Parameterized Algorithms for Kidney Exchange

Palash Dey\textsuperscript{1} and Arnab Maiti\textsuperscript{2}

calash.dey@cse.iitkgp.ac.in, arnabm2@uw.edu\textsuperscript{2}

Indian Institute of Technology Kharagpur\textsuperscript{1}, University of Washington\textsuperscript{2}

June 17, 2024

Abstract

In kidney exchange programs, multiple recipient-donor pairs each of whom are otherwise incompatible, exchange their donors to receive compatible kidneys. The Kidney Exchange problem is typically modelled as a directed graph where every vertex is either an altruistic donor or a pair of recipient and donor; directed edges are added from a donor to its compatible recipients. The computational task is to find if there exists a collection of disjoint cycles and paths starting from altruistic donor vertices of length at most $\ell_c$ and $\ell_p$, respectively, that covers at least some specific number $t$ of non-altruistic vertices (recipients). We study parameterized algorithms for the kidney exchange problem in this paper. Specifically, we design FPT algorithms parameterized by each of the following parameters: (1) the number of recipients who receive kidney, (2) treewidth of the input graph + max\{\ell_p, \ell_c\}, and (3) the number of vertex types in the input graph when $\ell_p \leq \ell_c$. We also present interesting algorithmic and hardness results on the kernelization complexity of the problem. Finally, we present an approximation algorithm for an important special case of the Kidney Exchange problem.

Keywords— Fixed-Parameter Tractability, Kidney Exchange, Vertex Type, Kernelization, Altruistic donors, Compatibility graph, Trading-cycle, Trading-chain

1 Introduction

Patients having acute renal failures are typically treated either with dialysis or with kidney transplantation. However, the quality of life on dialysis is comparatively lower and also the average life span of the recipients on dialysis is around 10 years \cite{usr}. For this reason, most recipients prefer a kidney transplantation over periodic dialysis. However, the gap between the demand and supply of kidneys, which can be obtained either from a deceased person or from a living donor, is so large that the average waiting time varies from 2 to 5 years at most centers \cite{uno, odt}. Moreover, even if a recipient is able to find a donor, there could be many medical reasons (like blood group or tissue mismatch) due to which the donor could not donate his/her kidney to the recipient.

The Kidney Paired Donation (KPD), a.k.a. Kidney Exchange program, allows donors to donate their kidneys to compatible other recipients with the understanding that their recipients will also receive medically compatible kidneys thereby forming some kind of barter market \cite{ABS07}. Since its inception in \cite{Rap86}, an increasing amount of people register in the kidney exchange program since, this way, recipients not only have a better opportunity to receive compatible kidneys, but also can get medically better matched kidneys which last longer \cite{SGW05}. The central problem in any kidney exchange program, also known as the clearing problem, is how to transplant kidneys among various recipients and donors so that a maximum number of recipients receive kidneys.

Typically, kidney exchanges happen in a cyclical manner. The problem is represented as a directed graph: a recipient along with his/her donor is a vertex; we add directed edges from a vertex $u$ to all other vertices whose recipients are compatible with the donor of the vertex $u$. In a (directed) cycle, the recipient of every vertex receives a kidney from the donor of the previous vertex along the cycle.
Donors do not have any legal obligation to donate kidneys and thus, technically speaking, a donor can leave the program as soon as his/her corresponding recipient receives a kidney [Alv05b, SGW+05]. This is not only unfair but also leaves a recipient without any donor. To avoid this problem, all kidney transplantations along a cycle are done simultaneously. As each transplantation involves two surgeries, logistic and human resource constraints allow only a few surgeries to be carried out simultaneously. For this reason, most kidney exchange platforms allow transplantation along cycles of very small length only [Alv05b, ABS07, MO15, MH17]. Sometimes we have altruistic donors (a.k.a non-directed donors (NDDs)) who do not have recipients paired with them. This allows us to have kidney transplantations along with chains (directed paths) also starting from some altruistic vertices — an altruistic donor donates his/her kidney to some compatible recipient whose paired donor donates his/her kidney to the next recipient along the chain and so on. Some platform allows non-simultaneous surgeries along a chain since a broken chain is less harmful than a broken cycle — it does not leave any recipient without donor [RAR15]. However, broken chains also lead to unfairness and thus platforms usually only allow small chains for transplantation. Thus, in the presence of altruistic donors, the fundamental problem of kidney exchanges becomes the following: find a collection of disjoint cycles and chains starting from altruistic vertices which covers a maximum number of non-altruistic vertices. We refer to Definition 1 in Section 2 for formal definition of the Kidney Exchange problem.

1.1 Related Work

To the best of our knowledge, Rapaport was the first person to introduce the idea of a kidney exchange [Rap63]. More than sixty thousands recipients await kidney transplantation and are listed on the United Network for Organ Sharing (UNOS) recipient registry in 2005 [SGW+05]. In September, 2004, the Renal Transplant Oversight Committee of New England approved the establishment of a clearing-house for kidney exchanges [Alv05b]. Roth et al. [Alv05a] discovered a wide class of constrained-efficient mechanisms that are strategy-proof when recipient–donor pairs and surgeons have 0–1 preferences. For example, a line of research allow only cycles [CKVR13, KAV14, SU14] while others allow kidney exchanges to happen along both cycles and chains [MO15, GvdKW14, XW18]. All versions of the Kidney Exchange problem can be formulated as some suitable version of the graph packing problem. Jai et al. [JTWZ17] discussed interesting relationship between barter market and set packing. Krivelevich et al. [KNS+07], Abraham et al. [ABS07], and Belmonte et al. [BHK+22] showed that the basic Kidney Exchange problem along with its various incarnations are NP-hard even when the maximum cycle length allowed is three or the maximum path length allowed is four. Hence, the Kidney Exchange problem is para-NP-hard parameterized by the maximum cycle length plus the maximum path length. Krivelevich et al. [KNS+07] and Jia et al. [JTWZ17] developed approximation algorithms for the Kidney Exchange problem by exploiting interesting connection with the set packing problem.

Practical heuristics and integer linear programming based algorithms have been extensively explored for the Kidney Exchange problem. For example, Manlove and O’Malley proposed an integer programming-based formulation in order to model the criteria that constitute the definition of optimality as per the laws of United Kingdom [MO15]. This formulation was used to build a software which was used by the National Health Service Blood and Transplant to find optimal sets of kidney exchanges for their National Living Donor Kidney Sharing schemes. Dickerson et al. [DMP+16] argued that although kidney donation along an arbitrarily long chain (path) is desirable in theory, there are many practical concerns because of which one would like to use only bounded length chains for kidney donation. Glorie, Klundert, and Wagelmans [GvdKW14] presented a generic iterative branch-and-price algorithm that can deal effectively with multiple criteria involving cycles and chains. Li et al. [LLHT14] proposed a practically efficient polynomial time algorithms for finding the Lorenz-dominant fractional matching. The Lorenz-dominant fractional matching, which can be implemented as a lottery of integral matchings, is in some sense the fairest allocation and also enjoys the property of being incentive compatible. Biro et al. [BMR09] designed approximation algorithms when cycles of length at most three are allowed and no chains are allowed. Rincasos-Alvarez et al. [KAV14] developed a decomposition method that is able to consider long cycles and long chains for projected large realistic instances. In particular, their algorithm also allows the prioritization of the solution composition, for example, chains over cycles or vice versa. Dickerson et al. [DMP+16] introduced the notion of “vertex type” and showed its usefulness as a graph parameter in real-world instances of a kidney exchange program. Two vertices are said to have
the same vertex type if their neighbourhoods are the same.

The closest predecessor of our work is by Xiao and Wang [XW18]. They proposed an exact algorithm with running time \(O(2^n n^3)\) where \(n\) is the number of vertices in the underlying graph. They also presented a fixed parameter tractable algorithm for the Kidney Exchange problem parameterized by the number of vertex types if we do not have any restriction on the length of cycles and chains. Lin et al. [LWFT19] studied the version of the Kidney Exchange problem which allows only cycles and developed a randomized parameterized algorithm with respect to the parameter being the number of patients receiving a kidney and the maximum allowed length of any cycle, combined.

1.2 Contribution

Designing exact algorithms for the Kidney Exchange problem has been a research focus in algorithmic game theory. We contribute to this line of research in this paper. Our specific contribution are the following.

We design FPT algorithms for the Kidney Exchange problem parameterized by the number of recipients receiving kidneys [Theorem 1], treewidth of the underlying graph + maximum length of path \(\ell_p\) + maximum length of cycle allowed \(\ell_c\) [Theorem 3], and the number of vertex types when \(\ell_p \leq \ell_c\) [Theorem 4]. We also present a Monadic second-order (MSO2) formula for the Kidney Exchange problem where the length of the formula is upper bounded by a function of \(\ell = \max\{\ell_p, \ell_c\}\) [Theorem 2].

We show that the Kidney Exchange problem admits a polynomial kernel with respect to the number of recipients receiving kidneys + maximum degree when \(\max\{\ell_p, \ell_c\}\) is a constant [Theorem 5]. We complement this result by showing that the Kidney Exchange problem does not admit any polynomial kernel parameterized by the number of recipients receiving kidneys+maximum degree+\(\max\{\ell_p, \ell_c\}\) unless \(\text{NP} \subseteq \text{co-NP/poly}\) [Theorem 6].

Finally, we also design a \((\frac{16}{9} + \epsilon)\)-approximation algorithm for the Kidney Exchange problem if only cycles of length at most 3 are allowed (and no paths are allowed) [Theorem 7]. We believe that our work substantially improves the current theoretical understanding of the Kidney Exchange problem.

A short version of this work has been published in IJCAI 2022 [MD22].

2 Preliminaries

For an integer \(k\), we denote the sets \(\{0, 1, \ldots, k\}\) and \(\{1, 2, \ldots, k\}\) by \([k]_0\) and \([k]\), respectively.

