \in \mathscr{A}^*$, then $\mu_{(val(a'),val(a))}$ can be subtracted from recursive expression (7b) to account for Constraint (9) in the RMP.

If the solution of the RMP is fractional, we branch, and then apply column generation to the resulting node. After optimally solving the RMP, the upper bound given by its objective value is compared to the best lower bound found. If the former is lower, that branch is pruned. Otherwise, a lower bound is obtained by granting integrality to the decision

variables (columns) in the RMP and re-solving it. Whenever a lower bound matches the best upper bound, optimality is achieved. If, due to time limitations, it is not possible to solve the RMP to optimality, (LR2) is solved (see Section (6)) to derive a valid upper bound. To this end, if $\mathscr{A}^* \neq \emptyset$, the second summation of (10) is subtracted from the objective function of (CC) and (CH).

6. General Solution Framework

In our solution framework, a combination of exact and restricted MDDs is used so that once built they are stored in computer memory and retrieved every time pricing problems need to be solved. As a result, we cannot solely rely on MDDs to prove optimality of the RMP. We introduce a three-phase solution framework consisting of a search through MDDs for both cycles and chains (Phase 1), a cutting plane algorithm to search for positive-price chains and cycles, whose final goal is to prove that no more positive-price chains exist (Phase 2), and a two-step search to find a positive-price cycle, if any (Phase 3). Figure 4 illustrates the framework.

6.1. Phase 1: Solving the pricing problems via MDDs

Building the MDDs is the first step. We parameterize some aspects to make a reasonable usage of computer memory (Section 7). Particularly, if $K \geq 4$ and $|\mathscr{P}| > 500$, we build restricted MDDs by considering a maximum cycle length of 3 in 90% of the decision diagrams, while in the remaining 10% we keep the true value of K. If $K \leq 4$ and $|\mathscr{P}| \leq 500$, we build exact MDDs. When constructing a transition state graph for chains, we explore at most 20% of vertices receiving an arc from $v \in \mathscr{V}$, unless $|\mathscr{N}| > 250$, in which case we explore only 10%. In all cases, the maximum length of chains (in terms of arcs) considered in the construction of MDDs is also 3, regardless of the true value of L. After the MDDs are built, we store them in memory and use them to solve the pricing problems as depicted by Figure 4, in integration with Phases 2 and 3.

6.2. Phase 2: A longest path formulation for chains and cycles

We use a longest path problem as a relaxation of (CH) in which the goal is to find an **s**-**t** path or a cycle, both corresponding to feasible positive-price columns. To this end, let us define $\mathscr{D}' = (\mathscr{V}', \mathscr{A}')$ as a digraph whose vertex set $\mathscr{V}' = \mathscr{V} \cup \{\mathbf{s}, \mathbf{t}\}$ has two dummy vertices **s** and **t** such that the set of arcs $\mathscr{A}' = \mathscr{A} \cup \{(\mathbf{s}, u) \cup (v, \mathbf{t}) \mid u \in \mathscr{N}, v \in \mathscr{P}\}$ connects dummy vertex **s** to NDDs and PDPs to vertex **t**. For arcs including a dummy vertex, their weight is set to zero. Moreover, let $\mathscr{C}'$ be the set of all simple cycles and $\mathscr{C}'_K \subseteq \mathscr{C}'$ be the set of feasible cycles in $\mathscr{D}'$. As defined before, $\mathscr{A}^*$ is the set of selected arcs. A decision variable y_{uv} takes the value 1 if arc $(u, v) \in \mathscr{A}'$ is selected, and 0 otherwise. Thus, a relaxed longest path formulation at some node in the branching tree is

$$\mathcal{Z}^{\text{(LPH)}} := \max \sum_{(u,v)\in\mathscr{A}':u\neq\mathbf{s}} (w_{uv} - \lambda_u) y_{uv} - \sum_{(u,v)\in\mathscr{A}^*} \mu_{(u,v)} y_{uv} \tag{LPH}$$

$$\text{s.t.} \sum_{v:(u,v)\in\mathscr{A}'} y_{uv} - \sum_{v:(v,u)\in\mathscr{A}'} y_{vu} = \begin{cases} 1, & u = \mathbf{s} \\ 0, & u \in \mathscr{V} \\ -1, & u = \mathbf{t} \end{cases} \tag{11a}$$

$$\sum_{(u,v)\in\mathscr{A}'} y_{uv} \leq 1 \qquad u \in \mathscr{V} \tag{11b}$$

$$y_{uv} \in [0, 1] \qquad (u, v) \in \mathscr{A}' \tag{11c}$$

Although a solution of (LPH) may lead to a path using more than L arcs, or a solution with subtours, or a non-integer solution, we see these downsides as an opportunity to either find efficiently a positive-price chain (cycle) missed in the first phase or prove that none exists. Because (LPH) is a relaxation of (CH), whenever the objective value of (LPH) is zero, so is the objective value of (CH). Note that even without subtour-elimination constraints, a solution of (LPH) guarantees a path (may be fractional) from vertex **s** to **t**

due to flow-balance Constraints (11a). The solution may also have subtours representing positive-price cycles useful for the RMP. Particularly, if y^* is an optimal solution of (LPH), we check arcs $(u,v) \in \mathscr{A}'$ for which $y^*_{uv} \geq 0.9$ when searching for positive-price columns. Lastly, enforcing two dummy arcs, one going out of $\mathbf{s}$ and one coming into $\mathbf{t}$, requires us to adjust the right-hand side of Constraints (12b), resulting in the following constraints:

$$\sum_{(u,v)\in c} y_{uv} \leq |c| - 1 \qquad c \in \mathscr{C}' \setminus \mathscr{C}'_K \tag{12a}$$

$$\sum_{(u,v)\in \mathscr{A}'} y_{uv} \leq L + 2 \tag{12b}$$

$$y_{uv} \in \{0,1\} \qquad (u,v) \in \mathscr{A}' \tag{12c}$$

Therefore, the goal is to solve first a linear program without Constraints (12), check the solution for positive-price and *feasible* columns, and only then add (12) if need be. Experimentally, we observed that the "warmed-up" Lagrange multipliers resulting from Phase 1 allow us to relax integrality constraints and yet obtain an integer solution in many cases. Figure 4 shows how (LPH) + Constraints (12) is solved via a cutting plane algorithm integrated with the other two phases during column generation.

6.3. Phase 3: A two-step procedure for cycles

If a positive-price cycle still exists in $\mathscr{D}$ but not found in Phases 1 and 2, we perform an exhaustive enumeration of cycles for as long as a predetermined time limit is not exceeded, otherwise, the exhaustive search is terminated and a MIP is solved instead. Note that Phase 2 guarantees to find any positive-price chain, if does exists. Therefore, in Phase 3, we only have to search positive-price cycles.

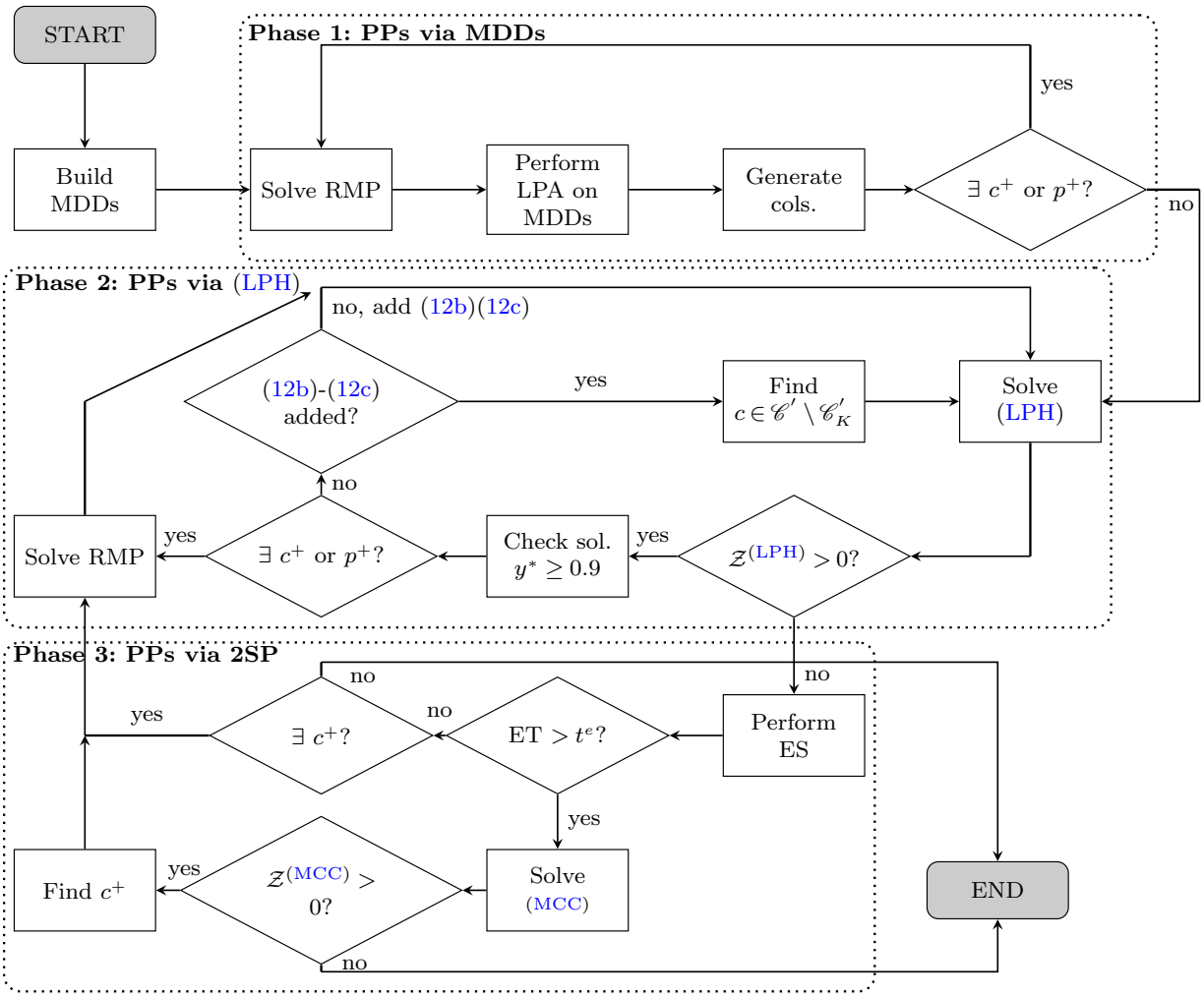


Figure 4 **Flow chart of column generation for $K \geq 3$, $L \in \mathbb{Z}_+$. PPs stands for pricing problems, LPA for longest-path algorithm, cols. for columns, ES for exhaustive search, ET for elapsed time, c^+ (p^+) for positive-price cycle (chain), and 2SP for the two-step procedure described in Phase 3.**

