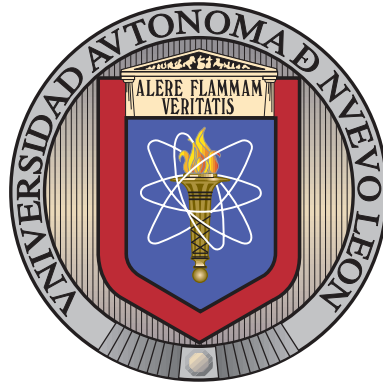


UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



FORMULATIONS AND ALGORITHMS FOR THE  
KIDNEY EXCHANGE PROBLEM

POR

LIZETH CAROLINA RIASCOS ALVAREZ

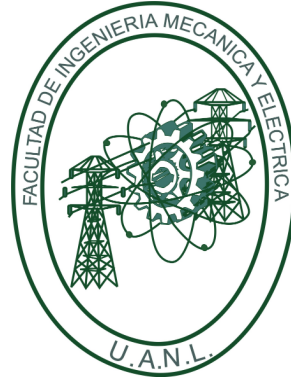
COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE  
MAESTRO EN CIENCIAS EN INGENIERÍA DE SISTEMAS

ABRIL 2017

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



FORMULATIONS AND ALGORITHMS FOR THE  
KIDNEY EXCHANGE PROBLEM

POR

LIZETH CAROLINA RIASCOS ALVAREZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE  
MAESTRO EN CIENCIAS EN INGENIERÍA DE SISTEMAS

ABRIL 2017

**Universidad Autónoma de Nuevo León**  
**Facultad de Ingeniería Mecánica y Eléctrica**  
**Subdirección de Estudios de Posgrado**

Los miembros del Comité de Tesis recomendamos que la Tesis “Formulations and Algorithms for the Kidney Exchange Problem”, realizada por la alumna Lizeth Carolina Riascos Alvarez, con número de matrícula 1770366, sea aceptada para su defensa como requisito parcial para obtener el grado de Maestro en Ciencias en Ingeniería de Sistemas.

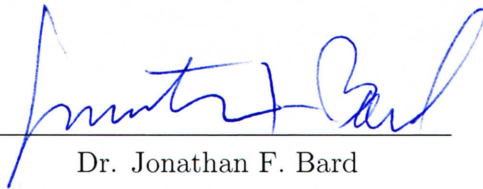
El Comité de Tesis



---

Dr. Roger Z. Ríos Mercado

Director



---

Dr. Jonathan F. Bard

Co-director

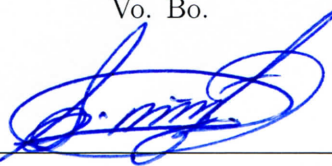


---

Dra. Ada Margarita Álvarez Socarrás

Revisor

Vo. Bo.



---

Dr. Simón Martínez Martínez

Subdirector de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, abril 2017

A

«DMT»

*una mente brillante*

*mi bonito amor*



# CONTENTS

---

<b>Acknowledgments</b>	<b>viii</b>
<b>Abstract</b>	<b>x</b>
<b>List of Figures</b>	<b>xii</b>
<b>List of Tables</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	5
1.2 Background . . . . .	7
1.3 Motivation . . . . .	9
1.4 Objectives . . . . .	10
1.5 Organization . . . . .	11
<b>2 The cycle packing variant of the Kidney Exchange Problem</b>	<b>12</b>
2.1 Problem Statement . . . . .	12
2.2 Integer Programming Formulations . . . . .	14

---

2.2.1	Cycle Formulation . . . . .	14
2.2.2	Edge Formulation . . . . .	15
2.2.3	Extended Edge Formulation . . . . .	17
2.2.4	Partitioned Edge Formulation . . . . .	18
2.2.5	Partitioned and Reduced Edge Formulation . . . . .	21
<b>3</b>	<b>Chains and Cycles variant of the Kidney Exchange Problem</b>	<b>29</b>
3.1	Problem statement . . . . .	29
3.2	Integer Programming Formulations . . . . .	30
3.2.1	Anderson’s Arc-based Formulation . . . . .	32
3.2.2	PC-TSP-based Formulation . . . . .	33
3.2.3	The Polynomial-sized SPLIT Formulation . . . . .	34
3.2.4	The Exponential-sized SPLIT Formulation . . . . .	36
3.2.5	The Reduced Exponential-sized SPLIT Formulation . . . . .	37
<b>4</b>	<b>Computational Experiments</b>	<b>39</b>
4.1	Description of Database Instances . . . . .	40
4.2	Experimental Conditions . . . . .	41
4.3	Comparing Search Algorithms to find length- $k$ paths in the KEP . . .	42
4.4	Selecting the best node-selection strategy for the SCC-Based Search Algorithm . . . . .	44
4.5	Assessment of KEP Formulations . . . . .	46

---

<b>5</b>	<b>Conclusions</b>	<b>62</b>
5.1	Summary of Research Contributions . . . . .	64
5.2	Future work . . . . .	65
	<b>Appendices</b>	<b>67</b>
<b>A</b>	<b>The Separation Problem</b>	<b>68</b>
<b>B</b>	<b>Description of Data Sets</b>	<b>71</b>
<b>C</b>	<b>Reduced instances for the Cycle Variant KEP Formulations</b>	<b>77</b>
<b>D</b>	<b>Compatibility Evaluation</b>	<b>81</b>
D.1	Blood Typing (ABO Compatibility) . . . . .	81
D.2	HLA Typing . . . . .	82
D.3	Cross-matching . . . . .	84

# ACKNOWLEDGMENTS

---

Gracias México, cuyo apoyo he visto reflejado en todas las personas e instituciones que han contribuido a mi exitoso fin de este programa de posgrado y a la investigación que he desarrollado en esta tesis.

A mi director el Profesor Roger Z. Ríos Mercado, gracias por su tiempo, guía y disponibilidad constante para el desarrollo y culminación de esta investigación. A mi co-director, el Profesor Jonathan Bard por el interés, tiempo y herramientas brindadas para fortalecer esta investigación durante mi estancia de investigación en la Universidad de Texas y a mi Revisora la Profesora Ada Álvarez, para quien me harían falta palabras que describan las múltiples maneras en que me ha apoyado a lo largo de estos dos años y poco más, incluso sin darse cuenta. Sus cursos han contribuido a este trabajo y sus palabras me han servido de motivación.

Agradezco también al conjunto de profesores del posgrado, en especial a la Profesora Yasmín Ríos, al profesor Igor Litvinchev, al profesor César Villarreal y al profesor Arturo Berrones, quienes a través de sus cursos contribuyeron a mi aprendizaje en áreas que despertaron mi interés. Gracias, a mis compañeros de generación, a mis compañeras y compañeros de otras generaciones, algunos ya ausentes, con quienes compartí momentos de estudio y de esparcimiento.

Agradezco también a los doctores Ross Anderson, de Google, y Vicky Mak-Hau, de Deakin University, quienes generosamente han compartido sus instancias de prueba, sin las cuales los experimentos llevados a cabo en esta tesis no habrían sido posibles.

A CONACyT, quien me ha otorgado dos becas, una de sostenimiento para el programa nacional y otra para llevar a cabo una estancia en el exterior, mi más sincero agradecimiento. Velaré por honrar los recursos que me han otorgado.

De igual manera agradezco a la Universidad Autónoma de Nuevo León y a la Facultad de Ingeniería Mecánica y Eléctrica por su apoyo económico en la forma de exención de pago de inscripción y colegiatura. A University of Texas at Austin por el acceso a sus instalaciones y equipos. A la Universidad Nacional de Colombia, por proveerme acceso a su base de datos de archivos digitales, desde donde pude consultar diversos textos científicos que contribuyeron a esta investigación.

Agradezco el apoyo de mis familiares, especialmente el de mi padre, quien me ha acompañado en esta travesía en la distancia y ha sabido aguardar por verme nuevamente cada vez que ha sido posible. A Lunita, mi mascota y compañera durante doce años fallecida mientras he estado lejos, gracias por hacer de los que fueron mis días de adolescencia y adultez, días de inolvidables alegrías.

*Finalmente, quiero agradecerte «DMT». Si hay alguien en el mundo que haya luchado junto a mí tantas batallas y permanecido conmigo para volver a intentarlo has sido tú. Has hecho mis sueños tuyos y procurado que nunca le falte una sonrisa a mis días. Para estos, precisamente, has llegado a rango “Leyenda”, aunque no pude estar atenta por causa de este trabajo, quiero que sepas que te guardo admiración hasta en los “pequeños” logros.*

# ABSTRACT

---

Lizeth Carolina Riascos Alvarez.

Candidato para obtener el grado de Maestro en Ciencias en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título de la tesis: FORMULATIONS AND ALGORITHMS FOR THE KIDNEY EXCHANGE PROBLEM.

Número de páginas: 90.

Important advances in healthcare management by means of Operations Research techniques have been achieved over the past few years. One area in particular is in helping patients in need of a kidney reduce their usually long waiting times. One way to do this is through a kidney exchange program. If a patient needing a kidney brings along a person (a relative or friend) willing to donate one of her/his kidneys and if they both are clinically compatible then the donation can be done immediately by mutual agreement. However, if this patient-donor pair (PDP) is not compatible, an exchange with another PDP could take place. This could happen for instance when the donor of one pair is compatible with the patient of another pair and vice-versa. If such a pair is found, then they can agree to have a simultaneous donation, have their kidney exchange surgeries relatively faster, avoiding the waiting list. In some other cases, altruist donors, those without requiring a kidney in return,

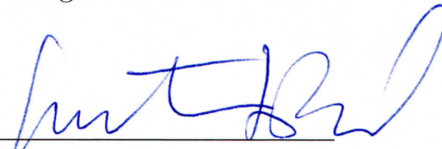
may trigger a sequence of donations involving a series of incompatible PDPs. Now, given a pool of PDPs with their compatibility information among pairs, finding the largest amount of matches implies finding the largest amount of people that could have their needed kidney surgeries in a timely fashion, rather than going to the waiting list. Formally, the kidney exchange problem is defined as finding the largest amount of matches (either in form of cycles or chains) among a pool of PDPs. In this thesis, we address the KEP by presenting and comparing a variety of existing models and algorithms that cover almost all the kidney exchange variants known so far. In particular, for one of the fundamental KEP formulations encountered in the literature, a reduction method for substantially decreasing the size of the problem is proposed. Moreover, we will set a natural partition on the directed graph based on graph theory concepts in order to model reduced-size instances, while keeping optimality. The proposed approach turns out to be very effective to solve the KEP in terms of quality of solution and computational effort, in many cases outperforming other KEP formulations.

Firma del Director: \_\_\_\_\_



Dr. Roger Z. Ríos Mercado

Firma del Co-director: \_\_\_\_\_



Dr. Jonathan F. Bard

# LIST OF FIGURES

---

1.1	Two-way cycle: This diagram illustrates a two-way cyclic exchange between two blood-type-incompatible recipient-donor pairs. . . . .	2
1.2	Length-3 chain: This diagram illustrates a three-transplant chain involving one non-directed donor and three blood-type-incompatible recipient-donor pairs. The last donor -Donor 3- will continue the chain. . . . .	3
1.3	NEAD: In the first period, a chain of length 3 is found and is divided into two segments. Donor 2 becomes a short-term bridge donor and Donor 3 becomes a (regular) bridge donor. In period 2, a chain beginning with Donor 3 is found. . . . .	4
1.4	An example of a KEP instance and its possible solutions. (a) Original compatibility graph; (b) Cycle-and-chain solution; (c) chain-only solution; (d) cycle-only solution. . . . .	6
2.1	Cycle packing variant: Feasible solution with $ P  = 8$ , $w_{ij} = 1$ and $k = 3$ . . .	13
2.2	Sample graph: Finding length-3 paths. . . . .	17
2.3	SCCs in $G$ : Each shaded region is a strongly connected component of $G$ . Each vertex belongs to exactly one SCC. Vertex $h$ forms a trivial SCC. . .	20
2.4	Removing sequentially a vertex from a strongly connected component. . .	23



---

3.1	Example of a KEP instance in presence of NDDs. . . . .	30
3.2	Example of cut set constraints for the PC-TSP model. . . . .	34
4.1	Performance of Algorithm 1. . . . .	43
4.2	Assessment of node selection strategy in the SCC-Based Search Algorithm. . . . .	45
4.3	Comparison of number of matches obtained from considering only cycles and considering both cycles and chains. . . . .	59
4.4	Solution composition for the chain-and-cycle variant of the KEP. . . . .	60

# LIST OF TABLES

---

2.1	Full set of length-3 paths for Figure 2.1 . . . . .	24
4.1	Notation of KEP formulations. . . . .	46
4.2	Experimental results for 3-way cyclic exchanges for data set DA1. . .	49
4.3	Experimental results for 3-way cyclic exchanges for data set DM. . . .	49
4.4	Experimental results for 3-way cyclic exchanges for data set DA1 $\cup$ DA2. . . . .	50
4.5	Size of formulations and savings on 3-path constraints for data set DM.	52
4.6	Size of formulations and savings on 3-path constraints for set DA1 $\cup$ DA2. . . . .	52
4.7	Assessment of eSPLIT and ReSPLIT formulations. . . . .	54
4.8	Comparison of formulations for 3-way cyclic exchanges and unbounded chains on data set DA. . . . .	55
4.9	Comparison of formulations for 3-way cyclic exchanges and unbounded chains on set DM. . . . .	56
4.10	Comparison of formulations for 3-way cyclic exchanges and unbounded chains on data set DA1 $\cup$ DA2. . . . .	57

---

4.11	Size of formulation AA for DM instances. . . . .	58
4.12	Size of AA formulation for $DA1 \cup DA2$ instances. . . . .	59
4.13	Comparison of solutions under different policies for cycles and/or chains allowed in the KEP. . . . .	61
B.1	Full set of original instances. . . . .	74
B.2	Codification of instances into sets PDPsR . . . . .	76
C.1	Instances for the cycle variant: Reduction on the input size. . . . .	80
D.1	Blood type compatibility chart: O is the universal donor and AB is the universal recipient. . . . .	82

## CHAPTER 1

# INTRODUCTION

---

Typically, a patient receives a kidney transplant from a deceased donor, or directly from a living donor who is frequently a relative. Unfortunately, deceased donors are scarce and patient–donor incompatibilities may occur, adding thousands of patients every year to the waiting lists around the world. In Mexico, 60% of people in need of transplant are renal disease sufferers and their waiting time for a deceased donor is up to almost 3 years [40]. As a small fraction of the demand is satisfied, some countries have adopted kidney exchange programs to increase the number of living-donor transplants by bringing together incompatible donors and recipients and conducting exchanges so that each recipient receives a compatible kidney [3, 29]. This is done in two ways. The first is when a living donor, who is incompatible with the intended recipient, donates a kidney to another patient as long as the donor’s recipient receives a compatible kidney from another donor (see Figure 1.1). Such exchanges are known as two-way, three-way, ...,  $k$ -way cyclic exchanges, depending on the number of incompatible patient-donor pairs (PDPs) involved in the cycle. The surgeries in cyclic exchanges are conducted simultaneously because, in a cycle, every patient-donor pair both gives a kidney and receives one, and so the cost of a broken link would be very high to a pair that first donated a kidney and later did not receive one in return. This simultaneity requirement increases substantially the operating rooms and surgical teams:  $2k$  in each case per every  $k$ -way cyclic exchange, i.e.

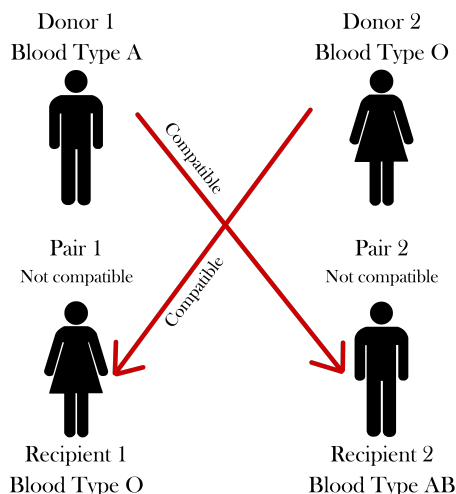


Figure 1.1: Two-way cycle: This diagram illustrates a two-way cyclic exchange between two blood-type-incompatible recipient-donor pairs.

3-way cycle involves the simultaneous coordination of 6 operating rooms and surgical teams. For these reason, cyclic exchanges with more than three patient-donor pairs are rarely conducted [3]. The second is when a non-directed donor (NDD) (i.e., an altruist donor who decides to donate without having an intended recipient) donates a kidney to a patient from an incompatible patient-donor pair. Then, the donor in this pair is further matched to another incompatible pair and so forth, forming a chain with a NDD and  $l$  recipients (see Figure 1.2). As a chain is initiated by a NDD, it can be organized so that no patient-donor pair has to donate a kidney before they have received one, allowing the simultaneity requirement to be relaxed.

A debate around chains is whether they should be performed simultaneously ‘domino-paired donation’ (DPD) or non-simultaneously ‘non-simultaneous extended altruistic donor’ chains (NEAD) [3, 6, 19]. In the first, the NDD triggers a short simultaneous chain with the donor in the last pair donating to a candidate on the waiting list for a deceased donor. In the second, the NDD initiates a long non-simultaneous extended altruistic donor (NEAD) chain consisting of several short segments, each carried out simultaneously. The last donor in each segment of a NEAD chain becomes a *bridge donor*, i.e., a donor whose intended recipient has

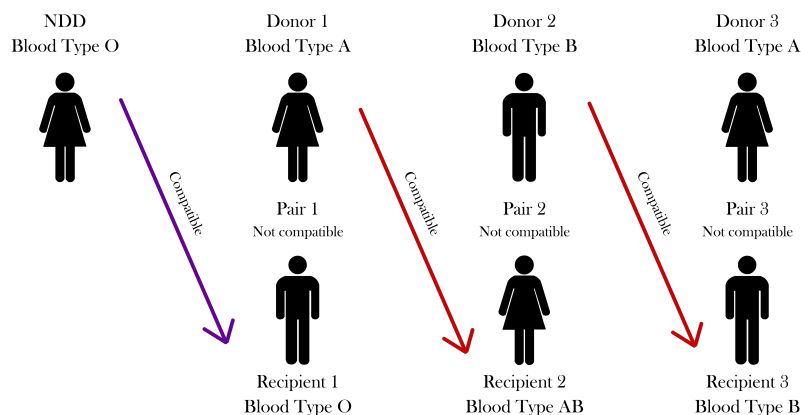


Figure 1.2: Length-3 chain: This diagram illustrates a three-transplant chain involving one non-directed donor and three blood-type-incompatible recipient-donor pairs. The last donor -Donor 3- will continue the chain.

received a kidney and becomes a NDD for the next segment (see Figure 1.3). When a bridge donor initiates a new segment within the current period is called a *short-term bridge donor* whereas a *regular bridge donor* may take months to continue the chain.

Gentry et al. [19] compared DPDs that involve at most two incompatible pairs and end with a simultaneous donation to a candidate on the deceased donor waiting list (three transplants) with NEAD chains in which each simultaneous segment has three or fewer incompatible pairs and ends with a bridge donor (also three transplants). Their simulations suggested that DPDs would provide as many or more transplants than NEAD chains. Ashlagi et al. [6] tested both the same assumptions as Gentry et al. [19] and new assumptions considering longer chains of length 4-6. In the latter, Ashlagi et al. [6] showed (see Figure 5 in [6]) that in approximately 80% of the instances, NEAD-6 provides more transplants than DPD, and in approximately 60–65% of the instances NEAD-6 produces more transplants than NEAD-5. Dickerson, Procaccia and Sandholm [16] also conducted simulations that let them to conclude that although NEAD chains result in more transplants than DPDs, NEAD chain segments should be constrained at four transplants whereas Anderson et al. [3] pointed out the benefit of long chains (unbounded) specially when the pool consists

of highly sensitized patients (i.e patients who have many anti-bodies and are thus not likely to accept a donor's kidney) since the compatibility graph becomes sparse, making short chains substantially suboptimal.