A Kidney Exchange problem is formally represented by a directed graph \(G = (V, A)\) which is known as the compatibility graph. A subset \(B \subseteq V\) of vertices denotes altruistic donors (also called non-directed donors); the other set \(V \setminus B\) of vertices denote a recipient-donor pair who wish to participate in the kidney exchange program. We have a directed edge \((u, v) \in A\) if the donor of the vertex \(u \in V\) has a kidney compatible with the recipient of the vertex \(v \in V \setminus B\). Kidney exchanges happen either (i) along a trading-cycle \(u_1, u_2, \ldots, u_k\) where the recipient of the vertex \(u_i \in V \setminus B\) receives a kidney from the donor of the vertex \(u_{i-1} \in V \setminus B\) for every \(2 \leq i \leq k\) and the recipient of the vertex \(u_1\) receives the kidney from the donor of the vertex \(u_k\), or (ii) along a trading-chain \(u_1, u_2, \ldots, u_k\) where \(u_1 \in B, u_i \in V \setminus B\) for \(2 \leq i \leq k\) and the recipient of the vertex \(u_j\) receives a kidney from the donor of the vertex \(u_{j-1}\) for \(2 \leq j \leq k\). Due to operational reasons, all the kidney transplants along a trading-cycle or a trading-chain should be performed simultaneously. This puts an upper bound on the length \(\ell\) of feasible trading-cycles and trading-chains. We define the length of a path or cycle as the number of edges in it. The Kidney exchange clearing problem is to find a collection of feasible trading-cycles and trading-chains which maximizes the number of recipients who receive a kidney. Formally, it is defined as follows.

**Definition 1 (Kidney Exchange).** Given a directed graph \(G = (V, A)\) with no self-loops, an altruistic vertex set \(B \subseteq V\), two integers \(\ell_p\) and \(\ell_c\) denoting the maximum length of, respectively, paths and cycles allowed, and a target \(t\), compute if there exists a collection \(C\) of disjoint cycles of length at most \(\ell_c\) and paths with starting from altruistic vertices only each of length at most \(\ell_p\) which cover at least \(t\) non-altruistic vertices. We denote an arbitrary instance of Kidney Exchange by \((G, B, \ell_p, \ell_c, t)\).
of the underlying undirected graph; two vertices \( t \) of width \( \ell_p \) We additionally require that every edge is introduced exactly once. One can show that a tree decomposition

\[ E \]

of a graph \( G \) if and only if either there is an edge from \( u \) to \( t \) or from \( v \) to \( t \). Also, refer to the Table 1 for the important notations.

| \( B \) | set of altruistic vertices |
| \( t \) | target number of recipients to receive kidneys |
| \( \ell_p \) | length of the longest path allowed |
| \( \ell_c \) | length of the longest cycle allowed |
| \( \tau \) | treewidth of underlying undirected graph |
| \( \theta \) | number of vertex types |
| \( \Delta \) | maximum degree of underlying undirected graph |
| \( \ell \) | max\{\( \ell_p \), \( \ell_c \)\} |

Table 1: Notation table.

### 2.1 Graph Theoretic Terminologies

In a graph \( G \), \( V[G] \) denotes the set of vertices in \( G \) and \( E[G] \) denotes the set of edges in \( G \). Let \( G[V'] \) denote the induced subgraph on \( V' \) where \( V' \subseteq V[G] \). Two vertices \( u \) and \( v \) in a directed graph \( G \) are called vertices of the same type if they have the same set of in-neighbors and the same set of out-neighbors. If there are no self loops in \( G \), vertices of the same type form an independent set. Treewidth measures how treelike an undirected graph is. We refer to [CFK+15] for an elaborate description of treewidth, tree decomposition, and nice tree decomposition.

**Definition 2 (Treewidth).** Let \( G = (V_G, E_G) \) be an undirected graph. A tree-decomposition of a graph \( G \) is a pair \( (T = (V_T, E_T), X = \{X_t\}_{t \in V_T}) \), where \( T \) is a tree where every node \( t \in V_T \) is assigned a subset \( X_t \subseteq V_G \), called a bag, such that the following conditions hold.

- For every edge \( \{x, y\} \in E_G \) there is a \( t \in V_T \) such that \( x, y \in X_t \), and
- For any \( v \in V_G \) the subgraph of \( T \) induced by the set \( \{t \mid v \in X_t\} \) is connected.

The width of a tree decomposition is \( \max_{t \in V_T} |X_t| - 1 \). The treewidth of \( G \) is the minimum width over all tree decompositions of \( G \) and is denoted by \( \tau(G) \).

A tree decomposition \( (T, X) \) is called a nice edge tree decomposition if \( T \) is a tree rooted at some node \( r \) where \( X_r = \emptyset \), each node of \( T \) has at most two children, and each node is of one of the following kinds:

- **Introduce node:** a node \( t \) that has only one child \( t' \) where \( X_t \supset X_{t'} \) and \( |X_t| = |X_{t'}| + 1 \).
- **Introduce edge node** a node \( t \) labeled with an edge between \( u \) and \( v \), with only one child \( t' \) such that \( \{u, v\} \subseteq X_{t'} = X_t \). This bag is said to introduce \( uv \).
- **Forget vertex node:** a node \( t \) that has only one child \( t' \) where \( X_t \subset X_{t'} \) and \( |X_t| = |X_{t'}| - 1 \).
- **Join node:** a node \( t \) with two children \( t_1 \) and \( t_2 \) such that \( X_t = X_{t_1} = X_{t_2} \).
- **Leaf node:** a node \( t \) that is a leaf of \( T \), and \( X_t = \emptyset \).

We additionally require that every edge is introduced exactly once. One can show that a tree decomposition of width \( t \) can be transformed into a nice tree decomposition of the same width \( t \) and with \( O(t|V_G|) \) nodes, see e.g. [BODT13, CFK+15]. For a node \( t \in T \), let \( T_t \) be the subtree of \( T \) rooted at \( t \), and \( V(T_t) \) denote the vertex set in that subtree. Then \( \beta(t) \) is the subgraph of \( G \) where the vertex set is \( \bigcup_{v \in V(T_t)} X_v \) and the edge set is the union of the set of edges introduced in each \( t', t' \in V(T_t) \). We denote by \( V(\beta(t)) \) the set of vertices in that subgraph, and by \( E(\beta(t)) \) the set of edges of the subgraph.

Since our graph is directed, whenever we mention treewidth of our graph, we refer to the treewidth of the underlying undirected graph; two vertices \( u \) and \( v \) are neighbors of the underlying undirected graph if and only if either there is an edge from \( u \) to \( v \) or from \( v \) to \( u \).
2.2 Parameterized Complexity

A tuple \((x, k)\), where \(k\) is the parameter, is an instance of a parameterized problem. Fixed parameter tractability (FPT) refers to solvability in time \(f(k) \cdot p(|x|)\) for a given instance \((x, k)\), where \(p\) is a polynomial in the input size \(|x|\) and \(f\) is an arbitrary computable function of \(k\). We use the notation \(O^*(f(k))\) to denote \(O(f(k)\text{poly}(|x|))\).

We say a parameterized problem is para-NP-hard if it is NP-hard even for some constant values of the parameter.

**Definition 3** (Kernelization). A kernelization algorithm for a parameterized problem \(\Pi \subseteq \Gamma^* \times \mathbb{N}\) is an algorithm that, given \((x, k) \in \Gamma^* \times \mathbb{N}\), outputs, in time polynomial in \(|x| + k\), a pair \((x', k') \in \Gamma^* \times \mathbb{N}\) such that (a) \((x, k) \in \Pi\) if and only if \((x', k') \in \Pi\) and (b) \(|x'|, k' \leq g(k)|\), where \(g\) is some computable function. The output instance \(x'\) is called the kernel, and the function \(g\) is referred to as the size of the kernel. If \(g(k) = k^{O(1)}\), then we say that \(\Pi\) admits a polynomial kernel.

It is well documented that the presence of a polynomial kernel for certain parameterized problems implies that the polynomial hierarchy collapses to the third level (or, more accurately, \(\text{NP} \subseteq \text{co-NP/poly}\)). As a result, polynomial-sized kernels are unlikely to be present in these problems. We use cross-composition to demonstrate kernel lower bounds.

**Definition 4** (Cross Composition). Let \(L \subseteq \Sigma^*\) be a language and \(Q \subseteq \Sigma^* \times \mathbb{N}\) be a parameterized language. We say that \(L\) cross-composes into \(Q\) if there exists a polynomial equivalence relation \(R\) and an algorithm \(A\), called the cross-composition, satisfying the following conditions. The algorithm \(A\) takes as input a sequence of strings \(x_1, x_2, \ldots, x_t \in \Sigma^*\) that are equivalent with respect to \(R\), runs in time polynomial in \(\sum_{i=1}^{t} |x_i|\), and outputs one instance \((y, k) \in \Sigma^* \times \mathbb{N}\) such that:

1. \(k \leq p(\max_{i=1}^{t} |x_i| + \log t)\) for some polynomial \(p(\cdot)\), and
2. \((y, k) \in Q\) if and only if there exists at least one index \(i\) such that \(x_i \in L\).

3 Results

We present our results in this section.

3.1 FPT Algorithms

We begin with presenting an FPT algorithm for the Kidney Exchange problem parameterized by the number of recipients who receive a kidney. Note that, although the number \(t\) of recipients who receive a kidney may not be small, this number is always smaller than the total number \(n\) of donors. Thus an FPT algorithm parameterized by \(t\) will always be efficient on some instances of the problem compared to the \(O(2^n n^3)\) time algorithm of Xiao and Wang [XW18]. We use the technique of color coding [AYZ94, AYZ98] to design our algorithm.

**Theorem 1.** There is an algorithm for the Kidney Exchange problem which runs in time \(O^*(2^{O(t)})\).

**Proof.** Let \((\mathcal{G}, \mathcal{B}, t_p, t_c, t)\) be an arbitrary instance of the Kidney Exchange problem. If \(t_p \geq t\), we check whether there exists a path starting from an altruistic vertex of length \(t\); if \(t_c \geq t\), then we check whether there exists a cycle of length \(t_c\) for some \(t \leq t_c \leq t_e\). Note that this can be checked by a deterministic algorithm in time \(O^*(2^{O(t)})\) (for path refer to [CFK+15] and for cycle refer to [Zeh16]). If there exists such a cycle or path, then clearly \((\mathcal{G}, \mathcal{B}, t_p, t_c, t)\) is a yes instance. So for the rest of the proof, let us update \(t_p\) and \(t_c\) as \(\min\{t_p, t-1\}\) and \(\min\{t_c, t-1\}\), respectively.

