Parameterized Complexity of Kidney Exchange Revisited

Úrsula Hébert-Johnson, Daniel Lokshtanov, Chinmay Sonar and **Vaishali Surianarayanan**
UC Santa Barbara
{ursula, daniello, csonar, vaishali}@ucsb.edu

Abstract

As of January 2023, there are more than 90,000 people on the national transplant waiting list in need of a kidney in the United States. These patients often have a friend or family member who is willing to donate, but whose kidney type might not be compatible. To help match these patients to suitable donors, patient-donor compatibility can be modeled as a directed graph. Specifically, in the Kidney Exchange problem, the input is a directed graph $\mathcal{G}$, a subset $\mathcal{B}$ of vertices (altruistic donors), and two integers ℓ_p and ℓ_c. An altruistic donor is a donor who is not paired with a patient, and the remaining vertices are patient-donor pairs. Whenever a donor is compatible with a patient from a patient-donor pair, we place a directed edge from the donor vertex to the patient-donor pair. Here the donor vertex can be either altruistic or non-altruistic. The goal is to find a collection of vertex-disjoint cycles and paths covering the maximum number of patients such that each cycle has length at most ℓ_c, and such that each path has length at most ℓ_p and begins at a vertex in $\mathcal{B}$. The path and cycle lengths are bounded so that the surgeries for a given path or cycle can be performed simultaneously.

Kidney Exchange has received a great deal of attention in recent years. We contribute to this line of work by closing two open problems from IJCAI '18 and IJCAI '22: "Is Kidney Exchange FPT when parameterized by (i) the treewidth (ω) of $\mathcal{G}$ and (ii) the number of vertex types in $\mathcal{G}$?" Two vertices have the same vertex type if they have the same in- and out-neighborhoods. We show that Kidney Exchange is FPT parameterized by the number of vertex types. On the other hand, we show W[1]-hardness with respect to ω. We also design a randomized $4^t n^{\mathcal{O}(1)}$-time algorithm parameterized by t, the number of patients helped, significantly improving upon the previous state of the art, which was $161^t n^{\mathcal{O}(1)}$.

1 Introduction

Kidney disease is a critical problem that affects tens of thousands of patients in the United States and hundreds of millions across the world [Saran *et al.*, 2020; Kovesdy, 2022]. To avoid a lifetime of dialysis appointments, patients with kidney failure generally join a waiting list to be matched with a compatible donor and receive a transplant. Unfortunately, the demand for kidney donations far exceeds the current number of transplants. In the United States, for example, about 40,000 people join the waiting list for a kidney transplant each year; in comparison, only about 20,000 patients find a match each year. This discrepancy has several implications. Naturally, the number of people on the waiting list is continually increasing. From the year 1995 to the present day, the number of people waiting for a kidney donor has more than doubled, from approximately 42,000 to more than 90,000 [Walsh, 2021]. Because of this, waiting times have become quite long. Currently, the median waiting time for a kidney transplant (from a deceased donor) is 4.05 years in the U.S. [Stewart *et al.*, 2023]. To make matters worse, many patients fail to ever find a match. Thousands of patients are removed from the waiting list each year because they either pass away or simply become too sick to undergo the surgery.

Patients in this scenario often have a friend or a family member who is willing to donate a kidney (a *paired donor*); however, this donor's kidney type might not be compatible with that of the patient. In 2000, the Kidney Paired Donation (KPD) or Kidney Exchange program was introduced [Rapaport, 1986; Roth *et al.*, 2004] to address this issue and increase the number of patients who receive a transplant. KPD is a centrally administered barter-exchange market for kidney donations. A patient with an incompatible donor can participate in the market in the hope of exchanging their donor with a compatible donor from another participating pair. Since its inception, the popularity of KPD has grown, both in terms of the number of participating pairs and in terms of the number of successful transplants. The program now operates in several countries including the United States, the United Kingdom, the Netherlands [Dickerson *et al.*, 2016], and India [Pahwa *et al.*, 2012]. With this program, patients have a higher chance of finding a compatible donor, and some patients can also receive a higher-quality match. For example, matching blood type and age can lead to a longer period of healthy kidney functionality [Segev *et al.*, 2005]. KPD gives

rise to a natural computational problem known as the *kidney exchange* problem, in which the goal is to maximize the number of patients who receive a transplant.

The goal in kidney exchange is to find *cycles* or *chains/paths* of compatible patient-donor pairs in order to maximize the number of possible transplants. Therefore, an instance of this problem can be represented as a directed graph: each patient-donor pair is a vertex, and there is a directed edge from a vertex v_i to v_j if the donor at v_i is compatible with the patient at v_j. The length of a cycle (resp. path) is defined as the number of edges in the cycle (resp. path). Each patient that belongs to a cycle (resp. path) receives a compatible kidney from the previous vertex on that cycle (resp. path). All transplants from a particular cycle must be done simultaneously as each donor is under no obligation to donate once their paired patient has received a kidney [Roth *et al.*, 2005; Mak-Hau, 2017]. Therefore, to manage the logistics simultaneous transplants, smaller cycles are required in many real-life settings [Abraham *et al.*, 2007; Manlove and O'Malley, 2015] including the United Network for Organ Sharing.

For a path to be feasible, it must begin with a donor having no paired patient. We refer to such donors as *altruistic donors*, and we refer to the corresponding vertices as *altruistic vertices*. We can afford much longer paths compared to the length of cycles [Ashlagi *et al.*, 2012; Dickerson *et al.*, 2016] as in some cases all of the transplants on a path need not be done simultaneously [Anderson *et al.*, 2015]. While sequentially operating on a path, if a donor leaves the program immediately after their paired patient receives a kidney, then in the remainder of the tail no patient is worse off than they were before joining the program. In particular, such patient-donor pairs can re-enter the KPD program.

We now define the Kidney Exchange problem formally. The input consists of a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ known as the *compatibility graph* (whose vertices are altruistic donors and patient-donor pairs), integers ℓ_p and ℓ_c, which denote the maximum permissible length for a path or cycle in the solution, respectively, and a subset $\mathcal{B} \subseteq \mathcal{V}$, denoting the altruistic vertices. The remaining set $\mathcal{V} \setminus \mathcal{B}$ contains the patient-donor pairs. We have a directed edge $(u, v) \in \mathcal{A}$ if the donor at the vertex $u \in \mathcal{V}$ has a kidney compatible with the patient at $v \in \mathcal{V} \setminus \mathcal{B}$.

The length of a path (resp. cycle) is defined as the number of edges on the path (resp. cycle). We say a path in $\mathcal{G}$ is *feasible* if it starts at an altruistic vertex and has length at least 1 and at most ℓ_p; a cycle in $\mathcal{G}$ is *feasible* if it has length at most ℓ_c. We say a path (resp. cycle) *covers* the non-altruistic vertices (patients) that appear on that path (resp. cycle).