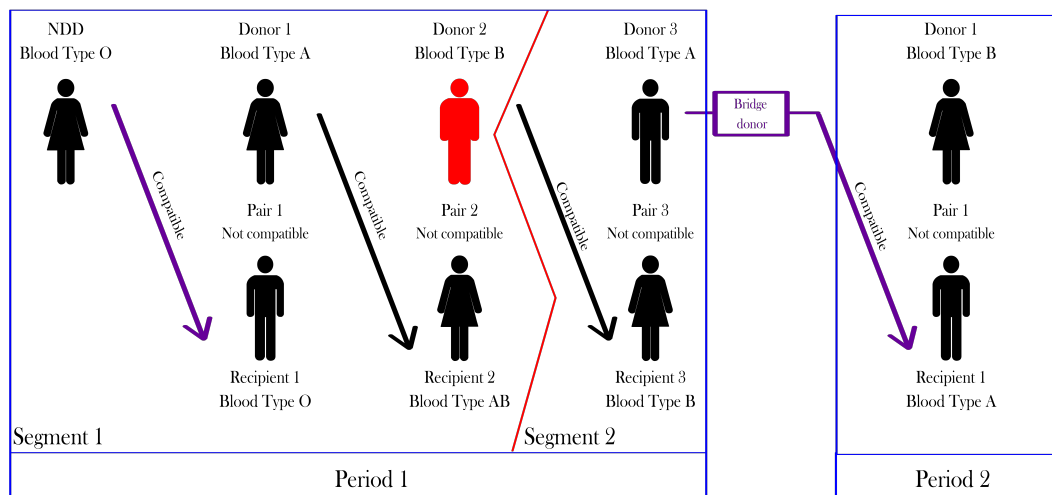


Figure 1.3: NEAD: In the first period, a chain of length 3 is found and is divided into two segments. Donor 2 becomes a short-term bridge donor and Donor 3 becomes a (regular) bridge donor. In period 2, a chain beginning with Donor 3 is found.

Mathematically, the longer the chains, the higher the number of matches. As the solution space is bigger when considering unconstrained chains, the optimal solution is at least as good or better than that of a model constraining the length of chains. However, in practice this may not be true. If broken chains become common, i.e bridge donors or incompatible pairs in a NEAD chain fail to donate a kidney, the broken link will make the rest of that chain to fail, affecting a large number of incompatible pairs in the match. Several kidney exchange matching services have adopted NEAD chains for arranging kidney exchanges in the United States, including the National Kidney Registry (NKR) and the United Network for Organ Sharing (UNOS). The former is the highest-volume kidney exchange program today. Anderson et al. [3] reported that of its more than 1,000 transplants, more than 67 percent of them have used NEAD chains and approximately 88 percent of all its transplants have been achieved through chains and although most chains are short the longest ones account for more than 11 percent of all transplants (see Figure

6 in [3]). The experience of UNOS program differs from that of the NKR. Persistent match-offer refusals and crossmatch failures after identifying matches (with only 8% of matches resulting in a transplant, see [27]), but prior to conducting transplants, led it to limit chain segments to 4. Note that this decision was not bridge donors-related because these failures occur before the transplant procedure takes place, not during or after it. Even with the restriction on chain length, most UNOS transplants have come from chains and three-way cycles.

## 1.1 PROBLEM STATEMENT

Given the list of NDDs and PDPs, along with their compatibility information, it is then possible to build the compatibility graph, which depicts the potential matches between donors and patients. PDP nodes represent patient-donor pairs biologically incompatible, while NDD nodes represent a single bridge or altruist donor, who can initiate a chain. Then, an edge going from one node to another, implies that the donor in the first node, either if it belongs to an incompatible pair (PDPs) or to a non-directed donor (NDDs), is compatible with the patient in the next node. Thus, each edge is a potential transplant, and has associated a weight determined by a medical board, to distinguish the priority given to that transplant. Moreover, a maximum cycle length  $k$  is established according to medical capacity in the transplant centers conducting nephrologies. Chains, on the other hand, may or may not be constrained. If they are, the maximum length  $l$  is also known in advance. The kidney exchange program in every transplant center is in charge of making decisions regarding the maximum length of cycles and chains.

All the previous information makes up an instance of the Kidney Exchange Problem (KEP). The objective is then to allocate donors to patients, organized into cycles (of length at most  $k$ ) and chains (of length at most  $l$  and triggered by NDDs) so that each donor gives a kidney once and each patient receives one also once, while maximizing the sum of the weights of all transplants conducted. When every



edge has unit weight, the aim is to perform as many transplants as possible so that patients can receive a compatible kidney.

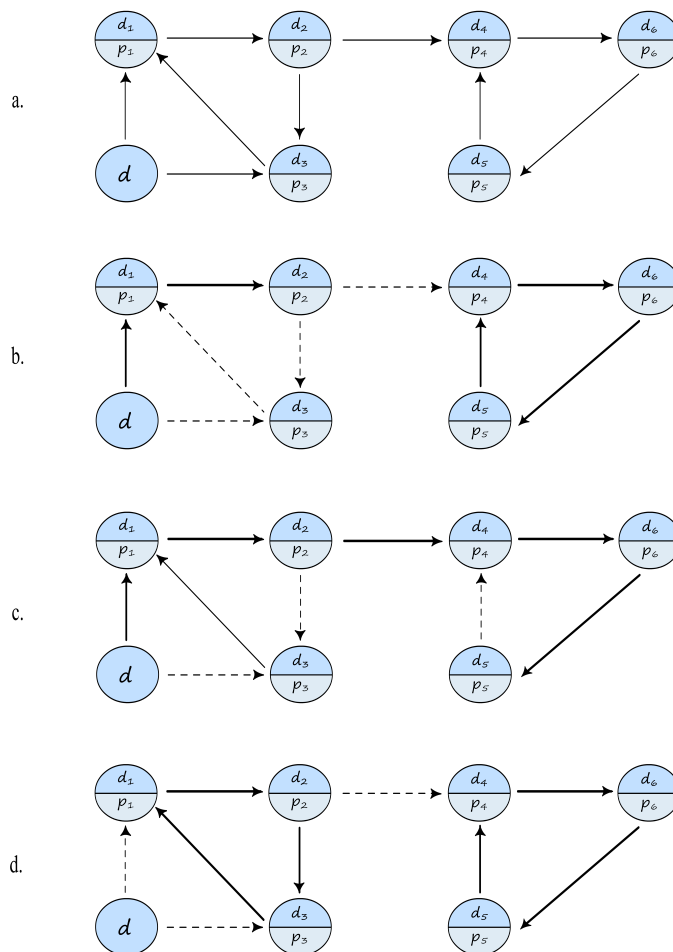


Figure 1.4: An example of a KEP instance and its possible solutions. (a) Original compatibility graph; (b) Cycle-and-chain solution; (c) chain-only solution; (d) cycle-only solution.

A KEP instance is illustrated in Figure 1.4. Figure 1.4a shows the initial compatibility graph with a single NDD (node  $d$ ) and six PDPs. Figure 1.4b depicts a solution (bold edges) when considering chains and cycles. Although the cycle and chain length for this solution is three and two, respectively, there are multiple values of  $k$  and  $l$  that can yield this solution. We know  $k$  must be at least three and  $l$  might be unbounded or constrained to two. The solution given in Figure 1.4c involves only a chain of length five, but again there are multiple variants of the KEP that can produce this solution. These variants may or may not allow cycles and

perhaps  $l$  is unbounded or constrained to five. In any case, the optimal arrangement turns out to be a unique chain. [Figure 1.4d](#), on the other hand, depicts a solution when only cycles of length three maximize the objective function. Similarly, chains may or may not be allowed in this model and still cycles of length three (thus,  $k \in \{3, 4, \dots\}$ ) produce the best solution.

When we are allowed to find unbounded chains and cycles, the KEP turns into the maximum weighted perfect matching problem on a bipartite graph, which can be solved in polynomial time. When only 2-cycle exchanges are allowed, this case is equivalent to the maximum matching problem, also solvable to optimality in polynomial time through Edmonds' maximum cardinality matching algorithm [\[17\]](#). The general problem with  $k$ -way cyclic exchanges corresponds to the KEP and is known to be NP-hard for  $k \geq 3$  [\[1, 9\]](#).

## 1.2 BACKGROUND

Roth, Sönmez, and Ünver [\[36\]](#) first proposed organizing kidney exchange on a large scale and first applied Operations Research (OR) methods to the Kidney Exchange Problem, also known as Kidney Paired Donation Problem (KPDP), including integrating cycles and chains [\[36, 37, 35\]](#).

Abraham, Blum, and Sandholm [\[1\]](#) and Roth, Sönmez, and Ünver [\[38\]](#) introduced the two fundamental Integer Programming (IP) models for kidney exchange: the cycle formulation and the edge formulation. The former, includes one binary decision variable for each feasible cycle or chain. The latter includes one decision variable for each compatible pair. In the cycle formulation, the number of constraints is polynomial in the input size, but the number of variables is exponential. In the edge formulation, the number of variables is linear but the number of constraints is exponential. They reported experimental results with simulated test instances (as proposed by Saidman et al. [\[39\]](#)) with up to 10,000 incompatible pairs.

Constantino et al. [11] introduced the first two compact IP formulations (i.e. that the number of variables and constraints are polynomial in the input size) for the kidney exchange considering only cycles: the edge-assignment formulation and the extended edge formulation. Although their extended edge formulation was empirically effective in finding the optimal solution where the length of cycles is greater than 3, both formulations have a weaker linear program (LP) relaxation than the cycle formulation, even when NDDs are not considered. They generated instances with low, medium and high density, the largest with 1000 incompatible patient-donor pairs.

Mak-Hau [28] introduced both a compact formulation integrating chains and cycles EE-MTZ by using Constantino's extended edge formulation to model cycles and a variant of the Miller-Tucker-Zemlin model for the traveling salesman problem to model chains; and an exponential version of the EE-MTZ, that modeled cycles like Roth, Sönmez, and Ünver [38]. The largest instance size reported was 256 PDPs and 6 NDDs.

Anderson et al. [4] introduced an exponential formulation in the number of variables and constraints based on the price-collecting salesman problem (as in the Traveling Salesman Problem we also must find a cycle visiting each city at most once, but now we may skip some cities by paying a penalty). The smallest instance reported had 162 agents (PDPs and NDDs) and the longest 1341 agents.

Dickerson et al. [14] introduced three new integer programming formulations combining Constantino's extended edge formulation and position-indexed variables for subtour elimination. Dickerson used both real instances from The United Network for Organ Sharing (UNOS) and the UK kidney exchange (NLDKSS); and simulated data. On average, the UNOS instances considered 231 PDPs and 2 NDDs and the NLDKSS instances considered 201 PDPs and 7 NDDs. Simulated data was based on all historical UNOS data, reflecting the expected instances size in the future, with instances up to 700 PDPs and 175 NDDs.

To avoid the need to keep the entire model in memory, column generation (Branch & Price) and constraint generation (Branch & Cut) algorithms have been implemented in literature. The fastest algorithms to date for the kidney exchange problem use column generation over the cycle formulation, see Abraham, Blum and Sandholm [1]; Dickerson Procaccia and Sandholm [15]; Glorie, Van de Klundert and Wagelmans [20]; Klimentova, Alvelos and Viana [25]; Plaut, Dickerson and Sandholm [34]; Dickerson et al. [14]. The only approach to date using constraint generation over a variant of the edge formulation is Anderson et al. [4], which is effective for solving instances where the cycle-length limit is 3 and chains are unconstrained in size, but it is outperformed by branch-and-price-based approaches when chain size is constrained [34]. Alternative objectives to those of finding the maximal number of exchanges or the maximal weighted sum of all exchanges for the kidney exchange problem include maximizing the expected number of transplants (Dickerson, Procaccia, and Sandholm [15]; Pedroso [33]; Alvelos et al. [2]) and lexicographic optimization of a hierarchy of objectives (Glorie, Van de Klundert, and Wagelmans [20]; Manlove and O'Malley [30]).

### 1.3 MOTIVATION

Kidney exchange programs implemented worldwide, based on the successful resolution of the KEP, have saved thousands of end-stage renal disease sufferers so far. Talking about the KEP implies, indeed, talking about a set of problems, each one adapted to the needs, regulations and experiences of every country and its corresponding kidney exchange program. Such differences make every KEP variant a unique problem with its own complexity. The KEP evolved from considering only cyclic exchanges in the last decade to integrating cycles and chains a few years ago, as a result of the increase in the number of altruists. Variations on the two fundamental IP models [1, 38] have laid the foundations of the current state of the art. Throughout the literature, the edge formulation is known to be impractical for

having exponentially many constraints, leading to an immediate need of large scale approaches (B&C and B&P), even for small to medium size instances. In this research, we show that only a subset of those many constraints keep the model correct and the performance is much better in almost all instances (based on clinical data from the National Kidney Registry program (NKR) in the USA) when compared with other IP formulations. Moreover, we show that the strongly connected components on the compatibility graph yield a natural partition of nodes and edges that allow us to model a subgraph containing only the PDPs that can be involved in a feasible solution, when either only cyclic exchanges are considered or chains and cycles are allowed so that  $l < k$ . The study of the KEP connectivity structure and concept applications of flow network theory on its resolution have not been addressed so far. This research aims at setting up a starting point in this direction.

## 1.4 OBJECTIVES

- Evaluate through computational experiments the impact on the number of exchanges when using the DPD and NEAD schemes.
- Compare experimentally the performance of current IP formulations encountered in the literature for several variants of the KEP on data from two reference papers (based on actual data from the NKR) and generated instances.
- Prove that the current number of constraints in the edge formulation can be substantially reduced keeping the model correct and design an efficient algorithm to this end.
- Propose a natural partition of nodes and edges to model a compatibility graph  $G$  as a set of split subgraphs so that the global solution is the sum over every subgraph optimal solution.

## 1.5 ORGANIZATION

To begin with, [Chapter 2](#) presents the IP formulations for the cycle variant of the KEP. Here, we introduce two new formulations as part of our contribution. In [Chapter 3](#), we address the chains and cycles KEP version. We show and analyze most IP formulations currently found in literature. We also apply some results from [Chapter 2](#) to enhance some of them. An assessment on model performance is presented in [Chapter 4](#) as well as a description of the algorithms used to solve the KEP. Additionally, the results are also used to measure the KEP impact on the number of exchanges. In [Chapter 5](#), we draw conclusions about our research and state some final thoughts. Finally, Appendixes containing relevant information to understand the KEP clinical background, details of the experiments conducted and in-depth results of this research are presented at the end.

## CHAPTER 2

# THE CYCLE PACKING VARIANT OF THE KIDNEY EXCHANGE PROBLEM

---

In this section we present the Kidney Exchange Problem (KEP) variant where feasible solutions can only take the form of cycles. When NDDs are present in the pool, besides cyclic exchanges it is also possible to find chains in form of fake cycles by adding dummy edges from each PDP to each NDD. In absence of NDDs, a cycle packing only corresponds to actual  $k$ -way cycles. This version is defined in [Section 2.1](#). Some existing and new IP formulations are then presented throughout [Section 2.2](#).

## 2.1 PROBLEM STATEMENT

Let  $P$  be the set of patient-donor pairs (PDPs) and  $N$  be the set of non-directed donors (NDDs). We model the Kidney Exchange Problem on a directed graph  $G = (V, E)$  where the set of vertices  $V = \{1, \dots, |V|\}$  is partitioned into  $P = \{1, \dots, |P|\}$  and  $N = \{|P| + 1, \dots, |P| + |N|\}$ . In absence of NDDs, as is the case with the cycle-only version,  $V = P$ . The set of edges  $E$  contains edge  $(i, j)$  if and only if the donor in node  $i$  is compatible with patient in pair  $j$  so that  $E = \{(i, j) \mid i \in V, j \in P\}$ . Note that  $\{(i, j) \mid i \in V, j \in N\} = \emptyset$  since NDDs do not have paired patients,

and therefore they do not have incoming edges. The digraph has no loops since we assume every PDP is incompatible. Each arc  $(i, j) \in E$  has a weight  $w_{ij} \in \mathbb{R}^+$  (set of non-negative real numbers), representing the priority given by the transplant center to that transplant. The weights are used to capture various prioritization schemes and other value judgments. There is a maximum cycle length limit given by  $k$  due to logistical issues as explained in [Chapter 1](#). The largest chain length is constrained to  $l$ ; however,  $l$  may be long or even unbounded. The objective is to find a maximum weight node-disjoint chain and cycle collection, bounded by  $l$  and  $k$ , respectively. When each arc has unit weight, the objective function is to maximize the number of transplants, otherwise, the objective is to maximize the weighted sum of the number of transplants.

When the KEP considers only cycles, it can be modeled as the problem known in graph theory as the Cycle Packing Problem in a directed graph [9]. [Figure 2.1](#) depicts a compatibility graph and a feasible solution when cycles of length at most  $k = 3$  are allowed and there are not NDDs in the pool. The feasible assignment is shown by the bold edges, while the dashed edges represent original compatible edges that are not part of the feasible solution.

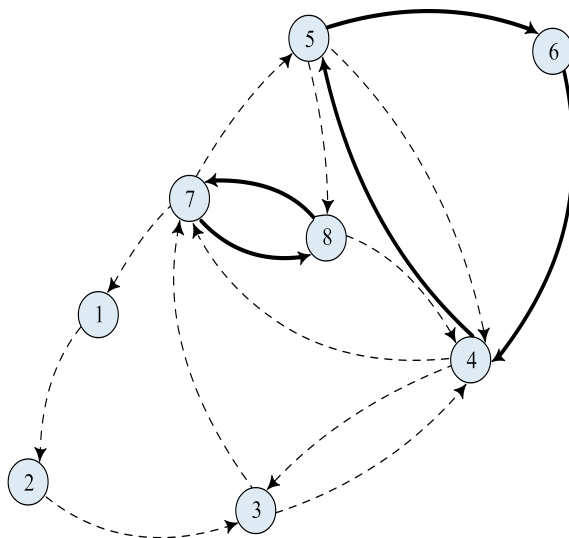


Figure 2.1: Cycle packing variant: Feasible solution with  $|P| = 8$ ,  $w_{ij} = 1$  and  $k = 3$ .



## 2.2 INTEGER PROGRAMMING FORMULATIONS

In this section, we present three existing integer programming formulations for the cycle packing variant of the KEP. The first two formulations are the well-known Cycle Formulation and Edge Formulation proposed independently by Abraham, Blum, and Sandholm [1] and Roth, Sönmez, and Ünver [38]. The former uses an exponential number of cycles and the latter an exponential number of constraints; later on, we will see how to reduce the size of constraints for this model. The last formulation is the Extended Edge Formulation, this is along with the Edge-assignment Formulation, the two first known compact formulations for the KEP, proposed by Constantino et al. [11]. As the Extended Edge Formulation dominates the Edge-assignment Formulation we consider only the Extended Edge Formulation in our analysis. Note that these formulations are easily scalable to introduce chains of length  $l < k$  by adding a dummy edge with zero weight from each vertex to every NDD, treating actual chains as cycles. Moreover, as part of our contribution we introduce the Partitioned Edge Formulation and the Partitioned and Reduced Edge Formulation, two more tractable versions of the Edge Formulation.

### 2.2.1 CYCLE FORMULATION

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

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

Define  $w_C = \sum_{(i,j) \in C} w_{ij}$ . The Cycle Formulation (C) can be written as follows:

$$\text{Maximize} \quad \sum_{C \in \zeta_k} w_C z_C \quad (2.1)$$

$$\text{subject to} \quad \sum_{C: i \in V(C)} z_C \leq 1 \quad i \in V \quad (2.2)$$

$$z_C \in \{0, 1\} \quad C \in \zeta(k) \quad (2.3)$$

The objective function (2.1) maximizes the weighted number of transplants. In the case of unitary weights,  $w_C$  equals the number of edges in  $C$ , i.e., the number of transplants associated with cycle  $C$ . Constraints (2.2) ensure that every vertex is in at most one of the selected cycles since each donor may donate, and each patient may receive only one kidney.