We observe that if \((\mathcal{G}, \mathcal{B}, t_p, t_c, t)\) is a yes instance, then there is a collection \(\mathcal{C} = (\mathcal{D}_1, \ldots, \mathcal{D}_k)\) of \(k \leq t\) disjoint cycles and paths starting from altruistic vertices each of length at most \(t_c\) and \(t_p\), respectively, which covers \(t_1\) non-altruistic vertices where \(t \leq t_1 \leq 2t\). Note that such a collection will exist as \(t_c < t\) and \(t_p < t\). We observe that the total number of vertices involved in \(\mathcal{D}_1, \ldots, \mathcal{D}_k\) is at most \(3t\) (at most one altruistic vertices in each \(\mathcal{D}_i, i \in [k]\)). We color each vertex of \(\mathcal{G}\) uniformly at random from a set \(S\).
of $3t$ colors. We say a coloring of $\mathcal{G}$ is good if every vertex in the $C$ gets a different color. A random coloring is good with probability at least
\[
\frac{(3t)!}{(3t)^3t} \geq \frac{1}{e^{3t}}.
\]

Let $C$ be a non-empty set of colors. Let $\mathcal{G}_C$ denote an induced subgraph of $\mathcal{G}$ on the set of vertices colored with one of the colors in $C$. We now solve the Kidney Exchange problem using dynamic programming. We maintain a dynamic programming table $D$ which is indexed by a set of colors $C$ and a number $t'$. $D(C, t')$ denotes whether there is a collection of disjoint cycles and paths starting from altruistic vertices of each length at most $\ell_c$ and $\ell_p$, respectively, which covers $t'$ non-altruistic vertices in the graph $\mathcal{G}_C$. Now we introduce a function $f$ which has a set of colors $C'$ and a number $i$ as its arguments. $f(C', i)$ decides whether there is a valid colourful cycle or path starting from altruistic vertex of length $i$ in $\mathcal{G}_C$. By valid colourful cycle (resp., path) we mean that no two vertices in the cycle (resp., path) have the same color and the length of the cycle (resp., path) is at most $\ell_c$ (resp., $\ell_p$). $f(C', i)$ can be computed in time $O^*(2^{O(i)})$ time (refer to [CFK+15]).

Now we present a recursive formula to compute each entry of the table.

\[
D(C, t') = \bigvee_{C' \subseteq C, C' \neq \emptyset, 1 \leq i \leq t'} (D(C \setminus C', t' - i) \land f(C', i))
\]

For base cases, let $D(C, 0) = 1$ and $D(\emptyset, t') = 0$ if $t' > 0$. Now argue the correctness of the above recursive equation. In one direction, let $D(C, t') = 1$. It implies that there is a collection of valid colourful disjoint cycles and paths starting from altruistic vertices which covers $t'$ non-altruistic vertices in the graph $\mathcal{G}_C$. Now consider once such disjoint cycle or path. Let the number of non-altruistic vertices in it be $i$ and the set of colors of the vertices in it be $C'$. Then, clearly $f(C', i) = 1$ and $D(C \setminus C', t - i) = 1$. In the other direction, let there exist a set $C' \subseteq C$ and a number $i \in [t']$ such that $D(C \setminus C', t' - i) \land f(C', i) = 1$. This implies that there is a collection of valid colourful disjoint cycles and paths starting from altruistic vertices which covers $t' - i$ non-altruistic vertices in the graph $\mathcal{G}_C$. And there is a valid colourful cycle or path starting from altruistic vertex of length $i$ in $\mathcal{G}_C$. Hence there is no a collection of valid colourful disjoint cycles and paths starting from altruistic vertices which covers $t'$ non-altruistic vertices in the graph $\mathcal{G}_C$. Therefore $D(C, t') = 1$.

If $(\mathcal{G}, B, \ell_p, \ell_c, t)$ is a yes instance and coloring is good, then $\bigvee_{t \leq t' \leq 2t} D(S, t') = 1$. For a no instance, $\bigvee_{t \leq t' \leq 2t} D(S, t') = 0$ for any coloring.

The total number of entries in the table $D$ is $2^{O(t)}$. Each entry can be computed in time $O^*(2^{O(t)})$. Hence, with probability at least $e^{-3t}$, our algorithm outputs the correct decision in $O^*(2^{O(t)})$ time. By repeating $O(e^{3t})$ times, we find the correct decision with constant probability. The overall running time of our algorithm is $O^*(2^{O(t)})$. The algorithm can be derandomized by using an $(n, 3t, 3t)$-splitter.

We next formulate a Monadic second-order ($MSO_2$) formula for the Kidney Exchange problem where the length of the formula is upper bounded by $\ell = \max\{\ell_c, \ell_p\}$.

**Theorem 2.** There is a Monadic second-order ($MSO_2$) formula for the Kidney Exchange problem where the length of the formula is upper bounded by a function of $\ell = \max\{\ell_c, \ell_p\}$.

**Proof.** Let $(\mathcal{G} = (V, E), B, \ell_p, \ell_c, t)$ be an arbitrary instance of the Kidney Exchange problem. We now define the atomic formulae $h(e, u)$ and $t(e, u)$ which will be used in the $MSO_2$ formula. If $u$ is a vertex variable and $e$ is an edge variable, then the formula $h(e, u)$ is true if and only if $u$ is the head of the edge $e$. Similarly, if $u$ is a vertex variable and $e$ is an edge variable, then the formula $t(e, u)$ is true if and only if $u$ is the tail of the edge $e$. Note that in an edge $(u, v)$, $u$ is the tail and $v$ is the head.

Let $u$ be a vertex variable, $e$ be an edge variable and $E$ be an edge set variable. We now define some formulae which will be useful to construct the $MSO_2$ formula. First, we define $out(u, E)$ which is true if and only if the outdegree of $u$ is at most 1 in the subgraph $\mathcal{G}' = (V, E)$.

\[
out(u, E) = \forall e_1 \in E[e_1(e_1, u) \rightarrow \neg(\exists e_2 \in E[(e_1 \neq e_2) \land t(e_2, u)])]
\]
Next, we define \( in(u, E) \) which is true iff the indegree of \( u \) is at most 1 in the subgraph \( G' = (V, E) \).

\[
in(u, E) = \forall e_1 \in E \left[ h(e_1, u) \rightarrow \neg(\exists e_2 \in E ((e_1 \neq e_2) \land h(e_2, u))) \right]
\]

Next, we define \( alt(u, E) \) which is true iff in the subgraph \( G' = (V, E) \), \( u \) is an altruistic vertex or \( u \) is an altruistic vertex or \( u \) has outdegree equal to 0 or \( u \) has indegree at least 1.

\[
alt(u, E) = (\neg \exists e \in E \ t(u, e)) \lor (\exists e \in E \ h(u, e)) \lor (\exists v \in E \ u = v)
\]

Next, for \( i \geq 2 \), we define \( dis_i(x_1, \ldots, x_i) \) which is true iff \( x_1, \ldots, x_i \) are pairwise distinct.

\[
dis_i(x_1, \ldots, x_i) = \bigwedge_{j \in [i]} \bigwedge_{k \in [i] \setminus \{j\}} (x_j \neq x_k)
\]

Let the subgraph \( G' = (V, E) \) be a collection of isolated vertices, disjoint paths and disjoint cycles. Then, for \( i \in \{1, 2, \ldots, \ell - 1\} \), we define \( path_i(e, E) \) which is true iff there is a disjoint path in the subgraph \( G' = (V, E) \) which contains the edge \( e \) and there are \( i \) edges between the head of \( e \) and the ending vertex of \( p \).

\[
path_i(e, E) = \exists_{x_1, x_2, \ldots, x_i} \in V \left[ dis_{i+2}(x_1, \ldots, x_{i+1}, x') \land (t(e, x') \land h(x, e_1)) \land (t(e_1, x_1) \land h(e_1, x_2)) \land \ldots \land (t(e_{i-1}, x_{i-1}) \land h(e_{i-1}, x_i)) \land \neg \exists v' \in E \ t(e', x_{i+1}) \right]
\]

Similarly, we define \( path_0(e, E) \) as follows.

\[
path_0(e, E) = \exists_{x_1} \in V \left[ h(x, e_1) \land \neg \exists v' \in E \ t(e', x_1) \right]
\]

Let the subgraph \( G' = (V, E) \) be a collection of isolated vertices, disjoint paths and disjoint cycles. Then, for \( i \in \{1, 2, \ldots, \ell - 1\} \), we define \( cycle_i(e, E) \) which is true iff there is a cycle \( c \) in the subgraph \( G' = (V, E) \) which contains the edge \( e \) and there are \( i + 1 \) edges in the cycle \( c \).

\[
cycle_i(e, E) = \exists_{x_1, x_2, \ldots, x_{i+1}} \in V \left[ dis_{i+1}(x_1, \ldots, x_{i+1}) \land (t(e, x_{i+1}) \land h(x, e_1)) \land (t(e_1, x_1) \land h(e_1, x_2)) \land \ldots \land (t(e_{i-1}, x_{i-1}) \land h(e_{i-1}, x_i)) \right]
\]

We also define \( path_{-1}(e, E), cycle_{-1}(e, E), cycle_0(e, E) \) as false for all the edge variables \( e \) and for all the edge set variables \( E \).

Next, we define our main \( MSO_2 \) formula which is \( \varphi(B, E) \) as follows

\[
\varphi(B, E) = \forall_{e \in V} \left( out(v, E) \land in(v, E) \land alt(v, E) \right) \land \left( \bigvee_{i \in \{-1, 0, \ldots, \ell_p - 1\}} path_i(e, E) \lor \bigvee_{i \in \{-1, 0, \ldots, \ell_e - 1\}} cycle_i(e, E) \right)
\]

We now show that \( \varphi(B, E) \) is true iff the subgraph \( G' = (V, E) \) excluding the isolated vertices is a collection of disjoint paths which start with an altruistic vertex and disjoint cycles of length at most \( \ell_p \) and \( \ell_e \), respectively.

In one direction, let \( \varphi(B, E) \) be true. Then \( \forall_{e \in V} \left( out(v, E) \land in(v, E) \land alt(v, E) \right) \) is true. This implies that the indegree and outdegree of every vertex is at most 1 and disjoint paths, if any begin with an altruistic vertex which implies that \( G' = (V, E) \) is a collection of isolated vertices, disjoint paths beginning with an altruistic vertex and disjoint cycles. Consider any disjoint path \( P \) with an edge \( e \) whose tail is the starting vertex of \( P \). Since \( \varphi(B, E) \) is true, \( \exists i \in \{0, \ldots, \ell_p - 1\} \) such that \( path_i(e, E) \) is true. This implies that the length of \( P \) is at most \( \ell_p \). Similarly consider any disjoint cycle \( C \) with an edge \( e \). Since \( \varphi(B, E) \) is true, \( \exists i \in \{1, \ldots, \ell_e - 1\} \) such that \( cycle_i(e, E) \) is true. This implies that the length of \( C \) is at most \( \ell_e \).