6.3.1. Exhaustive search We perform a depth-first search on every cycle copy $\hat{\mathscr{D}}^i$, where a feasible cycle is rooted at $v_i^* \in \mathscr{V}^*$. First, we sort and search over the graph copies in increasing order of their λ_v values. Next, we traverse $\hat{\mathscr{D}}^i$, and when v_i^* is the leaf node of a path from the root and it is found at a position between 3 and $K+1$, that path constitutes a feasible cycle c. If $\hat{r}_c^i > 0$, the cycle has a positive reduced cost and it is sent to the RMP. Abraham et al. (2007) and Lam and Mak-Hau (2020) implemented a similar search, although unlike them, our paths are rooted at vertices in FVS. Despite of searching

cycles on trees rooted only at a subset of vertices, exhaustive enumeration becomes a bottleneck for large instances. Therefore, whenever a time threshold, t^e, is surpassed we shift to solving the MIP provided next.

6.3.2. A MIP for cycles (MCC) finds a feasible cycle in $\mathscr{D}$ with maximum reduced cost at some branch-and-bound node. If found, the cycle is sent to the RMP. Note that Constraint (13c) guarantees at most K selected arcs, whether they are distributed in multiple (smaller) cycles or not. Thus, there is no need for subtour-elimination constraints. (MCC) is formulated as follows:

$$\mathcal{Z}^{\text{(MCC)}} = \max \sum_{(u,v)\in\mathscr{A}} (w_{uv} - \lambda_u) y_{uv} - \sum_{(u,v)\in\mathscr{A}^*} \mu_{(u,v)} y_{uv} \tag{MCC}$$

$$\text{s.t.} \sum_{v:(u,v)\in\mathscr{A}} y_{uv} - \sum_{v:(v,u)\in\mathscr{A}} y_{vu} = 0 \qquad u \in \mathscr{V} \tag{13a}$$

$$\sum_{v:(u,v)\in\mathscr{A}} y_{uv} \leq \begin{cases} 1, & u \in \mathscr{P} \\ 0, & u \in \mathscr{N} \end{cases} \tag{13b}$$

$$\sum_{(u,v)\in\mathscr{A}} y_{uv} \leq K \tag{13c}$$

$$y_{uv} \in \{0,1\} \qquad (u,v) \in \mathscr{A} \tag{13d}$$

6.3.3. Algorithmic details In the three-phase solution framework, after building the MDDs, the pricing problems are solved to optimality as follows. During Phase 1, we compute $\hat{\eta}^i$ and $\bar{\eta}^i$ for all $i \in \bar{I}$ and $i \in \hat{I}$, respectively, and add the positive-price columns found to the RMP after every iteration. We encourage the use of chains by returning up to 15 positive-price chain columns to the RMP, whereas only one from every cycle MDD. Note that in both cases, recursions (8a)-(8b) only need to be computed once. In Phase 2, we delay the inclusion of Constraints (12), until their addition is mandatory to find a positive-price chain column. Every time a solution is checked during Phase 2, a positive-price cycle

column is searched when failing to find a chain column. As for Phase 3, we set $t^e = 20$s to exhaustively find a positive-price cycle or reach the end with none. If the time threshold is exceeded, we proceed to solve (MCC) and resolve the RMP, if needed. When there are no NDDs present in the input graph $\mathscr{D}$, we simply skip Phase 2. Likewise, because MDDs are exact when $K \leq 4$ and $|\mathscr{P}| \leq 500$ and no NDDs, Phase 2 and 3 are dropped and a valid upper bound (LR2) is provided at the end, in case pricing problems are not solved to optimality within the time limit. In other cases, pricing problems could be solved, e.g., via MIP, but such implementation is out of the scope of our algorithm. Lastly, note that the procedure given in Figure 4 can be easily adapted to the case $L = \infty$, since it suffices to remove Constraint (12b).

7. Computational Experiments

MDDs as well as our B&P algorithm (BP_MDD) are implemented in C++ and experiments are conducted on a machine with Debian GNU/Linux as operating system and a 3.60GHz processor Intel(R) Core(TM) with 120 GB RAM, of which we allocated a maximum of 8 GB for BP_MDD (including the construction of MDDs) and 60 GB for state-of-the-art solution methods. CPLEX 12.10 is used as the LP/MIP solver. We investigate the extent up to which leading approaches could scale in practice, therefore, we allocated a larger RAM capacity. BP_MDD is compared against the state-of-the-art PICEF and HPIEF solution methods (Dickerson et al. 2016). The PICEF and HPIEF solvers are retrieved from the original authors, where HPIEF is the model with full reduction. They call Gurobi 7.5.2 to solve MIP models. Although different LP/MIP solvers may add noise to the analysis, in the latest history of benchmark sets, Gurobi is the lead (Mittelman 2020). It is worth noting that Lam and Mak-Hau (2020) hold the state-of-the-art solver for the cycle-only version. They tested their algorithm on the same library as ours, PrefLib (Mattei and

Walsh 2013), achieving total run-times up to 33s for instances with 2048 PDPs and mostly less than a second for instances with 1024 PDPs, when $K = 3$. For $K = 4$ with 1024 PDPs, their maximum run-time among the 9 instances optimally solved was 21.8min. The total run-time of our algorithm, although small, is larger than that reported by Lam and Mak-Hau (2020), particularly when $K = 3$. However, since Lam and Mak-Hau (2020) did not consider larger values of K, and their approach is for cycles only, we do not compare to them in this analysis. When $L = 0$, the formulation by McElfresh et al. (2019) reduces to the cycle formulation, and so does PICEF. Therefore, we compare BP_MDD with HPIEF and PICEF and we will refer to them as "solvers".

7.1. Instances

For our subsequent analysis, we used the PrefLib repository (Mattei and Walsh 2013), whose instances were generated by Dickerson et al. (2012b) based on data from KPDPs in the United States. The first group, referred to as KEP_N0, has 80 instances, split into 8 subsets, each with 10 instances and no NDDs; and hence, only cycles are considered. In this group, $|\mathscr{P}| \in \{16, 32, 64, 128, 256, 512, 1024, 2048\}$ and their arc density varies from 10% to 32%.

The second group of instances, referred to as KEP_N, is split into 23 subsets with 10 instances in each. NDDs are present, thus, both cycles and chains are considered in the solution. For these instances, $16 \leq |\mathscr{P}| \leq 2048$, $1 \leq |\mathscr{N}| \leq 307$ and the arc density is within 23% and 46%.

7.2. Computational performance

PICEF, HPIEF and BP_MDD are evaluated on the two groups of instances. Each instance in KEP_N0 is solved for $K \in \{3, 4, 5, 6\}$, totaling 320 runs per solver. For KEP_N , we set $K \in \{3, 4\}$ and $L \in \{3, 4, 5, 6\}$. The total number of runs per solver in this second group is

very large, $10 \times 23 \times 2 \times 4 = 1840$ runs. Therefore, we proceed as follows: For each run, if the RAM usage exceeds the limit, the solver stops and aborts the rest of instances in that run's subset, which we presume would lead to the same memory issues. Regardless of the group, for every run we set a time limit of 30min. Only one thread was used for all the experiments and the rule to find the K-limited FVS is the maximum in-degree. An analysis on multiple criteria to obtain a K-limited FVS from Algorithm 1 and their implications on performance and scalability is presented in Appendix C.

Figure 5 plots the number of instances in KEP_N0 solved to optimality up to discrete points in time before reaching the 30-min limit, which shows the outperformance of our algorithm over the state of the art. The time reported includes both the total MDD construction time and the B&P algorithm time.

When $K = 3$, BP_MDD and PICEF solve all 80 instances in under 20min, followed by HPIEF under 26min. When $K \in \{4, 5, 6\}$, BP_MDD solves all 80 instances under 25min. Both PICEF and HPIEF perform poorly as K increases. RAM usage of BP_MDD did not exceed 4 GB in these experiments, while PICEF and HPIEF surpassed 60 GB. Instances that run out of memory accumulated more than 37 million variables. By definition PICEF is exponential in the number of variables, thus, this result is not surprising. However, HPIEF is polynomial in terms of the number of variables and constraints, even more so the full-reduction HPIEF we compare our algorithm against, and yet dimensionality is clearly a challenge. In 94% of runs, BP_MDD solves instances to optimality at the root node and at most 8 nodes are explored in the branch-and-bound tree. On average, for instances with $|\mathscr{P}| \geq 1024$, 55% of the total run-time accounts for solving the pricing problems, with a minimum and maximum percentage of 19% and 84%, respectively. Phases 2 and 3 are responsible on average for 27% of the pricing time when $K \geq 4$ varying from 5% to 72%.

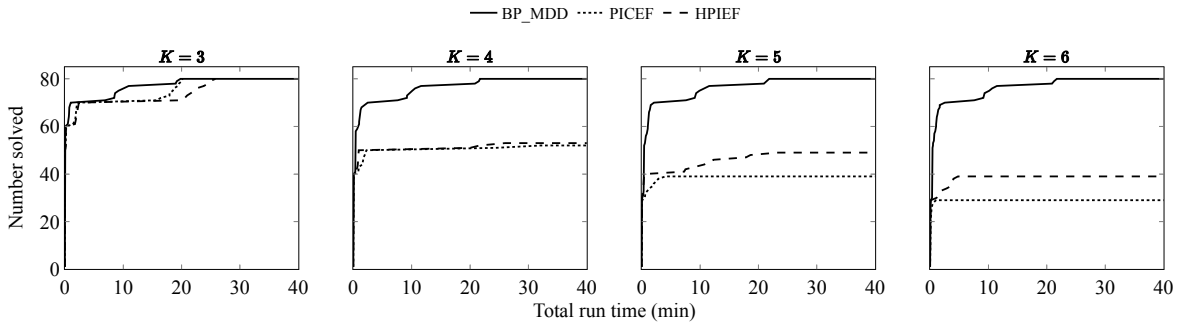


Figure 5 Performance profiles for the set KEP_N0.