### 2.2.2 EDGE FORMULATION

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

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

Then, the model can be expressed as follows:

$$\text{Maximize} \quad \sum_{(i,j) \in E} w_{ij} x_{ij} \quad (2.4)$$

$$\text{subject to} \quad \sum_{j: (i,j) \in E} x_{ij} - \sum_{j: (j,i) \in E} x_{ji} = 0 \quad i \in V \quad (2.5)$$

$$\sum_{j: (i,j) \in E} x_{ij} \leq 1 \quad i \in V \quad (2.6)$$

$$\sum_{1 \leq p \leq k} x_{i_p i_{p+1}} \leq k - 1 \quad (i_1, \dots, i_k, i_{k+1}) \in \Pi \quad (2.7)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E \quad (2.8)$$

The objective function (2.4) maximizes the weighted sum of matches – in the case of unit weights, it maximizes the total number of transplants. Constraints (2.5) guarantee that donor  $i$  donates a kidney if and only if patient  $i$  receives one back. Constraints (2.6) guarantee that a donor can only donate a single kidney and constraints (2.7) enforce the cycle-length to exclude cycles of cardinality longer than  $k$ . Any feasible cycle will always contain a path with at most  $k - 1$  edges (not repeating edges). So, if we preclude all length- $k$  paths from being part of a cycle, cycles of length greater than  $k$  are ruled out from feasible solutions. It is believed so far that all paths of length  $k$  are required to be considered explicitly in the model [1, 11, 38], but in Section 2.2.4 and Section 2.2.5 we will prove the model remains correct considering only a subset of such constraints, which can grow exponentially with  $k$ .

Let us look in detail at constraints (2.7). Consider Figure 2.2 and suppose  $k = 3$ . Observe that path  $(1 \rightarrow 2 \rightarrow 3 \rightarrow 5)$  cancels the infeasible cycle  $(1,2,3,5,1)$  since the sum of its edges have to be less or equal to 2, forming only cycles of size 2 and 3. Once we find all length-3 paths, we observe that path  $(2 \rightarrow 3 \rightarrow 5 \rightarrow 1)$  along with two more paths also delete cycle  $(1,2,3,5,1)$ . A similar situation arises with the other two infeasible cycles that can be covered for several paths. Then, one idea might be to add only one path per every infeasible cycle, but that implies doing an exhaustive search among all the exponentially many cycles that a graph may contain. Therefore, it is less computationally expensive to find all length- $k$  paths, although the number of such paths can also be exponential. This is why the Edge Formulation has shown to be impractical even in small instances [11].

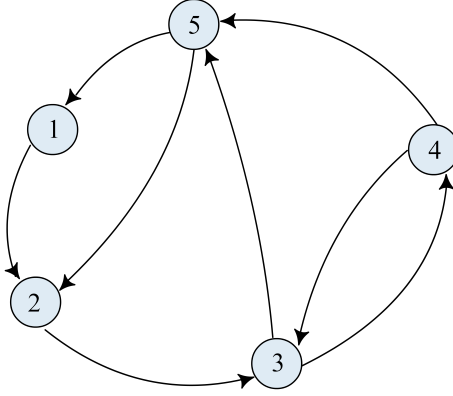


Figure 2.2: Sample graph: Finding length-3 paths.

### 2.2.3 EXTENDED EDGE FORMULATION

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

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

The Extended Edge Formulation (EE) is given as follows:

$$\text{Maximize} \quad \sum_{l \in L} \sum_{(i,j) \in E} w_{ij} x_{ij}^l \quad (2.9)$$

$$\text{subject to} \quad \sum_{j \in P: (i,j) \in E} x_{ij}^l = \sum_{j \in P: (j,i) \in E} x_{ji}^l \quad i \in V, l \in L \quad (2.10)$$

$$\sum_{l \in L} \sum_{j: (i,j) \in E} x_{ij}^l \leq 1 \quad i \in V \quad (2.11)$$

$$\sum_{(i,j) \in E} x_{ij}^l \leq k \quad l \in L \quad (2.12)$$

$$\sum_{j:(i,j) \in E} x_{ij}^l \leq \sum_{j:(i,j) \in E} x_{ij}^l \quad i > l, l \in L \quad (2.13)$$

$$\sum_{j:(i,j) \in E} x_{ij}^l = 0 \quad i < l, l \in L \quad (2.14)$$

$$x_{ij}^l \in \{0, 1\} \quad \in E, l \in L \quad (2.15)$$

The objective function (2.9) also maximizes the weighted number of transplants. Constraints (2.10) ensure the flow balance of a vertex, i.e., a paired donor will donate a kidney to another patient in the pool if and only if her intended recipient has received one in return. Constraints (2.11) guarantee that no more than one kidney transplant is involved for each PDP. Constraints (2.12) guarantee that the cardinality of each cycle and copy is not more than  $k$ . The Edge Extended Formulation has symmetry. Constraints (2.13) and (2.14) avoid multiplicity of solutions in the IP model induced by permutation of cycle indices. Symmetry elimination is achieved by restricting the index of a cycle to be exactly the smallest vertex, i.e., the smallest index among all vertices involved in the cycle.

#### 2.2.4 PARTITIONED EDGE FORMULATION

In this section, we show that when a graph  $G$  is not itself a strongly connected component (SCC), the partition of nodes and edges induced by the SCCs of  $G$  forms a collection of subgraphs that split the original KEP problem into several vertex-disjoint and arc-disjoint problems. A direct application of such a result is that now we can find the global optimum by optimizing each subgraph separately, without losing optimality. The Extended Edge Formulation is benefited from this, not only for handling smaller instances split into subgraphs but for having a fewer number of constraints (2.7). This is because such a decomposition discloses unnecessary paths, which is the base for us to introduce the Partitioned Edge Formulation. Therefore, the full set of all length- $k$  paths can be reduced as long as the graph contains more than one SCC. See Section 2.2.5 for a reduction method that works in any case.

## 2.2.4.1 CONNECTIVITY AND ITS RELATION WITH THE KEP

Now, we recall some concepts and properties [7, 12, 41] to prove that constraints (2.7) can be reduced guaranteeing optimality. In graph theory, a strongly connected component can be defined as follows:

**DEFINITION 2.1** *A strongly connected component of a directed graph  $G$  is a maximal subset of vertices  $S \subseteq V$  such that for every pair of vertices  $u$  and  $v$  in  $S$ , we have both  $u \rightsquigarrow v$  and  $v \rightsquigarrow u$ ; that is, there is a directed path from  $u$  to  $v$  and a directed path from  $v$  to  $u$ .*

One property of strong connectivity is that it partitions the vertices in such a way that each vertex belongs to exactly one SCC, as it is stated by the following lemma:

**LEMMA 2.2** *Let  $S$  and  $S'$  be distinct strongly connected components in a directed graph  $G = (V, E)$ . Let  $v \in S$ , let  $v' \in S'$ . Suppose that  $u \in S$  and  $u \in S'$ . Then  $S = S'$ .*

**PROOF:** If  $u \in S$  and  $u \in S'$ , then  $G$  contains paths  $v \rightarrow u \rightarrow v'$  and  $v' \rightarrow u \rightarrow v$ , thereby contradicting the assumption that  $S$  and  $S'$  are distinct strongly connected components. ■

At this point, we have proved that the compatibility graph for the KEP can be divided into vertex-disjoint and arc-disjoint subgraphs (see [Figure 2.3](#)). However, we still have to prove that when taking into account only the edges inside every SCC, we are not missing feasible space. Observe that if there exists edges going out from one SCC  $S$  to another distinct  $S'$ , they will never be part of a cycle since  $G$  does not contain a path from  $S'$  to  $S$  that closes a potential cycle. This means, in fact, that directed cycles are contained only in the SCCs of  $G$ . Therefore, adding constraints

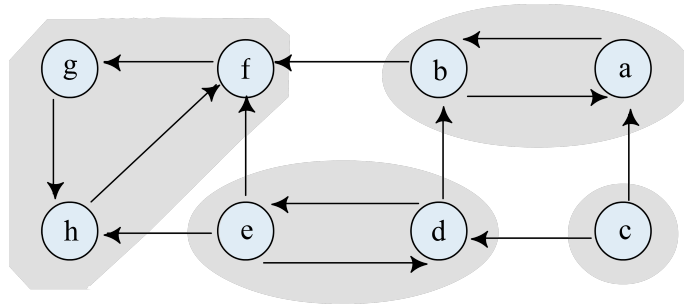


Figure 2.3: SCCs in  $G$ : Each shaded region is a strongly connected component of  $G$ . Each vertex belongs to exactly one SCC. Vertex  $h$  forms a trivial SCC.

of type (2.7) containing such edges turns out to be unnecessary and confirms our decomposition correctness, as proved below.

**LEMMA 2.3** *Let  $S$  and  $S'$  be distinct strongly connected components in a directed graph  $G = (V, E)$ , let  $u, v \in S$ . Also, let  $u', v' \in S'$ , and suppose that  $G$  contains a path  $u \rightarrow v'$ . Then  $G$  cannot contain a path  $u' \rightarrow v$ .*

**PROOF:** If  $G$  contains a path  $u' \rightarrow v$ , then it contains paths  $u \rightarrow v' \rightarrow u'$  and  $u' \rightarrow v \rightarrow u$ . Thus,  $u$  and  $u'$  are reachable from each other, therefore contradicting the assumption that they belong to distinct strongly connected components. ■

Consider again Figure 2.3, savings on constraints (2.7) come from disregarding edges  $(c,b)$ ,  $(f,e)$ ,  $(f,b)$ ,  $(g,c)$ ,  $(h,g)$  and  $(h,d)$ . Ruling out length- $k$  paths that use these edges does not remove feasible cycles at all, and therefore optimality is guaranteed.

#### 2.2.4.2 INTEGER PROGRAMMING FORMULATION

Let  $Q$  be the set of subgraphs induced by the SCCs of  $G = (V, E)$  and  $q$  be the number of non-trivial SCCs, so that  $Q = \{Q_1, \dots, Q_h, \dots, Q_q\}$ . Also, let  $G_h = (V_h, E_h)$  be the  $h$ -th subgraph in  $Q$  and  $\Lambda_h$  be the full set of length- $k$  paths in the  $h$ -th SCC.

We keep the same variable  $x_{ij}$ , now associated with each arc  $(i, j) \in E_h$ , defined exactly as in [Section 2.2.2](#).

Then, the Partitioned Edge Formulation (PE) can be expressed as follows:

$$\text{Maximize } \sum_{(i,j) \in E_h} w_{ij} x_{ij} \quad (2.16)$$

$$\text{subject to } \sum_{j:(i,j) \in E_h} x_{ij} - \sum_{j:(j,i) \in E_h} x_{ji} = 0 \quad i \in V_h \quad (2.17)$$

$$\sum_{j:(i,j) \in E_h} x_{ij} \leq 1 \quad i \in V_h \quad (2.18)$$

$$\sum_{1 \leq p \leq k} x_{i_p i_{p+1}} \leq k - 1 \quad (i_1, \dots, i_k, i_{k+1}) \in \Lambda_h \quad (2.19)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E_h \quad (2.20)$$

Constraints (2.19) are as before, all length- $k$  paths ( $k + 1$  vertices) inside every subgraph  $G_h = (V_h, E_h)$ .

### 2.2.5 PARTITIONED AND REDUCED EDGE FORMULATION

The Partitioned Edge Formulation significantly reduces the number of constraints (2.7) (see [Chapter 4](#)), but it can not be applied when  $G$  is itself a SCC and still every infeasible cycle is covered by several paths. For these reasons, we went further with our analysis.

From [Definition 2.1](#), we can draw that *two vertices are strongly connected if and only if there exists a general directed cycle that contains them both* since by definition all vertices are reachable from each other. A general directed cycle can be either a circuit or a simple cycle. While the latter only repeats the first and last vertex in the cycle, the former allows repeating several vertices. In other words, any path within a SCC is part of a general directed cycle. This fact is very useful



because once we choose one vertex  $u$  as a potential starting and ending point of a cycle and compute its length- $k$  paths we are sure those paths actually lead to a certain type of cycle containing node  $u$ . Notice that the next time we choose another vertex to check for its paths, we can remove vertex  $u$  from the SCC as we only want to find paths for infeasible cycles we have not covered yet. When removing a node from a SCC, remaining vertices may form a new smaller SCC while some others are no longer reachable. Even though graph in [Figure 2.1](#) is itself a SCC, the previous result can be applied to it as shown in [Figure 2.4](#). Consider  $k = 3$ . In quadrant I, vertex 4 is chosen, it gives rise to 7 paths (encoding by markers). In quadrant II, after removing vertex 4, a new SCC is computed (shaded areas). In quadrant III, vertex 7 is chosen, generating 1 path. In quadrant IV, after removing vertex 7, SCCs are recomputed, but this time they are made up of a single node. Then, the process ends with 8 length-3 paths.

The full set of length-3 paths in [Figure 2.1](#) has 50 paths, which are illustrated in [Table 2.1](#). Thus, with our method the number of paths and, consequently the number of constraints (2.7) decreases by 86% (see [Chapter 4](#) for more results). However, within the 8 paths we found only 7 are actually needed. As we said before, such paths are part of either a circuit or a simple cycle. The path  $4 \rightarrow 3 \rightarrow 7 \rightarrow 1$  becomes a circuit. Note that, the only way this path leads to a cycle is repeating (besides node 4) node 3. Constraints (2.6) eliminate circuits as they are infeasible solutions, so adding the above-mentioned path is unnecessary because circuits are not allowed. One idea to get rid of circuits when we are looking for simple cycles is to perform a depth-first search to check whether or not an intended path becomes a simple cycle. In [Algorithm 2](#) we present the Double-Search Algorithm, which is such as the SCC-Based Search Algorithm, except because now we have to determine if a path leads or not to a simple cycle before storing it. In [Chapter 4](#), we discuss the trade-off between time and path reduction.

In order to choose a specific node  $u$  at every step, we tested three priority rules: (a) the maximal in-degree, (b) the maximal out-degree, and (c) the maximal degree

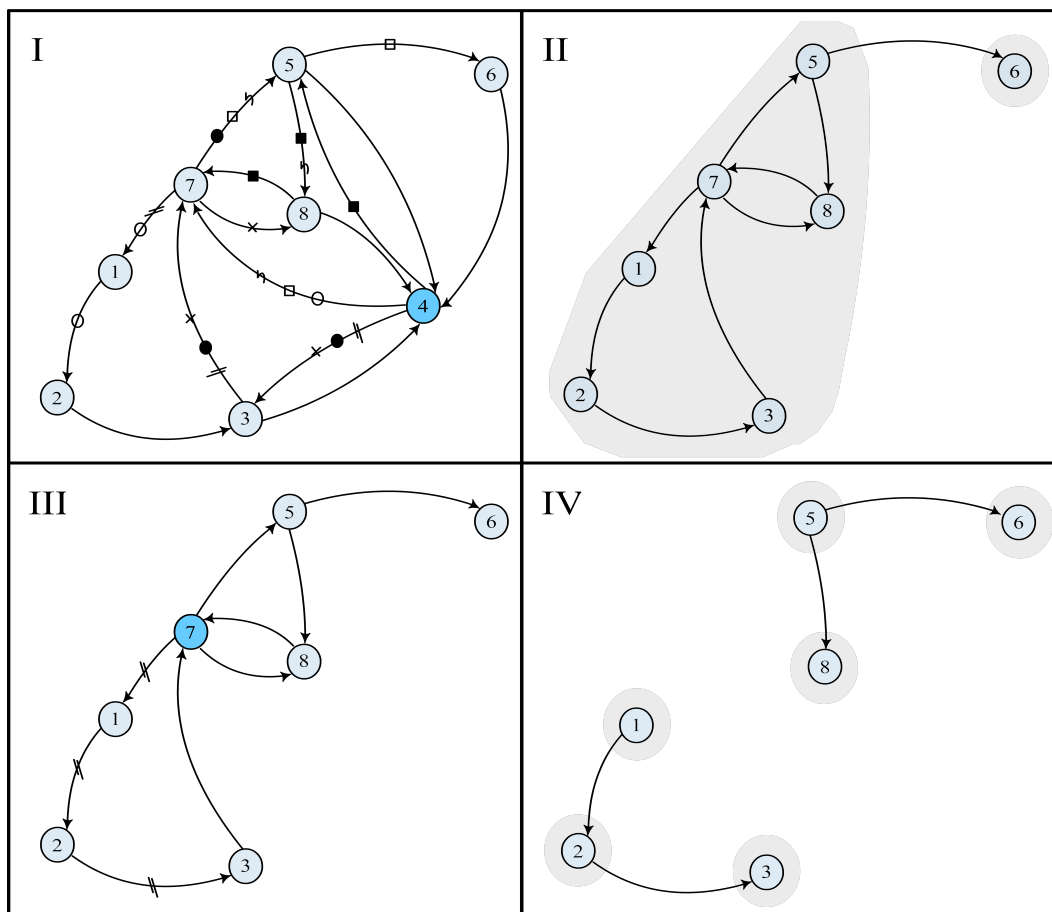


Figure 2.4: Removing sequentially a vertex from a strongly connected component.

of a vertex. Generally, the maximal in-degree yielded the highest reduction of paths. This can be explained in part, because the number of general directed cycles  $u$  can be part of, highly depends on the number of its predecessor vertices. As  $u$  can reach all of them and viceversa, we know that at least one general directed cycle can be formed per every predecessor of  $u$ . Thus, the larger the number of predecessors, the higher the likelihood of covering more cycles and nodes so that once we remove  $u$  from the graph, fewer nodes keep strongly connected and then fewer paths remain to be found.

Algorithm 1 formally defines the process just described. Let  $StrCC(G)$  be a procedure that finds the non-trivial SCCs of a directed graph  $G$ . To do so, we implemented Kosaraju's Algorithm [42], also known as the Kosaraju-Sharir algo-

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

 Table 2.1: Full set of length-3 paths for [Figure 2.1](#)

rithm which runs in linear time. Other efficient algorithms for finding SCCs are Tarjan’s strongly connected components algorithm [43] and the path-based strong component algorithm [18]. Let  $Pool$  be the set of connected components in which a strongest connected component is decomposed. Let  $T$  and  $T'$  be auxiliary graphs. Let  $A^u$  be the set of edges that are incident to  $u$ . Additionally, let  $choose(T, g)$  be a procedure for finding a node in  $T$  to be removed depending on the node selection strategy given by ‘g’, where  $g = \{\text{indegree, outdegree, total degree}\}$ , i.e.,  $choose(T, o)$  returns the node in  $T$  with highest outdegree. Ties are broken-up arbitrarily. Let  $DepthFirstSearch(k, u, T)$  be the well-known algorithm for traversing graph data structures, that, starting at node  $u$ , traverses  $T$  in the search of length- $k$  paths. Thus, it returns the set of length- $k$  paths starting at node  $u$ . Also, let  $Pool\_max(Pool)$  be a function that returns the subgraph in  $Pool$  with the largest cardinality node set. Finally, let  $\Omega_h$  be the set of length- $k$  paths found by Algorithm 1 in each  $Q_h$  and  $\Omega = \sum_{1 \leq h \leq q} \Omega_h$ .

---

**Algorithm 1** SCC-Based Search for length- $k$  paths in the KEP

---

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

```

1:  $Q = \{Q_1, \dots, Q_q\} \leftarrow StrCC(G)$ 
2:  $\Omega_h \leftarrow \emptyset$ 
3:  $\Omega \leftarrow \emptyset$ 
4: for  $h = 1$  to  $q$  do
5:    $Pool \leftarrow \{Q_h\}$ 
6:    $T = (\bar{V}, \bar{E}) \leftarrow Pool\_max(Pool)$ 
7:   while  $(|\bar{V}| \geq k + 1)$  do
8:      $Pool \leftarrow Pool \setminus T$ 
9:      $u \leftarrow choose(T, g)$ 
10:     $\Omega_h \leftarrow \Omega_h \cup DepthFirstSearch(k, u, T)$ 
11:     $\bar{V} \leftarrow V \setminus \{u\}$ 
12:     $\bar{E} \leftarrow E \setminus A^u$ 
13:     $T' \leftarrow StrCC(T)$ 
14:     $Pool \leftarrow Pool \cup T'$ 
15:     $T = (\bar{V}, \bar{E}) \leftarrow Pool\_max(Pool)$ 
16:   end while
17:    $\Omega \leftarrow \Omega \cup \Omega_h$ 
18: end for
19: return  $\Omega$ 

```

---

In the following algorithm, let  $LeadtoCycle(\omega, T)$  be a Boolean function that determines if a length- $k$  path  $\omega$  in graph  $T$  leads to a cycle.