In the other direction, let the subgraph \( G' = (V, E) \) excluding the isolated vertices is a collection of disjoint paths which start with an altruistic vertex and disjoint cycles of length at most \( \ell_p \) and \( \ell_e \).
This implies that the indegree and outdegree of every vertex is at most 1 and disjoint paths begin with an altruistic vertex which implies \( \forall v \in V(\text{out}(v, E) \land \text{in}(v, E) \land \text{alt}(v, E)) \) is true. Consider any edge \( e \) in \( G' = (V, E) \). If \( e \) is part of a path \( P \), then there are \( i \) edges between the head of \( e \) and the ending vertex of \( P \) where \( i \in \{0, \ldots, \ell_p - 1\} \) as the length of \( P \) is at most \( \ell_p \). Similarly, if \( e \) is part of a path \( C \), then there are \( i + 1 \) edges in the cycle \( c \) where \( i \in \{1, \ldots, \ell_c - 1\} \) as the length of \( C \) is at most \( \ell_c \). Therefore, \( \forall e \in E \left( \bigvee_{i \in \{-1, 0, \ldots, \ell_p - 1\}} \text{paths}_i(e, E) \lor \bigvee_{i \in \{-1, 0, \ldots, \ell_c - 1\}} \text{cycles}_i(e, E) \right) \) is true. This implies that \( \varphi(B, E) \) is true, where \( g \) is some computable function [ALS91]. If the maximum value of \( \alpha(E) \) is at least \( t \), we have a YES instance otherwise we have a NO instance. Note that \( ||\varphi|| \) is upper bounded by function of \( \ell \) only. This concludes the proof of this theorem.

It follows immediately from Theorem 2 and [ALS91] that the Kidney Exchange problem is FPT parameterized by \( \tau \) and \( \ell = \max\{\ell_p, \ell_c\} \). However, the running time that we get is not practically useful. Next we proceed to design an efficient dynamic programming based algorithm with running time \( O^*(\ell^{O(\tau)} \cdot \ell^{O(\tau)}) \).

**Theorem 3.** There is an algorithm for the Kidney Exchange problem which runs in time \( O^*(\ell^{O(\tau)} \cdot \ell^{O(\tau)}) \). In particular, Kidney Exchange admits an FPT algorithm parameterized by \( \max\{\ell_p, \ell_c\}, \tau \).

**Proof.** Let \( (G, B, \ell_p, \ell_c, t) \) be an arbitrary instance of the Kidney Exchange problem. Let \( T = (T, \{X_i \subseteq V(G) | i \in V(T)\}) \) be a nice tree decomposition of the input \( u \)-vertex graph \( G \) that has width at most \( \tau \). For each \( t' \) in \( T \), let \( V_{t'} \) be the union of all the bags present in the subtree of \( T \) rooted at \( t' \), including \( X_{t'} \). Let \( G_{t'} = (V_{t'}, E_{t'} = \{e : e \text{ is introduced in the subtree rooted at } t'\}) \) be a subgraph of \( G[V_{t'}] \). We solve the Kidney Exchange problem using dynamic programming. At every node \( t' \) of the tree decomposition, we maintain a dynamic programming table \( D \) which is indexed by the tuple \((u, (P_i)_{i \in [\ell_p]}, (Q_i)_{i \in [\ell_c]}, (E_i)_{i \in [\ell]}), g : [\ell] \rightarrow \{p, c\}, \ell : [\ell] \rightarrow [\ell], a : [\ell] \rightarrow [0, 1], s : [\ell] \rightarrow X_{u} \cup \{\bot\}, e : [\ell] \rightarrow X_{u} \cup \{\bot\})\) where \( u \) is the node in the tree decomposition, \( \mu \in [\tau + 1] \), \((P_i)_{i \in [\ell_p]} \) is a partition of \( X_u \), \( Q_i \) is a permutation of \( P_i \), and \( E_i \) is a set of edges from \( E[G_u] \) having both their end points in \( P_i \) for every \( i \in [\ell_c] \). Here \( \bot \) is a special symbol which is not part of \( V[G] \). We define \( D(u, (P_i)_{i \in [\ell_p]}, (Q_i)_{i \in [\ell_c]}, (E_i)_{i \in [\ell]}), g, \ell, a, s, e \) to be the maximum number of edges present in any subgraph of \( G_u \) such that all the following conditions hold.

- \( \mathcal{H} \) excluding the isolated vertices is a collection of disjoint paths and disjoint cycles.
- Disjoint paths starting with an altruistic vertex have length at most \( \ell_p \) whereas paths without an altruistic vertex have length at most \( \ell_c \).
- Disjoint cycles have length at most \( \ell_c \).
- Disjoint paths which do not have a vertex from \( X_u \) must begin with an altruistic vertex.
- \( \forall[\mathcal{H}] \cap X_u = X_u \setminus P_0 \)
- For each path \( P \in \mathcal{H} \), if \( \exists i, j \in [\ell] \) such that \( P_i \cap V[P] \neq \emptyset \) and \( P_j \cap V[P] \neq \emptyset \), then \( i = j \).
- For each cycle \( C \in \mathcal{H} \), if \( \exists i, j \in [\ell] \) such that \( P_i \cap V[C] \neq \emptyset \) and \( P_j \cap V[C] \neq \emptyset \), then \( i = j \).
- For every \( i \in [\ell_p] \), if \( g(i) = p \), then either \( P_i \) consists of only one vertex which is isolated or \( E[\mathcal{H}] \cap E[G_u[P_i]] \subseteq E[P] \) where \( P \) is a disjoint path in \( \mathcal{H} \), \( Q_i \) is a topological order of \( P_i \), \( E_i = E[\mathcal{H}] \cap E[G_u[P_i]], P \) has \( \ell(i) \) edges and exactly \( a(i) \) altruistic vertices. If \( s(i) = 1 \), then starting vertex of \( P \) is not part of \( P_i \). If \( s(i) = v \), then starting vertex of \( P \) is \( v \) where \( v \in P_i \). If \( \epsilon(i) = 1 \), then ending vertex of \( P \) is not part of \( P_i \). If \( \epsilon(i) = v \), then ending vertex of \( P \) is \( v \) where \( v \in P_i \).
- For every \( i \in [\ell_c] \), if \( g(i) = c \) and \( Q_i = x_1 > \cdots > x_\nu \), then \( E[\mathcal{H}] \cap E[G_u[P_i]] \subseteq E[C] \) where \( C \) is a disjoint cycle in \( \mathcal{H} \), \( Q_i \) is a topological order of \( P_i \), \( \mathcal{H} \) is the path created by removing the edge \((x_\nu, x_1)\) from \( C \), \( E_i = E[\mathcal{H}] \cap E[G_u[P_i]], C \) has \( \ell(i) \) edges.
Formally, we update the table entry as follows.

- There exists an \( i \in [\mu] \) such that \( g(i) = c \) and \( a(i) = 1 \).
- There exists an \( i \in [\mu] \) such that \( g(i) = p \), \( a(i) = 0 \) and \( s(i) = \perp \).
- There exists an \( i \in [\mu] \) such that \( g(i) = c \) and \( \mathcal{L}(i) < \max\{2, |P_i|\} \).
- There exists an \( i \in [\mu] \) such that \( g(i) = p \) and \( \mathcal{L}(i) < |P_i| - 1 \).
- There exists an \( i \in [\mu] \) such that \( P_i \) contains more than \( a(i) \) altruistic vertices.
- There exists an \( i \in [\mu] \) such that \( \mathcal{L}(i) > \ell \).
- There exists an \( i \in [\mu] \) such that \( \mathcal{L}(i) > \ell_c \) and \( g(i) = c \).
- There exists an \( i \in [\mu] \) such that \( \mathcal{L}(i) > \ell_p \) and \( a(i) = 1 \).
- There exists an \( i \in [\mu] \) such that \( Q_i = x_1 > \ldots > x_\lambda \), \( g(i) = p \), \( E_i \not\subseteq \{(x_i, x_{i+1}) : i \in \{1, \ldots, \lambda - 1\}\} \).
- There exists an \( i \in [\mu] \) such that \( Q_i = x_1 > \ldots > x_\lambda \), \( g(i) = c \), \( E_i \not\subseteq \{(x_i, x_{i+1}) : i \in \{1, \ldots, \lambda - 1\}\} \) \( \cup \{(x_\lambda, x_1)\} \).

We do this because none of the DP Table indices satisfying the above conditions can lead to a valid solution. We now use the following formulas to compute table entries in a bottom-up fashion.

**Leaf node:** Since the tree-decomposition \((\mathcal{T}, \{X_u\}_{u \in V[\mathcal{T}]})\) is nice, for every leaf node \( u \) of \( \mathcal{T} \), we have \( X_u = \emptyset \) and thus all the DP table entries at \( u \) is set to 0.

**Introduce vertex node:** Let \( u \in \mathcal{T} \) be an introduce-vertex-node with child \( u' \in \mathcal{T} \) such that \( X_u = X_{u'} \cup \{w\} \) for some vertex \( w \in V[G] \setminus X_{u'} \). Since no edge is introduced in an introduce-vertex-node, \( w \) is an isolated vertex in \( \mathcal{G}_u \). We set the table entry \( \mathcal{D}(u, (P_i)_{i \in [\mu]}), (Q_i)_{i \in [\mu]}, (E_i)_{i \in [\mu]}; g, \mathcal{L}, a, s, e) \) to \( \mathcal{D}(u, (P_i)_{i \in [\mu]}), (Q_i)_{i \in [\mu]}, (E_i)_{i \in [\mu]}; g, \mathcal{L}, a, s, e) \) if \( w \in P_j \) where \( j = 0 \) or \( P_j = \{w\} \), \( g(j) = p \), \( \mathcal{L}(j) = 0 \), \( s(j) = \{w\} \), \( e(j) = \{w\} \), \( a(j) = 1 \) for every \( \mathbb{1}(w) = 1 \) otherwise we set it to \(-\infty\).