Kidney Exchange Problem (KEP)
Input: A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ with no self-loops, an altruistic set $\mathcal{B} \subseteq \mathcal{V}$ such that each vertex in $\mathcal{B}$ is a source in $\mathcal{G}$, and two nonnegative integers ℓ_p and ℓ_c.
Goal: Find a set of vertex-disjoint feasible cycles and feasible paths covering the maximum number of non-altruistic vertices in $\mathcal{G}$.

For the rest of the paper, we denote instances of KEP by $(\mathcal{G}, \mathcal{B}, \ell_p, \ell_c)$, and the number of vertices in $\mathcal{G}$ by n.

1.1 Previous Work on KEP and Our Contributions

Many variants of Kidney Exchange have been studied in recent years, most of which boil down to finding a maximum packing of cycles and paths in a directed graph [Akbarpour *et al.*, 2014; Ashlagi and Roth, 2014; Roth *et al.*, 2005; Segev *et al.*, 2005]. Since this underlying combinatorial problem is NP-hard, many variants of Kidney Exchange are known to be NP-hard even for very restricted cases [Krivelevich *et al.*, 2007; Abraham *et al.*, 2007; Biro *et al.*, 2009]. To cope with this NP-hardness, Kidney Exchange has been studied through the lens of approximation algorithms [Jia *et al.*, 2017; Krivelevich *et al.*, 2007] and parameterized complexity [Xiao and Wang, 2018; Maiti and Dey, 2022]. Our work focuses on the latter, exploring the parameterized complexity of this problem.

The Kidney Exchange problem (KEP) as described above was first introduced in [Glorie *et al.*, 2014]. For the special case of KEP where ℓ_p and ℓ_c are unrestricted, [Xiao and Wang, 2018] gave an algorithm with running time $f(\theta)n^{\mathcal{O}(1)}$, where θ is the number of vertex types in $\mathcal{G}$ (two vertices have the same vertex type if they have the same in- and out-neighborhoods[1]) and f is a computable function of θ. (Such an algorithm is known as an FPT algorithm parameterized by θ.) Subsequently, [Maiti and Dey, 2022] gave a $f(\theta)n^{\mathcal{O}(1)}$ time algorithm for instances of KEP where $\ell_p \leq \ell_c$, and posed as an open problem whether there exists an $f(\theta)n^{\mathcal{O}(1)}$-time algorithm for KEP without any assumptions on ℓ_p, ℓ_c. Additionally, [Maiti and Dey, 2022] obtain a $f(\omega + \max\{\ell_p, \ell_c\})n^{\mathcal{O}(1)}$-time algorithm for KEP, where ω is the treewidth of $\mathcal{G}$ (a measure of the "treelikeness" of $\mathcal{G}$), and asked whether KEP can be solved in time $f(\omega)n^{\mathcal{O}(1)}$. Finally, [Maiti and Dey, 2022] considered the number of patients helped (t) as a parameter, and gave an algorithm with running time $\sim 161^t n^{\mathcal{O}(1)}$.

In this work we make a step forwards towards a more complete understanding of the parameterized complexity of KEP. In particular, we prove the following results:

1. KEP admits a randomized algorithm with running time $4^t n^{\mathcal{O}(1)}$ (Theorem 2). This improves over the $\sim 161^t n^{\mathcal{O}(1)}$ time algorithm of [Maiti and Dey, 2022].
2. KEP admits an algorithm with running time $f(\theta)n^{\mathcal{O}(1)}$ (Theorem 1). This generalizes the previous algorithms of [Maiti and Dey, 2022; Xiao and Wang, 2018].
3. Unless FPT = W[1], KEP does not admit an algorithm with running time $f(\omega)n^{\mathcal{O}(1)}$ (Theorem 3).

Our second and third result resolves two open problems of [Maiti and Dey, 2022]. We note that we defer some of our proofs (marked with $\star$) to the supplementary material.

Other work on computing exact solutions: Apart from the recent developments in parameterized algorithms, the main approaches in the literature are based on Integer Linear Programming [Roth *et al.*, 2007; Constantino *et al.*, 2013; Mak-Hau, 2017] and the branch and price approach [Barnhart *et al.*, 1998; Plaut *et al.*, 2016]. While these algorithms

[1]For undirected graphs, the number of types is also referred to as "neighborhood diversity" in the literature [Lampis, 2012].

are guaranteed to eventually find an optimal solution, no running time guarantees are given. Finally, [Xiao and Wang, 2018] gave a dynamic-programming-based algorithm running in time $\mathcal{O}(2^n n^3)$.

Motivation for studying the parameters θ, t, and ω: Kidney compatibility depends on a few health parameters such as blood type and age. [Dickerson *et al.*, 2017] observed that in real-world Kidney Exchange instances, these take on only few different values. Thus the number of vertex types (θ) in $\mathcal{G}$ is typically small. Our other parameters, the number t of patients helped and the treewidth ω of $\mathcal{G}$, are well studied parameters that have improved the understanding of many well-known graph problems. These parameters, being much smaller than $n = |V(\mathcal{G})|$, help us design algorithms that are faster than the existing exact exponential time algorithms, e.g., the $\mathcal{O}(2^n n^3)$ time algorithm by [Xiao and Wang, 2018]. All three parameters, t, θ and ω, have been studied in the past for KEP and we continue this work.

2 Preliminaries

We represent by $\mathbb{N}$ the set of natural numbers including zero. For $k \in \mathbb{N}$, we denote the set $\{1, 2, \ldots, k\}$ by $[k]$. Given a graph G and a subset $X \subseteq V(G)$, we denote the induced subgraph of G on X by $G[X]$. We say a partition of a multiset $\mathcal{X}$ into ℓ parts is a collection $\mathcal{X}_1 \uplus \cdots \uplus \mathcal{X}_\ell$ of non-overlapping, possibly empty submultisets of $\mathcal{X}$ whose union is $\mathcal{X}$. These multisets are non-overlapping in the following sense: for every $x \in \mathcal{X}$, the multiplicity of x in $\mathcal{X}$ is equal to the sum over i of the multiplicity of x in $\mathcal{X}_i$.

The treewidth of an undirected graph is a measure of how close the graph is to a tree. We refer the reader to [Cygan *et al.*, 2015] for a complete description of treewidth and tree decompositions. Whenever we mention the treewidth of a directed graph, we refer to the treewidth of the underlying undirected graph. We say two vertices in $\mathcal{G}$ have the same *vertex type* if they are (i) both altruistic or both non-altruistic and (ii) have the same set of in-neighbors and out-neighbors.