---

**Algorithm 2** Double-Check Search for length- $k$  paths in the KEP

---

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

```

1:  $Q = \{Q_1, \dots, Q_q\} \leftarrow StrCC(G)$ 
2:  $\Omega_h \leftarrow \emptyset$ 
3:  $\Omega \leftarrow \emptyset$ 
4: for  $h = 1$  to  $q$  do
5:    $Pool \leftarrow \{Q_h\}$ 
6:    $T = (\bar{V}, \bar{E}) \leftarrow Pool\_max(Pool)$ 
7:   while  $(|\bar{V}| \geq k + 1)$  do
8:      $Pool \leftarrow Pool \setminus T$ 
9:      $u \leftarrow choose(T, g)$ 
10:     $\omega \leftarrow DepthfirstSearch(k, u, T)$ 
11:    while  $(\omega \neq \emptyset)$  do
12:      if  $(LeadtoCycle(\omega, T) = true)$  then
13:         $\Omega_h \leftarrow \Omega_h \cup \{\omega\}$ 
14:      end if
15:       $\omega \leftarrow DepthfirstSearch(k, u, T)$ 
16:    end while
17:     $\bar{V} \leftarrow V \setminus \{u\}$ 
18:     $\bar{E} \leftarrow E \setminus A^u$ 
19:     $T' \leftarrow StrCC(T)$ 
20:     $Pool \leftarrow Pool \cup T'$ 
21:     $T = (\bar{V}, \bar{E}) \leftarrow Pool\_max(Pool)$ 
22:  end while
23:   $\Omega \leftarrow \Omega \cup \Omega_h$ 
24: end for
25: return  $\Omega$ 

```

---

## 2.2.5.1 ALGORITHM 1 TIME COMPLEXITY

We now analyze the time complexity of Algorithm 1. Notice that our algorithm performs three core tasks: find the SCCs, choose a new node, and form length- $k$  paths. These tasks are repeated as long as non-trivial SCCs remain in the graph or until the number of remaining nodes in a SCC is  $\geq k + 1$  (otherwise the graph cannot contain infeasible cycles), whichever condition is reached first. Then, the worst case would be when the digraph is complete, (i.e. every pair of distinct vertices is connected by a pair of unique edges, one in each direction), because every time we remove a single node, all the remaining ones form a new SCC.

For finding the SCCs, we know that Kosaraju's Algorithm time complexity is  $\mathcal{O}(|V| + |E|)$ . For selecting a node, basically we arrange vertices in non-decreasing order of in-degree. To this end, we used the sorting function *sort* from the algorithm library in C++, that runs in  $\mathcal{O}(|V| \log_2(|V|))$ . For finding paths, the depth-first search complexity time depends on  $k$ . For instance, if  $k = 2$  and we have already chosen a starting node, the first time we have  $(|V| - 1)(|V| - 2)$  ways to form paths of size 2 (measured by the number of edges), if  $k = 3$  we have  $(|V| - 1)(|V| - 2)(|V| - 3)$  ways to form paths of size 3 and so on. The next time, we will have  $|V| - 1$  vertices and  $|E| - 2(|V| - 1)$  arcs. Notice that at any step Kosaraju's Algorithm complexity is  $\mathcal{O}((|V| - c) + (|E| - 2(|V| - c)))$  where  $c$  is the number of times we have removed a node. Such complexity actually decreases. For simplicity let's consider Kosaraju's Algorithm complexity as  $\mathcal{O}(|V| + |E|)$  at any moment. Similarly, the sorting function time complexity is  $\mathcal{O}((|V| - c) \log_2(|V| - c))$  at a specific round during the algorithm. As the total number of vertices decrease, so does the expression  $(|V| - c) \log_2(|V| - c)$ , but again let us consider  $c = 0$ . As we saw above, finding paths of size  $k$  needs to perform approximately  $|V|^k$  operations every time. The previous three main tasks are repeated  $(|V| - k)$  times for  $|V| \geq k$ . Therefore, the global time complexity of our algorithm is:

$$\left. \begin{array}{l} \text{Finding SCCs : } \mathcal{O}(|V| + |E|) \\ \text{Choosing a node : } \mathcal{O}(|V| \log_2(|V|)) \\ \text{Finding } k\text{-paths : } \mathcal{O}(|V|^k) \end{array} \right\} (|V| - k) \simeq \mathcal{O}(|V|^{k+1})$$

### 2.2.5.2 INTEGER PROGRAMMING FORMULATION

As stated before, let  $\Omega_h$  be the set of length- $k$  paths in the  $h$ -th SCC encountered by the SCC-based search Algorithm (Algorithm 1), and let  $\omega_h \in \Omega_h$ . Then, the Partitioned and Reduced Edge Formulation (PRE) can be expressed as follows:

$$\text{Maximize} \quad \sum_{(i,j) \in E_h} w_{ij} x_{ij} \quad (2.21)$$

$$\text{subject to} \quad (2.17) - (2.18) \quad (2.22)$$

$$\sum_{(i,j) \in \omega} x_{ij} \leq k - 1 \quad \omega \in \Omega_h \quad (2.23)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E_h \quad (2.24)$$

Notice that we can set a formulation either for every SCC ( $G_h = (V_h, E_h)$ ) of the original graph by using  $\Omega_h$  or a unique formulation for the full graph  $G = (V, E)$  by using  $\Omega$ . In the Partitioned Formulation, however, we can only take advantage of splitting the full graph if possible and then establishing a formulation for each SCC. We might keep a single IP formulation either because the graph  $G$  is itself a strongly connected component or because we desire to keep the full graph. In both cases,  $V_h = V$  and  $E_h = E$ . In Chapter 4, however, we do split the graph when considering the cycle packing version as our hypothesis is that coping with smaller instances might make the original problem more tractable.

## CHAPTER 3

# CHAINS AND CYCLES VARIANT OF THE KIDNEY EXCHANGE PROBLEM

---

In this chapter we present a generalization of the KEP in which the solution involves cycles and chains, although an optimal solution may be made up by either cycles or chains. We need to find such paths in the same graph we find cycles. This implies that, besides including cardinality-infeasible-cycle elimination constraints, we also need to prevent chains from forming cycles. Throughout the chapter, we discuss how some existing IP formulations deal with these new requirements. Most of them only consider unbounded chains that can be performed in practice as NEAD chains. Finally, we will see that results obtained in Chapter 2 can be applied to one of such formulations.

### 3.1 PROBLEM STATEMENT

Consider the same definition given in [Section 2.1](#), noting that  $V = P \cup N$ , where  $N$  is the set of altruists or bridge donors and  $G = (V, E)$  unless stated otherwise. Likewise, the objective is to find arc-and-vertex disjoint cycles and chains maximizing the weighted number of transplants or, the total number of exchanges. [Figure 3.1](#) depicts an example of a KEP instance in presence of NDDs and its optimal solution



when considering different values of  $w_{ij}$ . To the left, the compatibility graph is shown. To the right, the optimal solution is represented by bold arcs, with an optimal value of 8.0.

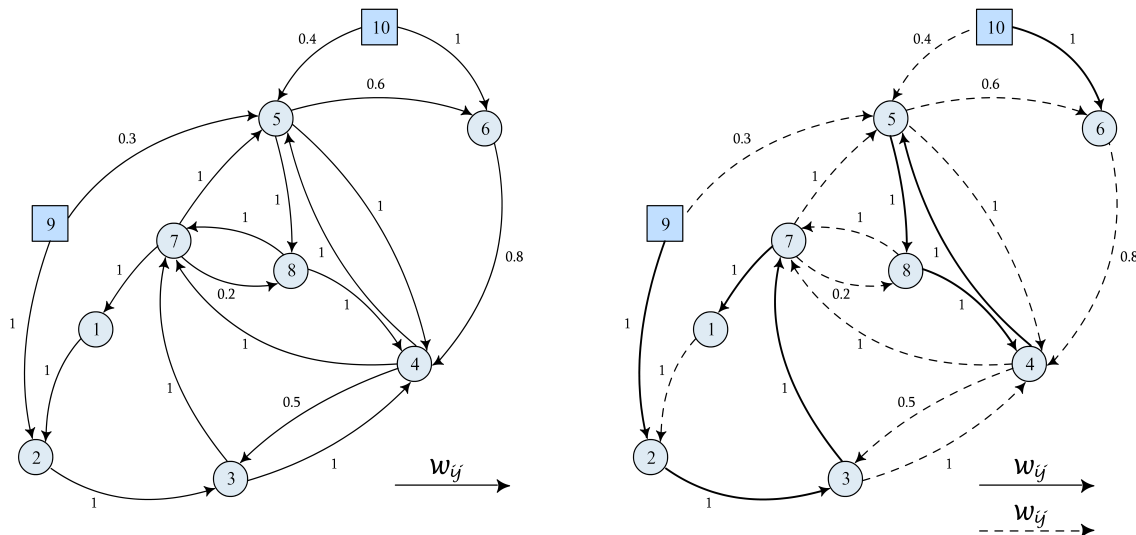


Figure 3.1: Example of a KEP instance in presence of NDDs.

## 3.2 INTEGER PROGRAMMING FORMULATIONS

In this section we present five IP formulations. The first two were proposed by Anderson et al. [4], the next two by Mak-Hau [28] and the last one is a contribution of this thesis.

The first model is an arc-based formulation, in which a binary variable represents chains as well as cycles. It also contains exponentially many constraints since a family type of the Dantzig, Fulkerson, and Johnson formulation [26] constraints of the TSP are introduced to cancel every infeasible cycle. Unlike the Edge Formulation, it is impractical to exhaustively enumerate and explicitly add those constraints to the model even for small instances, although they can be added as needed. The most efficient way known so far in Integer Programming to deal with this type of constraints is to relax those constraints and then add only those that are needed by

solving an associated separation sub-problem. In Appendix A, we describe a solution scheme for this formulation.

The second model is inspired by the Prize Collecting Traveling Salesman Problem (PC-TSP) [8, 21]. The substantial difference with the TSP is that now, we are allowed to form a cycle excluding some cities by paying a penalty. The PC-TSP is similar to the KEP in that we want NDDs to form a long path, which the PC-TSP closes off as a cycle, without being required to visit every node. This analogy is reflected on the way chains are held back from inducing cycles; a cut-set type of Subtour Elimination Constraints (SEC) similar to the one used for the Asymmetric Traveling Salesman Problem. The model borrows cycle variables from the Cycle Formulation. Therefore, this formulation contains exponentially many constraints as well as exponentially many variables. A similar solution algorithm to the arc-based formulation can be used for this formulation. See Appendix A for more details on this.

The third model is an extension of the Extended Edge Formulation. The original variable used in that model, now is split into two to allow chains. Cycles are treated as usual. For chains, however, constraints from the Miller–Tucker–Zemlin (MTZ) Formulation [13, 26, 31] of the TSP are used as SECs. To reduce the number of them, continuous variables are added to the model, and serve to indicate a position index in which a node, if part of a chain, is visited; keeping the formulation polynomial in the input size of variables and chains.

While the previous formulation uses cycle variables as in the Cycle Formulation, the fourth model, uses binary variables representing edges that belong to a cycle as in the Edge Formulation. In fact, strong cardinality-infeasible-cycle elimination constraints (2.7) are added to the model. Again, MTZ-type constraints are used as SECs. As the number of paths in (2.7) can grow exponentially with  $k$ , this formulation is exponential in the number of constraints.

The last formulation is based on the previous one. Here, we include the results

obtained in [Chapter 2](#) in order to reduce the number of exponential constraints of the Edge Formulation.

### 3.2.1 ANDERSON'S ARC-BASED FORMULATION

Keeping a similar notation to that used in [Section 2.2.1](#), let  $\zeta$  be the set of all cycles in  $G$ , let also  $\zeta_k$  be the set of all cycles in  $G$  with length at most  $k$ . Additionally,  $f_i^e$  and  $f_i^o$  stand for the flow entering to  $i$  and the flow going out from  $i$ , respectively. This formulation uses a variable  $x_{ij}$  associated with each edge  $(i, j) \in E$ , defined as follows:

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

Notice that this variable is used indistinctly for chains as well for cycles. Then, the Anderson's Arc-based Formulation (AA) can be expressed as follows:

$$\text{Maximize} \quad \sum_{(i,j) \in E} x_{ij} w_{ij} \quad (3.1)$$

$$\text{subject to} \quad \sum_{(j,i) \in E} x_{ji} = f_i^e \quad i \in V \quad (3.2)$$

$$\sum_{(i,j) \in E} x_{ij} = f_i^o \quad i \in V \quad (3.3)$$

$$f_i^o \leq f_i^e \leq 1 \quad i \in P \quad (3.4)$$

$$f_i^o \leq 1 \quad i \in N \quad (3.5)$$

$$\sum_{(i,j) \in C} x_{ij} \leq |C| - 1 \quad C \in \zeta \setminus \zeta_k \quad (3.6)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E \quad (3.7)$$

Constraints (3.4) say that patient–donor pair nodes give a kidney as long as they have received at most one. Constraints (3.5) say that NDD nodes can donate only one kidney. Constraints (3.6) rule out cycles of length greater than  $k$ . As chains are considered unbounded, there are no additional constraints on this.

### 3.2.2 PC-TSP-BASED FORMULATION

Let  $S$  be a set of nodes,  $S \subset V$ , and let  $\bar{S} = V \setminus S$ . For each  $i \in V$ , let  $\zeta_k(i)$  be the set of cycles from  $\zeta_k$  containing an edge incident to  $i$ . For this formulation Anderson et al. [4] split chain and cycle variables. Now,  $z_C$  is a variable for every feasible cycle  $C$  and  $x_{ij}$  is a variable for edges in unbounded chains, as defined below:

$$z_C = \begin{cases} 1 & \text{if cycle } C \text{ is selected for the exchange} \\ 0 & \text{otherwise} \end{cases}$$

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

Thus, the PC-TSP-based Formulation (PC-TSP) can be expressed as follows:

$$\text{Maximize } \sum_{(i,j) \in E} x_{ij} w_{ij} + \sum_{C \in \zeta_k} z_C w_C \quad (3.8)$$

$$\text{subject to } \sum_{(j,i) \in E} x_{ji} = f_i^e \quad i \in V \quad (3.9)$$

$$\sum_{(i,j) \in E} x_{ij} = f_i^o \quad i \in V \quad (3.10)$$

$$f_v^o + \sum_{C \in \zeta_k(i)} z_C \leq f_i^e + \sum_{C \in \zeta_k(i)} z_C \leq 1 \quad i \in P \quad (3.11)$$

$$f_i^o \leq 1 \quad i \in N \quad (3.12)$$

$$\sum_{(j,m):j \in \bar{S}, m \in S} x_{jm} \geq f_i^e \quad S \subseteq P, i \in S \quad (3.13)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E \quad (3.14)$$

$$z_C \in \{0, 1\} \quad C \in \zeta_k \quad (3.15)$$

Constraints (3.11) assure that every patient-donor pair, if involved in the solution, must belong to either a cycle or a chain and the flow out of any node  $i$  is at most the flow coming into this node. Constraints (3.13) say that if a node  $i$  is to be involved in any chain, there must exist a flow coming from a NDD. See Figure 3.2 for a clarifying example. The graph contains a single NDD denoted by the square node. Notice that if node  $i$  is to be involved in any chain, then  $f_i^e = 1$ . As a result, we must use at least one of the edges  $a$  or  $b$  that go across the cut separating  $S$  from the remaining nodes and  $n$ .

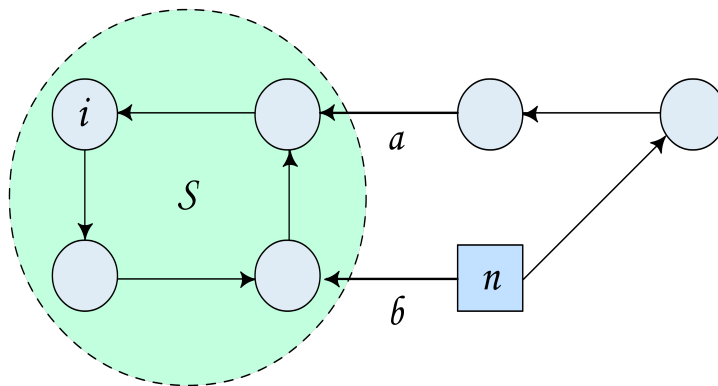


Figure 3.2: Example of cut set constraints for the PC-TSP model.

### 3.2.3 THE POLYNOMIAL-SIZED SPLIT FORMULATION

This is a Mixed Linear Integer Programming model, which is an extension of the Extended Edge Formulation to allow chains. To this end, an auxiliary sink node  $\tau$  is introduced that serves for implementing Subtour Elimination Constraints, in particular MTZ-constraints, broadly used in the context of the Asymmetric Traveling

Salesman Problem. This dummy node introduces some changes in our notation. Let  $E' = \{(i, j) \mid i \in V, j \in P^\tau\}$  where  $P^\tau = P \cup \{\tau\}$  and let us define  $G'$  as  $G' = (V \cup \{\tau\}, E')$ . This model has three types of variables: one set of continuous variables  $t_i$  to set a time stamp for vertex  $i$  should it be part of a chain, and two sets of decision variables as defined below:

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

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

Thus, the Polynomial-sized SPLIT Formulation (pSPLIT) is defined below:

$$\text{Maximize} \quad \sum_{(i,j) \in E'} x_{ij} w_{ij} + \sum_{l \in L} \sum_{(i,j) \in E} z_{ij}^l w_{ij} \quad (3.16)$$

$$\text{subject to} \quad \sum_{j \in P: (i,j) \in E} z_{ij}^l = \sum_{j \in P: (j,i) \in E} z_{ji}^l \quad i \in P, l \in L \quad (3.17)$$

$$\sum_{j \in P^\tau: (i,j) \in E'} x_{ij} = \sum_{j \in V: (j,i) \in E'} x_{ji} \quad i \in P \quad (3.18)$$

$$\sum_{j \in P^\tau: (i,j) \in E'} x_{ij} \leq 1 \quad i \in N \quad (3.19)$$

$$\sum_{j \in P^\tau: (i,j) \in E'} x_{ij} + \sum_{l \in L} \sum_{j \in P: (i,j) \in E} z_{ij}^l \leq 1 \quad i \in P \quad (3.20)$$

$$(2.12) - (2.14) \quad \text{replacing } x_{ij}^l \text{ by } z_{ij}^l \quad (3.21)$$

$$t_i - t_j + |P|x_{ji} + (|P| + 2)x_{ij} \leq |P| + 1 \quad i \in V, j \in P^\tau \quad (3.22)$$

$$t_i = 0 \quad i \in N \quad (3.23)$$

$$t_i \geq 0 \quad i \in P^\tau \quad (3.24)$$

$$t_\tau \leq |P| + 1 \quad i \in P^\tau \quad (3.25)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E' \quad (3.26)$$

$$z_{ij}^l \in \{0, 1\} \quad (i, j) \in E, l \in L \quad (3.27)$$

Constraints (3.17) and (3.18) are flow-balance constraints. The flow in must be equal to the flow out for edges that belong either to a cycle or a chain and flow out is at most one. Recall that  $\tau$  is a sink node that can be reached for every node, thus guaranteeing that these constraints can be also satisfied by chain variables. Again, constraints (3.19) say that a NDD can donate a single kidney. Constraints (3.20) assure patient-donor pairs to be part of either a chain or a cycle. Constraints (3.21) are borrowed from the Extended Edge Formulation to respect cycle feasibility and avoid symmetry issues. Constraints (3.22) prevent chain-edge variables to induce cycles. Constraints from (3.23) to (3.25) are bound constraints.

In order to see how constraints (3.22) operate, suppose there was a subtour  $(i_1, i_2, \dots, i_r, i_1)$  with  $2 \leq r \leq |P|$ . Writing constraints (3.22) for every edge of that subtour gives

$$\begin{aligned} t_{i_1} - t_{i_2} + 2(|P| + 1) &\leq |P| + 1 \\ t_{i_2} - t_{i_3} + 2(|P| + 1) &\leq |P| + 1 \\ &\vdots \\ t_{i_r} - t_{i_1} + 2(|P| + 1) &\leq |P| + 1 \end{aligned}$$

Adding up these constraints yield  $2r(|P| + 1) \leq r(|P| + 1)$ , a contradiction.