**Introduce edge node:** Let \( u \in \mathcal{T} \) be an introduce-edge-node introducing an edge \((x, y) \in \mathcal{E}[G]\) and \( u' \) the child of \( u \). That is, we have \( X_u = X_{u'} \). For the table entry \( \mathcal{D}(u, (P_i)_{i \in [\mu]}), (Q_i)_{i \in [\mu]}, (E_i)_{i \in [\mu]}; g, \mathcal{L}, a, s, e) \), if any of the following conditions hold

- there does not exist any \( i \in [\mu] \) such that \( x, y \in P_i \),
- there exists an \( i \in [\mu] \) such that \( x, y \in P_i \) and \( (x, y) \notin E_i \),

then we set the entry to \( \mathcal{D}(u, (P_i)_{i \in [\mu]}), (Q_i)_{i \in [\mu]}, (E_i)_{i \in [\mu]}; g, \mathcal{L}, a, s, e) \). Otherwise, let us assume that there exists a \( j \in [\mu] \) such that \( x, y \in P_j \) and \( (x, y) \notin E_j \). Intuitively, we compute the appropriate DP table indices at node \( u' \) that we can get when we remove the edge \{x, y\} from the current DP table index at node \( u \). Then, we choose the maximum value among the DP table entries corresponding to DP table indices at node \( u' \) that we computed and add 1 to it to get the value for the current table entry. Formally, we update the table entry as follows.

1. If \( g(j) = p \), then suppose we have \( Q_j = x_1 > \cdots > x_\lambda = y \) \( \cdots > x_\mu \) (note that \( y \) should appear immediately after \( x \) in \( Q_j \), otherwise we would have already set the entry to \(-\infty\)). Let us define \( P'_j = (P'_i)_{i \in [\mu+1]}; Q'_j = (Q'_i)_{i \in [\mu+1]}; E'_j = (E'_i)_{i \in [\mu+1]}; g' : [\mu + 1] \rightarrow \{p, c\}, a' : [\mu + 1] \rightarrow \{0, 1\}, s' : [\mu + 1] \rightarrow X_u \cup \{\perp\}, e' : [\mu + 1] \rightarrow X_u \cup \{\perp\}, \) and \( \mathcal{L}' : [\mu + 1] \rightarrow [\ell] \) for any integer \( k \in [\ell]_0 \) as follows.

   - \( P'_j = \{x_1, \ldots, x_\lambda\}, P'_{j+1} = \{x_{\lambda+1}, \ldots, x_\mu\} \) \( P'_i = P_i \) for every \( i \in [\mu] \), \( i \neq j \)
   - \( Q'_j = x_1 > \cdots > x_\lambda, Q'_{j+1} = x_{\lambda+1} > \cdots > x_\mu, Q'_i = Q_i \) for every \( i \in [\mu] \), \( i \neq j \)
   - \( E'_j = E_j \cap \{(x_1, \ldots, x_\lambda) \times (x_1, \ldots, x_\lambda)\}, E'_{j+1} = E_j \cap \{(x_{\lambda+1}, \ldots, x_\mu) \times (x_{\lambda+1}, \ldots, x_\mu)\} \), \( E'_i = E_i \) for every \( i \in [\mu] \), \( i \neq j \)
   - \( g'(j) = g'(\mu + 1) = p \) and \( g'(i) = g(i) \) for every \( i \in [\mu] \), \( i \neq j \)
   - \( a'(\mu + 1) = 0 \) and \( a'(i) = a(i) \) for every \( i \in [\mu] \)
\[-s'(j) = s(j), s'(\mu + 1) = y \text{ and } s'(i) = s(i) \text{ for every } i \in [\mu]_0, i \neq j \]
\[-e'(j) = e, e'(\mu + 1) = e(j) \text{ and } e'(i) = e(i) \text{ for every } i \in [\mu]_0, i \neq j \]
\[-L^k(j) = k, L^k(\mu + 1) = L(j) - 1 - k, L^k(i) = L(i) \text{ for every } i \in [\mu], i \neq j \]

We now set the entry to \(1 + \max_{k \in [\mu]} D(u', P', Q', L', g', L', a', s', e')\)

- If \(g(j) = c\), then let us define \(g': [\mu] \rightarrow \{p, c\}\) as \(g'(j) = p\) and \(g'(i) = g(i)\) for every \(i \in [\mu], i \neq j\).
- Let us also define \(s': [\mu] \rightarrow X_u \cup \{\perp\}\) as \(s'(j) = y\) and \(s'(i) = s(i)\) for every \(i \in [\mu], i \neq j\).
- Similarly, let us define \(e': [\mu] \rightarrow X_u \cup \{\perp\}\) as \(e'(j) = x\) and \(e'(i) = e(i)\) for every \(i \in [\mu], i \neq j\).
- We set the entry to \(1 + D(u', P_i) \in [\mu]_0, (Q_i) \in [\mu], (E_i) \in [\mu], g, L, a, s, e)\) where \(\forall i \neq j, L'(i) = L(i), L'(j) = L(j) - 1\).

**Forget vertex node:** Let \(u \in T\) be a forget node with a child \(u'\) such that \(X_u = X_{u'} \setminus \{w\}\) for some \(w \in X_{u'}\). In this case, we update the table entry \(D(u, (P_i)\in [\mu]_0, (Q_i)\in [\mu], (E_i)\in [\mu], g, L, a, s, e)\) as follows. For an index \((u', (P''_i)\in [\mu]_0, (Q''_i)\in [\mu], (E''_i)\in [\mu], g'', L'', a'', s'', e'')\) of the DP table at node \(u'\), we define \(\text{DEL}(u', (P''_i)\in [\mu]_0, (Q''_i)\in [\mu], (E''_i)\in [\mu], g'', L'', a'', s'', e'') = (u, P', Q', L', a', s', e')\) as follows.

- If \(w \in P''_0\), then \(P' = (P''_i)\in [\mu]_0, Q' = (Q''_i)\in [\mu], L' = (E''_i)\in [\mu], g'' = g'', L'' = a'', s'' = s', e' = e''\) for every \(i \in [\mu], i \neq j\).
- If there exists \(j \in [\mu]\) such that \(P''_j = \{w\}\) and \(P''_j > 1\), then
  \[-P' = (P'_i)\in [\mu] \text{ where } P'_j = P''_j \setminus \{w\}, P'_i = P''_i \text{ for every } i \in [\mu], i \neq j \]
  \[-\text{We define } Q' = (Q'_i)\in [\mu]. \text{ Suppose we have } Q''_j = x_1 > \cdots > x_\lambda(= w) > \cdots > x_\nu. \text{ Then, we define } Q'_j = x_1 > \cdots > x_\lambda - 1 > x_{\lambda + 1} > \cdots > x_\nu \text{ if } \lambda \notin \{1, \nu\}; Q'_j = x_2 > \cdots > x_\nu \text{ if } \lambda = 1; \]
  \[-E' = (E'_i)\in [\mu] \text{ where } E'_j = E''_j \cap (P'_j \times P'_j), E'_i = E''_i \text{ for every } i \in [\mu], i \neq j \]
  \[-g'(i) = g''(i) \text{ for every } i \in [\mu] \]
  \[-L'(j) = L''(j) - (|E''_j| - |E'_j|), L'(i) = L''(i) \text{ for every } i \in [\mu], i \neq j \]
  \[-a'(i) = a''(i) \text{ for every } i \in [\mu] \]
  \[-s'(i) = s''(i) \text{ for every } i \in [\mu], i \neq j. \text{ If } s'(j) = w, \text{ then } s'(j) = \perp \text{ otherwise } s'(j) = s''(j). \]
  \[-e'(i) = e''(i) \text{ for every } i \in [\mu], i \neq j. \text{ If } e'(j) = w, \text{ then } e'(j) = \perp \text{ otherwise } e'(j) = e''(j). \]
- If there exists \(j \in [\mu]\) such that \(P''_j = \{w\}\), then
  \[-a(j) = 1 \text{ and } g''(j) = p \text{ or } a(j) = 0 \text{ and } g''(j) = c, \text{ then } P' = (P''_i)\in [\mu]_0\{j\}, Q' = (Q''_i)\in [\mu] \{j\}, L' = (E''_i)\in [\mu] \{j\}, g' = g'', L' = L'', a' = a'', s' = s'', e' = e'' \]
  \[-a(j) = 0 \text{ and } g''(j) = p, \text{ then } P' = Q' = g' = L' = a' = s' = e' = \perp. \text{ This ensures that the } \text{DEL}(u', (P''_i)\in [\mu], (Q''_i)\in [\mu], (E''_i)\in [\mu], g'', L'', a'', s'', e'') \text{ doesn't correspond to any DP table index.} \]

Intuitively, applying \(\text{DEL}(\cdot)\) on a DP table index at node \(u'\) leads to a DP table index at node \(u\) which we get when we remove \(w\) from the DP table index at node \(u'\).

We update the table entry \(D(u, (P_i)\in [\mu]_0, (Q_i)\in [\mu], (E_i)\in [\mu], g, L, a, s, e)\) as

\[
\min\{D(u', (P''_i)\in [\mu], (Q''_i)\in [\mu], (E''_i)\in [\mu], g'', L'', a'', s'', e'') : \text{DEL}(u', (P''_i)\in [\mu], (Q''_i)\in [\mu], (E''_i)\in [\mu], g'', L'', a'', s'', e'') \}
\]

**Join node:** For a join node \(u\), let \(u_1, u_2\) be its two children. Note that, we have \(X_u = X_{u_1} = X_{u_2}\). We say that the following pair of tuple \(((u_1, (P''_i)\in [\mu]_0, (Q''_i)\in [\mu], (E''_i)\in [\mu], g'', L'', a'', s'', e'') (u_2, (P''_i)\in [\mu]_0, (Q''_i)\in [\mu], (E''_i)\in [\mu], g'', L'', a'', s'', e'')\) is said to be compatible with \((a, (P_i)\in [\mu]_0, (Q_i)\in [\mu], (E_i)\in [\mu], g, L, a, s, e)\) if the following happens.

\[
\text{For every } i \in [\mu], i \neq j, L'(i) = L(i), L'(j) = L(j) - 1.
\]

\[
\text{If } g(j) = c, \text{ then let us define } g': [\mu] \rightarrow \{p, c\} \text{ as } g'(j) = p \text{ and } g'(i) = g(i) \text{ for every } i \in [\mu], i \neq j.
\]

\[
\text{Similarly, let us define } e': [\mu] \rightarrow X_u \cup \{\perp\} \text{ as } e'(j) = x \text{ and } e'(i) = e(i) \text{ for every } i \in [\mu], i \neq j.
\]

\[
\text{We set the entry to } 1 + D(u', (P_i)\in [\mu]_0, (Q_i)\in [\mu], (E_i)\in [\mu], g, L, a, s, e).
\]
Intuitively, a pair of tuples is said to be compatible with the current DP table index at node $u_1$ construct the paths corresponding to the partition in the current DP table index at node $u$. Proof.

Let $I$ be an instance of the Kidney Exchange problem. We show that $I$ admits a solution of size $\ell(I)$ in time $O^*(2^n \cdot n^3)$ by reducing it to an instance of the problem with a given FPT algorithm parameterized by $\ell$, using the reduction of Xiao and Wang [XW18].