Let G be an undirected simple graph (graph with no self-loops and multi-edges). A vertex coloring of G using at most c colors is a function from $\chi_V : V(G) \to [c]$ such that for each edge $e = (u, v) \in E(G)$, $\chi_V(u) \neq \chi_V(v)$. For a vertex v in G, we refer to $\chi_V(v)$ as the color of the vertex v. An edge coloring of G using at most c colors is a function from $\chi_E : E(G) \to [c]$ such that for each pair of distinct edges $e_1, e_2 \in E(G)$ having an common end point, $\chi_E(e_1) \neq \chi_E(e_2)$. For an edge e in G, we refer to $\chi_E(e)$ as the color of the edge e. We will use greedy coloring and Vizing's theorem [Lewis, 2015] to efficiently obtain a vertex coloring and an edge coloring for our hardness result.

Now let H be a directed graph. A *walk* in H is a sequence of vertices $v_1, v_2, \ldots, v_k \in V(H)$ such that there is a directed edge from v_i to v_{i+1} for all $1 \leq i \leq k-1$. A *closed or cyclic walk* in H is a walk such that $v_1 = v_k$. In other words, a walk is a path in which we are allowed to repeat vertices, and a closed walk is a cycle in which we are allowed to repeat vertices. Closed walks with distinct starting points are considered to be distinct: for example, the closed walk v_1, v_2, v_3, v_4, v_1 is distinct from v_2, v_3, v_4, v_1, v_2, assuming the vertices v_1, v_2, v_3, v_4 are distinct.

3 FPT Algorithm Parameterized by θ

We fix an instance $(\mathcal{G}, \mathcal{B}, \ell_p, \ell_c)$ of KEP. In this section, we outline our FPT algorithm for KEP parameterized by θ, the number of vertex types in $\mathcal{G}$. Vertices of the same type form an independent set since $\mathcal{G}$ has no self-loops[2].

Theorem 1 ($\star$)**.** *KEP admits an algorithm running in time $2^{2^{\mathcal{O}(\theta^2 \log \theta)}} n^{\mathcal{O}(1)}$, where θ is the number of vertex types in the input graph $\mathcal{G}$.*

In our algorithm, we reduce the problem to an ILP with few ($\sim 2^{\mathcal{O}(\theta^2 \log \theta)}$) variables and use an existing [Lenstra Jr, 1983] FPT algorithm for ILPs parameterized by the number of variables.

Proposition 1 ([Lenstra Jr, 1983])**.** *There is an algorithm for computing a feasible as well as an optimal solution of an ILP which is FPT by the number of variables t and runs in time $2^{\mathcal{O}(t)} t^t L^{\mathcal{O}(1)}$, where L is the total number of bits required to encode the ILP.*

For our algorithm, we show that there exists an optimal solution to the problem that can be represented succinctly. We then design an ILP with few variables to find such a succinct representation of an optimal solution. The former is the *crux* of our result. Due to space constraints, we now give an overview of how to obtain this succinct representation and briefly discuss the ILP. The precise ILP formulation and other details are deferred to the supplementary material.

We first show how to represent a solution to KEP in terms of the types in $\mathcal{G}$. We introduce the quotient graph Q to represent $\mathcal{G}$ in terms of the types in $\mathcal{G}$. We relate paths (cycles) in the solution to walks (cyclic walks) in Q and extend this relationship to solutions, which are sets of paths and cycles.

Definition 1. *The **quotient graph** $Q = (V_Q, A_Q, w_Q)$ of $\mathcal{G}$ is a directed graph with vertex weights given by $w_Q : V_Q \to \mathbb{Z}^+$. Each vertex $v \in V_Q$ represents a type in $\mathcal{G}$, and the weight $w_Q(v)$ is the number of vertices of that type. There is an arc from a vertex u to a vertex v in A_Q if and only if there is an arc from a type u vertex to a type v vertex in $\mathcal{G}$. Let $B_Q \subseteq V_Q$ be the set of altruistic types.*

We can succinctly represent a walk in $\mathcal{G}$ by recording how many times it transitions between each pair of vertex types. This is captured by the notion of configuration graphs.

Definition 2. *A **configuration graph**[3] $H = (V_H, A_H, w_H)$ of Q is an edge-weighted connected graph arc set with $w_H : A_H \to \mathbb{Z}^+$ such that $V_H \subseteq V_Q$ and $A_H \subseteq A_Q$.*

The next definition captures configuration graphs that encode feasible paths and cycles in $\mathcal{G}$.

[2]Our result can also be extended to allow types that are cliques (for the case with self-loops), but we give an algorithm for the case with only independent sets for ease of exposition.

[3]This is slightly different from the definition in the supplementary material for ease of exposition.

Definition 3. *A configuration graph H is **path-feasible (cycle-feasible)** if there is a feasible path (cycle) R in $\mathcal{G}$ that only uses vertex types from V_H, such that for each $e = (s, t) \in A_H$, R contains exactly $w_H(e)$ arcs from vertices of type s to vertices of type t. We say such an R **realizes** H and denote the configuration realized by R by $H(R)$.*

Observe that every path (cycle) R in $\mathcal{G}$ realizes the unique configuration $H(R)$, but a configuration can be realized by multiple paths (cycles) covering the same set of vertices in $\mathcal{G}$. Furthermore, a configuration can either be path-feasible or cycle-feasible but not both. One can use Eulerian paths (cycles) to identify path-feasible (cycle feasible) configurations.

An *Eulerian path (Eulerian cycle)* of an edge weighted directed graph G with weight function $w : A(G) \to \mathbb{Z}^+$ on the arc set of G is a walk (cyclic walk) in G that travels through every edge e in G exactly $w(e)$ times. An Eulerian path starts and ends at different vertices in G. The existence of an Eulerian path or Eulerian cycle in G can be verified by checking the connectivity of G and the weighted degrees of each vertex in G. There exists an Eulerian cycle in G if and only if G is connected and for each vertex in G, its weighted in-degree is equal to its weighted out-degree. A slightly modified check can be done for Eulerian paths too.

Lemma 1 ($\star$). *Let H be a configuration graph of Q. H is path-feasible if and only if H contains an Eulerian path of length at most ℓ_p that starts at an altruistic type and in which each vertex $v \in V_H$ occurs at most $w_Q(v)$ times. H is cycle-feasible if and only if H contains an Eulerian cycle of length at most ℓ_c in which each vertex $v \in V_H$ repeats at most $w_Q(v)$ times.*

We note that we will have degree constraints in our ILP to check the Eulerian property of configuration graphs, in turn checking the feasibility of paths and cycles by Lemma 1. We are now ready to define how to represent solutions in terms of configuration graphs.