### 3.2.4 THE EXPONENTIAL-SIZED SPLIT FORMULATION

Unlike the previous model, this one simply uses a binary variable  $z_{ij}$  for each  $(i, j) \in E$  to indicate  $(i, j)$  is being used in a cycle. Additionally, MTZ-constraints are replaced by a stronger version of cardinality-infeasible-cycle elimination constraints: length- $k$  paths constraints as in equation (2.7). In Chapter 4, we present results for this model, also considering paths from constraints (2.23). Let us define the

Exponential-sized SPLIT Formulation (eSPLIT) as follows:

$$\text{Maximize } \sum_{(i,j) \in E'} x_{ij} w_{ij} + \sum_{(i,j) \in E} z_{ij} w_{ij} \quad (3.28)$$

$$\text{subject to } (3.18) - (3.19), (3.22) - (3.25) \quad (3.29)$$

$$(2.7) \quad \text{replacing } x_{ij} \text{ by } z_{ij} \quad (3.30)$$

$$\sum_{j \in P^r: (ij) \in E'} x_{ij} + \sum_{j \in P: (i,j) \in E} z_{ij} \leq 1 \quad i \in P \quad (3.31)$$

$$\sum_{j \in P: (i,j) \in E} z_{ij} = \sum_{j \in P: (j,i) \in E} z_{ji} \quad i \in P \quad (3.32)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E \quad (3.33)$$

$$z_{ij} \in \{0, 1\} \quad (i, j) \in E \quad (3.34)$$

Constraints (3.29) and (3.30) are borrowed from the Polynomial-sized SPLIT Formulation and the Edge Formulation, respectively. Every node belongs to either a cycle or a chain and the flow out of any node is at most one, which is met by constraints (3.31). Constraints (3.32) say that for each patient-donor pair, the flow in and the flow out are equal.

### 3.2.5 THE REDUCED EXPONENTIAL-SIZED SPLIT FORMULATION

A natural extension to the Exponential-sized SPLIT Formulation is to replace constraints (3.30) by constraints (2.23). Therefore, instead of finding the full set of  $k$ -paths we only aim at finding a subset, as small as possible, while keeping the model correctness. Notice that in this case, we cannot set different IP formulations for each SCC because now we also have to find a collection of vertex-and-arc disjoint paths. Recall that when we split the graph some edges are removed, losing feasible paths and therefore optimality. The advantage of constraints (2.23) is that they can



be applied regardless connectivity characteristics of the graph.

It follows that, the Reduced Exponential-sized SPLIT Formulation (ReSPLIT) can be stated as follows:

$$\text{Maximize } \sum_{(i,j) \in E'} x_{ij} w_{ij} + \sum_{(i,j) \in E} z_{ij} w_{ij} \quad (3.35)$$

$$\text{subject to } (3.18) - (3.19), (3.22) - (3.25) \quad (3.36)$$

$$(2.23) \quad \text{replacing } x_{ij} \text{ by } z_{ij} \quad (3.37)$$

$$(3.31) - (3.32) \quad (3.38)$$

$$x_{ij} \in \{0,1\} \quad (i, j) \in E \quad (3.39)$$

$$z_{ij} \in \{0,1\} \quad (i, j) \in E \quad (3.40)$$

## CHAPTER 4

# COMPUTATIONAL EXPERIMENTS

---

The purpose of this chapter is to investigate a few key issues and to provide an empirical assessment of the models and algorithmic solution strategies. In the first experiment, we carry out a comparison between the SCC-based search algorithm (see Algorithm 1 in Chapter 2) for computing subsets of length- $k$  paths and the well-known Depth-First Search Algorithm for computing the full set of length- $k$  paths. Then, a complete formulation assessment is performed considering the formulations for the cycle packing version (seen in Chapter 2) of the KEP and formulations for the chains and cycles version of the problem (seen in Chapter 3). In the former, we include experiments for assessing our two proposed models when considering only cycles, namely, the Partitioned Edge Formulation and the Partitioned and Reduced Edge Formulation. In the latter, we include experiments for assessing our new model when we are allowed to find cycles and chains, namely, the Reduced Exponential-sized SPLIT Formulation. Finally, we present a computational study to illustrate the difference and practical impact of allowing a list of altruist donors (list of NDD pairs).

## 4.1 DESCRIPTION OF DATABASE INSTANCES

We collected data from two sources: Anderson et al. [4] and Mak-Hau [28]. They gave us some of the instances studied in their papers. In Table B.1, descriptive information is presented in detail for the full set of instances.

- Anderson et al. [4] simulated the National Kidney Registry (NKR) Kidney Pair Donation (KPD) pool over a two year time period from May 24, 2010 to May 24, 2012. The initial pool contained 63 patient-donor pairs, and an additional number of 410 pairs arrive over the course of their simulation. The dataset also contained 75 altruistic donors. Compatibility between donors and patients was determined primarily by blood type and HLA compatibility rules (see Appendix D), although some patient preferences were also taken into account. To create representative snapshots of actual instances encountered by a KPD program they considered the fact that easy-to-match patients tend to wait little time to be involved in a solution, leaving in the pool the most hard-to match patients after each match run. Based on this, they estimated statistical parameters to reproduce this behavior. For full details, we refer the reader to [4, 5]. It is worth mentioning that some instances included negative weights for some academic experiments they performed, but the compatibility graphs were still accurate, so we took the absolute value of those weights. In overall, we study in this research 86 instances from their simulations.
- Mak-Hau gave us instances that were initially meant to consider only PDPs. They were adapted to generate instances with the same graph density and number of vertices in the PDPs and NDDs reported in Anderson et al. [4]. Thus, even though they are not the same problems, the graph density and sizes are the same. She created some vertices on the graphs (NDDs or PDPs) and allocated their blood type randomly, according to the proportions of the blood type population in the USA. Arcs on the graph were created if a donor's

kidney is by blood type compatible with a patient from any other PDP. In total, we received 7 instances of this type.

In spite of having similar descriptive information, Mak-Hau's and Anderson's instances have significant structural differences. Particularly, Mak-Hau's instances induce subgraphs that are all or about to be a single SCC, while Anderson's could be all partitioned into several non-trivial strong components. As we will see, this fact affects deeply the performance of the different formulations.

To evaluate the formulations we split the data into three subsets:

- DM1: corresponds to the set formed by 7 instances provided by Mak-Hau [28].
- DA1: This set is made up of those instances that Anderson et al. [4] found difficult and reported in their paper.
- DA2: These instances were not reported in [4] but are also part of the instances generated by their simulations.

## 4.2 EXPERIMENTAL CONDITIONS

For the following experiments, CPU times and bounds were obtained with CPLEX 12.7 in Concert Technology for C++ applications on a computer with an Intel Core i7 processor at 2.00 gigahertz, 8 gigabytes of RAM. In general, all our implementations were coded using C++. Only one core of the processor was assigned to these experiments.

### 4.3 COMPARING SEARCH ALGORITHMS TO FIND LENGTH- $k$ PATHS IN THE KEP

As seen in [Chapter 2](#), for *The Edge Formulation* we are required to feed the full set of length- $k$  paths (constraints (2.7) and constraints (2.19), respectively) into the MILP solver, which can be found by using the well-known Depth-First Search algorithm, while for the Partitioned and Reduced Edge Formulation we only need to provide a subset of them given by the proposed SCC-Based Search Algorithm explained in [Chapter 2](#). Since our algorithm cannot differentiate from paths leading to circuits and simple cycles, it is desirable to know what proportion accounts for each one. This is because the larger the number of paths leading to simple cycles, the more non-redundant length- $k$  paths are added to the MILP solver. Recall that circuits, regardless of the KEP formulation, are always excluded from feasible solutions by the balance constraints, thus we only need to focus on paths leading to simple cycles. Therefore, in this experiment we aim at reaching the following goals:

- Determine whether the SCC-Based Search Algorithm is more efficient than the Depth-First Search Algorithm in terms of computational time.
- Determine in which proportion length- $k$  paths found by the SCC-Based Search Algorithm actually conduct to simple cycles.
- Compare the computational time spent by the Double-Check Search Algorithm (see [Chapter 2](#)), which finds only leading-to-simple-cycles paths, to that of the SCC-Based Search Algorithm.

In order to accomplish these goals, we plot in [Figure 4.1](#) the following performance measures where the horizontal axis represents the full set of instances in [Table B.1](#) and the vertical axis represents the ratio of the different performance measures.:

- Rt-SCC-DEPTH: The ratio between the running time for the SCC-Based Search Algorithm to the time of the Depth-First Search Algorithm (line with diamond markers).
- Rp-SCC-DOUBLE: The ratio of the number of  $k$ -paths found by the SCC-Based Search Algorithm to the number of  $k$ -paths found by the Double-Check Search Algorithm (line with asterisk markers).
- Rt-SCC-DOUBLE: The ratio of the running time of the SCC-Based Search Algorithm to the time of the Double-Check Search Algorithm (line with circle markers).

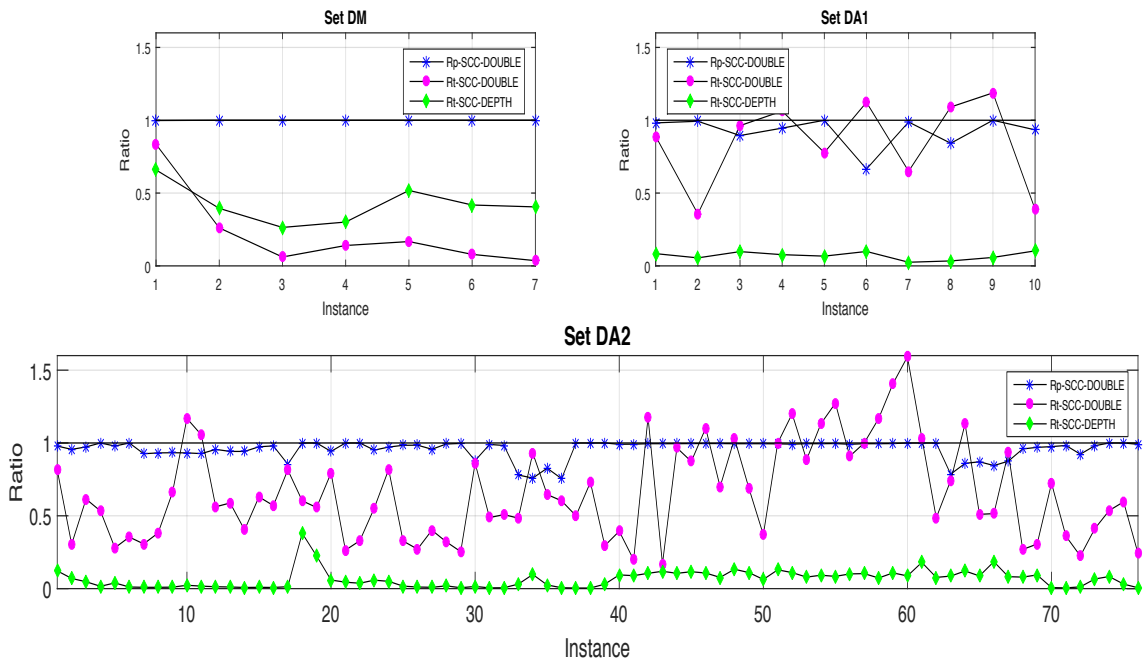


Figure 4.1: Performance of Algorithm 1.

As we can see the curve for Rt-SCC-DEPTH is almost always below  $y = 0.2$ , which means that the time spent by the SCC-Based Search Algorithm is about 20% the time of searching all length- $k$  paths in the graph by the Depth-First Search Algorithm. Consequently, it also means that time needed to fully introduce the Edge Formulation in the MILP solver takes significantly longer than that employed by

the Partitioned and Reduced Edge Formulation. We also see that for almost all instances, more than 90% of paths found by the SCC-Based Search Algorithm lead to simple cycles in the graph, as it can be drawn from Rp-SCC-DOUBLE. This means that the number of non-redundant paths (leading to simple cycles) found by our algorithm are a vast majority and the process to find them can be done very efficiently. Lastly, we see that Rt-SCC-DOUBLE is almost always below  $y = 1$ , meaning that for most instances the Double-Search Algorithms takes more time. Judging by the number of leading-to-simple-cycles paths found the SCC-Based Search Algorithm, it is not worthwhile to check if length- $k$  paths are actually part of a simple cycle.

For the Partitioned Edge Formulation we do not compare time needed to find paths because we also need to find the full set of length- $k$  paths like in the Edge Formulation. The only difference is that now, the graph is split into several SCCs and some vertexes may not be part of any, making the full set of paths for the Partitioned Edge Formulation smaller than the full set for the Edge Formulation.

#### 4.4 SELECTING THE BEST NODE-SELECTION STRATEGY FOR THE SCC-BASED SEARCH ALGORITHM

Now that we have determined the efficiency and accuracy of the SCC-Based Search Algorithm, in this experiment we assess different node-selection strategies. It is crucial to assess different node selection strategies since the algorithm relies on the picked nodes to choose the number and type of paths that finally will be part of the set of constraints. Specifically, we aim to find out the reduction in the number of paths when each strategy is applied compared to the full set of constraints and to determine how sensitive the number of paths is regarding each strategy. To this end we plot in [Figure 4.2](#) the number of paths of size  $k = 3$  and  $k = 4$  when considering the full set of paths in the original graph (dashed line) and the paths within the SCCs found by the SCC-Based Search Algorithm using different node selection criterion:

maximal outdegree (dotted line), that selects the node with the largest number of outgoing edges among the nodes forming a SCC; maximal indegree (dashed-dotted line), that selects the node with the smallest number of incoming edges within a SCC; and maximal total degree (solid line), that selects the node whose number of incoming and outgoing edges is maximum in a SCC. The horizontal axis represents instances in sets DM and DA1. The vertical axis represents the number of length- $k$  paths found by applying each strategy.

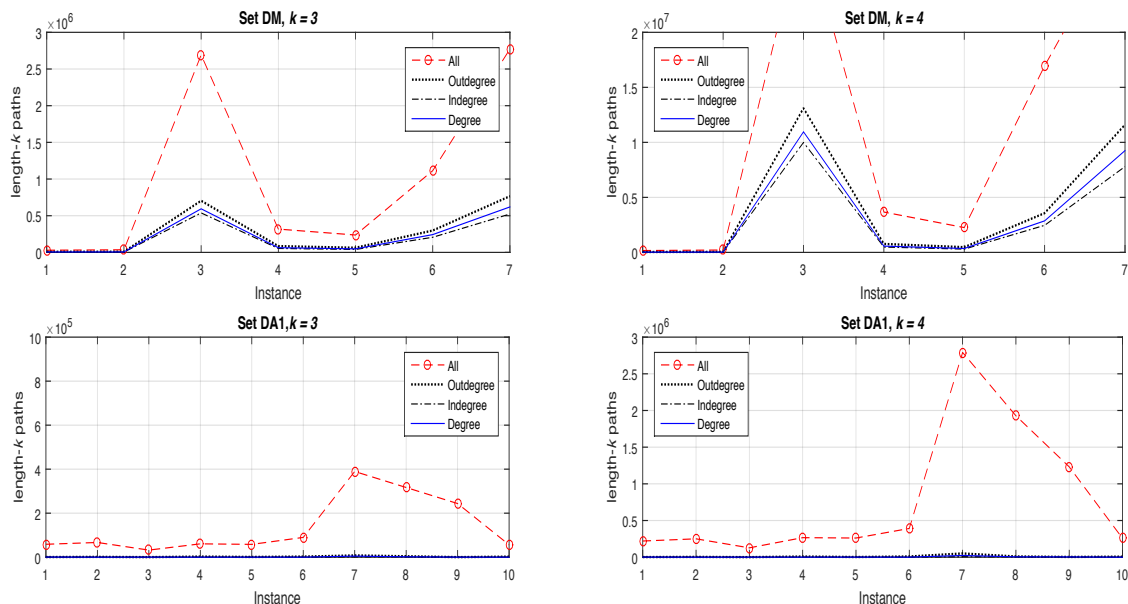


Figure 4.2: Assessment of node selection strategy in the SCC-Based Search Algorithm.

As we can see, the growth of the full set of length- $k$  paths is explosive for some DM instances. For example, generating the length-4 paths for instances 3 and 7 was so large (taking over 1.5 gigabytes each) we were unable to open the corresponding files. On the other hand, the number of paths in set DA1 remains relatively tractable, although one of them almost reaches 3 million paths when  $k=4$ . It can be clearly seen that our proposed algorithm under any of the node selection strategies reduces the needed number of paths considerably, specially under strategies of maximal indegree and maximal degree. However, the nature of the number of paths continues to be exponential. Due to this, we test the different formulations considering only



Formulation Name	Notation	Authors	KEP variant
Cycle Formulation	C	Abraham et al. [1] and Roth et al. [38]	Cycles
Edge Formulation	E	Abraham et al. [1] and Roth et al. [38]	Cycles
Extended Edge Formulation	EE	Constantino et al. [11]	Cycles
Partitioned Edge Formulation	PE	This thesis	Cycles
Partitioned and Reduced Edge Formulation	PRE('g')	This thesis	Cycles
Anderson Arc-based Formulation	AA	Anderson et al. [4]	Cycles and Chains
PC-TSP-based Formulation	PC-TSP	Anderson et al. [4]	Cycles and Chains
Polynomial-sized SPLIT Formulation	pSPLIT	Mak-Hau [28]	Cycles and Chains
Exponential-sized SPLIT Formulation	eSPLIT	Mak-Hau [28]	Cycles and Chains
Reduced Exponential-sized SPLIT Formulation	ReSPLIT('g')	This thesis	Cycles and Chains

Table 4.1: Notation of KEP formulations.

$k = 3$ . It remains to be seen the performance when other values of  $k$  are considered.

## 4.5 ASSESSMENT OF KEP FORMULATIONS

The following notation (see Table 4.1) is used to refer to each formulation studied so far. The IP formulations that consider chains and cycles were presented in their respective papers considering no upper bound on the chain size, although the authors also pointed out how a bounded chain model can be transformed into an unbounded chain model. For our experiments, we also consider unbounded chains. In Table 4.1, the argument  $g$  in  $\text{PRE}(g)$  and  $\text{ReSPLIT}(g)$  refers to the node selection strategy used by the SCC-based Search Algorithm to find the length- $k$  paths in each formulation. Let us formally define this notation as  $g \in \{i, d, o\}$  with (i)-indegree, (d)-degree, and (o)-outdegree.

### Results for the cycle packing variant of the KEP

In this section we present results for the cycle packing variant considering  $k = 3$ . As the KEP is solvable in polynomial time when  $k = 2$  [1, 9], only values of  $k \geq 3$  are challenging and therefore of interest for this thesis.

Computational results for this experiment are provided from Table 4.2 to Ta-

ble 4.6 where:

- PDPsR: Set of PDPs in which instances of sets DA1  $\cup$  DA2 were grouped. Instances were ordered in non-decreasing order of the number of PDPs to form such ranges. Every group, but the last, contains 10 instances. The last group has 6 instances. See Appendix B for details on this.
- PRE(i): Partitioned and Reduced Edge Formulation when the node selection strategy for the SCC-Based Search Algorithm is the maximal indegree.
- nf: Number of instances that failed to obtain a feasible solution, either because CPLEX was unable to solve the initial LP relaxation after the time limit ( $q/*$ ) or because before starting branching CPLEX displayed a run-out-of-memory status ( $*/r$ ).
- nVars and aVars: Number of variables in a single instance and average number of variables in a subset of instances, respectively, both according to a specific IP formulation.
- nConsts and aConsts: Number of constraints for a single instance and average number of constraints for a subset of instances, respectively, in a specific IP Formulation.
- rConsts and rgCon: Relative decrease in the number of length- $k$  paths associated to a single instance or to a group of instances, respectively, passing from formulation E to PE or PRE(i). The former is given by  $(nConstsE - nConstsX)/nConstsE$ , where  $nConstsE$  ( $nConstsX$ ) is the number of constraints obtained by using the E (PE or PRE(i)) formulation. In the latter, the reduction is given by  $(\sum_{(n)} nConstsE - \sum_{(n)} nConstsX) / \sum_{(n)} nConstsE$ , where  $n$  stands for each instance.
- tc and tp: Time employed for finding feasible cycles in order to determine the variables needed in the Cycle Formulation and the time employed for finding length- $k$  paths in the graph according to each formulation, respectively. For

searching feasible cycles, we implemented an adaptation of Johnson’s Algorithm [24], which is as far as we know one of the fastest to do so.