Let $\mathcal{T}$ denote the number of vertex types in a graph $G(V, E)$. Xiao and Wang [XW18] presented an FPT algorithm parameterized by $\theta$ when $\ell_p = \ell_c = |V|$. We improve the result by presenting an FPT algorithm parameterized by $\theta$ when $\ell_p \leq \ell_c$. Towards that, we first present an important lemma on the structure of an optimal solution.

**Lemma 1.** In every Kidney Exchange problem instance when $\ell_p \leq \ell_c$, there exists an optimal solution where the length of every path and cycle in that solution is at most $\theta + 3$.

**Proof.** Let $\mathcal{C}$ (which is a collection of paths starting from altruistic vertices and cycles) be an optimal solution of an instance $I$ of the Kidney Exchange problem with $\ell_p \leq \ell_c$. If the length of every path and cycle in $\mathcal{C}$ is at most $\theta + 3$, then there is nothing to prove. So let us assume otherwise. That is, there either exists a path starting from an altruistic vertex of length more than $\theta + 3$ or a cycle of length more than $\theta + 3$ in $\mathcal{C}$ (or both). We now construct another solution $\mathcal{C}_1$ as follows.

**Case I:** Suppose there exists a path $p = v_1v_2\ldots v_k$ in $\mathcal{C}$ where $v_1$ is an altruistic vertex and $\ell_p \geq k > \theta + 3$. By pigeonhole principal, there exists a vertex type $\gamma$ such that the vertices $v_a$ and $v_b$ are of type $\gamma$ for some indices $1 < a < b < k$. Note that $b \neq a + 1$ as vertices of the same type form an independent set. Consider a path $p_1$ defined as $p_1 = v_1v_2v_{b+1}\ldots v_k$ and cycle $q$ defined as $q = v_{a+1}\ldots v_kv_{a+1}$. Since there exists an edge from $v_1$ to $v_{b+1}$ and the vertices $v_a$ and $v_b$ are of same type (of type $\gamma$), there exists an edge from $v_a$ to $v_{a+1}$. Similarly, since there exists an edge from $v_a$ to $v_{a+1}$ and the vertices $v_a$ and $v_b$ are of same type (of type $\gamma$), there exists an edge from $v_b$ to $v_{b+1}$. Hence, the path $p_1$ and cycle $q$ is well-defined. We now consider the solution $\mathcal{C}_1 = (\mathcal{C} \setminus \{p\}) \cup \{p_1, q\}$. We observe
that both $\mathcal{C}$ and $\mathcal{C}_1$ cover the same number of non-altruistic vertices. $\mathcal{C}_1$ is a valid solution as length of $p_1$ and $q$ is at most $\ell_p$ and we have $\ell_p \leq \ell_c$.

**Case II:** Suppose there exists a cycle $q = v_1v_2\ldots v_kv_1$ in $\mathcal{C}$ where $\ell_c \geq k > \theta + 3$. By pigeonhole principal, there exists a vertex type $\gamma$ such that the vertices $v_a$ and $v_b$ are of type $\gamma$ for some indices $1 < a < b < k$. Note that $b \neq a + 1$ as vertices of the same type form an independent set. Consider cycles $q_1 : v_a v_{a+1} \ldots v_{b-1} v_a$ and $q_2 : v_1 \ldots v_{b-1} v_b v_{b+1} \ldots v_k v_1$. Since there is an edge from $v_{a-1}$ to $v_a$ and the vertices $v_a$ and $v_b$ are of same type, there exists an edge from $v_{b-1}$ to $v_b$. Similarly, since there exists an edge from $v_{a-1}$ to $v_a$ and the vertices $v_a$ and $v_b$ are of same type, there exists an edge from $v_{a-1}$ to $v_b$. Hence, the cycle $q_1$ and $q_2$ are well-defined. We now consider the solution $\mathcal{C}_1 = (\mathcal{C} \setminus \{q\}) \cup \{q_1, q_2\}$. We observe that both $\mathcal{C}$ and $\mathcal{C}_1$ cover the same number of non-altruistic vertices. $\mathcal{C}_1$ is a valid solution as length of $q_1$ and $q_2$ is at most $\ell_c$.

If $\mathcal{C}_1$ has any path or cycle of length more than $\theta + 3$, then we apply the above procedure to construct another optimal solution $\mathcal{C}_2$. We repeat the procedure as long as the current optimal solution has any path or cycle of length more than $\theta + 3$. The above process terminates after finite number of steps with an optimal solution $\mathcal{C}_s$ since we are always replacing longer path/cycle with shorter paths and cycles. Hence, in the optimal solution $\mathcal{C}_s$, the length of every path and cycle is at most $\theta + 3$ which proves the result.

We use the following useful result by Lenstra to design our FPT algorithm.

**Lemma 2** (Lenstra’s Theorem [LJ83]). There is an algorithm for computing a feasible as well as an optimal solution of an integer linear program which is fixed parameter tractable parameterized by the number of variables.

Using Lemmata 1 and 2 we develop an FPT algorithm for Kidney Exchange parameterized by the number $\theta$ of types of vertices in Theorem 4.

**Theorem 4.** There exists an FPT algorithm for the Kidney Exchange problem parameterized by $\theta$ when $\ell_p \leq \ell_c$.

**Proof.** Let $(\mathcal{G} = (V, E), B, \ell_p, \ell_c, t)$ be an arbitrary instance of the Kidney Exchange problem where $\ell_p \leq \ell_c$. We denote the set of types in $\mathcal{G}$ by $\Gamma$ and the type of vertex $v$ by $\gamma(v)$. For a type $\gamma \in \Gamma$, let the set of vertices of type $\gamma$ be $V_\gamma \subseteq V$. If there is a $\gamma \in \Gamma$ such that $V_\gamma$ contains both altruistic and non-altruistic vertices, then we remove the non-altruistic vertices as they have in degree 0 and can’t be part of any feasible solution. For a path/cycle $p = v_1\ldots v_k$, “signature of $p$” is defined as $\gamma(v_1)\ldots\gamma(v_k)$ and we denote it by $\gamma(p)$. For a type $\gamma \in \Gamma$, we denote the number of vertices of type $\gamma$ in $\mathcal{G}$ by $n(\gamma)$. For a type $\gamma \in \Gamma$ and a path/cycle/signature-sequence $p$, we denote the number of vertices of type $\gamma$ in $p$ by $n_p(\gamma)$. For a path/cycle/signature-sequence $p$, we denote the number of non-altruistic vertices in $p$ by $\lambda(p)$. Let $\mathcal{A}$ denote the set of signatures of paths starting from altruistic vertices of length at most $\min\{\theta + 4, \ell_p\}$ and signatures of cycles of length at most $\min\{\theta + 3, \ell_c\}$ in $\mathcal{G}$. Since there are $\theta$ types, we have $|\mathcal{A}| = O(\theta^\theta)$. We can compute the set $\mathcal{A}$ in time $O^*(\theta^\theta)$ as follows. For each possible signature-sequence $p = \gamma_1\ldots\gamma_k$ of a path, we check if there is an edge from a vertex in $V_{\gamma_1}$ to a vertex in $V_{\gamma_{i+1}}$ for all $i \in [k-1]$ and $n_p(\gamma) \leq n(\gamma)$ for all $\gamma \in \Gamma$. If the conditions hold true, then we add $p$ to the set $\mathcal{S}$. Similarly, for each possible signature-sequence $p = \gamma_1\ldots\gamma_k$ of a cycle, we check if there is an edge from a vertex in $V_{\gamma_i}$ to a vertex in $V_{\gamma_{i \text{ mod } k}+1}$ for all $i \in [k]$ and $n_p(\gamma) \leq n(\gamma)$ for all $\gamma \in \Gamma$. If the conditions hold true, then we add $p$ to the set $\mathcal{A}$.

We consider the following integer linear program; its variables are $x(p)$ for every $p \in \mathcal{A}$.

\[
\begin{align*}
\max & \sum_{p \in \mathcal{A}} \lambda(p)x(p) \\
\text{subject to:} & \sum_{p \in \mathcal{A}} n_p(\gamma)x(p) \leq n(\gamma) & \forall \gamma \in \Gamma \\
x(p) & \in \{0, 1, \ldots, |V|\} & \forall p \in \mathcal{A}
\end{align*}
\]

We claim that the Kidney Exchange instance is a yes instance if and only if the optimal value of the above ILP is at least $t$. 


In one direction, suppose the Kidney Exchange instance is a yes instance. Let \( C \) be an optimal solution (a multi-set of paths and cycles) of the Kidney Exchange instance. For signature sequence \( p \in A \), we define \( x(p) = \sum_{q \in C} \delta(p = \gamma(q)) \) where \( \delta() \) is the indicator function. As \( C \) is a subgraph of \( G \), we have \( \sum_{p \in A} n_p(\gamma) \sum_{q \in C} \delta(p = \gamma(q)) \leq n(\gamma), \forall \gamma \in \Gamma \). Hence \( x(p)_{p \in A} \) is a feasible solution. Since \( C \) covers at least \( t \) non-altruistic vertices, it follows that \( \sum_{p \in A} \lambda(p)x(p) \geq t \). Hence the optimal value of the above ILP is at least \( t \).

On the other direction, suppose there exists a solution \((x^*(p))_{p \in A}\) to the ILP such that \( \sum_{p \in A} \lambda(p)x^*(p) \geq t \). We now describe an iterative approach to construct a solution \( C \) for the Kidney Exchange instance from \((x^*(p))_{p \in A}\). We initialize \( C \) to the empty set and initialize \( x'(w) \) to 0 for all \( w \in A \). Let \( n_C(\gamma) \) denote the number of vertices of type \( \gamma \) in \( C \). Till there exists a \( w \in A \) such that \( x'(w) < x^*(w) \), we add to \( C \) a path/cycle \( q \) of signature \( w \) belonging to \( G \setminus C \) and increase \( x'(w) \) by 1. Now we show that if there exists a \( w \in A \) such that \( x'(w) < x^*(w) \), then there is always a path/cycle \( q \) of signature \( w \) in \( G \setminus C \). Due to the way \( A \) is defined and the fact that \( w \in A \), it suffices to show that \( n_w(\gamma) \leq n(\gamma) - n_C(\gamma), \forall \gamma \in \Gamma \). Since \((x^*(p))_{p \in A}\) is a feasible solution, \( \sum_{p \in A} n_p(\gamma)x^*(p) \leq n(\gamma), \forall \gamma \in \Gamma \). Therefore \( n_w(\gamma) \leq \sum_{p \in A} n_p(\gamma)(x^*(p) - x'(p)) \leq n(\gamma) - \sum_{p \in A} n_p(\gamma)x'(w) = n(\gamma) - n_C(\gamma), \forall \gamma \in \Gamma \). Hence there is always a path/cycle \( q \) of signature \( w \). Now when the iterative procedure terminates, the number of non-altruistic vertices covered by \( C \) is \( \sum_{p \in A} \lambda(p)x^*(p) \) which is at least \( t \). Hence, the Kidney Exchange instance is a yes instance. Now the result follows from Lemma 2.\( \square \)

### 3.2 Para-NP-Hardness Result

In this section, we present an intractability result for Kidney Exchange parameterized by \((\Delta + \ell_p + \ell_c)\).