On the same runs ($|\mathscr{P}| \geq 1024$ and $K \geq 4$), the average and maximum time for building the MDDs is 109.8s and 257.5s, respectively. For only 10 instances with $K \geq 4$, PICEF and HPIEF report a feasible solution within the time limit, meaning that for the others the RAM threshold is exceeded. The optimality gap reported by these solvers varies between 0% and 30%, with respect to the best solution found among all the three solvers. Overall, PICEF is slightly better than HPIEF in terms of optimality gap but not in terms of the number of instances that could fit into memory.

Figure 6 shows the solution time (without preprocessing or MDD construction) taken for every run and the state-of-the-art solver when $K \in \{3, 4\}$ for all chain lengths. Markers above the diagonal indicate BP_MDD is faster. Particularly, BP_MDD is faster in 20% of runs when $|\mathscr{P}| < 512$, yet the maximum run-time taken for BP_MDD to solve these instances is < 4s. When $|\mathscr{P}| \geq 512$, BP_MDD is faster in 80% of runs.

Figure 7 shows the performance profiles for the instances in KEP_N solved to optimality by the three solvers, which demonstrates that our algorithm outperforms the others, especially when long chains are allowed. For $K = 3$ and $L = 3$, the performance at the time limit is similar for the three algorithms, although BP_MDD does not solve one instance (2048 PDPs and 307 NDDs) to optimality. For that instance, BP_MDD's optimality gap is 0.2%. The same instance remains suboptimal across the other K-L combinations. The

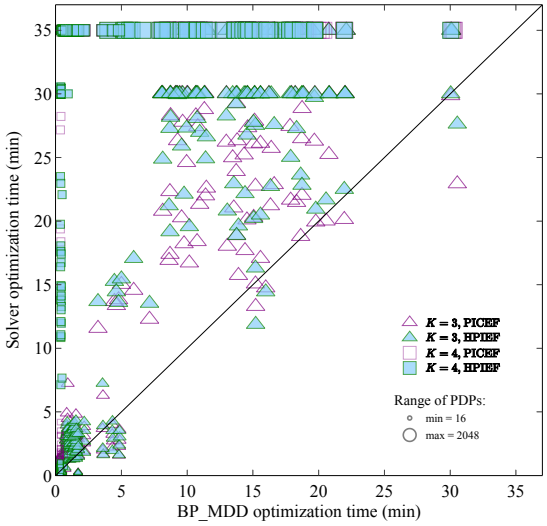


Figure 6 **Performance comparison when $L \in \{0,3,4,5,6\}$. Triangle and square shaped markers correspond to instances with $K = 3$ and $K = 4$, respectively. The size of the markers are correlated with the number of PDPs in the instances. Markers located at the y-axis value of 35 on the top of the plot indicate instances that were not solved to optimality within the time limit.**

performance of PICEF and HPIEF decreases as K and L increases. The maximum optimality gap provided by PICEF and HPIEF, among the 95 suboptimal runs at the time limit is 11%. Among these two solvers, HPIEF provided 67% of the suboptimal solutions.

On average, BP_MDD solved 94% of runs to optimality at the root node, with a maximum of two explored branch-and-bound nodes. For instances with $|\mathscr{P}| \geq 1024$, solving the pricing problems accounts for 66% of the total run-time, and on average 53% of the pricing time is spent in Phase 1. For the same instances, the time spent on building the MDDs accounts on average for 25% of the total run-time, with a maximum of 35%. On average, in more than 90% of the cases, all NDDs were used in the final solution.

Lastly, we note that the majority of columns across all runs are found in Phase 1 (Appendix C).

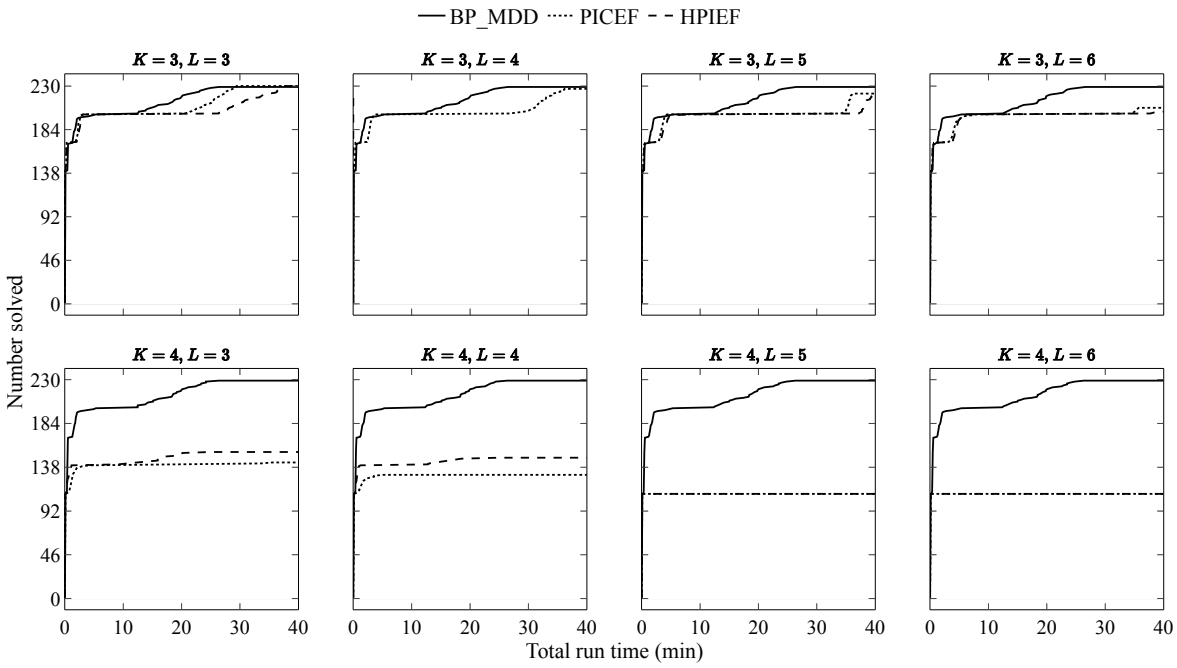


Figure 7 **Performance profiles for the set KEP_N.**

7.3. Sensitivity analysis on the solution structure

In this section, we first investigate the benefit of long chains for different instance structures, according to the presence of highly sensitized patients. In the second part of our analysis, we show the change in the composition of different optimal solutions when cycles are prioritized over chains and vice versa. To the best of our knowledge, BP_MDD is the only exact algorithm for which such a preference can be selected a priori.

7.3.1. The value of long chains An important question from a practical perspective is, whether the same optimal matching value achieved when allowing chains can be obtained by relying only on cycles, and thus disregarding the need of chains. Ashlagi et al. (2012) and Dickerson et al. (2012b) showed analytically that, when the graph is sufficiently sparse, chains benefit the number of matched patients. A standard methodology to study this and other practical issues in the KEP is the use of analytical models. These models, generate random graphs, e.g., vertices and arcs, by following probability estimates on

inherent characteristics to KPDP participants, such as a patient's sensitization degree and the number of highly-sensitized patients in the graph. For our further analysis, we use a similar random graph model to that of Ashlagi et al. (2012) and Ding et al. (2018). Particularly, we also assume the existence of two patient-donor pair categories: highly-sensitized and low-sensitized. Instead of building completely random graphs, we take a sample of instances from KEP_N and preserve arcs $(u, v) \in \mathscr{A}$ with probability p_h if the patient (and thus the pair) in vertex v is highly-sensitized or probability p_ℓ if that patient is not. Our goal is to perform a sensitivity analysis on the original instances. Therefore, in Figure 8 we use the notation (σ, p_ℓ) to represent an instance where the proportion of low-sensitized pairs over the total number of pairs is σ and p_ℓ is the compatibility probability for low-sensitized pairs. In all cases, we set $p_h = 5\%$, taken from Ding et al. (2018).

We selected 3 small-size instances (indexed by S-181, S-182 and S-183) and 3 medium-size instances (indexed by M-201, M-202, M-203) from the group KEP_N. The former have 256 pairs and 38 singleton donors whereas the latter have 512 pairs and 25 singleton donors. For each instance, we generated 6 random graphs according to the tuples (σ, p_ℓ) depicted in Figure 8. The average graph densities from left to right are $4.2\%, 6.8\%, 5.6\%, 9.2\%, 8.9\%$ and 15.7%. Overall, the higher the percentage of low-sensitized patients, the higher the arc density.

These instances were solved within an optimality gap of $\leq 3\%$, with 93.5% solved under $\leq 1\%$ optimality gap. Overall, these instances are substantially more challenging. BP_MDD solved optimally 57% of them in 151.9 seconds, on average. The feasible solutions took on average 1850.3 seconds, with the largest solution time exceeding an hour.