Definition 4. *The **configuration multiset of a solution** S to the instance $(\mathcal{G}, \mathcal{B}, \ell_p, \ell_c)$ is the multiset $\mathcal{H}(S) = \{H(R) : R \in S\}$ of configuration graphs realized by the paths and cycles in S.*

Observe that for a solution S, $|\mathcal{H}(S)|$ can be much larger than θ, and can even be as large as $\mathcal{O}(n)$. So we still need to find a way to express solutions more succinctly.

We define an edge e in a configuration graph H to be **light** if $0 < w_H(e) < 2\theta$, **medium** if $2\theta \leq w_H(e) \leq \theta^3$, and **heavy** if $w_H(e) > \theta^3$. It would be nice if there always exist an optimal solution S whose configuration multiset $\mathcal{H}(S)$ only contained configuration graphs with light and medium edges. There are only few ($\sim f(\theta)$) such configurations graphs in total and we can fairly easily construct an ILP with few variables to find such a solution. Unfortunately there exist instances for which such an optimal solution does not exist. We now develop tools to handle heavy edges.

Definition 5. *A cycle in H is called **non-light** if it does not contain any light edges. Based on the number of non-light cycles in H, we call H a* 0-NL, 1-NL *or* $>_1$-NL *configuration. If H is* 1-NL *then we denote the unique non-light cycle in H by $C(H)$.*

In Definition 5 the number of non-light cycles refers to the number of distinct non-light cycles in H. We do not care whether these cycles overlap or not.

Lemma 2. *Every heavy edge in a path-feasible (cycle-feasible) configuration belongs to a non-light cycle.*

Proof. Let $\mathcal{H}$ be a path-feasible (cycle-feasible) configuration and let $e = (u, v)$ be a heavy edge in A_H. Recall that by Definition 2, $w(e) > \theta^3$. Suppose e does not belong to a non-light cycle. Then each path from v to u contains a light edge, which by definition has weight less than 2θ. Furthermore, there can be at most $\binom{\theta}{2}$ light edges in H. Thus there is an edge cut S from v to u having weight $w(S) < \theta^3$. H contains an Eulerian path (cycle) R because H is path-feasible (cycle-feasible). Then $R \setminus S$ is a disjoint union of $|S| - 1$ walks. But since $w(S) - 1 < \theta^3$ and $w(e) > \theta^3$, one of these walks uses the edge e at least two times. Thus the walk contains a subwalk from v to u disjoint from S. This contradicts that S cuts u from v. □

Lemma 2 immediately implies the following corollary.

Corollary 1. 0-NL *configurations have no heavy edges.*

A cycle C in Q is a *common cycle* of two configurations H and H' of Q if C is a cycle in both H and H'. C is a *common non-light cycle* in H and H' if C is a common cycle of H and H' and it is non-light in both H and H'. The following Lemma helps bound the number of $>_1$-NL configurations.

Lemma 3 ($\star$). *There exists an optimal solution S to $(\mathcal{G}, \mathcal{B}, \ell_p, \ell_c)$ such that for every $R_1, R_2 \in S$, $H(R_1)$ and $H(R_2)$ do not contain more than one common non-light cycle. Furthermore, there are at most $2^{2\theta \log(\theta)}$ $>_1$-NL configurations in $\mathcal{H}(S)$.*

Let S^* be an optimal solution as provided by Lemma 3. We can represent S^* using $\mathcal{H}(S^*)$, the configuration multiset of S^*. However we want to find a way to represent S^* succinctly so that we can design an ILP with few variables that can find it. Configurations in $\mathcal{H}(S^*)$ can be partitioned into 0-NL, 1-NL and $>_1$-NL configurations. We first discuss how we handle 0-NL and $>_1$-NL configurations in $\mathcal{H}(S^*)$.

There are at most $2^{\mathcal{O}(\theta^2 \log \theta)}$ 0-NL configurations — by Definition 2 and Corollary 1, each such configuration H has no heavy edges and has a subgraph of the quotient graph as its underlying graph (V_H, A_H). Let $\mathcal{F}_0$ be the set of all path- or cycle-feasible 0-NL configurations. $|\mathcal{F}_0| \leq 2^{\mathcal{O}(\theta^2 \log \theta)}$ and so we can represent all 0-NL configurations in $\mathcal{H}(S^*)$ succinctly by using a function $h_0 : \mathcal{F}_0 \to Z^*$ where $h_0(H)$ is the number of occurrences of H in $\mathcal{H}(S^*)$. In the ILP we construct, we will have a nonnegative variable to compute this count for each $H \in \mathcal{F}_0$.

Let $\mathcal{F}_{>1}$ be the set of all path- or cycle-feasible $>_1$-NL configurations. There can be many $>_1$-NL configurations since they have heavy edges, so we cannot use the trick we had for 0-NL configurations. Let $O := \{G_H = (V_H, A_H) : H \in \mathcal{H}(S^*) \cap \mathcal{F}_{>1}\}$ be the set of underlying graphs of all $>_1$-NL configurations in $\mathcal{H}(S^*)$. We can enumerate all of O because there are at most $2^{2\theta \log(\theta)}$ $>_1$-NL configurations in $\mathcal{H}(S^*)$ by the definition of S^*. Then for each $G_H = (V_H, A_H) \in O$

and each arc $e \in A_H$, we will have a nonzero variable for the weight $w_H(e)$ of e in the ILP. Furthermore, to check feasibility of the configurations, we will add constraints in the ILP to ensure that the conditions of Lemma 1 hold.

Let $\mathcal{F}_1$ be the set of all path- or cycle-feasible 1-NL configurations and let $\mathcal{F}'_1$ be the set of 1-NL configurations with edge weight at most θ^3. We now handle 1-NL configurations in $\mathcal{H}(S^*)$ which can potentially be many. We obtain a succinct representation for these configurations using $\mathcal{F}'_1$.

Given a nonnegative integer x and a 1-NL configuration $H' \in \mathcal{F}'_1$, we define the x-derived configuration of H' to be the configuration obtained from H' by adding weight x to each edge e on the non-light cycle $C(H')$. We then show that every 1-NL path(cycle)-feasible configuration H is the x-derived configuration of some $H' \in \mathcal{F}'_1$ and some $x \geq 0$ — to prove this we use the Eulerian property of the feasible configuration H (Lemma 1) and the fact that all heavy edges in H lie on the unique non-light cycle $C(H)$ (Lemma 2).

Next we partition all the 1-NL configurations in $\mathcal{H}(S^*)$ into groups $\cup_{H' \in \mathcal{F}'_1} \mathcal{S}_{H'}$ based on which configuration $H' \in \mathcal{F}'_1$ they can be derived from. If a configuration can be derived from many configurations $H' \in \mathcal{F}'_1$, we arbitrarily select one such $H' \in \mathcal{F}'_1$ and add H' to $\mathcal{S}_{H'}$.