**Definition 5** (X3C'). Given a finite set \( X = \{x_1, x_2, \ldots, x_{3b}\} \) and a collection of 3-element subsets \( C = \{(x_{ib}, x_{jb}, x_{kb})| 1 \leq b \leq m\} \) of \( X \) in which no element in \( X \) appears in more than three subsets, compute if there exists a sub-collection \( C' \) of \( C \) such that every element in \( X \) occurs in exactly one member of \( C' \).

It turns out that the proof of Abraham et al. [ABS07, Theorem 1], which shows \( \text{NP} \)-completeness of the Kidney Exchange problem, can be appropriately modified to get the Observation 1. We use X3C problem which is known to be \( \text{NP} \)-complete [Con85].

**Observation 1.** The Kidney Exchange problem, parameterized by \((\Delta + \ell_p + \ell_c)\), is para-NP-hard.

**Proof.** Consider an instance \((X, C)\) of X3C'. For each subset in \( C \), create a gadget as mentioned in the proof of Abraham et al. [ABS07, Theorem 1]. Now set \( t = 9|C| + |X|, \ell_p = 0 \) and \( \ell_c = 3 \). This completes the construction of the instance of the Kidney Exchange problem. The correctness of the reduction follows from proof of Abraham et al. [ABS07, Theorem 1]. The instance of Kidney Exchange problem which gets created has \( \ell_p = 0, \ell_c = 3 \) and \( \Delta \leq 6 \). Hence, the Kidney Exchange problem, parameterized by \( \Delta + \ell_p + \ell_c \) is para-NP-hard.\( \square \)

### 3.3 Kernelization Results

We now present our result on kernelization for Kidney Exchange. Note that kernelization algorithms are often found quite useful in practice for drastically reducing the size of the input instance. We show that the Kidney Exchange problem admits a polynomial kernel for the parameter \( t + \Delta \) for every constant \( \ell_p \) and \( \ell_c \). Formally we show the following result.

**Theorem 5.** For the Kidney Exchange problem, there exists a vertex kernel of size \( O(t \Delta^{\text{max}(\ell_p, \ell_c)}) \).

**Proof.** Let \((G, B, \ell_p, \ell_c, t)\) be an arbitrary instance of the Kidney Exchange problem. We apply the following reduction rule:

- If there is a vertex \( v \in G \) which participates neither in any cycle of length at most \( \ell_c \) nor in any path of length at most \( \ell_p \) starting from any vertex in \( B \), then delete \( v \).

The correctness of the reduction rule follows from the fact that any vertex that the above reduction rule deletes never participates in any solution (which is a collection of cycles and paths starting from any vertex in \( B \) of length at most \( \ell_c \) and \( \ell_p \), respectively).

In order to find all the vertices which get removed due to the reduction rule, we do the following:
1. Initialize an empty set $S$.

2. For each vertex $v$, using Dijkstra’s algorithm, we find the shortest cycle and the shortest path starting from any vertex in $B$ which contains $v$. If the length of such a cycle and path is greater than $\ell_c$ and $\ell_p$, respectively, then we add $v$ to the set $S$.

3. Delete all the vertices in $S$ and the graph induced on $V \setminus S$ is our reduced instance.

These steps can be done in polynomial time as the shortest directed path and cycle in a graph can be found in polynomial time using Dijkstra’s algorithm. Note that the reduction rule is no longer applicable as no vertex from set $S$ was part of any cycle or path starting from any vertex in $B$ of length at most $\ell_c$ and $\ell_p$, respectively, and their removal doesn’t affect such cycles and paths.

When the reduction rule is no longer applicable, we show that the number of vertices in the graph is at most $O(\Delta^{\max(\ell_p, \ell_c)})$. To see this, let us choose any maximal collection $C$ of cycles and paths (starting from an altruistic vertex) of length at most $\ell_c$ and $\ell_p$. Such a collection can be found by iteratively finding the shortest cycles and paths (starting from an altruistic vertex), adding it to the collection if they satisfy the constraints on the length and deleting them from the input graph. We stop this iterative process when we can’t find any such cycle and path. This step can be done in polynomial time as the shortest directed path and cycle in a graph can be found in polynomial time using Dijkstra’s algorithm. Let $W$ be the set of vertices in $C$. Note that the number of altruistic vertices in $W$ is at most $|W|/2$ as each path starting with an altruistic vertex has at least one non-altruistic vertex and there are no isolated vertices in $W$. We assume $|W| < 2t$, otherwise we output a trivial yes instance as the number of non-altruistic vertices in $W$ would be at least $t$. Since $C$ is maximal and the fact that the reduction rule is no longer applicable, every vertex in $V \setminus W$ should be within a distance of at most $\max\{\ell_p, \ell_c\}$ from at least one vertex of $C$ in the underlying undirected graph. It follows that the number of vertices in $V \setminus C$ is at most $O(\Delta^{\max(\ell_p, \ell_c)})$, which proves the result.

Hence, Theorem 5 shows that we can significantly reduce the size of the Kidney Exchange problem instance where both $\ell_p$ and $\ell_c$ are small, and the maximum degree is also small, to contain only $O(t)$ vertices. Note that there could be practical reasons to ensure both $\ell_p$ and $\ell_c$ to be small as pointed out by Dickerson et al. [DMP+16]. Is it possible to get rid of the exponential dependence on $\max\{\ell_p, \ell_c\}$ in the kernel size in Theorem 5? In particular, does the Kidney Exchange problem admit a polynomial kernel parameterized by $t + \Delta + \ell_p + \ell_c$? We answer this question negatively in Theorem 6.

**Theorem 6.** The Kidney Exchange problem does not admit any polynomial kernel with respect to the parameter $t + \Delta + \ell_p + \ell_c$ unless $\text{NP} \subseteq \text{co-NP}/\text{poly}$.

**Proof.** We show that the Hamiltonian Path problem on simple (no self loop) directed graphs cross-composes into the Kidney Exchange problem. First, we take a polynomial equivalence relation $R$ that places all malformed instances into one equivalence class, while all the well-formed instances are in the same equivalence class if the instances have the same number of vertices. Now, we present the cross-composition algorithm. Given a set of malformed instances of Hamiltonian Path, we provide a trivial no instance of the Kidney Exchange problem. Now let us assume that we are given $k$ well-formed instances $G_1, \ldots, G_k$ of Hamiltonian Path from an equivalence class of $R$ as an input to the cross-composition algorithm. Let the number of vertices in each of these $k$ instances be $n$. Let $k'$ be smallest power of 2 greater than or equal to $k$. Let $\mathcal{T}$ be a complete directed binary tree with $k'$ leaf nodes, root being the only altruistic vertex and the direction of edges in the tree is from the parent to the child. Let $h$ be the height (largest number of edges in a path from the root to a leaf node) of $\mathcal{T}$. Note that $h = O(\log k)$ as $k' \leq 2k$. The cross-composition algorithm creates an instance $\mathcal{H}$ of the Kidney Exchange problem by taking a disjoint union of $G_1, \ldots, G_k$, adding the tree $\mathcal{T}$, and adding an outgoing edge from $i$-th leaf node to every vertex of the $G_i$. Note that the maximum degree $\Delta$ of $\mathcal{H}$ is $n + 1$. Clearly, the Kidney Exchange problem instance $(\mathcal{H}, B = \{u\}, \ell_p = h + n, \ell_c = 0, t = h + n)$ is a yes instance if and only if there exists an integer $i \in [k]$ such that the Hamiltonian Path instance $G_i$ is a yes instance. Since Hamiltonian Path on simple directed graphs is an NP-complete problem [GJ79], it follows that the Kidney Exchange problem does not admit a polynomial kernel with respect to the parameter $t + \Delta + \ell_p + \ell_c$ unless $\text{NP} \subseteq \text{co-NP}/\text{poly}$ [CFK+13].

Hence, if any of $\ell_p$ or $\ell_c$ is large, which can be the case sometimes in practice also, Theorem 6 shows that we cannot hope to always significantly reduce the size of the input instance.
3.4 Approximation Algorithm

We now present our approximation algorithm for the Kidney Exchange problem when only cycles of length at most 3 are allowed; no path is allowed. Biro et al. [BMR09] studied this problem with the name MAX SIZE ≤3-WAY EXCHANGE and proved APX-hardness. A trivial extension from the result on MAX CYCLE WEIGHT ≤k-WAY EXCHANGE in Biro et al. [BMR09] leads to a 2+ε approximation algorithm for MAX SIZE ≤3-WAY EXCHANGE. Now, towards designing the approximation algorithm, we use the 3-Set Packing problem which is defined as follows.

**Definition 6 (3-Set Packing).** Given a universe $U$, a family $F \subseteq 2^U$ of sets of size at most 3, and an integer $k$, compute if there exists a subfamily $W \subseteq F$ of pairwise disjoint sets such that $|W| \geq k$. We denote any instance of 3-Set Packing by $(U, F, k)$.

The following result is due to [Cyg13].

**Lemma 3.** [Cyg13]. For every $\varepsilon > 0$, there is a $(4/3 + \varepsilon)$-approximation algorithm for the 3-Set Packing problem for optimizing $k$.

**Theorem 7.** If there is a $\alpha$-approximation algorithm for 3-Set Packing problem, then there is a $4\alpha/3$-approximation algorithm for MAX SIZE ≤3-WAY EXCHANGE.

**Proof.** Let $G$ denote the input graph associated with the problem instance of MAX SIZE ≤3-WAY EXCHANGE. The high level idea of our algorithm is as follows. Using the graph $G$, we create two instances of the 3-Set Packing problem, solve them using the approximation algorithm for the 3-Set Packing problem, and output the best of the two (corresponding MAX SIZE ≤3-WAY EXCHANGE) solutions.