The empirical results depicted in Figure 8 complement previous findings on the benefit of chains (Ashlagi et al. 2012, Dickerson et al. 2012b, Ding et al. 2018). For sparse graphs,

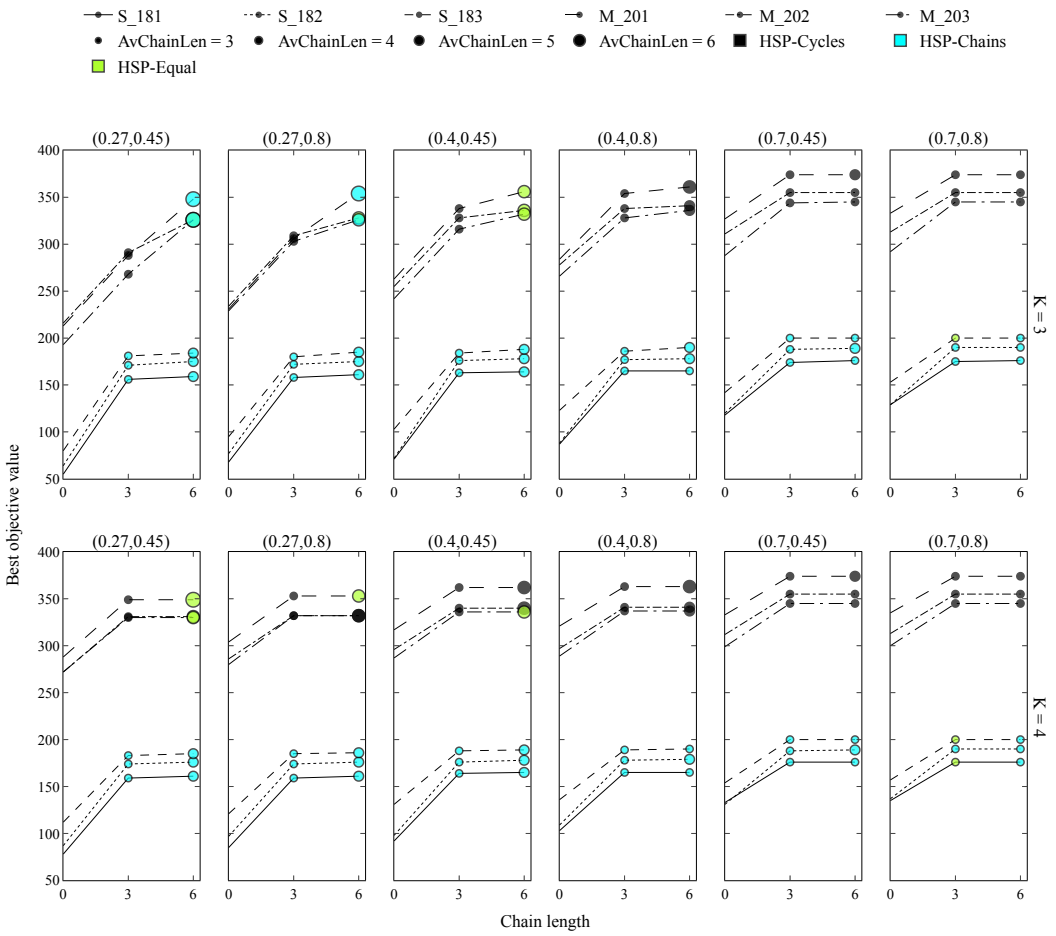


Figure 8 **Change in the number of matches by sensitization category and chain length. AvChainLen stands for the average chain length of an instance solution, rounded to the closest integer. HSP-Chains (Cycles) indicates that chains (cycles) included a higher percentage of the matched and highly sensitized patients. HSP-Equal indicates that the difference between the percentage of matched and highly-sensitized patients in chains and cycles is smaller than** 5%**.**

chains, particularly long chains (involving more than three transplants) are able to reach more patients when the opportunities of cyclic exchanges are scarce. In all cases there is a significant benefit from a cycle-only solution to one allowing chains of size at least three. Chains become shorter, stabilizing around three, as the percentage of low-sensitized patients increase. This behavior is also observed for the vast majority of instances in KEP_N, which are significantly denser than the ones presented in Figure 8. On average,

each instance in KEP_N sees an increase of 20% when passing from a solution with $K = 3$ and no chains to one with $K = L = 3$, and about 19% from no chains and $K = 4$ to $K = 4, L = 3$. From this point on, increasing the size of chains beyond three, seems only significant for a handful of the small and medium size instances.

For the small size instances in Figure 8, allowing chains seem to be critical to include the difficult to match patients in a solution, even when the arc density increases. When the length of cycles is four, chains still provide benefit, although it seems that an important number of pairs that were reached before by long chains can be matched in these long cycles. Interestingly, chains seem to have a more dramatic effect as the size of the instances increase.

7.3.2. Solution composition under prioritization by exchange type Another relevant question is, whether for the same optimal matching value, one can opt for a solution that prioritizes the presence of chains over cycles or vice versa. For instance, chains can be prioritized over cycles since even when failures occur in arcs or vertices of the graph, the original chain can simply turn into a shorter one, whereas a cyclic exchange falls apart altogether (Klimentova et al. 2016, Dickerson et al. 2019).

With this concern in mind, we added two modes under which BP_MDD can be executed. The default mode, referred to as "CH" prioritizes chains over cycles and it corresponds to returning up to 15 positive-price chain columns to the RMP as described in section 6.3.3. The alternative mode, referred to as "CY" prioritizes cycles over chains by restricting the number of chain columns returned to the RMP at every iteration of the column generation algorithm to up most one. Since the size of a K-limited FVS is generally much bigger than the number of singleton donors (see Appendix C), under this mode, there are high chances of discovering a larger number of positive-price cycle columns, as opposed to

those corresponding to chains. Therefore, if possible, the resulting feasible solution consists mostly of cycles.

Table 2 shows the solution composition for the instances shown in Figure 8, in terms of the average chain length (AvChainLen), the average cycle length (AvCycleLen) and the number of cycles (nCycleSol). The header with the average ratios between CH and CY indicates the division between the value obtained under the CH mode (e.g., AvChainLen) and the value corresponding to the same feature under the CY mode. The figures are aggregated by the two types of instances (small and medium size) and all cycle lengths. It is worth noting that *all* singleton donors were used under both modes. As observed in Table 2, the difference relies on the average chain length and the number of cycles. Under the CH mode, the algorithm finds longer chains and fewer cycles than its counterpart. The opposite behavior occurs under the CY mode. In both cases, the length of cycles remains steady around three. The difference in the chains length becomes more evident as the instances become denser. The overall behavior of these instances is also observed in the set KEP_N, especially when comparing the ratio of the average chain length (see Appendix C).

In terms of computational performance, the solution time for KEP_N instances under the CY mode, was on average 8 times slower when compared to CH. This result can be explained due to the restriction on chain columns in the RMP. Thus, forcing BP_MDD to go over more iterations to prove optimality of the RMP. Moreover, about 90% of the KEP_N instances under the CY mode were solved to optimality. The average gap was 0.3%.

Note that BP_MDD can be extended to encompass other priorities. It is known that highly-sensitized patients tend to wait longer for a match offer than those who are not.

Table 2 Solution composition by execution modes

		Averages on CH mode			Average Ratios CH/CY		
(σ, p_ℓ)	L	AvChainLen	AvCycleLen	nCycleSol	AvChainLen	AvCycleLen	nCycleSol
(0.27,0.45)	3	2.93	3.08	46.25	1.01	1.00	0.99
(0.27,0.45)	6	4.65	2.93	37.33	1.06	1.01	0.81
(0.27,0.8)	3	2.92	2.99	50.58	1.05	1.00	0.93
(0.27,0.8)	6	4.47	2.88	40.42	1.08	1.02	0.79
(0.4,0.45)	3	2.93	2.96	54.75	1.04	1.00	0.95
(0.4,0.45)	6	4.35	2.80	44.08	1.09	0.98	0.83
(0.4,0.8)	3	2.89	2.91	57.58	1.09	1.02	0.90
(0.4,0.8)	6	3.91	2.84	49.33	1.19	1.01	0.78
(0.7,0.45)	3	2.85	2.82	64.08	1.18	1.01	0.86
(0.7,0.45)	6	3.41	2.79	58.75	1.26	1.02	0.78
(0.7,0.8)	3	2.85	2.84	64.33	1.29	1.02	0.83
(0.7,0.8)	6	3.19	2.83	60.92	1.30	1.00	0.80

If exchanges of a given size are preferred, the longest path for the MDDs can focus on finding, if any, positive-price chains (cycles) among the targeted exchanges. Phases two and three also offer alternatives to choose which and how many of those exchanges to return to the RMP. These prioritization schemes work in a similar fashion to having hierarchical objectives (Glorie et al. 2014), with the advantage that multiple runs of the algorithm are not required.

8. Conclusion

We addressed the problem of finding a matching in kidney exchange, considering long chains. We first introduced a Lagrange relaxation that allows its decomposition into inde-

pendent sub-problems, and showed that the Lagrangian dual provides an upper bound as tight as the so-called cycle formulation. We then proposed a B&P algorithm in which the pricing problems for both cycles and chains are primarily solved via MDDs, giving us the advantage of finding positive-price columns by means of a shortest-path algorithm. Although the time complexity can be exponential (due to the size of the input), we showed experimentally that by combining exact and restricted decision diagrams, positive-price cycle and chain columns can be found efficiently. Given the large and realistic dataset we used to test our algorithm and the state-of-the art approaches, the experimental evidence suggests a remarkable performance of our solution algorithm for real match runs, making it, to the best of our knowledge, the first one to optimally solve nearly all instances from the PrefLib library (Mattei and Walsh 2013), with the one unsolved instance having a 0.2% optimality gap. Moreover, our algorithm allows in a single run the prioritization of an exchange type as a secondary objective. Lastly, we performed a sensitivity analysis on a sample of instances to empirically determine the benefit of chains and show that they are able to reach a high percentage of highly-sensitized patients.

For projected multi-hospital (countries) initiatives of KPDPs, instances may grow even larger than the largest instances considered here. In such a case, *relaxed* decision diagrams can be used to solve the Lagrangian subproblems in order to obtain the new upper bound, whereas restricted decision diagrams can still be used to efficiently find columns.

Acknowledgments

This work has been supported by the University of Toronto Centre for Healthcare Engineering.

References

Abraham DJ, Blum A, Sandholm T (2007) Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. *Proceedings of the 8th ACM conference on Electronic commerce*, 295–304.

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

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

Ashlagi I, Gamarnik D, Rees MA, Roth AE (2012) The need for (long) chains in kidney exchange. Working paper, National Bureau of Economic Research.

Ashlagi I, Roth AE (2021) Kidney exchange: An operations perspective. Working Paper 28500, National Bureau of Economic Research, `http://www.nber.org/papers/w28500`.

Awasthi P, Sandholm T (2009) Online stochastic optimization in the large: Application to kidney exchange. 405–411.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research* 46(3):316–329.

Bergman D, Cire AA, Hoeve WJv, Hooker JN (2014) Optimization bounds from binary decision diagrams. *INFORMS Journal on Computing* 26(2):253–268.