Finally we represent each group $\mathcal{S}_{H'}$ in the partition by a triple containing - (i) the configuration $H' \in \mathcal{F}'_1$ that the part corresponds to, (ii) $|\mathcal{S}_{H'}|$ and (iii) the sum $t = \sum_{H \in \mathcal{S}_{H'}} \sum_{x \in \mathbb{N}:\ H \text{ is } x\text{-derived from } H'} x$. In the ILP we will add two variables to find $|\mathcal{S}_{H'}|$ and t for each group $\mathcal{S}_{H'}$. Lastly in the ILP we will also add constraints to ensure the number of vertices of each type used in the solution does not exceed the number of vertices of each type in $\mathcal{G}$.

In total, we solve at most $2^{2^{\mathcal{O}(\theta \log \theta)}}$ instances of our ILP, each having $2^{\mathcal{O}(\theta^2 \log \theta)}$ variables. Using Proposition 1, our algorithm will run in time $2^{2^{\mathcal{O}(\theta^2 \log \theta)}} n^{\mathcal{O}(1)}$. We formalize this discussion and provide our ILP formulation and algorithm in the supplementary material.

4 Faster FPT Algorithm Parameterized by t

In this section, we describe a randomized FPT algorithm for KEP parameterized by t, the number of patients (i.e., non-altruistic vertices) covered. This algorithm runs in time $\mathcal{O}^*(4^t)$, where the $\mathcal{O}^*$ notation suppresses polynomial factors. Since self-loops do not cause any complications in this algorithm, we write this section in a way that allows $\mathcal{G}$ to have self-loops.

If $\ell_p \geq t$ and $\mathcal{G}$ contains a path of length t, then we immediately have a solution to our problem. The same is true if $\ell_c \geq t$ and $\mathcal{G}$ contains a cycle of length ℓ, where $t \leq \ell \leq \ell_c$. Therefore, we begin by checking whether $\mathcal{G}$ contains such a path or cycle. This can be done in randomized time $\mathcal{O}^*(4^k)$ [Williams, 2009; Zehavi, 2016].[4] For the remainder of our algorithm, we update ℓ_p and ℓ_c to the values $\min\{\ell_p, t-1\}$ and $\min\{\ell_c, t-1\}$, respectively, and thus we can assume $\ell_p < t$ and $\ell_c < t$. This first step is similar to [Maiti and Dey, 2022].

In our algorithm, we will work with formal polynomials that have no coefficients. This means we simply think of a polynomial as a string composed of the characters, '+', '×', and the characters representing the variables. If x and y are variables, we abbreviate the string $x \times y$ as xy, and we abbreviate the string $x \times x \times \cdots \times x$, where there are k copies of x, as x^k. We do not plug in values for the variables; each variable is only a symbol. From now on, we will refer to a formal polynomial without coefficients as simply a *polynomial*, since all of our polynomials will be of that form.

Definition 6. *Suppose $P(x_1, \ldots, x_n)$ is a polynomial. A* multilinear *term in P is a term (monomial) that is a product of distinct variables.*

For example, in the polynomial $P(x_1, x_2) = x_1x_2 + x_1^2x_2$, the term x_1x_2 is multilinear, but $x_1^2x_2$ is not. It is also important to note that in our formal polynomials, addition is commutative, but multiplication is not. For instance, $x_1 + x_2 = x_2 + x_1$, but $x_1x_2 \neq x_2x_1$.

The key idea of our algorithm is as follows: We construct a polynomial P_t that contains a multilinear term if and only if there is a solution covering at least t patients, and then we check whether P_t contains a multilinear term using the following result (Theorem 3.1 in [Williams, 2009]).

Proposition 2. *There is a randomized algorithm that takes as input a polynomial $P(x_1, \ldots, x_n)$ of degree at most k, represented by an arithmetic circuit of size $s(n)$ with $+$ gates (of unbounded fan-in), $\times$ gates (of fan-in two), and no scalar multiplications. This algorithm outputs* yes *with high probability (at least $\frac{1}{2}$) if there is a multilinear term in the sum-product expansion of P, and always outputs* no *if there is no multilinear term. The algorithm runs in time $\mathcal{O}^*(2^k s(n))$.*

Upon reading the proof of Proposition 2, it becomes apparent that this algorithm only distinguishes between the following two cases:

1. There is no multilinear term in $P(x_1, \ldots, x_n)$,
2. There exists a multilinear term that appears *exactly once* in $P(x_1, \ldots, x_n)$.

This is usually not an issue because for many problems, the polynomial that one would naturally construct has no repeating terms (for example, in the case of k-path [Williams, 2009]). However, for KEP one must be slightly more careful, since in this case a solution can consist of multiple paths and cycles, and thus there can be distinct solutions that correspond to the same list of vertices, in the same order. Therefore, we will take care to construct a polynomial that has no repeating terms. To apply Proposition 2, we describe a polynomial P_t of degree at most $2t$ that can be represented by a circuit of size $\mathcal{O}(t^4n^3)$. Once we have constructed such a polynomial, Proposition 2 immediately yields the following theorem. Recall that a solution to KEP is a set of vertex-disjoint feasible paths and feasible cycles.

[4] The algorithm in [Zehavi, 2016] is designed to detect a cycle of length at least t. However, with the following very simple modification, we can use that algorithm to detect a cycle of length ℓ, where $t \leq \ell \leq \ell_c$: In [Zehavi, 2016], line 6 of Algorithm 1, which says "if the path P' exists then Accept," should be modified to "if the path P' exists and is of length at most $\ell_c - k$ then Accept."

Theorem 2. *There is a randomized algorithm that, given an instance of KEP and a positive integer t, outputs* yes *with high probability (at least $\frac{1}{2}$) if there is a solution covering at least t patients, and always outputs* no *if there is no such solution. This algorithm runs in time $\mathcal{O}^*(4^t)$.*

Fix an instance $(\mathcal{G}, \mathcal{B}, \ell_p, \ell_c)$ of KEP. To define P_t, suppose the vertex set of $\mathcal{G}$ is $\{1, \ldots, n\}$, and let $x_1, \ldots, x_n$ be variables corresponding to the vertices of $\mathcal{G}$. We also define t additional variables $y_1, \ldots, y_t$. We call a walk in $\mathcal{G}$ an *altruistic walk* if its first vertex belongs to $\mathcal{B}$ and it has length (number of edges) between 1 and ℓ_p, inclusive. Suppose $\mathcal{D}$ is an ordered collection of (possibly overlapping) altruistic walks and closed walks in $\mathcal{G}$, in which each closed walk has length at most ℓ_c. We call such a collection $\mathcal{D}$ a *donation plan* if it contains at most t cycles. Note that a donation plan is not required to be feasible — the same vertex might be covered multiple times in the same walk or across different walks. In the definition of $\mathcal{D}$, we only consider collections with at most t cycles because we know that if there exists a feasible solution, then there exists a feasible solution with at most t cycles.