**Algorithm 1:** Let us define $F_1 = \{\{a, b, c\} \subseteq V[G] : a, b, c \text{ forms a triangle in } G\}$. Using the $\alpha$-approximation algorithm, we compute a solution $S_1$ for the 3-Set Packing instance $(V[G], F_1)$. Let $C_1$ be the set of cycles corresponding to the sets in $S_1$. Suppose the total number of vertices covered by the cycles in $C_1$ be $A_1$.

**Algorithm 2:** Let us define $F_2 = \{\{a, b, c\} \subseteq V[G] : a, b, c \text{ forms a triangle in } G\} \cup \{\{a, b\} \subseteq V[G] : a, b \text{ forms a 2 cycle in } G\}$. Using the $\alpha$-approximation algorithm, we compute a solution $S_2$ for the 3-Set Packing instance $(V[G], F_2)$. Let $C_2$ be the set of cycles corresponding to the sets in $S_2$. Suppose the total number of vertices covered by the cycles in $C_2$ be $A_2$.

If $A_1 > A_2$, then our algorithm returns $C_1$; otherwise our algorithm returns $C_2$. Since $|F_1|, |F_2| = \mathcal{O}(n^3)$, our algorithm runs in polynomial time.

We now prove approximation ratio of our algorithm. Towards that, let us fix an optimal solution $C_{\text{OPT}}$ which covers OPT number of vertices. Let OPT$_i$ denote the number of vertices covered by cycles of length $i$ in $C_{\text{OPT}}$ for $i = 2, 3$. Then, we have $\text{OPT} = \text{OPT}_2 + \text{OPT}_3$. The $\alpha$-approximation algorithm used in Algorithm 1 gives us the following.

$$A_1 \frac{\alpha}{3} \geq \frac{\text{OPT}_3}{3\alpha} \Rightarrow A_1 \geq \frac{\text{OPT}_3}{\alpha}$$

Similarly, the $\alpha$-approximation algorithm used in Algorithm 2 gives us the following.

$$A_2 \frac{\alpha}{2} \geq \frac{1}{\alpha} \left( \frac{\text{OPT}_3}{3} + \frac{\text{OPT}_2}{2} \right)$$

$$\Rightarrow A_2 \geq \frac{2}{\alpha} \left( \frac{\text{OPT}_3}{3} + \frac{\text{OPT}_2}{2} \right)$$

$$= \frac{2}{\alpha} \left( \frac{\text{OPT}_2}{2} + \frac{\text{OPT}_3}{6} \right)$$

$$= \frac{1}{\alpha} \left( \text{OPT} - \frac{\text{OPT}_3}{3} \right)$$

[OPT = OPT$_2$ + OPT$_3$]

Hence, we can bound $\max\{A_1, A_2\}$ as follows which proves the claimed approximation guarantee of our algorithm.

$$\max\{A_1, A_2\} \geq \frac{\text{OPT}}{4\alpha/3} \square$$
Theorem 7 immediately gives us the following corollary.

**Corollary 1.** For every \( \varepsilon > 0 \), there is a \( \left( \frac{40}{9} + \varepsilon \right) \)-approximation algorithm for the Kidney Exchange problem if only cycles of length at most 3 are allowed (and no paths are allowed).

## 4 Conclusion and Open Questions

In this paper, we have presented FPT algorithms for the Kidney Exchange problem with respect to some natural parameters, namely (i) solution size \( t \) (the number of recipients receiving a kidney), (ii) treewidth+max\{\( \ell_p, \ell_c \)\} and (iii) the number of vertex types when \( \ell_p \leq \ell_c \). We also depicted an MSO_2 formula for the Kidney Exchange problem where the length of the formula is upper bounded by a function of max\{\( \ell_p, \ell_c \)\}. For kernelization, we have exhibited a polynomial kernel w.r.t \( \Delta + t \) when max\{\( \ell_p, \ell_c \)\} is \( O(1) \). We have complemented this result by refuting existence of a polynomial kernel parameterized \( \Delta + t + \max\{\ell_p, \ell_c\} \) unless \( \text{NP} \subseteq \text{co-NP/poly} \). We have finally presented an \( \left( \frac{40}{9} + \varepsilon \right) \)-approximation algorithm for the Kidney Exchange problem in the special case when cycles of length at most 3 are allowed and no path is allowed.

Our work leaves many interesting open questions. One such question is whether the Kidney Exchange problem is FPT w.r.t treewidth only. Another such question is whether the Kidney Exchange problem is FPT w.r.t number of vertex types (without any assumption on \( \ell_c \) and \( \ell_p \)). Another important question is the existence of a polynomial kernel for the Kidney Exchange problem parameterized by the solution size alone when the maximum allowed length of paths and cycles are the same. Our hardness proof in Theorem 6 breaks down if we want to allow paths and cycles of the same length. Finally, it is interesting to study whether the running time of our FPT algorithms can be improved further or are they best possible assuming standard complexity-theoretic assumptions like ETH or SETH.

## References

- [ABS07] David J. Abraham, Avrim Blum, and Tuomas Sandholm. Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. In Jeffrey K. MacKie-Mason, David C. Parkes, and Paul Resnick, editors, Proc. 8th ACM Conference on Electronic Commerce (EC-2007), pages 295–304. ACM, 2007.
- [ALS91] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.
- [Alv04] Alvin Roth, Tayfun Sömmez, Utku Ünver. Kidney exchange. *Quarterly Journal of Economics*, 119(2):457–488, 2004.
- [Alv05a] Alvin Roth, Tayfun Sömmez, Utku Ünver. A kidney exchange clearinghouse in new england. *Am. Econ. Rev.*, 95(2):376–380, 2005.
- [Alv05b] Alvin Roth, Tayfun Sömmez, Utku Ünver. Pairwise kidney exchange. *J. Econ. Theory*, 125(2):151–188, 2005.
- [Alv07] Alvin Roth, Tayfun Sömmez, Utku Ünver. Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *Am. Econ. Rev.*, pages 828–851, 2007.
- [AYZ94] Noga Alon, Raphy Yuster, and Uri Zwick. Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs. In Proc. 26th annual ACM symposium on Theory of computing, pages 326–335, 1994.
- [AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995.
- [BHK+22] Rémy Belmonte, Tesshu Hanaka, Masaaki Kanzaki, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Michael Lampis, Hirotaka Ono, and Yota Otachi. Parameterized complexity of \( (a, 1) \)-path packing. *Algorithmica*, 84(4):871–895, 2022.
[BMR09] Péter Biro, David F Manlove, and Romeo Rizzi. Maximum weight cycle packing in directed graphs, with application to kidney exchange programs. *Discrete Math Algorithms Appl*, 1(04):499–517, 2009.

[BOD18] A faster parameterized algorithm for pseudoforest deletion. *Discrete Applied Mathematics*, 236:42–56, 2018.

[CFK+15] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[CKVR13] Miguel Constantino, Xenia Klimentova, Ana Viana, and Abdur Rais. New insights on integer-programming models for the kidney exchange problem. *Eur. J. Oper. Res.*, 231(1):57–68, 2013.

[Cyg13] Marek Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 509–518. IEEE, 2013.

[DMP+16] John P Dickerson, David F Manlove, Benjamin Plaut, Tuomas Sandholm, and James Trimble. Position-indexed formulations for kidney exchange. In *Proc. 2016 ACM Conference on Economics and Computation*, pages 25–42, 2016.

[GJ79] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[Gon85] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[GvdKW14] Kristiaan M Gorie, J Joris van de Klundert, and Albert PM Wagelmans. Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manuf. Serv. Oper. Manag.*, 16(4):498–512, 2014.

[JTWZ17] Zhipeng Jia, Pingzhong Tang, Ruosong Wang, and Hanrui Zhang. Efficient near-optimal algorithms for barter exchange. In *Proc. 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 362–370, 2017.

[KAV14] Xenia Klimentova, Filipe Alvelos, and Ana Viana. A new branch-and-price approach for the kidney exchange problem. In *Proc. International Conference on Computational Science and Its Applications*, pages 237–252. Springer, 2014.

[KNS+07] Michael Krivelevich, Zeev Nutov, Mohammad R Salavatipour, Jacques Verstraete, and Raphael Yuster. Approximation algorithms and hardness results for cycle packing problems. *ACM T Algorithms*, 3(4):48–es, 2007.

[LJ83] Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[LLHT14] Jian Li, Yicheng Liu, Lingxiao Huang, and Pingzhong Tang. Egalitarian pairwise kidney exchange: fast algorithms via linear programming and parametric flow. In *AAMAS*, pages 445–452, 2014.

[LWFF19] Mugang Lin, Jianxin Wang, Qilong Feng, and Bin Fu. Randomized parameterized algorithms for the kidney exchange problem. *Algorithms*, 12(2):50, 2019.

[MD22] Arnab Maiti and Palash Dey. Parameterized algorithms for kidney exchange. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022*, Vienna, Austria, 23-29 July 2022, pages 405–411. ijcai.org, 2022.

[MH17] Vicky Mak-Hau. On the kidney exchange problem: cardinality constrained cycle and chain problems on directed graphs: a survey of integer programming approaches. *J Comb Optim*, 33(1):35–59, 2017.

[MO15] David F Manlove and Gregg O’malley. Paired and altruistic kidney donation in the uk: Algorithms and experimentation. *J. Exp. Algorithmics*, 19:1–21, 2015.
Organ Donation and Transplantation Statistics. [link]

Felix T Rapaport. The case for a living emotionally related international kidney donor exchange registry. In Transplantation proceedings, volume 18, page 5, 1986.

David Gamarnik Ross Anderson, Itai Ashlagi and Alvin E Roth. Finding long chains in kidney exchange using the traveling salesman problem. Proc. National Academy of Sciences, 112(3):663–668, 2015.

Dorry L Segev, Sommer E Gentry, Daniel S Warren, Brigitte Reeb, and Robert A Montgomery. Kidney paired donation and optimizing the use of live donor organs. Jama, 293(15):1883–1890, 2005.

Tayfun Sönmez and M Utku Ünver. Altruistically unbalanced kidney exchange. J. Econ. Theory, 152:105–129, 2014.

United Network for Organ Sharing (UNOS). [link]

United States Renal Data System (USRDS). [link]

Mingyu Xiao and Xuanbei Wang. Exact algorithms and complexity of kidney exchange. In IJCAI, pages 555–561, 2018.

Meirav Zehavi. A randomized algorithm for long directed cycle. Information Processing Letters, 116(6):419–422, 2016.