- *ts*: Time needed to compute the SCCs of the graph using Kosarajus’s Algorithm [42].
- *t*: Total CPU time employed for reaching an optimal solution in a single instance. In case of a group of instances, *t* represents the average running time. The CPU time limit was set to 1,200.00 seconds for all formulations. Notice that the Partitioned Edge Formulation splits the original problem into independent subproblems (see Section 2.2.4). Thus, times reported for instances under the PE formulation are the sum of times for such subproblems.
- *opt*: Number of instances solved to optimality. In all cases, the weighted edge sum optimization function was considered. This column does not appear when all instances were optimally solved.
- *gap*(%): is the average relative optimality gap associated to a formulation, defined by  $gap(\%) = \frac{(UB - LB)}{LB} * 100$ , where *UB* is the upper bound provided by the linear relaxation of the formulation and *LB* is the best found lower bound, replaced by the optimal value when known. Whenever a symbol “inf” appears, it means CPLEX could not find an upper bound for that problem. This column does not appear when all instances were optimally solved.

In Table 4.2 we present the computational results for the formulations considered in Chapter 2 and the set of instances DA1. the Cycle Formulation outperforms the others, specially the Extended Edge Formulation, taking substantially larger than the others. Although this formulation is not dominated by the Cycle Formulation (see [11]), we suspect there are two factors that may be provoking this result: first, the size of the formulation is considerably larger in comparison to the others (see Table 4.6) and second, the structure of these instances can be specially hard to solve. The Partitioned Edge Formulation and the Partitioned and Reduced Edge Formulation are the second best ranked, outperforming the Edge Formulation, to

such an extent that their running times are frequently in the same order of magnitude than those of the Cycle Formulation. Comparing the performance of PE and PRE(i), we see they are similar, with PE being slightly better.

Instance	C		E		EE		PE			PRE		
	tc	t	tp	t	gap(%)	t	ts	tp	t	ts	tp	t
1	0.09	0.02	0.38	0.22	0.0	10.69	0.02	0.04	0.08	0.02	0.03	0.06
2	0.03	0.001	0.46	0.17	0.0	6.14	0.02	0.03	0.06	0.02	0.03	0.13
3	0.02	0.02	0.27	0.09	0.0	6.86	0.02	0.01	0.05	0.02	0.02	0.06
4	0.07	0.001	0.44	0.20	0.0	43.80	0.03	0.05	0.08	0.03	0.02	0.06
5	0.04	0.001	0.42	0.41	0.0	45.86	0.03	0.03	0.06	0.03	0.04	0.09
6	0.07	0.02	0.65	0.27	0.0	81.38	0.06	0.07	0.09	0.06	0.03	0.06
7	0.10	0.03	2.58	42.53	5.9	1200.00	0.05	0.25	8.30	0.05	0.06	36.16
8	0.10	0.03	2.12	0.75	0.0	112.56	0.05	0.11	0.33	0.05	0.03	0.09
9	0.07	0.02	1.73	0.67	0.0	337.56	0.07	0.10	0.17	0.07	0.01	0.11
10	0.11	0.02	0.48	0.33	0.0	783.17	0.07	0.07	0.09	0.07	0.01	0.05

Table 4.2: Experimental results for 3-way cyclic exchanges for data set DA1.

In [Table 4.3](#) the experimental results for 3-way cyclic exchanges for set DM are shown. The symbol “-” in columns gap(%) and t represents instances that were not solved by PE as their underlying graph is a SCC. On the other hand, all the edge-based formulations perform poorly in set DM. The Extended Edge Formulation performs much better than the others, since this formulation could reach three optimal solutions out of seven and gave fairly good gap values. However, the Cycle Formulation clearly outperforms all others. We elaborate on this behavior when analyzing [Table 4.5](#) and [Table 4.6](#).

Instance	C			E			EE		PE				PRE			
	tc	gap(%)	t	tp	gap(%)	t	gap(%)	t	ts	tp	gap(%)	t	ts	tp	gap(%)	T
1	0.08	0.00	0.08	0.21	64.0	1200.00	0.0	5.11	0.02	0.21	-	-	0.02	0.09	93.7	1200.00
2	0.06	0.00	0.11	0.22	67.3	1200.00	0.0	12.50	0.02	0.20	94.7	1200.00	0.02	0.15	97.6	1200.00
3	0.44	0.00	10.89	15.57	inf	1200.00	10.6	1200.00	0.07	15.57	-	-	0.07	3.98	inf	1200.00
4	0.19	0.00	0.67	1.83	37.8	1200.00	2.0	1200.00	0.03	1.83	-	-	0.03	0.61	40.3	1200.00
5	0.34	0.00	0.25	1.46	66.1	1200.00	0.0	198.56	0.03	1.46	-	-	0.03	0.64	70.2	1200.00
6	0.66	0.00	81.56	6.48	inf	1200.00	inf	1200.00	0.04	6.48	-	-	0.04	2.96	34.5	1200.00
7	1.93	0.80	1200.00	16.58	inf	1200.00	inf	1200.00	0.05	16.58	-	-	0.05	6.69	inf	1200.00

Table 4.3: Experimental results for 3-way cyclic exchanges for data set DM.

PDPsR	C				E				EE				PE				PRE																																																																																																																																																																																													
	tc	opt	gap(%)	t	tp	opt	gap(%)	t	nf	opt	gap(%)	t	ts	tp	opt	gap(%)	t	nf	opt	gap(%)	t	ts	tp	opt	gap(%)	t																																																																																																																																																																																				
114-216	0.04	10	0.0	0.02	1.57	9	0.4	130.15	9	0.7	15.39	9	0.4	180.23	0.02	0.53	9	0.4	180.23	0.02	0.06	9	0.7	189.30	0.06	10	0.0	0.02	2.53	10	0.0	80.38	10	0.0	179.42	10	0.0	4.23	0.03	0.02	9	1.0	160.64	0.04	0.02	0.02	3.18	8	0.9	334.04	8	1.1	300.40	0.04	0.09	10	0.0	6.62	0.04	0.02	10	0.0	21.12	0.08	10	0.0	0.02	3.92	10	0.0	82.22	7	3.2	229.71	0.06	0.12	10	0.0	1.09	0.06	0.03	10	0.0	4.00	0.10	10	0.0	0.02	4.86	8	0.0	240.67	8	1.3	317.71	(1/0)	0.07	1.26	9	0.0	124.14	0.07	0.04	9	0.1%	162.14	0.08	10	0.0	0.01	0.56	10	0.0	0.52	0.06	0.08	10	0.0	0.15	0.08	10	0.0	0.01	2.99	10	0.0	1.25	0.08	0.16	10	0.0	0.48	0.08	0.01	10	0.0	0.07	0.09	10	0.0	0.02	1.79	10	0.0	19.95	7	15.3	485.78	0.08	0.11	10	0.0	0.33	0.08	0.02	10	0.0	0.23	0.09	10	0.0	0.02	43.62	3	0.0	474.42	(1/2)	1	23.1	377.16	(1/3)	0.12	9.22	4	16.2	401.05	(2/0)	0.12	0.27	4	1.1	407.64	0.64	6	0.0	0.10	43.62	3	0.0	474.42	(1/2)	1	23.1	377.16	(1/3)	0.12	9.22	4	16.2	401.05	(2/0)	0.12	0.27	4	1.1	407.64

Table 4.4: Experimental results for 3-way cyclic exchanges for data set DA1  $\cup$  DA2.

In [Table 4.4](#), we present the results for the set  $DA1 \cup DA2$ . First of all, we want to point out that the reduction in instance size, after finding the SCCs in the graph, is impressive (see [Table C.1](#)). This reduction has a positive impact, specially on the PE performance, which is after the Cycle Formulation, the best one. However, CPLEX could not solve the LP relaxation, and in consequence, it could not find a relative gap for two instances in the last subset. PRE(i), on the other hand, could solved those instances, obtaining a gap of only 1.1%. We observe that PRE(i) performs better than PE as the number of PDPs increases, which suggests that a deep reduction on the number of length- $k$  paths for small instances might weaken the IP formulation. Surprisingly, CPLEX ran out of memory three times with the Extended Edge Formulation, which again is outperformed by the other formulations. We will see below that despite mathbb a polynomial formulation, the size of EE can be large enough so as to run out of memory. Constantino et al. [\[11\]](#) also proposed a reduction scheme. It remains to be established whether that reduction improves significantly EE performance on graphs with multiple SCCs.

In [Table 4.5](#), the size of the different formulations is shown. We observe that instances in DM cannot be partitioned, as they are a single SCC, except the second instance, where the induced subgraph is a minor partition. The Cycle Formulation has by far the smallest size, it means that the number of feasible cycles is relatively small, making the constraints for the edge formulation families a large set. However, the reduction on constraints, even when it is not possible to partition the original graph, is substantial. The constraints decrease in PRE(i) regarding constraints in E is at least 80%. Even though the number of constraints and variables for PRE(i) is frequently less than those for the EE, the latter performs better on this subset as we saw previously, which may suggest that the difficulty for edge-formulation families when it comes to solve a graph that is itself a SCC relies heavily on the structural nature of the graph rather than on dimension.

The size of formulations for instances in  $DA1 \cup DA2$  are presented in [Table 4.6](#). Now, we see that the reduction on the number of constraints for PE falls between

Instance	C		E		EE		PE			PRE(i)		
	nVars	nConsts	nVars	nConsts	nVars	nConsts	nVars	nConsts	rConsts(%)	nVars	nConsts	rConsts(%)
1	80	152	897	29859	136344	46360	897	29859	0.0	897	4528	84.8
2	94	156	954	36130	148824	48362	948	35977	0.4	948	5443	84.9
3	4960	198	4740	2692311	938520	78606	4740	2692311	0.0	4740	538600	80.0
4	642	199	2341	318195	465859	79401	2341	318195	0.0	2341	54386	82.9
5	353	269	2583	237551	694827	144991	2583	237551	0.0	2583	40753	82.8
6	1369	310	4760	1114834	1475600	192510	4760	1114834	0.0	4760	206751	81.5
7	2609	389	7535	2772762	2931115	303031	7535	2772762	0.0	7535	523932	81.1

Table 4.5: Size of formulations and savings on 3-path constraints for data set DM.

66% and 97%. For the PRE model, this reduction is over 98% in all cases. The reduction in the number of constraints has been shown to have a valuable impact on this set of instances.

PDPsR	C		E		EE		PE			PRE(i)		
	aVars	aConsts	aVars	aConsts	aVars	aConsts	aVars	aConsts	rgCon(%)	aVars	aConsts	rgCon(%)
114-216	274.4	170.1	3959.1	270128.4	740889.0	54468.9	661.5	91659.7	66.1	661.5	4527.4	98.3
221-236	162.7	229.9	6853.1	436337.7	1575163.6	98460.4	544.5	20308.9	95.3	544.5	1098.10	99.7
241-285	146.0	264.1	9100.1	542560.2	2405233.3	115846.2	446.5	16201.8	97.0	446.5	1022.3	99.8
289-317	133.4	305.7	11907.3	655658.6	3658318.1	160640.6	592.7	21579.9	96.7	592.7	754.7	99.9
324-343	201.8	334.9	13718.7	797499.4	4587889.1	135198.9	1376.9	220374.5	72.4	1376.90	1606.6	99.8
343-348	99.4	346.5	13191.5	72391.1	4557000.5	90040.4	513.6	14706.4	79.7	513.6	303.8	99.6
348-365	162.1	355.0	13591.2	480167.3	4828165.4	135416.5	587.9	27975.8	94.2	587.9	384.20	99.9
365-379	110.5	371.2	15135.3	279233.2	5618812.2	141954.9	831.5	19399.4	93.1	831.5	667.2	99.8
386-474	904.0	425.3	19973.3	7079568.2	10636474.5	295032.0	4090.7	1603992.8	77.3	4090.7	18959.8	99.7

Table 4.6: Size of formulations and savings on 3-path constraints for set DA1  $\cup$  DA2.

### Results for the cycle and chains variant of the KEP

The following notation is used to interpret the results of this section:

- `nlazyC` and `alazyC`: Total number of violated lazy constraints found and added to the AA formulation while solving a single problem and the average number of all violated lazy constraints found and added in the same way, when solving a subset of instances, respectively.
- `tsep` and `atsep`: Time needed to find and add all the violated lazy constraints (i.e. `nlazyC`) to the AA formulation while solving a single problem and the

average time used to find and add the average of all the violated lazy constraints found in a subset of instances, respectively.

In the following experiment the goal is to compare the eSPLIT Formulation with ReSPLIT('g') formulations under the three different strategies for 'g'. Recall that ReSPLIT('g') is obtained by replacing constraints (2.7) by (2.23). To this end, we run those formulations on instances from DM and DA1 data sets. The results are shown in Table 4.7. First of all, the new instances studied in this section are the same as those just studied, except for the inclusion of NDDs. If we compare running times given in Table 4.7 to Table 4.2 and Table 4.3, we can see that instances in set DA1 become harder to solve in presence of NDDs, but still times are reasonably small, while instances in set DM become easier to solve by edge-based formulations. This improvement might be caused by the strengthening of the edge-based formulations by means of MTZ-type constraints or by the fact that cycles and chains are both allowed. Regarding the performance of the different strategies, the ReSPLIT families outperform the eSPLIT formulation in terms of lower running times and greater number of optimum values, specially when the node selection strategy is the maximal indegree and maximal degree. Although the maximal degree strategy solves one more problem than the maximal indegree strategy, we prefer the indegree strategy in an attempt of solving the largest instances without running out of memory. From now on, ReSPLIT stands for ReSPLIT(indegree).

In the following experiment, the goal is to compare the performance of formulations seen in Chapter 3, in terms of relative optimality gap and running time, for instances in set DA1 (see Table 4.8). As can be seen, the ReSPLIT formulation outperforms the others as it reaches 9 optimal values out of 10, running significantly faster. It is worthwhile mentioning that our implementation of the lazy constraint scheme to solve AA is very competitive with respect to that of Anderson et al. [4], solving to optimality one more instance in set DA1 (see Table S3 in [4]). However, it is not possible to establish a direct comparison in terms of running time since there are technological differences. As we can see PC-TSP performs poorly when it is



Data Set	Instance	eSPLIT		ReSPLIT(i)		ReSPLIT(o)		ReSPLIT(d)	
		t	gap(%)	t	gap(%)	t	gap(%)	t	gap(%)
DM	1	12.70	0.0	1.44	0.0	5.81	0.0	0.67	0.0
	2	16.03	0.0	1.14	0.0	6.86	0.0	0.92	0.0
	3	1200.00	inf	1200.00	inf	1200.00	inf	1200.00	27.5
	4	289.44	0.0	120.41	0.0	131.00	0.0	135.84	0.0
	5	274.28	0.0	125.44	0.0	138.61	0.0	113.67	0.0
	6	1200.00	inf	776.28	0.0	1200.00	0.6	904.27	0.0
	7	1200.00	inf	1200.00	inf	1200.00	inf	1200.00	inf
DA1	1	146.42	0.0	107.27	0.0	140.25	0.0	118.14	0.0
	2	0.59	0.0	0.49	0.0	0.55	0.0	0.39	0.0
	3	13.67	0.0	15.69	0.0	16.39	0.0	16.99	0.0
	4	55.47	0.0	52.81	0.0	62.56	0.0	51.83	0.0
	5	14.59	0.0	10.05	0.0	11.22	0.0	10.39	0.0
	6	21.75	0.0	16.45	0.0	22.99	0.0	21.81	0.0
	7	1200.00	25.1	1200.00	25.1	1200.00	25.1	1200.00	25.1
	8	107.19	0.0	40.09	0.0	78.17	0.0	55.77	0.0
	9	37.41	0.0	18.95	0.0	22.59	0.0	27.22	0.0
	10	134.28	0.0	115.14	0.0	156.05	0.0	154.09	0.0
Average time:		348.94		294.29		329.11		306.64	

Table 4.7: Assessment of eSPLIT and ReSPLIT formulations.

solved by a lazy constraint scheme. In [4] a branch-and-cut algorithm was proposed, instead, that is able to solve these instances in less than 9 seconds. The  $gap(\%)$  column does not appear for AA nor PC-TSP because any solution found by the lazy constraint scheme, different from the optimal, is infeasible. Therefore, it is not worthwhile analyzing solutions out of the feasible space. Additionally, it seems that pSPLIT works better than AA and PC-TSP for the smallest instances. As we can see in Table 4.12, even for small instances, the number of violated constraints in AA exceeds 6,000 constraints. This implies that it was necessary to optimize the model anew that number of times before reaching an optimal solution. Similarly, for the largest instances, the number of violated constraints are roughly as many as those of the smallest instances. Therefore, we suspect that pSPLIT converges faster than AA and PC-TSP for small instances, because it is more efficient solving only once a

small-sized polynomial formulation than solving thousands of times an increasingly larger formulation.

Instance	AA	PC-TSP	pSPLIT		ReSPLIT	
	t	t	gap(%)	t	gap(%)	t
1	1200.00	1200.00	0.0	448.48	0.0	107.27
2	0.13	1.14	0.0	6.83	0.0	0.49
3	119.09	1200.00	0.0	63.19	0.0	15.69
4	622.84	1200.00	0.0	251.44	0.0	52.81
5	48.27	449.34	0.0	92.22	0.0	10.05
6	55.70	721.03	0.0	215.42	0.0	16.45
7	1200.00	1200.00	25.1	1200.00	25.1	1200.00
8	320.75	1200.00	0.0	878.53	0.0	40.09
9	313.72	1200.00	0.0	1 110.55	0.0	18.95
10	1200.00	1200.00	87.5	1200.00	0.0	115.14

Table 4.8: Comparison of formulations for 3-way cyclic exchanges and unbounded chains on data set DA.

By contrast, AA and PC-TSP perform better on the DM instances (see [Table 4.9](#)). Looking at [Table 4.11](#) we can notice that the number of violated lazy constraints added by AA are small, reaching the optimal value quickly. Unlike the cycle packing variant, now the extension of the edge formulation families outperform the pSPLIT formulation, specially the ReSPLIT.

In [Table 4.10](#), we compare four formulations in the set  $DA1 \cup DA2$ , up to the subset 324-343 and then we choose the best two ranked formulations in terms of time and optimal values to continue to assess those two best formulations. Thus, symbol “-” means the results are not shown for the less competitive formulations. Again, the eSPLIT outperforms the other formulations, finding the highest number of optimal solutions and spending in almost all cases the shortest running time. Hence, we can conclude that the indegree selection node strategy works competitively, enhancing

Instance	AA	PC-TSP	pSPLIT		ReSPLIT(i)	
	t	t	gap(%)	t	gap(%)	t
1	0.02	0.14	0.0	6.03	0.0	1.44
2	0.03	0.14	0.0	8.13	0.0	1.14
3	0.11	0.38	inf	1200.00	inf	1200.00
4	0.11	0.42	0.0	187.59	0.0	120.41
5	0.17	0.45	0.0	223.39	0.0	125.44
6	0.59	4.72	inf	1200.00	0.0	776.28
7	0.44	0.89	inf	1200.00	inf	1200.00

Table 4.9: Comparison of formulations for 3-way cyclic exchanges and unbounded chains on set DM.

the original eSPLIT formulation. Most importantly, we see that for current realistic instances, the ReSPLIT formulation provides in several cases the optimal solution or gives a feasible one, with a gap of 22% in the worst case.