For each altruistic walk $W_1 = i_1, i_2 \ldots, i_\ell$ in a donation plan $\mathcal{D}$, we define $\phi(W_1) = x_{i_1} x_{i_2} \cdots x_{i_\ell}$, and for each closed walk $W_2 = j_1, j_2, \ldots, j_{\ell'}, j_1$ in $\mathcal{D}$, we define $\phi(W_2) = y_k x_{j_1} x_{j_2} \cdots x_{j_{\ell'}}$, where k is such that W_2 is the k^{th} closed walk in $\mathcal{D}$. The *degree* of a donation plan $\mathcal{D}$ is the degree of the term $\prod_{W \in \mathcal{D}} \phi(W)$.

For a walk W in a donation plan, let $p(W)$ denote the number of patients covered by W, where patients that are covered multiple times along the walk are counted multiple times. Note that this is equal to the edge-length of W. For $\ell, t' \geq 1$, let $\Pi_\ell^{t'}$ be the collection of all donation plans $\mathcal{D}$ such that $\sum_{W \in \mathcal{D}} p(W) = t'$, and such that the degree of $\mathcal{D}$ is exactly ℓ. We define the polynomial

$$P_t(x_1, \ldots, x_n, y_1, \ldots, y_t) = \sum_{t'=t}^{2t} \sum_{\ell=2}^{2t} \sum_{\mathcal{D} \in \Pi_\ell^{t'}} \prod_{W \in \mathcal{D}} \phi(W).$$

In the product over $W \in \mathcal{D}$, we multiply the terms $\phi(W)$ in order, according to the ordering of the walks in $\mathcal{D}$.

First, to ensure that we can apply Proposition 2, we observe that distinct donation plans give rise to distinct terms in P_t:

Observation 1 ($\star$). *If $\prod_{W \in \mathcal{D}_1} \phi(W) = \prod_{W \in \mathcal{D}_2} \phi(W)$ for donation plans $\mathcal{D}_1, \mathcal{D}_2$, then $\mathcal{D}_1 = \mathcal{D}_2$.*

As mentioned above, proofs of results marked with a star can be found in the supplementary material. Now, we need to show that P_t contains a multilinear term if and only if there is a solution covering at least t patients:

Lemma 4 ($\star$). *There is a solution to $(\mathcal{G}, \mathcal{B}, \ell_p, \ell_c)$ covering at least t patients if and only if P_t contains a multilinear term.*

The following lemma (Lemma 5) is a key step in the proof of Lemma 4. For the statement of Lemma 5, we need one more definition: Let S be a solution to the given instance of KEP. The *degree* of S is defined as $\sum_{W \in S}(p(W) + 1)$. This is equal to the degree of the monomial $\prod_{W \in \mathcal{D}} \phi(W)$, where $\mathcal{D}$ is any ordering of S.

Lemma 5 ($\star$). *If there exists a solution to the instance $(\mathcal{G}, \mathcal{B}, \ell_p, \ell_c)$ that covers at least t patients, then there exists a solution S with the following properties: the number of patients covered by S is at least t and at most $2t$, and the degree of S is at most $2t$.*

Finally, to properly apply Proposition 2, we must be able to build a circuit that computes P_t. This can be done in a similar way to the algorithm for k-path [Williams, 2009]. The details can be found in the supplementary material.

5 W[1]-Hardness of KEP Parameterized by ω

In this section, we will show that KEP does not admit an algorithm with running time $f(\omega)n^{\mathcal{O}(1)}$, where ω is the treewidth of the compatibility graph, unless FPT = W[1]. We prove this by showing that KEP is W[1]-hard parameterized by ω.

Theorem 3. *KEP is* W[1]*-hard. Unless* FPT = W[1]*, KEP does not admit an algorithm with running time $f(\omega)n^{\mathcal{O}(1)}$.*

To prove Theorem 3, we give a reduction from the following W[1]-hard variant of the Multicolored Clique problem. [5]

Dense k-Multicolored Clique **Parameter:** k
Input: A positive integer $k > 3$, a k-partite graph $G = (V = V_1 \cup V_2 \cup \cdots \cup V_k, E)$, and a graph $G_{aux} = (V_{aux}, E_{aux})$ constructed from G with the following properties:
(i) In G, $|V_i| = n$ for all $i \in [k]$, where $n \in \mathbb{N}$, **(ii)** for all i, $G[V_i \cup V_j]$ is not a complete bipartite graph for at most three $j \neq i$, **(iii)** $V_{aux} = \{v_1, \ldots, v_k\}$, where v_i for $i \in [k]$ corresponds to the vertex set V_i in G, **(iv)** for $i, j \in [k]$ with $i \neq j$, $(v_i, v_j) \in E_{aux}$ if and only if $G[V_i \cup V_j]$ is not a complete bipartite graph, and **(v)** G_{aux} is triangle free.
Question: Does there exist a k-clique $S \subseteq V$ with $|S \cap V_i| = 1$ for all i?

Let $(G = (V_1 \cup V_2 \cup \cdots \cup V_k, E), k)$ be an instance of Dense k-Multicolored Clique. We refer to each partition V_i of G as a color class of G. Let u_i^j be the j^{th} vertex in V_i for $1 \leq j \leq n$ and $1 \leq i \leq k$. We choose an arbitrary ordering of the vertices and edges of G, G_{aux}.

We observe that the maximum degree of a vertex in G_{aux} is 3; therefore, using Vizing's Theorem [Lewis, 2015] we can obtain a proper vertex coloring $\chi_V : V(G_{aux}) \to \{1, 2, 3, 4\}$ and a proper edge coloring $\chi_E : E(G_{aux}) \to \{5, 10, 15, 20\}$ in polynomial time. For a vertex $v_i \in V_{aux}$, we denote the set of edges incident to v_i in G_{aux} by $E(v_i)$.

We now give the construction of a $(0, n^{25})$-KEP instance $(\mathcal{G}, \emptyset, 0, n^{25})$, where $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, using G, G_{aux}, χ_V, and χ_E. We will first construct an edge-weighted directed graph G_{kep}. We will then construct $\mathcal{G}$ from G_{kep} by replacing every edge e with weight w_e from u to v in G_{kep} by a new path from u to v having $w_e - 1$ internal vertices. Our construction has the following two types of gadgets:

[5] A simple reduction from the W[1]-hard problem Partitioned Subgraph Isomorphism (PSI) [Lokshtanov *et al.*, 2020, Proposition 3.1] [Marx, 2007, Corollary 6.3] shows W[1]-hardness for our variant of MCC. See the supplementary material for further details. Also, we define MCC as above to get a cleaner reduction.