Biró P, Haase-Kromwijk B, Andersson T, Ásgeirsson EI, Baltesová T, Boletis I, Bolotinha C, Bond G, B́ohmig G, Burnapp L, Cechlárová K, Di Ciaccio P, Fronek J, Hadaya K, Hemke A, Jacquelinet C, Johnson R, Kieszek R, Kuypers DR, Leishman R, Macher MA, Manlove D, Menoudakou G, Salonen M, Smeulders B, Sparacino V, Spieksma FC, Valentín MO, Wilson N, van der Klundert J (2019) Building Kidney Exchange Programmes in Europe—An Overview of Exchange Practice and Activities. *Transplantation* 103:1514–1522.

Biró P, Manlove DF, Rizzi R (2009) Maximum weight cycle packing in directed graphs, with application to kidney exchange programs. *Discrete Mathematics, Algorithms and Applications* 01(04):499–517.

Cantwell L, Woodroffe C, Holdsworth R, Ferrari P (2015) Four years of experience with the Australian kidney paired donation programme. *Nephrology (Carlton, Vic.)* 20(3):124–131.

Carvalho M, Klimentova X, Glorie K, Viana A, Constantino M (2020) Robust models for the kidney exchange problem. *INFORMS Journal on Computing* .

Castro MP, Cire AA, Beck JC (2020) An MDD-based Lagrangian approach to the multicommodity pickup-and-delivery TSP. *INFORMS Journal on Computing* 32(2):263–278.

CBS (2019) Interprovincial organ sharing national data report: Kidney Paired Donation Program 2009–2018. Technical report, Canadian Blood Services, https://profedu.blood.ca/sites/msi/files/kpd-eng_2018.pdf.

Chisca D, Lombardi M, Milano M, O'Sullivan B (2018) From offline to online kidney exchange optimization. *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 587–591, URL http://dx.doi.org/10.1109/ICTAI.2018.00095.

CIHI (2019) Annual statistics on organ replacement in Canada: Dialysis, transplantation and donation, 2009 to 2018. Ottawa, ON: CIHI; 2019. Technical report, Canadian Institute for Health Information, https://www.cihi.ca/sites/default/files/document/corr-snapshot-2019-en.pdf.

Cire AA, van Hoeve WJ (2013) Multivalued decision diagrams for sequencing problems. *Operations Research* 61(6):1411–1428.

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

Dickerson JP (2014) Robust dynamic optimization with application to kidney exchange. *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, 1701–1702 (Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems).

Dickerson JP, Manlove DF, Plaut B, Sandholm T, Trimble J (2016) Position-indexed formulations for kidney exchange. *Proceedings of the 2016 ACM Conference on Economics and Computation*, 25–42.

Dickerson JP, Procaccia AD, Sandholm T (2012a) Dynamic Matching via Weighted Myopia with Application to Kidney Exchange. *AAAI*.

Dickerson JP, Procaccia AD, Sandholm T (2012b) Optimizing kidney exchange with transplant chains: Theory and reality. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, 711–718.

Dickerson JP, Procaccia AD, Sandholm T (2019) Failure-aware kidney exchange. *Management Science* 65(4):1768–1791.

Ding Y, Ge D, He S, Ryan CT (2018) A nonasymptotic approach to analyzing kidney exchange graphs. *Operations Research* 66(4):918–935.

Flechner SM, Thomas AG, Ronin M, Veale JL, Leeser DB, Kapur S, Peipert JD, Segev DL, Henderson ML, Shaffer AA, Cooper M, Hil G, Waterman AD (2018) The first 9 years of kidney paired donation through

the National Kidney Registry: Characteristics of donors and recipients compared with National Live Donor Transplant Registries. *American Journal of Transplantation* 18(11):2730–2738.

Gao I (2019) Fair Matching in Dynamic Kidney Exchange. `https://arxiv.org/abs/1912.10563`.

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

Hooker JN (2013) Decision diagrams and dynamic programming. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 94–110 (Springer).

Karp R (1972) *Reducibility among combinatorial problems*, 85–103 (Boston, MA: Springer).

Kinable J, Cire AA, van Hoeve WJ (2017) Hybrid optimization methods for time-dependent sequencing problems. *European Journal on Operations Research* 259(3):887–897.

Klimentova X, Alvelos F, Viana A (2014) A new branch-and-price approach for the kidney exchange problem. *Computational Science and Its Applications – ICCSA 2014*, 237–252 (Springer).

Klimentova X, Pedroso JaP, Viana A (2016) Maximising expectation of the number of transplants in kidney exchange programmes. *Computers & Operations Research* 73:1–11.

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

Lozano L, Smith JC (2018) A binary decision diagram based algorithm for solving a class of binary two-stage stochastic programs. *Mathematical Programming* 1–24.

Mak-Hau V (2015) 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):35–39.

Malik S, Cole E (2014) Foundations and Principles of the Canadian living donor paired exchange program. *Canadian Journal of Kidney Health and Disease* 1(1):6.

Mattei N, Walsh T (2013) Preflib: A Library for Preferences http://www.preflib.org. Perny P, Pirlot M, Tsoukiàs A, eds., *Algorithmic Decision Theory*, 259–270 (Berlin, Heidelberg: Springer Berlin Heidelberg).

McElfresh D, Bidkhori H, Dickerson J (2019) Scalable robust kidney exchange. *Proceedings of the AAAI Conference on Artificial Intelligence* 33:1077–1084.

Mittelman HD (2020) Benchmarking optimization software-a (Hi)story. *SN Operations Research Forum* 1(2).

Monteiro T, Klimentova X, Pedroso JP, Viana A (2020) A comparison of matching algorithms for kidney exchange programs addressing waiting time.

Morrison DR, Sewell EC, Jacobson SH (2016) Solving the pricing problem in a branch-and-price algorithm for graph coloring using zero-suppressed binary decision diagrams. *INFORMS Journal on Computing* 28(1):67–82.

Plaut B, Dickerson J, Sandholm T (2016a) Hardness of the pricing problem for chains in barter exchanges `https://arxiv.org/abs/1606.00117`.

Plaut B, Dickerson JP, Sandholm T (2016b) Fast optimal clearing of capped-chain barter exchanges. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 601–607 (AAAI Press).

Raghunathan AU, Bergman D, Hooker J, Serra T, Kobori S (2018) Seamless multimodal transportation scheduling `https://arxiv.org/abs/1807.09676`.

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

Saidman SL, Roth AE, Sönmez T, Unver MU, Delmonico FL (2006) Increasing the opportunity of live kidney donation by matching for two- and three-way exchanges. *Transplantation* 5(81):773–82.

Wall AE, Veale JL, Melcher ML (2018) Advanced donation programs and deceased donor-initiated chains-2 Innovations in kidney paired donation. *Transplantation* 101(12):2818–2824.

Appendix

A. On the completeness of two existing formulations

In this section, we present the edge assignment formulation and the extended edge formulation including NDDs, as proposed by Constantino et al. (2013). We show that for these formulations to model chains correctly, the inclusion of infeasible-cycle-breaking constraints is required, which was not mentioned in their paper.

A.1. Edge Assignment Formulation

Following an equivalent notation to that used by Constantino et al. (2013), we proceed to introduce some notation. Let $D = (V, A)$ be a digraph representing compatibilities among donors (single or paired) and PDPs. The set of vertices $V = \{1, ..., |\mathscr{P}| + |\mathscr{N}|\}$ has $|\mathscr{P}|$-many PDPs and $|\mathscr{N}|$-many NDDs. Let vertices $\{1, ..., |\mathscr{N}|\}$ represent NDDs and vertices $\{|\mathscr{N}| + 1, ..., |\mathscr{N}| + |\mathscr{P}|\}$ represent PDPs. An arc $(i, j) \in A$ exists if the donor in vertex i is compatible with patient in vertex j. Assume that a dummy patient is associated to each NDD, so that paired donors $j \in \{|\mathscr{N}| + 1, ..., |\mathscr{N}| + |\mathscr{P}|\}$ are compatible with each dummy patient $i \in \{1, ..., |\mathscr{N}|\}$. Also, consider $|V|$ as an upper bound on the number of cycles and chains in any feasible solution. For each vertex $\ell \in \{|\mathscr{N}| + 1, ..., |\mathscr{P}| + |\mathscr{N}|\}$, let $V^\ell = \{i \in V \mid i \geq \ell\}$ be the set of vertices forming cycles with index higher or equal to ℓ, whereas for each index $\ell \in \{1, ..., |\mathscr{N}|\}$ let $V^\ell = \{i \in \mathscr{P}\} \cup \{\ell\}$ be the set of vertices forming part of a chain started by the ℓ-th NDD. Notice that only vertices $i \geq \ell$ are included in each vertex set to remove multiplicity of solutions. Moreover, it can happen that for some $\ell \in \{1, ..., |V|\}$, $V^\ell = \emptyset$, e.g., if all vertices pointing or receiving an arc from the vertex with the lowest index are removed. Thus, denote by $\mathscr{L}$ the set of indices for which $V^\ell \neq \emptyset$. Lastly, consider the following decision variables:

$$x_{ij} = \begin{cases} 1, \text{ if arc } (i,j) \text{ is selected in a cycle or chain} \\ 0, \text{ otherwise} \end{cases}$$

$$y_i^\ell = \begin{cases} 1, \text{ if node } i \text{ is assigned to the } \ell\text{-th cycle (chain)} \\ 0, \text{ otherwise} \end{cases}$$

Then, the edge assignment formulation is defined as follows:

$$\max \sum_{(i,j)\in A} w_{ij} x_{ij} \tag{14a}$$

$$\sum_{j:(j,i)\in A} x_{ji} = \sum_{j:(i,j)\in A} x_{ij} \qquad i \in V \tag{14b}$$

$$\sum_{j:(i,j)\in A} x_{ij} \leq 1 \qquad i \in V \tag{14c}$$

$$\sum_{i\in\{\ell\}\cup\{|\mathscr{N}|+1,\ldots,|\mathscr{P}|+|\mathscr{N}|\}} y_i^\ell \leq L \qquad \ell \in \{1,\ldots,|\mathscr{N}|\} \tag{14d}$$

$$\sum_{i\geq|\mathscr{N}|+1} y_i^\ell \leq K \qquad \ell \in \{|\mathscr{N}|+1,\ldots,|\mathscr{N}|+|\mathscr{P}|\} \tag{14e}$$

$$\sum_{\ell\in\mathscr{L}: i\in V^\ell} y_i^\ell = \sum_{j:(i,j)\in A} x_{ij} \qquad i \in V \tag{14f}$$

$$y_i^\ell + x_{ij} \leq 1 + y_j^\ell \qquad (i,j) \in A, \ell \in \mathscr{L}, i \in V^\ell \tag{14g}$$

$$y_i^\ell \leq y_\ell^\ell \qquad \ell \in \mathscr{L}, i \in V^\ell \tag{14h}$$

$$y_i^\ell \in \{0,1\} \qquad \ell \in \mathscr{L}, i \in V^\ell \tag{14i}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{14j}$$

Constraints (14b) assure that patient i receives a kidney if and only if donor j donates a kidney. Constraints (14c) allow at most one donation. Constraints (14d) and (14e) limit the length of chains and cycles. Constraints (14f) ensure that vertex i is in a cycle (chain) if and only if there is an assignment of i to some ℓ. Constraints (14g) state that if vertex i

is in cycle (chain) ℓ and donor i gives a kidney to recipient j, then vertex j must also be in the ℓ-th cycle (chain). Constraints (14h) establish that a vertex $i \in V^\ell$ can be assigned to the ℓ-th cycle (chain) only if vertex ℓ is also assigned. Constraints (14i) and (14j) indicate decision variables' domain.