In [Table 4.11](#) and [Table 4.12](#) the size of the AA Formulation is shown for DM and  $DA1 \cup DA2$  instances, respectively. The nVars values were queried once CPLEX ended, thus it may differ slightly from the total number of edges. The column named NDDsR represents the range of the number of NDDs according to each group of instances. As seen in [Table 4.11](#), DM instances violate lazy constraints a few numbers of times, more precisely there are three instances reaching the optimal solution without violating any constraint and only one instance reaches up to 25 violated constraints. These figures are smaller compared with the violated constraints detected in the set  $DA1 \cup DA2$ . Looking at [Table 4.12](#), we notice that for the full set of Anderson’s instances, AA Formulation needs to add a high number of violated lazy constraints. In the event of PC-TSP is even worse, since it has to add constraints per every node contained in  $S$ , where  $S$  is every single cycle. Here, we do not provide the size of the PC-TSP formulation, but certainly is much higher than that of AA. More importantly is the fact that the separation problem was solved

PDPsR	NDDsR	AA		PC-TSP		pSPLIT			ReSPLIT(i)		
		opt	t	opt	t	opt	gap(%)	t	opt	gap(%)	t
114-216	3-21	7	371.95	6	480.35	8	11.7	302.43	8	11.7	262.92
221-236	22-34	5	600.07	5	600.28	5	13.2	606.95	5	13.2	600.11
241-285	5-39	5	674.07	4	841.62	5	11.8	670.33	5	11.4	608.43
289-317	6-39	4	721.40	4	724.24	5	31.6	730.62	7	21.7	507.42
324-343	6-46	6	596.46	2	960.38	4	33.6	998.93	9	1.7	178.68
343-348	46-46	2	1 018.03	-	-	-	-	-	10	0.0	172.10
348-365	43-49	6	543.09	-	-	-	-	-	10	0.0	90.64
365-379	39-50	9	127.03	-	-	-	-	-	9	5.9	121.98
386-474	49-50	4	401.90	-	-	-	-	-	4	11.0	415.46

Table 4.10: Comparison of formulations for 3-way cyclic exchanges and unbounded chains on data set  $DA1 \cup DA2$ .

very efficiently in any case, we can see, for instance, the worst case was separating 18,965 infeasible solutions, for which the lazy constraint scheme took less than 5 seconds to identify the violated constraints and add them to the model.

### Comparing the different KEP versions in terms of pairs matched

In the next experiment we selected 17 instances, that form part of sets DM and DA1. We took optimal solutions and contrasted them to determine whether the inclusion of NDDs to allow chains has an important impact on the number of exchanges. To this end we plot the increase in the number of exchanges when passing from the cycle variant to the cycles and chains variant, computed by Equation (4.1) and the contribution of chains to solutions considering cycles and chains, as stated by Equation (4.2). In Figure 4.3, the horizontal axis represents instances in sets DM and DA1 and the vertical axis shows performance measures in percentages. It can be observed the number of matches grows gently (line with square markers) regardless of the set, although the growth seems to be more significant in the set DA1 than in the set DM. Regarding chains contribution to the solution composition (line with circle markers), there is a clear difference between both sets. Solutions in set DM

Instance	nVars	nlazyC	tsep
1	960	0	$1 \times 10^{-4}$
2	992	0	$1 \times 10^{-4}$
3	4927	3	$1 \times 10^{-4}$
4	2382	7	$1 \times 10^{-4}$
5	2607	5	$1 \times 10^{-4}$
6	4776	23	$1 \times 10^{-4}$
7	7565	9	$1 \times 10^{-4}$

Table 4.11: Size of formulation AA for DM instances.

are formed mainly by chains, while in set DA1 most solutions are reached in the form of cycles.

$$\frac{\text{NumberMatchesChainsandCycles} - \text{NumberMatchesCycles}}{\text{NumberMatchesCycles}} \quad (4.1)$$

$$\frac{\text{NumberChains}}{\text{NumberChains} + \text{NumberCycles}} \quad (4.2)$$

For the same set, we present in [Figure 4.4](#) the average length of cycles and chains. The horizontal axis represents instances in sets DM and DA1 and the vertical axis represents the average length of chains and cycles involved in a solution, when  $k = 3$  and chain length is unbounded. We can see that the average length of chains for instances in the set DA1 is around three 3, with one exception. This result is interesting because having such size the transplants can be conducted as DPD chains. On the contrary, the average length of chains for instances in the set DM is by far larger than 3. For not too long chains, it is possible and likely advantageous to perform them as NEAD chains, but for chains exceeding hundreds of transplants, it seems unlikely to perform them all successfully, even under the scheme of NEAD chains, because there are multiple complications related to health condition, renegeing, last-minute incompatibilities and others, that may occur, making the chain fail

PDPsR	NDDsR	aVars	alazyC	atsep
114-216	3-21	3 980.00	8 311.00	0.93
221-236	22-34	6 884.40	6 383.40	2.63
241-285	5-23	9 136.50	9 429.40	3.28
289-317	6-27	11 975.90	8 403.00	3.17
324-343	6-46	13 746.80	13 682.00	2.76
343-348	46-46	13 191.60	18 964.80	4.87
348-365	43-48	13 602.00	8 125.00	3.52
365-379	39-50	15 154.20	1 546.30	0.64
386-474	49-50	20 893.42	2 126.33	1.34

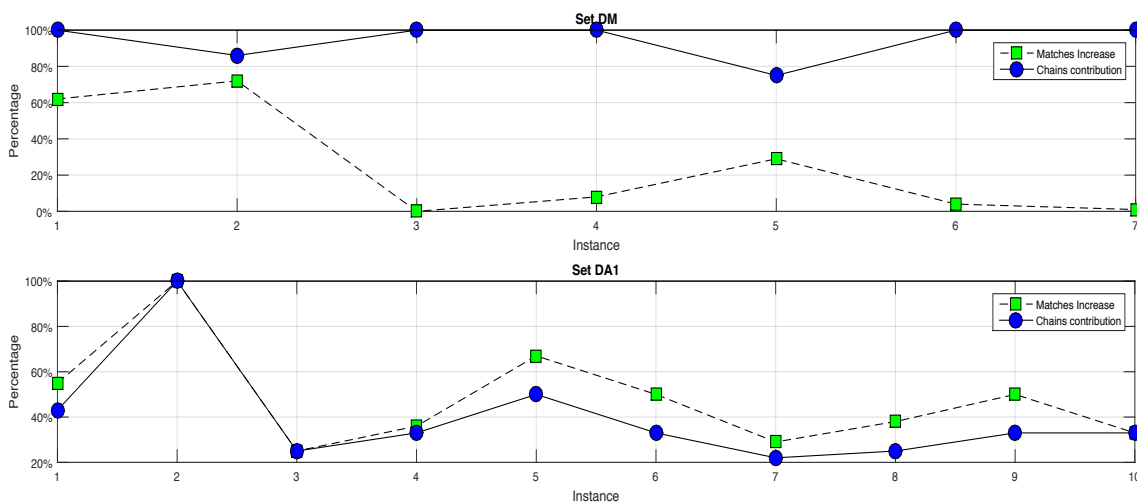
Table 4.12: Size of AA formulation for  $DA1 \cup DA2$  instances.

Figure 4.3: Comparison of number of matches obtained from considering only cycles and considering both cycles and chains.

and affecting kidney-failure patients and living donors beyond the broken link.

In our last experiment, we have assessed the variation in the objective function (the sum of weights) for instances in sets DA1 and DM considering different versions of the KEP seen throughout this thesis. Columns  $k = 2$  and  $k = l = 2$  in [Table 4.13](#),

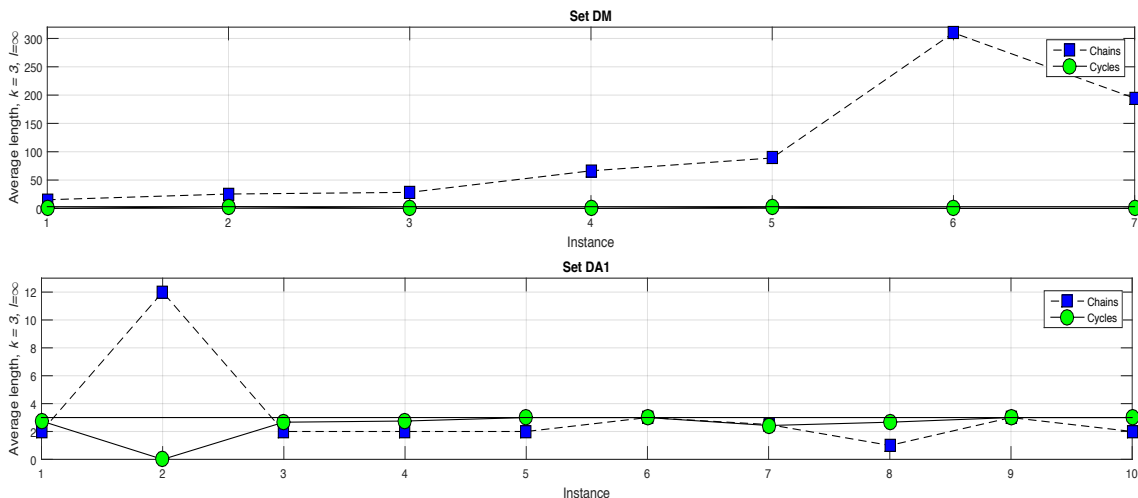


Figure 4.4: Solution composition for the chain-and-cycle variant of the KEP.

represent a maximum matching problem, which can be solved in polynomial time. The models were solved with a time limit of 1,200.00 CPU seconds. The table displays optimal objective function values, or best feasible solution found after the time limit, indicated by a star (\*). It is worth mentioning that although  $k = l$ , the number of real exchanges in the graph arranged in chains are actually  $l - 1$  as we added a dummy edge (with zero weight) from each node to each NDD to treat chains as cycles (see [Chapter 2](#)). Comparing the objective values of columns  $k = 2$  and  $k = l = 2$  to the others, we can see that the introduction of altruist donors (NDDs) gives a significant benefit in the DA1 instances. This benefit is even higher in the DM instances. Additionally, if we keep  $k$  constant and allow chains, then we observe NDDs have a positive impact on the number of transplants conducted. For set DA1, allowing chains of maximum length equal three has roughly the same effect in the objective function than having unbounded chains. On the contrary, for set DM the increase is noticeable when chains can be longer than three. Moreover, instances in set DA1 seem to be more sensitive to  $k$ , i.e., the objective improves as  $k$  gets large. For instances in set DM, however, chains seem to have a heavier effect on the number of exchanges than cycles do.

	Instance	Only Cycles			Cycles and Chains				
		$k = 2$	$k = 3$	$k = 4$	$k = l = 2$	$k = l = 3$	$k = l = 4$	$k = 2, l = \infty$	$k = 3, l = \infty$
DA1	1	6.000	11.005	15.005	9.110	14.115	18.115	9.110	14.115
	2	4.000	6.005	8.015	5.005	7.010	9.020	11.020	11.020
	3	4.000	8.010	10.015	5.000	9.010	11.015	5.000	9.010
	4	4.000	11.010	13.005	7.000	14.010	16.005	7.000	14.010
	5	4.000	6.005	8.500	7.000	9.005	11.500	7.000	9.005
	6	4.000	6.205	7.305	5.005	9.205	10.305	7.000	9.205
	7	12.005	17.005	21.015	14.015	20.025	24.035	15.025*	20.025*
	8	6.005	8.100	10.100	7.010	10.105	12.105	8.010	10.105
	9	4.100	6.005	8.005	6.100	8.105	10.105	6.200	8.105
	10	5.000	7.005	14.010	8.000	10.005	17.010	8.000	10.005
DM	1	92.000	289.000	426.000	133.000	358.000	497.000	626.000	626.000
	2	181.000	552.000	779.000	228.000	626.000	875.000	1 202.000	1 203.000
	3	627.000	888.000	934.000*	656.000	903.000	947.000	988.000	988.000
	4	279.000	662.000	817.000*	292.000	681.000	838.000	952.000	952.000
	5	429.000	1 338.000	1 849.000*	455.000	1 380.000	1 893.000*	2 354.000	2 354.000
	6	953.000	2 130.000	2 474.000*	961.000	2 140.000	2 514.000*	2 909.000	2 909.000
	7	691.000	1 542.000	1 716.000*	701.000	1 549.000*	1 723.000*	1 929.000	1 929.000

Table 4.13: Comparison of solutions under different policies for cycles and/or chains allowed in the KEP.



## CHAPTER 5

# CONCLUSIONS

---

In this thesis we addressed the Kidney Exchange Problem, also called the Kidney Paired Donation Problem, which arises as an alternative solution for patients with non-living compatible donors. Traditionally, patients in need of a kidney have two sources to obtain one: a living compatible donor or a deceased donor from the waiting list. In almost all countries around the world, deceased donors' kidneys cannot cover the high demand of patients in need, making the wait very long for many of them and compromising their survival chances. The former, although preferable, was limited to relatives and close friends, sometimes incompatible with the intended recipient. A decade ago, Operations Research techniques were used for the first time to solve the underlying optimization problem of a new centralized system of kidney transplantation in the USA. This has been spread out years later in many developed countries around the world, in the form of National Kidney Exchange Programs. These kidney exchange programs have a primary role of gathering a collection of biological incompatible patient-donor pairs and use KEP models for finding the best possible ways of performing the kidney exchanges. Those swaps were initially simultaneously performed in such a way that exchanges formed a cycle, limited by a small number of pairs mainly for logistical issues. It was based on these conditions that the first IP formulations started to emerge. But then, a new element, besides incompatible pairs, came into play in this problem: altruist donors, that is, people

willing to donate a kidney to anybody in need without requiring one in return. The direct impact of this new element was that now chains, in addition to cyclic exchanges, were also possible. The discussion became on how long these chains should be and whether they should be conducted simultaneously or not, giving rise to the current versions of the KEP, most of them addressed in this research.

Particularly, the Edge Formulation and the Cycle Formulation were the two pioneering IP formulations. From the literature review, we observed how the “curse of dimensionality” affected specially the Edge Formulation and other IP formulations based on it. Therefore, we oriented our research to the compatibility KEP graph structure. As a result, we found its close relation with connectivity properties that, to the best of our knowledge, were never applied before to the KEP. Based on this, we proposed two new edge-related formulations when either the graph can be partitioned into several SCCs or in any case, even when the graph is itself a strongly connected component.

In this thesis we assess ten different IP formulations for several variants of the KEP: seven from the existing literature (the Cycle Formulation, the Edge Formulation, the Extended Edge Formulation, the Exponential-sized SPLIT Formulation, the Polynomial-sized SPLIT Formulation, the Anderson’s Arc-based Formulation and the PC-TSP Formulation) and three new formulations proposed in this work (the Partitioned Edge Formulation, the Partitioned and Reduced Edge Formulation and the Reduced Exponential-sized SPLIT Formulation).

The PC-TSP Formulation, which is based on the Cycle Formulation and the cut-set families seen in [Section 3.2.2](#) performed poorly, although the Cycle Formulation by itself showed outstanding performance. For instances in the set AN, we ran out of memory when implementing the Edge Formulation for the largest instances and it took longer to find feasible and optimal solutions than the Partitioned Edge Formulation and the Partitioned and Reduced Edge Formulation. The latter worked better as the size of the instance increased. Overall, edge-based IP formulations

failed to solve instances in set DM, only the Cycle Formulation could do it, finding six optimal solutions out of seven. Additionally, when considering unbounded chains and cycles, the proposed Reduced Exponential-sized SPLIT Formulation performed clearly better than the other formulations, including the original Exponential-sized SPLIT Formulation.

We also observed, when comparing cycle-only formulations (with  $k=2$  and  $k=3$ ) and cycle-and-chain formulations, that the benefit of allowing cycles of size 3 over cycles of size 2 resulted in a gain of twice as much the objective function value (see [Table 4.13](#)). In addition, when we also allow cycles ( $k=2$ ) and chains we generally obtained substantial benefits in terms of weighted matches when chains are unbounded. This suggests that it is very important to set public policies that encourage people the culture of altruistic donation.

## 5.1 SUMMARY OF RESEARCH CONTRIBUTIONS

- We conducted a vast empirical comparison between existing and new IP formulations for several variants of the KEP on data from realistic instances based on the NKR pool and randomly-generated instances, as well.
- We apply graph theory concepts to produce a natural partition of the graph, that allows us to solve separately each subgraph induced by the non-trivial strongly connected components when we are concerned on finding only cycles. In fact, any formulation can be used to solve separately such subproblems. To this end, we presented three new formulations based on the Edge Formulation, one of the pioneering KEP formulations, reducing significantly the number of constraints needed by the initial formulation. Two of the proposed formulations are flexible enough as they do not depend on whether the graph is a single SCC or not to be useful, that is, these formulations can always be applied to any graph. These findings helped us to show that if we model the Edge For-

mulation as we illustrated throughout this thesis, we can frequently replace the exponentially sized set of constraints into a set of significantly smaller size, making the problem more tractable, and allowing the resolution of larger instances.

- We designed an efficient algorithm, the SCC-Based Search Algorithm, to reduce the set of constraints, i.e., length- $k$  paths needed by the Edge Formulation while keeping model equivalence. As we saw previously, the running time of our algorithm is substantially lower than that of finding the full set of length- $k$  paths while providing paths that in more than 70% of the cases led to infeasible cycles, guaranteeing that each cycle is forbidden by such constraints only once.
- We provided insight on how the problem structure can affect IP formulations and conclude in which cases it is recommended the use of one or another.
- We implemented a lazy constraint scheme or recursive algorithm to solve the AA and PC-TSP formulation. Our implementation showed to be very competitive compared to the original version by Anderson et al. [4], even finding an additional optimal solution when applying the AA formulation. Additionally, we provided results on the PC-TSP performance, not found in the literature to the best of our knowledge, when a similar scheme is used to solve that formulation.

## 5.2 FUTURE WORK

For current realistic instances it is possible to provide if not optimal a near-optimal solution regardless the variant studied here of the KEP. However, as the size of instances increases it is noticeable the problem becomes harder to solve and even worst if we are allowed to find both chains and cycles. Therefore, the need of large

scale optimization techniques is broadly justified. To the best of our knowledge the size and complexity of current instances have not exceeded the best exact solution methods in existence; however, some initiatives of multi-hospital kidney exchange programs or even international collaborations may lead to the need of meta-heuristics or specialized large-scale decomposition techniques in the near future.

On the other hand, there is still a huge potential of developing new KEP versions as the needs of every country are very specific. These variants may include considering only chains or multiple objectives. For instance, an associated problem in practice is that patients and donors may live miles away one from each other, making in practice a major logistic problem. Currently, this fact and some related are taken into account within the scoring rubrics used to determine the weight of edges. Recently, some advances on the stochastic version of the KEP have been done in literature [2, 15, 33]. An interesting study is to compare the impact on the number of matches passing from the deterministic version to the stochastic one in terms of matches and to analyze which pairs are critic to get robust solutions.

From the practical standpoint, these models and algorithms can be used in local/nation-wide databases as a tool for implementing similar kidney exchange programs in Mexico.

# APPENDICES

---

## APPENDIX A

# THE SEPARATION PROBLEM

---

In many combinatorial optimization problems there are models with an exponential set of constraints that may yield a stronger IP formulation. Natural, these exponential number of constraints cannot be explicitly written out and feed them to an MILP solver; as it is the case for the Anderson’s Arc-based Formulation, the PC-TSP-based Formulation, and the Edge Formulation, although in the latter we can do so for small values of  $k$ . Instead, we start with a problem relaxation by dropping the “difficult” constraints, and add them only when needed. In order to detect violated constraints and add them iteratively to the model we need to solve a separation problem [22].

Consider the following problem OP: Maximize  $\{c^T x \mid x \in X \subseteq \mathbb{R}^n\}$

The separation problem associated with OP is, given  $\hat{x} \in \mathbb{R}^n$  decide if  $\hat{x} \in \text{conv}(X)$ , and if not, find an inequality  $\pi^T x \leq \pi_0$  satisfied by all points in  $X$ , but violated by  $\hat{x} : (\pi^T \hat{x} \geq \pi_0)$ .

Depending on whether  $\hat{x}$  is fractional or integer, there are two methods in integer programming algorithms to add such violated constraints [5, 23]: *lazy constraints* and *user cuts*.