(i) Edge gadgets (X_e): For each edge $e \in E_{aux}$, we add an edge gadget X_e to G_{kep}.
— *Vertices of X_e:* Suppose $e = (v_i, v_j)$. Let $m = |E(G[V_i, V_j])| + 1$. We construct two ordered sets of vertices X_e^i and X_e^j, each of size m. We set $V(X_e) = X_e^i \cup X_e^j$.

Let $x_{e,\ell}^i$ and $x_{e,\ell}^j$ denote the ℓ^{th} vertex in X_e^i and X_e^j, respectively. For $1 \leq \ell < m$, the pair $(x_{e,\ell}^i, x_{e,\ell}^j)$ corresponds to the ℓ^{th} edge in $E(G[V_i, V_j])$. We now define two mappings $f_e^i \colon [m-1] \to V_i$ and $f_e^j \colon [m-1] \to V_j$. If $(u_i \in V_i, u_j \in V_j)$ is the ℓ^{th} edge in $E(G[V_i, V_j])$, then $f_e^i(\ell) := u_i$ and $f_e^j(\ell) := u_j$. These mappings will later be used to assign weights to the edges of G_{kep}.
— *Edges of X_e:* For each $1 \leq \ell < m$, we add the following four types of directed edges:
(a) $\left(x_{e,\ell}^i, x_{e,\ell+1}^i\right)$ and (b) $\left(x_{e,\ell}^j, x_{e,\ell+1}^j\right)$ both with weight 1,
(c) $\left(x_{e,\ell}^i, x_{e,\ell+1}^j\right)$ with weight $n^{\chi_V(i)\chi_E(e)} + \ell \cdot n^{\chi_E(e)}$,
(d) $\left(x_{e,\ell}^j, x_{e,\ell+1}^i\right)$ with weight $n^{\chi_V(j)\chi_E(e)} + \ell \cdot n^{\chi_E(e)}$.

We call the edges of type (c) and (d) *crossing* edges.

(ii) Vertex gadgets (Y_i): For each vertex $v_i \in V_{aux}$, we add a vertex gadget Y_i to G_{kep}.
— *$V(Y_i)$:* Let $V(Y_i) = \{start_i(s_i), y_1^i, \ldots, y_n^i, end_i(e_i)\}$, where each y_ℓ^i for $1 \leq \ell \leq n$ corresponds to the ℓ^{th} vertex $u_\ell \in V_i$. Note that $|V(Y_i)| = n + 2$.
— *Edges of Y_i:* Recall that $E(v_i) \subseteq E_{aux}$ is the set of edges incident to v_i. We add the following edges to Y_i:
(a) $\left(y_\ell^i, start_i\right)$ with weight $n^{25} - sum_\ell$ for each $1 \leq \ell \leq n$, where $sum_\ell = \Sigma_{e \in E(v_i)} m_e + (n^{\chi_V(v_i)} + \ell)(\Sigma_{e \in E(v_i)} n^{\chi_E(e)}) + n - \ell + 2$, and where m_e is the number of vertices in set X_e^i. Recall that m_e is one plus the number of edges between the color classes V_i, V_j.
(b) $\left(y_\ell^i, y_{\ell-1}^i\right)$ with weight 1 for each $2 \leq \ell \leq n$.
(c) $\left(end_i, y_n^i\right)$ with weight 1.

Linking the edge and vertex gadgets: We now link our gadgets together to complete the construction of G_{kep}. For each $v_i \in V_{aux}$, we consider the following three cases according to the cardinality of $E(v_i)$:

1. $|E(v_i)| = 1$. Write $E(v_i) = \{e_1\}$. We add the following edges to G_{kep}:
   $(start_i, x_{e_1,1}^i)$ and $(x_{e_1,m_e}^{j_1}, end_i)$, both with weight 1.
2. $|E(v_i)| = 2$. Write $E(v_i) = \{e_1, e_2\}$, and let $e_1 < e_2$ be the ordering of the edges. We add the following edges to G_{kep}:
   (i) $(start_i, x_{e_1,1}^i)$ and $(x_{e_2,m_{e_2}}^{j_2}, end_i)$, both with weight 1. (ii) $(x_{e_1,m_{e_1}}^{j_1}, x_{e_2,1}^i)$ with weight 1.
3. $|E(v_i)| = 3$. Write $E(v_i) = \{e_1, e_2, e_3\}$, and suppose $e_1 < e_2 < e_3$ is the ordering of those edges. We add the following edges to G_{kep}: (i) $(start_i, x_{e_1,1}^i)$ and $(x_{e_3,m_{e_3}}^{j_3}, end_i)$, and (ii) $(x_{e_1,m_{e_1}}^{j_1}, x_{e_2,1}^i)$ and $(x_{e_2,m_{e_2}}^{j_2}, x_{e_3,1}^i)$, all with weight 1.

Since G_{aux} has k vertices each of degree at most 3, G_{kep} has k vertex gadgets and at most $3k/2$ edge gadgets. Recall that $\mathcal{G}$ is constructed by replacing each edge in G_{kep} with a path. Since replacing an edge with a path does not increase the treewidth of the graph, to show that the treewidth of $\mathcal{G}$ is $\mathcal{O}(k)$, it suffices to show that the treewidth of G_{kep} is $\mathcal{O}(k)$.

Observation 2 ($\star$). *The treewidth of G_{kep} is $\mathcal{O}(k)$.*

There exists a k-MCC in G if and only if there is a solution to the constructed $(0, n^{25})$-KEP instance $(\mathcal{G}, \emptyset, 0, n^{25})$ which covers at least kn^{25} patients. Due to limited space, *we defer the formal proof of equivalence to the supplementary material*. We will now describe the structure of cycles in the solution and give a high-level idea of our proof. We say a cycle *intersects* an edge gadget X_e (resp. a vertex gadget Y_i) if it has nonempty intersection with $V(X_e)$ (resp. $V(Y_i)$). Consider X_e, where $e = (v_i, v_j) \in E_{aux}$. Suppose $(u_i, u_j) \in E(G[V_i \bigcup V_j])$, where $u_i \in V_i$ and $u_j \in V_j$ is the ℓ^{th} edge in X_e. We say a cycle intersecting X_e *selects the edge* (u_i, u_j) if it contains one of the two crossing edges $(x_{e,\ell}^i, x_{e,\ell+1}^j)$ or $(x_{e,\ell}^j, x_{e,\ell+1}^i)$, and no other crossing edge in X_e. A cycle intersecting a vertex gadget Y_i must follow a path of the form $end_i, y_n^i, \ldots, y_\ell^i, start_i$ for some $n \geq \ell \geq 1$ since the only way to enter and exit a vertex gadget is through end_i and $start_i$, respectively. We say such a cycle *selects the vertex* u_ℓ (corresponding to y_ℓ), where $u_\ell \in V_i$. In our proof, we show that *(i)* each cycle in the solution for $(\mathcal{G}, \emptyset, 0, n^{25})$ intersects one vertex gadget and at most three edge gadgets, *(ii)* if the vertex gadget corresponds to $v_i \in V_{aux}$, then the three edge gadgets correspond to the edges incident to v_i in G_{aux}, and *(iii)* all of the edges selected across the three edge gadgets are incident to the vertex selected in the vertex gadget. Finally, we show that the k vertices selected by the k cycles in the solution form a clique in G.