Now, we proceed to show a solution example satisfying (14) and yet infeasible to the KEP. Consider Figure 9 and assume $K = 3$ and $L = 6$. Notice that in the solution, $y_2^2 = y_3^2 = y_4^2 = y_5^2 = y_6^2 = y_3^3 = y_4^3 = y_5^3 = y_6^3 = y_4^4 = y_5^4 = y_6^4 = y_5^5 = y_6^5 = x_{46} = x_{54} = x_{56} = x_{65} = x_{52} = x_{62} = x_{26} = 0$ and $y_1^1 = y_2^1 = y_3^1 = y_4^1 = y_5^1 = y_6^1 = x_{53} = x_{36} = x_{64} = x_{45} = x_{12} = x_{21} = 1$.

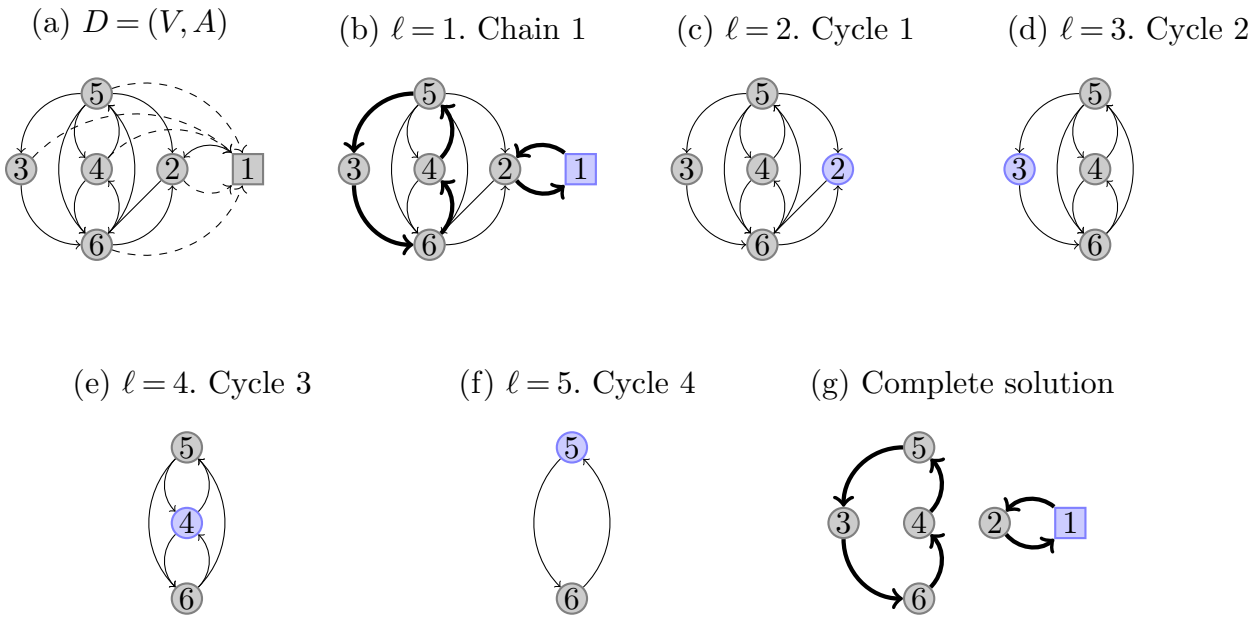


Figure 9 **Counter example where (14) and (15) provide an infeasible solution. An altruistic donor is represented by a square and blue vertices correspond to the vertex with lowest index in the ℓ-th cycle or chain. Dashed arcs indicate compatibility of paired donors with a dummy patient associated to the altruistic donor, solid arcs indicate compatibilities among real donors and patients, and bold arcs represent the ones selected in a solution.**

A.2. EEF: Extended Edge Formulation

Consider $\mathscr{P} + \mathscr{N}$ copies of the graph D, $D^\ell = (V^\ell, A^\ell)$, where V^ℓ is as defined in Section A.1 and $A^\ell = \{(i, j) \in \mathscr{A} \mid i, j \in V^\ell\}$ is the set of arcs in the ℓ-th copy. Recall that $\mathscr{P}$ and $\mathscr{N}$

is an upper bound on the number of cycles and chains in a feasible solution, respectively. In each copy $\ell \in \{1, ..., |\mathscr{N}|\}$ chains are triggered by the ℓ-th NDD and at most L arcs can be selected. In each copy $\ell \in \{|\mathscr{N}| + 1, ..., |\mathscr{N}| + |\mathscr{P}|\}$, all cycles include the vertex with lowest index in that copy (e.g., see Figure 9). Let x_{ij}^{ℓ} be an arc variable taking the value one if the arc $(i, j) \in A^{\ell}$ is selected in the ℓ-th copy and zero, otherwise. Similarly to EAF, consider $\mathscr{L}$ as the set of indices for which $V^{\ell} \neq \emptyset$. Then, the extended edge formulation can be formulated as follows:

$$\max \sum_{\ell \in \mathscr{L}} \sum_{(i,j) \in A^{\ell}} w_{ij} x_{ij}^{\ell} \tag{15a}$$

$$\sum_{j:(j,i) \in A^{\ell}} x_{ji}^{\ell} = \sum_{j:(j,i) \in A^{\ell}} x_{ij}^{\ell} \qquad \ell \in \mathscr{L}, i \in V^{\ell} \tag{15b}$$

$$\sum_{\ell \in \mathscr{L}} \sum_{j:(i,j) \in A^{\ell}} x_{ij}^{\ell} \leq 1 \qquad i \in V \tag{15c}$$

$$\sum_{(i,j) \in A^{\ell}} x_{ij}^{\ell} \leq L \qquad \ell \in \{1, ..., |\mathscr{N}|\} \tag{15d}$$

$$\sum_{(i,j) \in A^{\ell}} x_{ij}^{\ell} \leq K \qquad \ell \in \{|\mathscr{N}| + 1, ..., |\mathscr{P}| + |\mathscr{N}|\} \tag{15e}$$

$$\sum_{j:(i,j) \in A^{\ell}} x_{ij}^{\ell} \leq \sum_{j:(\ell,j) \in A^{\ell}} x_{\ell j}^{\ell} \qquad \ell \in \mathscr{L}, i \in V^{\ell} \tag{15f}$$

$$x_{ij}^{\ell} \in \{0, 1\} \qquad \ell \in \mathscr{L}, (i, j) \in A^{\ell} \tag{15g}$$

Constraints (15b) assure that a patient in the ℓ-th graph copy receives a kidney if his or her paired donor donates one. Constraints (15c) allow every donor (paired or singleton) to donate at most one kidney in only one copy of the graph. Constraints (15d) and (15e) guarantee the maximum length allowed for chains and cycles in terms or arcs. Constraints (15f) assure that a copy is selected, only if the vertex with the lowest index is selected. Lastly, constraints (15g) define the nature of the decision variables.

The same counter example used for the EAF defined in Section A.1 applies to the EEF. For the example given in Figure (9), consider $K = 3$ and $L = 6$. Note that only the chain copy is selected, but due to the presence of subtours of length superior to the cycle size limit, both formulations can provide an infeasible solution. Therefore, infeasible-cycle-breaking constraints are required for every chain copy.

B. Proofs

In this section, we provide the proofs on the validity of IEEF and the complexity of finding a positive-price column via MDDs. For the sake of consistency, the numeration of the following propositions coincides with that used in the main body of the paper.

PROPOSITION 2. IEEF is a correct formulation of the matching problem in the KEP.