6 Discussion and Open Problems

In this work, we considered KEP, a natural problem of matching kidney patients to donors. We made significant progress in understanding its parameterized complexity. Our randomized algorithm from Theorem 2 can be derandomized to run in time 14.34^t using a deterministic version of Proposition 2 [Fomin *et al.*, 2014, Theorem 5.1]. Furthermore, we believe our algorithms can be used for designing local-search-style heuristics. For instance, algebraic algorithms for MOTIF FINDING led to a highly parallelized implementation for this problem on GPUs [Kaski *et al.*, 2018].

In Section 5, we show that KEP is W[1]-hard parameterized by ω. Lampis [Lampis, 2014] introduced a technique to design approximation schemes running in $f(\omega)n^{\mathcal{O}(1)}$ time for problems that are W[1]-hard with respect to ω. This technique uses tools from approximation algorithms over tree decompositions. We remark that a direct application of this technique would yield a bicriteria approximation scheme for t and $max(\ell_p, \ell_c)$ for KEP running in time $f(\omega)n^{\mathcal{O}(1)}$. Approximating just the solution size t within a similar running time would also be interesting. Another nice open problem that stems from our work would be to ask whether there exists a single-exponential FPT algorithm parameterized by the number of vertex types.

Contribution Statement

All authors contributed equally to this work. Their names are ordered alphabetically by last name.

References

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

[Akbarpour *et al.*, 2014] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. Dynamic matching market design. *arXiv preprint arXiv:1402.3643*, 2014.

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

[Ashlagi and Roth, 2014] Itai Ashlagi and Alvin E Roth. Free riding and participation in large scale, multi-hospital kidney exchange. *Theoretical Economics*, 9(3):817–863, 2014.

[Ashlagi *et al.*, 2012] Itai Ashlagi, David Gamarnik, Michael A Rees, and Alvin E Roth. The need for (long) chains in kidney exchange. Technical report, National Bureau of Economic Research, 2012.

[Barnhart *et al.*, 1998] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[Biro *et al.*, 2009] Péter Biro, David F Manlove, and Romeo Rizzi. Maximum weight cycle packing in directed graphs, with application to kidney exchange programs. *Discrete Mathematics, Algorithms and Applications*, 1(04):499–517, 2009.

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

[Cygan *et al.*, 2015] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.

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

[Dickerson *et al.*, 2017] John P. Dickerson, Aleksandr M. Kazachkov, Ariel D. Procaccia, and Tuomas Sandholm. Small representations of big kidney exchange graphs. In *The Workshops of the The Thirty-First AAAI Conference on Artificial Intelligence, Saturday, February 4-9, 2017, San Francisco, California, USA*, volume WS-17 of *AAAI Technical Report*. AAAI Press, 2017.

[Fomin *et al.*, 2014] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative sets of product families, 2014.

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

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

[Kaski *et al.*, 2018] Petteri Kaski, Juho Lauri, and Suhas Thejaswi. Engineering motif search for large motifs. In Gianlorenzo D'Angelo, editor, *17th International Symposium on Experimental Algorithms, SEA 2018, June 27-29, 2018, L'Aquila, Italy*, volume 103 of *LIPIcs*, pages 28:1–28:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Kovesdy, 2022] Csaba P Kovesdy. Epidemiology of chronic kidney disease: An update 2022. *Kidney International Supplements*, 12(1):7–11, 2022.

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

[Lampis, 2012] Michael Lampis. Algorithmic metatheorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[Lampis, 2014] Michael Lampis. Parameterized approximation schemes using graph widths. In *Automata, Languages, and Programming: 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I 41*, pages 775–786. Springer, 2014.

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

[Lewis, 2015] Rhyd Lewis. *A guide to graph colouring*, volume 7. Springer, 2015.

[Lokshtanov *et al.*, 2020] Daniel Lokshtanov, MS Ramanujan, Saket Saurab, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2181–2200. SIAM, 2020.

[Maiti and Dey, 2022] Arnab Maiti and Palash Dey. Parameterized algorithms for kidney exchange. In Lud De

Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 405–411. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

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

[Manlove and O'Malley, 2015] David F Manlove and Gregg O'Malley. Paired and altruistic kidney donation in the uk: Algorithms and experimentation. *Journal of Experimental Algorithmics (JEA)*, 19:1–21, 2015.

[Marx, 2007] Dániel Marx. Can you beat treewidth? In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 169–179. IEEE, 2007.

[Pahwa *et al.*, 2012] Mrinal Pahwa, Yusuf Saifee, Vipin Tyagi, Sudhir Chadha, and Harsh Jauhari. Paired exchange kidney donation in india: A five-year single-center experience. *International urology and nephrology*, 44(4):1101–1105, 2012.

[Plaut *et al.*, 2016] Benjamin Plaut, John P Dickerson, and Tuomas Sandholm. Fast optimal clearing of capped-chain barter exchanges. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[Rapaport, 1986] Felix T Rapaport. The case for a living emotionally related international kidney donor exchange registry. In *Transplantation proceedings*, volume 18, pages 5–9, 1986.

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

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

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

[Saran *et al.*, 2020] Rajiv Saran, Bruce Robinson, Kevin C Abbott, Jennifer Bragg-Gresham, Xiaoying Chen, Debbie Gipson, Haoyu Gu, Richard A Hirth, David Hutton, Yan Jin, et al. US renal data system 2019 annual data report: Epidemiology of kidney disease in the United States. *American Journal of Kidney Diseases*, 75(1):A6–A7, 2020.

[Segev *et al.*, 2005] 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.

[Stewart *et al.*, 2023] Darren Stewart, Tatenda Mupfudze, and David Klassen. Does anybody really know what (the kidney median waiting) time is? *American Journal of Transplantation*, 23(2):223–231, 2023.

[Walsh, 2021] Dylan Walsh. A beautiful application: Using economics to make kidney exchanges more efficient and fair. *Stanford Graduate School of Business*, 2021.

[Williams, 2009] Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Information Processing Letters*, 109(6):315–318, 2009.

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

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