Proof. First, we need to show that a feasible matching to the KEP, $M(K,L)'$, corresponds to a feasible solution to IEEF whose objective function value is equal to the sum of the arc weights in $M(K,L)'$, which is

$$\sum_{(u,v)\in\mathscr{A}(M(K,L)')} w_{uv}.$$

Formally, let $M(K,L)' = \mathscr{C}'_K \cup \mathscr{C}'_L$ where $\mathscr{C}'_K \subseteq \mathscr{C}_K$ and $\mathscr{C}'_L \subseteq \mathscr{C}_L$. Since there are as many chain copies as NDDs and a single NDD per copy, for a chain $p = (v_1, ..., v_\ell) \in \mathscr{C}'_L$, there exists a unique copy $i \in \bar{I}$ for which the i-th NDD, u_i, corresponds to v_1. We then denote by $\bar{I}' \subseteq \bar{I}$ the set of chain copies corresponding to altruistic donors in $\mathscr{N} \cap \mathscr{V}(M(K,L)) \neq \emptyset$, which can be empty, i.e., there are no chains in a feasible solution. Thus, we define $\bar{x}'$ as

$$\bar{x}^{i\prime}_{uv} = \begin{cases} 1 \text{ if } (u,v) \in \mathscr{A}(p) \\ 0 \text{ o.w.} \end{cases} \qquad \forall i \in \bar{I}', p = (u_i, v_2, ..., v_\ell) \in \mathscr{C}'_L$$

and

$$\bar{x}^{i\prime} = 0 \qquad \forall i \in \bar{I} \setminus \bar{I}'.$$

In the case of cycles, suppose $\mathscr{V}^* = \{v_1, ..., v_{|\mathscr{V}^*|}\}$ is a K-limited FVS provided by Algorithm 1. Then, for a cycle $c = (v_1, ..., v_k, v_1) \in \mathscr{C}'_K$ at least one of its vertices is also in $\mathscr{V}^*$ by the definition of a K-limited FVS (see Section 4.1). Let e be the smallest position in $\mathscr{V}^*$ where a feedback vertex $v^*_e \in \mathscr{V}(c)$ is at, which coincides with the position of that copy in $\hat{I}$. Then, for the cycle $c \in \mathscr{C}'_K$ we define

$$\hat{x}^{e\prime}_{uv} = \begin{cases} 1 \text{ if } (u,v) \in \mathscr{A}(c) \\ 0 \text{ o.w.} \end{cases} \qquad \forall (u,v) \in \hat{\mathscr{A}}^e.$$

We set the other components of $\hat{x}'$ vector similarly. At $(\hat{x}', \bar{x}')$, constraints (1a) are satisfied since $M(K,L)'$ is feasible, i.e., every donor can donate at most once. This unitary flow satisfies flow-balance constraints in cycle copies (1b) and chain copies (1c). By the construction of $(\hat{x}', \bar{x}')$, we know that at most one cycle/chain is selected in each copy, and they are the same cycles and chains of $M(K,L)'$. Thus, it is easy to see that the sum of arc variables for cycle (1d) and chain (1e) copies does not exceed K and L, respectively. Constraints (1f) and (1g) are also satisfied since the left-hand side can take one as its maximum value, by (1a), and only arcs from the chain and cycle copies whose critical vertices u_i and v^*_i appear in $M(K,L)'$ are selected while constructing $(\hat{x}', \bar{x}')$. Observe that constraints (1h) are applied to cycles present in chain copies. They are satisfied by $\bar{x}'$ since only the set of arcs forming a chain is selected in $\bar{x}'$ for copies in $\bar{I}'$, and no arc is selected from copies in $\bar{I} \setminus \bar{I}'$.

Therefore, $(\hat{x}', \bar{x}')$ is a feasible solution to IEEF. Lastly, it is easy to see that its objective value is equal to

$$\sum_{i \in \hat{I}} \sum_{(u,v) \in \hat{\mathscr{A}}^i} w_{uv} \hat{x}^{i\prime}_{uv} + \sum_{i \in \bar{I}} \sum_{(u,v) \in \bar{\mathscr{A}}^i} w_{uv} \bar{x}^{i\prime}_{uv} = \sum_{(u,v) \in \mathscr{A}(M(K,L))} w_{uv},$$

Again, by construction, we need to show for the reverse direction that any feasible solution of IEEF corresponds to a feasible matching of the KEP, $M(K,L)^\star$. Let $(\hat{x}^\star, \bar{x}^\star)$

be a feasible solution of IEEF. Because IEEF can have symmetric solutions, a cycle $c \in \mathscr{C}_K$ can be selected in (a) the cycle copy corresponding to its associated feedback vertex as determined by Algorithm 1, (b) another cycle copy, or (c) a chain copy. However, constraints (1f) allow case (b) only when the feedback vertex of that copy is also selected. Similarly, constraints (1g) allow case (c) only when the NDD associated to a chain copy is also selected. A chain, on the other hand, can only be selected in a unique chain copy by construction of the chain copies. From (1a), we guarantee that the flow leaving every vertex is at most one across all copies, whether they are cycle or chain copies. Thus, an arc traversal in IEEF is vertex-disjoint.

To build feasible cycles from $(\hat{x}^\star, \bar{x}^\star)$ to include in $M(K,L)^\star$, we start by inspecting the arc flow leaving feedback vertices in their corresponding copies, i.e., $\sum_{v:(v_i^\star,v)\in\mathscr{A}^i} \hat{x}_{v_i^\star v} = 1, i \in \hat{I}$. Constraints (1b) enforce flow preservation, i.e., if a vertex gives one unit of flow, it must also receive one. Thus, the arc traversal starting from $v_i^\star$ must be a cycle and it is also simple by (1a). Constraint (1d) enforces the number of selected arcs in a cycle copy to be at most K. Thus, under case (a), the arc traversal is guaranteed to be a feasible cycle $c \in \mathscr{C}_K$. If $|\mathscr{A}(c)| \leq K-2$, then we evaluate case (b), since a cycle can still arise. We proceed to inspect the outgoing flow of other feedback vertices also present in that copy. It is easy to see that due to constraints (1a), (1b) and (1d) the new arc traversal is also a simple cycle $\tilde{c} \in \mathscr{C}_K$, such that $|\mathscr{A}(\tilde{c})| \leq K - |\mathscr{A}(c)|$. Thus, there can be as many cycles inside a cycle copy as long as the total number of arcs selected does not exceed K. Under case (c), when an NDD is selected in a chain copy, a subtour may also be selected. By constraints (1a) and (1c) one can see that such subtours correspond to simple cycles. Nonetheless, their cardinality can exceed K when $L > K$. Constraints (1h) remove all $c \in \bar{\mathscr{C}}^i \setminus \bar{\mathscr{C}}^i_K,\ i \in \bar{I}$, thus, guaranteeing that if a subtour is selected it corresponds to a feasible cycle.

To include chains in $M(K,L)^\star$, constraints (1a), (1c) and (1e) guarantee the arc traversal in a chain copy to contain a feasible chain. Note that all the cycles and chains included in $M(K,L)^\star$ are vertex-disjoint due to (1a). Lastly, the weight of each arc variable in IEEF has a one-to-one correspondence with a transplant weight in $M(K,L)^\star$. Thus, the objective value of $(\hat{x}^\star, \bar{x}^\star)$ corresponds to $\sum_{(u,v)\in\mathscr{A}(M(K,L)^\star)} w_{uv}$, which completes the proof. □

PROPOSITION 4. Given the reduced costs $\hat{r}^i_c$ and $\bar{r}^i_p$ expressed as an arc-separable function for all $(n_s, n_{s'}) \in \hat{\mathcal{A}}^i$ and $(n_s, n_{s'}) \in \bar{\mathcal{A}}^i$, a positive-price cycle, if one exists, can be found in time $\mathcal{O}(\sum_{i\in\hat{I}} \sum_{(n_s,n_{s'})\in\hat{\mathcal{A}}^i} |\delta^i_-(n_s)|)$. Similarly, a positive-price chain can be found in $\mathcal{O}(\sum_{i\in\bar{I}} \sum_{(n_s,n_{s'})\in\bar{\mathcal{A}}^i} |\delta^i_-(n_s)|)$.

Proof. For every arc $(n_s, n_{s'}) \in \hat{\mathcal{A}}^i$ and $(n_s, n_{s'}) \in \bar{\mathcal{A}}^i$, $|\delta^i_-((n_s, n_{s'}))|$ comparisons need to be performed to obtain $\hat{\eta}^i((n_s, n_{s'}))$ and $\bar{\eta}^i((n_s, n_{s'}))$, respectively in (7). Therefore, for the i-th MDD of a cycle copy, $\sum_{(n_s,n_{s'})\in\hat{\mathcal{A}}^i} |\delta^i_-(n_s)|$ comparisons are required to compute $\hat{\eta}^i$, whereas for the i-th MDD of a chain copy, there are the same number of comparisons plus $|\bar{\mathcal{A}}^i|$ comparisons of all arcs, in (8b), before obtaining $\bar{\eta}^i$. Because there are $|\hat{I}|$ cycle MDDs and $|\bar{I}|$ chain MDDs, it follows that the time complexity is as shown above. □

PROPOSITION 5. The size of the input $\sum_{i\in\hat{I}} \sum_{(n_s,n_{s'})\in\hat{\mathcal{A}}^i} |\delta^i_-(n_s)|$ grows as $|\hat{\mathscr{V}}^i|^{K+1}$ does.

Proof. Without lost of generality, assume $\mathscr{D}$ is complete. As stated before, the layer of an arc $a : (n_s, n_{s'}) \in \hat{\mathcal{A}}^i$ is the layer to which its source node belongs, i.e., if node $n_{s'}$ is on layer k, then $\ell(a) = k$. Moreover, let $\hat{\mathcal{A}}^i_k := \{a \in \hat{\mathcal{A}}^i \mid \ell(a) = k\}$ be the set of arcs that belong to layer k. By construction of the MDDs, $\mathbf{r}$ has only one outgoing arc such that $val((\mathbf{r}, n_1)) = v^*_i$. In the second layer, $\mathcal{L}_2$, the cardinality of $\hat{\mathcal{A}}^i_2$ can be up to $|\hat{\mathscr{V}}^i| - 1$ corresponding to the vertices $v \in \hat{\mathscr{V}}^i \setminus \{v^*_i\}$ that can be selected in the second position of a cycle. Note that since there can be the same number of 2-way cycles, an arc $a \in \hat{\mathcal{A}}^i_2$ has a sink node n such that $(n, \mathbf{t}) \in \hat{\mathcal{A}}^i$. If the length of a cycle is larger than two, then the

cardinality of $\hat{\mathcal{A}}_3^i$ can be up to $|\hat{\mathscr{V}}^i|-2$, representing the $|\hat{\mathscr{V}}^i|-2$ vertices $v \in \hat{\mathscr{V}}^i \setminus \{v_i^*\}$ that can be selected in the third position of a cycle. The same process is repeated until in layer $\mathcal{L}_K$ there are $|\hat{\mathscr{V}}^i|-(K-1)$ vertices $v \in \hat{\mathscr{V}}^i$ to choose, and thus, $|\hat{\mathscr{V}}^i|-(K-1)$ arcs $a \in \hat{\mathcal{A}}^i$ pointing to n.

Thus, $|\hat{\mathcal{A}}^i|$ equals

$$\prod_{k=2}^{K} |\hat{\mathscr{V}}^i| - (k-1) + \sum_{k=2}^{K-1} (|\hat{\mathscr{V}}^i| - (k-1)) < |\hat{\mathscr{V}}^i|^{K-1} + K|\hat{\mathscr{V}}^i| \tag{16a}$$

The second sum on the left-hand side corresponds to the number of arcs at every layer whose sink node is n, thus, closing up cycles using fewer than K arcs. Therefore, under a worst-case scenario in which $|\delta_-^i(n_s)|$ and $|\hat{I}|$ tend to $|\hat{\mathscr{V}}^i|$, the complexity of finding a positive-price cycle becomes $\mathcal{O}(|\hat{\mathscr{V}}^i|^{K+1})$. □

Proposition 6. The size of the input $\sum_{i\in\bar{I}} \sum_{(n_s,n_{s'})\in\bar{\mathcal{A}}^i} |\delta_-^i(n_s)|$ grows as $|\bar{\mathscr{V}}^i|^{L+2}$ does for bounded chains and as $|\bar{\mathscr{V}}^i|!$ when $L \to \infty$.

Proof. A similar reasoning to proposition 5 can be followed, except that a chain can be cut short if by visiting a new vertex $v \in \bar{\mathscr{V}}^i$ in a sequence of the state transition graph at least one PDP is present more than once, thereby violating the condition of being a simple path. We know that for a path to have L-many arcs, it is necessary to have a sequence with L PDPs, thus, $|\hat{\mathcal{A}}^i|$ tends to $|\bar{\mathscr{V}}^i|^L$ and $\sum_{i\in\bar{I}} \sum_{(n_s,n_{s'})\in\hat{\mathcal{A}}^i} |\delta_-^i(n_s)| \approx |\hat{\mathscr{V}}^i|^{L+2}$. Therefore, for bounded chains, finding a positive-price chain can be done in time $\mathcal{O}(|\hat{\mathscr{V}}^i|^{L+2})$. The second part follows by the fact that after we visit ℓ vertices, there are still $|\hat{\mathscr{V}}^i|-\ell$ ways to choose the next one, until only one can be chosen, thus, the time to find a positive-price column when L is unbounded is exponential. □

C. Additional Results

Columns by phase

Figure 10 depicts the type and number of columns found in each phase for individual

runs across all the K-L combinations. The x-axis represents the total time (in minutes) to solve the pricing problem of a single run, i.e., an instance on a K-L setting, during the three phases. For every x-value there may be multiple y-points representing the number of columns found in a specific phase (sub phase) in thousands and whether they are cycle or chain columns. Therefore, for the same x-value, multiple y-values may correspond to the same instance, maybe on different settings, if the markers share the same size and color, regardless of the shape. For similar total pricing-problem times, markers may overlap. As an example, consider the time interval from 2min to 10min. In the cycle subplot there are big (purple) circle markers indicating that from 10k up to 30k cycle columns were found in Phase 1, as apposed to the chain subplot where no such markers appear. This means that this subset of instances correspond to $L = 0$. In another example, some instances with 2252 PDPs and 204 NDDs whose pricing time is 15.6min and have a green circle around 12k and 6k in the cycle and chain subplots, respectively, plus a triangle indicating one chain found through (LPH). Thus, the total number of columns of a run is the sum over the number of columns indicated by all markers in the y-axis with respect to the same x-value, provided that markers share color and size. In this particular case of 2048 PDPs and 204 NDDs there are 17,396 columns. Overall, most markers are circles, indicating that the majority of columns are found via MDDs across all runs, while Phase 2 and Phase 3 mostly certify that no more positive-price columns exist.

Effect of vertex-selection rules on BP_MDD

For instances in the set KEP_N0, Table 3 provides average values on the performance of BP_MDD in terms of the number of states (i.e., the number of nodes in MDDs) and the solution time when different selection rules are used in Algorithm 1 and the function *CycleCopy* is replaced by the construction of MDDs. Selection rule Incr.Index corresponds

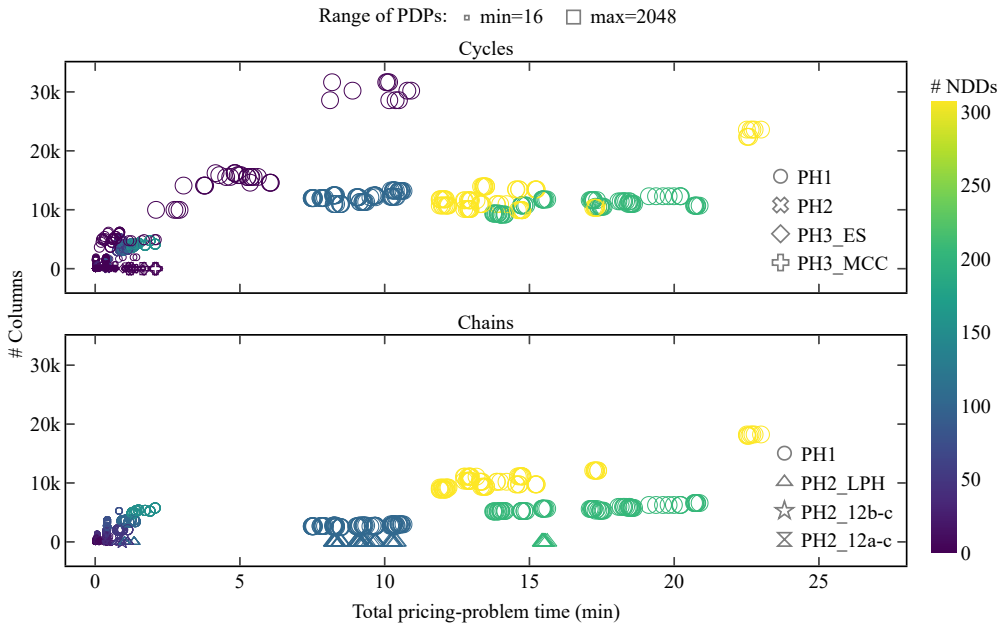


Figure 10 **Classification of columns by run, phase (PH1, PH2, PH3) and type. Marker types indicate different (sub) phases of column generation. Marker sizes are correlated with the number of PDPs, while marker colors indicate the number of NDDs. ES stands for the Exhaustive Search in PH2.**

to selecting every pair in the graph in increasing order of their index, as in (Constantino et al. 2013). The column K-FVS corresponds to finding the K-FVS with smallest cardinality through a MIP formulation. This formulation was given a time limit of 1hr. There is a hyphen if a feasible solution was not available after this time.

Even though, Incr.Index reports the smallest solution time on average, the number of states throughout all MDDs double those of the Indegree selection rule. This selection rule offers the best trade-off between scalability and computational performance.

Prioritization for instances in KEP_N

Table 4 shows the solution composition under the two prioritization schemes for a sample of the instance sets in KEP_N. Header names are defined as in Table 2 of the Results Section. It was worth noting that,except for 23 runs in the set N588_A76, all altruistic donors were used to trigger chains. For those runs, the percentage of used altruistic donors was 98.7%.

Table 3 Trade-off between performance and scalability when $K = 3$

	Average # States				Solution Time (s)			
Set	K-FVS	Incr.Index	Indegree	Total degree	K-FVS	Incr.Index	Indegree	Total degree
N16_A0	10.4	20.3	12.0	11.4	0.00	0.00	0.00	0.00
N32_A0	74.9	124.1	73.2	73.4	0.00	0.00	0.00	0.00
N64_A0	283.7	479.8	229.5	282.4	0.01	0.01	0.01	0.01
N128_A0	1,281.5	1,753.6	933.9	1,192.2	0.03	0.02	0.02	0.02
N256_A0	6,270.3	7,542.1	4,038.2	5,646.7	0.22	0.14	0.15	0.15
N512_A0	29,805.3	31,531.5	16,158.9	25,537.5	1.62	0.96	1.44	1.34
N1024_A0	-	129,608.1	65,816.1	112,277.8	-	8.35	22.84	42.40
N2048_A0	-	513,423.8	252,659.7	467,831.7	-	85.29	521.25	484.82

Table 4 Solution composition by execution modes

			Averages on CH mode			Average Ratio CH/CY		
Set	K	L	AvChainLen	nCyclesSol	AvCycleLen	AvChainLen	nCycleSol	AvCycleLen
N147_A19	3	3	2.97	17.22	2.81	1.50	0.71	1.02
N147_A19	3	4	2.99	17.00	2.83	1.51	0.71	1.02
N147_A19	3	5	3.00	17.11	2.79	1.55	0.71	1.00
N147_A19	3	6	3.06	16.50	2.83	1.58	0.68	1.02
N147_A19	4	3	3.00	17.21	2.81	1.49	0.71	1.03
N147_A19	4	4	3.00	17.00	2.83	1.52	0.71	1.02
N147_A19	4	5	3.00	17.12	2.79	1.55	0.71	1.00
N147_A19	4	6	3.00	16.55	2.83	1.58	0.68	1.02
N294_A38	3	3	3.00	30.63	2.84	1.76	0.63	1.02
N294_A38	3	4	3.00	30.43	2.85	1.67	0.64	1.02
N294_A38	3	5	3.00	30.57	2.84	1.58	0.67	1.01
N294_A38	3	6	3.00	30.70	2.83	1.61	0.66	1.01
N294_A38	4	3	3.00	30.67	2.84	1.74	0.63	1.02
N294_A38	4	4	3.00	30.44	2.85	1.68	0.64	1.03
N294_A38	4	5	3.00	30.52	2.84	1.58	0.67	1.01
N294_A38	4	6	3.00	30.71	2.83	1.60	0.66	1.02
N588_A76	3	3	3.00	61.30	2.93	1.54	0.69	1.03
N588_A76	3	4	3.00	61.30	2.93	1.56	0.68	1.03
N588_A76	3	5	3.00	61.30	2.93	1.36	0.75	1.04
N588_A76	3	6	3.00	61.30	2.93	1.41	0.73	1.03
N588_A76	4	3	3.00	61.30	2.93	1.53	0.69	1.03
N588_A76	4	4	3.00	61.30	2.93	1.50	0.70	1.03
N588_A76	4	5	3.00	61.30	2.93	1.36	0.75	1.04
N588_A76	4	6	3.00	61.30	2.93	1.41	0.73	1.03