- *Lazy constraints*: When a model has a large number of constraints, sometimes

it is not possible to enumerate them all explicitly. One strategy to deal with a large set of constraints is to solve initially a relaxed model leaving out those constraints and then add them to the model only when they are violated. The augmented model is resolved and the process continues until no violated constraints remain. Those constraints added iteratively are called lazy constraints. They represent a subset of constraints of the original model. Thus, some of such constraints or all of them, in the worst case, are needed in order to keep model correctness. Unlike cuts, which we review below, lazy constraints only rule out unfeasible integer solutions for the original problem. Once the augmented model, whose solution is integer, has no more violations of lazy constraints, that solution is optimal for the original model.

- *Cuts*: These are valid inequalities, that is, constraints that are satisfied by the feasible integer solutions in a full model but violated by an infeasible point in the solution space, so that we can eliminate that solution while guaranteeing optimality. Cuts are not part of the original set of constraints, and therefore they do not compromise model correctness. However, they do strengthen the formulation and as a result speed up the convergence to optimal solutions of the branch-and-bound algorithm. Moreover, cuts can eliminate solutions at nodes in the branch-and-bound tree regardless of the values taken by the variables: fractional or integer. There exists general cuts, those which can be added to the entire model and local cuts, those which are only applicable to tree nodes.

To get optimal solutions, Anderson et al. [4] proposed an iterative procedure that uses branch and bound and a lazy constraint generation scheme for the Anderson's Arc-based Formulation and, a branch-and-cut approach for the PC-TSP-based Formulation. In this thesis we implemented our version of that lazy constraint generation scheme within a branch-and-bound framework in order to solve the above mentioned formulations and for the remaining ones we used the branch-and-bound method as usual, so that we can compare all the approaches under the same conditions. Then, the idea of the lazy constraint generation scheme is as follows: By



relaxing constraints (3.6) and (3.13), we obtain a problem that can be solved by branch and bound as normal. The solution to this relaxed problem is checked for cycles of size greater than  $k$  when dealing with Anderson's Arc-based Formulation and for any single cycle for the PC-TSP-based Formulation. Then, a separation problem is solved. Anderson's implementation takes  $\mathcal{O}(|V|)$  when the input given is the edges involved in the solution or  $\mathcal{O}(|V| + |E|)$  when it is necessary to inquire all edges in the graph to see which ones are part of the solution and then build an adjacency list in order to obtain the violated constraints. This is justified since in a solution every node has indegree and outdegree of at most one, triggering the violated constraint to be added as the scheme goes on. In our case this lazy-constraint scheme takes  $\mathcal{O}(|E|)$  because our data structure is shaped as an adjacency list, thus, as we query the variable values (zero or one), we build the violated constraints. Once our procedure finds a unit weight, it goes directly to the adjacent-node row in the list to continue checking the values until unfeasible cycles, if any, are detected. Hence, our implementation is more efficient when a query on the variables is required, as it is the case when optimizing a model. In order to obtain consistent times with Anderson's we also implemented a Lazy Constraint Callback Function within CPLEX to add lazy constraints while the problem is being solved, without restarting the branch and bound.

## APPENDIX B

# DESCRIPTION OF DATA SETS

---

The following notation was used to characterize the instances:

- PDPs: the number of patient-donor pairs in the graph.
- NDDs: the number of altruists or bridge donors in the graph.
- Edges and EdgesPDP: the number of edges in the original graph and the number of edges in the graph in absence of NDDs, respectively.
- $w^{\min}$ : the minimum value assigned to arc weights.
- $w^{\max}$ : the maximum value assigned to arc weights.
- File name: name given to instances by the authors who generated them.
- DM1: corresponds to the set formed by 7 instances provided by Mak-Hau [28].
- DA1: This set is made up of those instances that Anderson et al. [4] found difficult and reported in their paper.
- DA2: These instances were not reported in [4] but are also part of the instances generated by their simulations.

Data set	Instance	File name	$w^{\min}$	$w^{\max}$	PDPs	NDDs	Edges
DM	1	Clin152-10-ME.dat	1.00	5.00	152	10	897
	2	CLIN156-10.dat	1.00	10.00	156	10	9993
	3	Clin198-7.dat	1.00	5.00	198	7	4927
	4	Clin199-3.dat	1.00	5.00	199	3	2382
	5	Clin269-3.dat	1.00	10.00	269	3	2607
	6	Clin310-1.dat	1.00	10.00	310	1	4776
	7	Clin389-2.dat	1.00	5.00	389	2	7565
DA1	1	BinputData198_7.csv	1.00	2.00	198	7	4882
	2	BinputData202_3.csv	1.00	1.05	202	3	4706
	3	BinputData215_6.csv	1.00	1.05	215	6	6145
	4	BinputData261_6.csv	1.00	2.00	261	6	8915
	5	BinputData263_6.csv	1.00	2.00	263	6	8939
	6	BinputData284_5.csv	1.00	2.00	284	5	10126
	7	BinputData312_6.csv	1.00	1.05	312	6	13045
	8	BinputData324_6.csv	1.00	2.00	324	6	13175
	9	BinputData328_6.csv	1.00	2.00	328	6	13711
	10	BinputData330_6.csv	1.00	3.00	330	6	13399
	1	r10p114t0e1785time196.0.csv	0.10	2.00	114	10	1785
	2	r13p115t0e1519time224.0.csv	1.01	2.00	115	13	1519
	3	r13p120t0e1744time224.0.csv	1.01	2.00	120	13	1744
	4	r17p155t0e2629time364.0.csv	1.01	2.00	155	17	2629
	5	r17p175t0e5608time343.0.csv	1.10	2.00	175	17	5608
	6	r21p191t0e3965time490.0.csv	1.01	2.00	191	21	3965
	7	r22p216t0e6817time532.0.csv	0.15	2.00	216	22	6817
	8	r22p221t0e7184time539.0.csv	0.15	2.00	221	22	7184
	9	r22p224t0e7365time546.0.csv	0.15	2.00	224	22	7365
	10	r22p230t0e7687time553.0.csv	0.15	2.00	230	22	7687
	11	r23p230t0e7592time560.0.csv	0.15	2.00	230	23	7592
	12	r23p236t0e8015time567.0.csv	0.15	2.00	236	23	8015
	13	r23p241t0e8595time574.0.csv	1.10	2.00	241	23	8595
	14	r23p246t0e8967time581.0.csv	1.10	2.00	246	23	8967
	15	r23p248t0e9104time609.0.csv	0.15	2.00	248	23	9104
	16	r23p254t0e9547time616.0.csv	0.15	2.00	254	23	9547
	17	r27p274t0e10127time707.0.csv	0.10	2.00	274	27	10127
	18	r27p285t0e9006time784.0.csv	1.01	2.00	285	27	9006
	19	r27p289t0e9771time784.0.csv	1.01	6.00	289	27	9771
	20	r34p230t0e6015time672.0.csv	1.01	2.00	230	34	6015
	21	r34p231t0e6309time672.0.csv	1.01	2.00	231	34	6309
	22	r34p232t0e6125time672.0.csv	1.01	2.00	232	34	6125
	23	r34p232t0e6138time672.0.csv	1.01	2.00	232	34	6138
	24	r34p233t0e6414time672.0.csv	1.01	2.00	233	34	6414
	25	r34p304t0e12925time756.0.csv	0.10	2.00	304	34	12925
	26	r34p310t0e13539time763.0.csv	0.10	2.00	310	34	13539

Continued on next page.

Data set	Instance	File name	Lowest $w_{ij}$	Greatest $w_{ij}$	PDPs	NDDs	Edges
	27	r34p334t0e18109time763.0.csv	1.10	2.00	334	34	18 109
	28	r35p305t0e10941time840.0.csv	1.01	2.00	305	35	10 941
	29	r35p312t0e14001time812.0.csv	0.10	2.00	312	35	14 001
	30	r35p317t0e13895time791.0.csv	0.10	2.00	317	35	13 895
	31	r35p317t0e14600time812.0.csv	0.15	2.00	317	35	14 600
	32	r35p326t0e16889time812.0.csv	1.10	2.00	326	35	16 889
	33	r36p300t0e8683time868.0.csv	1.01	2.00	300	36	8 683
	34	r39p285t0e8039time903.0.csv	1.01	2.00	285	39	8 039
	35	r39p291t0e8359time903.0.csv	1.01	2.00	291	39	8 359
	36	r39p367t0e18615time938.0.csv	0.10	2.00	367	39	18 615
	37	r43p364t0e14665time1015.0.csv	1.01	2.00	364	43	14 665
	38	r43p365t0e14772time1015.0.csv	1.01	2.00	365	43	14 772
	39	r45p342t0e11474time1050.0.csv	1.01	2.00	342	45	11 474
	40	r46p339t0e12528time1001.0.csv	1.00	1.00	339	46	12 528
	41	r46p341t0e12619time1001.0.csv	1.01	2.00	341	46	12 619
	42	r46p342t0e12932time1001.0.csv	1.01	2.00	342	46	12 932
	43	r46p343t0e12632time1001.0.csv	1.00	1.01	343	46	12 632
	44	r46p343t0e13072time1001.0.csv	1.01	2.00	343	46	13 072
	45	r46p345t0e12827time1001.0.csv	1.00	1.01	345	46	12 827
	46	r46p346t0e13498time1001.0.csv	1.01	2.00	346	46	13 498
	47	r46p346t0e13835time1001.0.csv	1.01	6.00	346	46	13 835
	48	r46p347t0e12682time1001.0.csv	1.00	1.01	347	46	12 682
	49	r46p347t0e13057time1001.0.csv	1.00	1.01	347	46	13 057
	50	r46p347t0e13209time1001.0.csv	1.01	6.00	347	46	13 209
	51	r46p348t0e13189time1001.0.csv	1.00	1.01	348	46	13 189
	52	r46p348t0e13201time1008.0.csv	1.01	2.00	348	46	13 201
	53	r46p348t0e13346time1008.0.csv	1.01	2.00	348	46	13 346
	54	r46p348t0e13484time1001.0.csv	1.01	6.00	348	46	13 484
	55	r46p348t0e13490time1001.0.csv	1.01	6.00	348	46	13 490
	56	r46p349t0e12545time1001.0.csv	1.00	1.00	349	46	12 545
	57	r46p349t0e13269time1001.0.csv	1.01	2.00	349	46	13 269
	58	r46p350t0e13112time1001.0.csv	1.00	1.01	350	46	13 112
	59	r46p350t0e13614time1001.0.csv	1.01	6.00	350	46	13 614
	60	r46p366t0e15573time1036.0.csv	1.01	2.00	366	46	15 573
	61	r47p379t0e15206time1071.0.csv	1.01	2.00	379	47	15 206
	62	r47p379t0e15246time1071.0.csv	1.01	2.00	379	47	15 246
	63	r48p365t0e14014time1092.0.csv	1.01	2.00	365	48	14 014
	64	r48p365t0e14133time1092.0.csv	1.01	2.00	365	48	14 133
	65	r48p366t0e14295time1092.0.csv	1.01	2.00	366	48	14 295
	66	r48p368t0e13651time1089.0.csv	1.01	2.00	368	48	13 651
	67	r48p374t0e15390time1092.0.csv	1.01	2.00	374	48	15 390
	68	r48p377t0e15918time1092.0.csv	1.01	2.00	377	48	15 918
	69	r49p362t0e13055time1113.0.csv	1.00	1.01	362	49	13 055
	70	r49p414t0e22616time1092.0.csv	0.10	2.00	414	49	22 616

Continued on next page.

Data set	Instance	File name	Lowest $w_{ij}$	Greatest $w_{ij}$	PDPs	NDDs	Edges
	71	r49p434t0e27100time1092.0.csv	0.15	2.00	434	49	27 100
	72	r49p457t0e32295time1092.0.csv	1.10	2.00	457	49	32 295
	73	r50p371t0e13515time1127.0.csv	1.01	2.00	371	50	13 515
	74	r50p386t0e14856time1127.0.csv	1.00	1.01	386	50	14 856
	75	r50p387t0e15771time1134.0.csv	1.01	2.00	387	50	15 771
	76	r50p474t0e34772time1127.0.csv	1.10	2.00	474	50	34 772

Table B.1: Full set of original instances.

Table B.2 shows instances in Table B.1 ordered by increasing number of PDPs and allocated within subsets named PDPsR. Each of them contains 10 instances, except the last one with 6 instances.

PDPsR	Data set	Instance	File name
114-216	DA2	1	r10p114t0e1785time196.0.csv
	DA2	2	r13p115t0e1519time224.0.csv
	DA2	3	r13p120t0e1744time224.0.csv
	DA2	4	r17p155t0e2629time364.0.csv
	DA2	5	r17p175t0e5608time343.0.csv
	DA2	6	r21p191t0e3965time490.0.csv
	DA1	1	BinputData198_7.csv
	DA1	2	BinputData202_3.csv
	DA1	3	BinputData215_6.csv
	DA2	7	r22p216t0e6817time532.0.csv
221-236	DA2	8	r22p221t0e7184time539.0.csv
	DA2	9	r22p224t0e7365time546.0.csv
	DA2	10	r22p230t0e7687time553.0.csv
	DA2	11	r23p230t0e7592time560.0.csv
	DA2	20	r34p230t0e6015time672.0.csv
	DA2	21	r34p231t0e6309time672.0.csv
	DA2	22	r34p232t0e6125time672.0.csv
	DA2	23	r34p232t0e6138time672.0.csv
	DA2	24	r34p233t0e6414time672.0.csv
	DA2	12	r23p236t0e8015time567.0.csv
241-285	DA2	13	r23p241t0e8595time574.0.csv
	DA2	14	r23p246t0e8967time581.0.csv
	DA2	15	r23p248t0e9104time609.0.csv
	DA2	16	r23p254t0e9547time616.0.csv
	DA1	4	BinputData261_6.csv
	DA1	5	BinputData263_6.csv
	DA2	17	r27p274t0e10127time707.0.csv

Continued on next page

PDPsR	Data Set	Instance	File name
	DA1	6	BinputData284_5.csv
	DA2	18	r27p285t0e9006time784.0.csv
	DA2	34	r39p285t0e8039time903.0.csv
289-317	DA2	19	r27p289t0e9771time784.0.csv
	DA2	35	r39p291t0e8359time903.0.csv
	DA2	33	r36p300t0e8683time868.0.csv
	DA2	25	r34p304t0e12925time756.0.csv
	DA2	28	r35p305t0e10941time840.0.csv
	DA2	26	r34p310t0e13539time763.0.csv
	DA1	7	BinputData312_6.csv
	DA2	29	r35p312t0e14001time812.0.csv
	DA2	30	r35p317t0e13895time791.0.csv
	DA2	31	r35p317t0e14600time812.0.csv
324-343	DA1	8	BinputData324_6.csv
	DA2	32	r35p326t0e16889time812.0.csv
	DA1	9	BinputData328_6.csv
	DA1	10	BinputData330_6.csv
	DA2	27	r34p334t0e18109time763.0.csv
	DA2	40	r46p339t0e12528time1001.0.csv
	DA2	41	r46p341t0e12619time1001.0.csv
	DA2	39	r45p342t0e11474time1050.0.csv
	DA2	42	r46p342t0e12932time1001.0.csv
343-348	DA2	44	r46p343t0e13072time1001.0.csv
	DA2	45	r46p345t0e12827time1001.0.csv
	DA2	46	r46p346t0e13498time1001.0.csv
	DA2	47	r46p346t0e13835time1001.0.csv
	DA2	48	r46p347t0e12682time1001.0.csv
	DA2	49	r46p347t0e13057time1001.0.csv
	DA2	50	r46p347t0e13209time1001.0.csv
	DA2	51	r46p348t0e13189time1001.0.csv
	DA2	52	r46p348t0e13201time1008.0.csv
	DA2	53	r46p348t0e13346time1008.0.csv
348-365	DA2	54	r46p348t0e13484time1001.0.csv
	DA2	55	r46p348t0e13490time1001.0.csv
	DA2	56	r46p349t0e12545time1001.0.csv
	DA2	57	r46p349t0e13269time1001.0.csv
	DA2	58	r46p350t0e13112time1001.0.csv
	DA2	59	r46p350t0e13614time1001.0.csv
	DA2	69	r49p362t0e13055time1113.0.csv
	DA2	37	r43p364t0e14665time1015.0.csv
	DA2	38	r43p365t0e14772time1015.0.csv
	DA2	63	r48p365t0e14014time1092.0.csv
	DA2	64	r48p365t0e14133time1092.0.csv

Continued on next page

PDPsR	Data Set	Instance	File name
	DA2	60	r46p366t0e15573time1036.0.csv
	DA2	65	r48p366t0e14295time1092.0.csv
	DA2	36	r39p367t0e18615time938.0.csv
	DA2	66	r48p368t0e13651time1089.0.csv
	DA2	73	r50p371t0e13515time1127.0.csv
	DA2	67	r48p374t0e15390time1092.0.csv
	DA2	68	r48p377t0e15918time1092.0.csv
	DA2	61	r47p379t0e15206time1071.0.csv
	DA2	62	r47p379t0e15246time1071.0.csv
	DA2	74	r50p386t0e14856time1127.0.csv
	DA2	75	r50p387t0e15771time1134.0.csv
386-474	DA2	70	r49p414t0e22616time1092.0.csv
	DA2	71	r49p434t0e27100time1092.0.csv
	DA2	72	r49p457t0e32295time1092.0.csv
	DA2	76	r50p474t0e34772time1127.0.csv

Table B.2: Codification of instances into sets PDPsR

## APPENDIX C

# REDUCED INSTANCES FOR THE CYCLE VARIANT KEP FORMULATIONS

---

The following instances are those from [Table B.1](#) but considering that the graphs contain PDPs only. These instances were used to test the cycle variant IP formulations.

The following notation is used to characterize the instances:

- EdgesPDP: Number of remaining edges in the graph after removing NDDs.
- EdgesPDP(p): Number of edges in all the non-trivial SCCs, in which the original graph could be partitioned.
- PDP(p): number of pair nodes in all the non-trivial SCCs, in which the original graph could be partitioned.
- nSCC: number of SCCs in which the original graph could be partitioned.

Data set	Instance	File name	PDPs	EdgesPDP	nSCC	PDP(p)	EdgesPDP(p)
	1	Clin152-10-ME.dat	152	897	1	152	897
	2	CLIN156-10.dat	156	954	1	154	948
	3	Clin198-7.dat	198	4740	1	198	4740
DM	4	Clin199-3.dat	199	2341	1	199	2341

Continued on next page



APPENDIX C. REDUCED INSTANCES FOR THE CYCLE VARIANT KEP FORMULATIONS 78

Data Set	Instance	File Name	PDPs	EdgesPDP	nSCC	PDP(p)	EdgesPDP(p)
	5	Clin269-3.dat	269	2 583	1	269	2 583
	6	Clin310-1.dat	310	4 760	1	310	4 760
	7	Clin389-2.dat	389	7 535	1	389	7 535
DA1	1	BinputData198_7.csv	198	4 874	1	67	474
	2	BinputData202_3.csv	202	4 704	1	55	266
	3	BinputData215_6.csv	215	6 137	1	48	183
	4	BinputData261_6.csv	261	8 902	1	60	437
	5	BinputData263_6.csv	263	8 926	1	55	286
	6	BinputData284_5.csv	284	10 113	1	68	547
	7	BinputData312_6.csv	312	13 041	1	95	791
	8	BinputData324_6.csv	324	13 172	1	65	311
	9	BinputData328_6.csv	328	13 692	1	75	393
	10	BinputData330_6.csv	330	13 381	1	63	583
DA2	1	r10p114t0e1785time196.0.csv	114	1 779	2	42	358
	2	r13p115t0e1519time224.0.csv	115	1 500	1	58	356
	3	r13p120t0e1744time224.0.csv	120	1 722	1	63	489
	4	r17p155t0e2629time364.0.csv	155	2 608	1	87	678
	5	r17p175t0e5608time343.0.csv	175	5 538	1	121	3 036
	6	r21p191t0e3965time490.0.csv	191	3 945	1	81	335
	7	r22p216t0e6817time532.0.csv	216	6 784	2	49	440
	8	r22p221t0e7184time539.0.csv	221	7 143	2	56	482
	9	r22p224t0e7365time546.0.csv	224	7 324	2	58	534
	10	r22p230t0e7687time553.0.csv	230	7 644	2	62	599
	11	r23p230t0e7592time560.0.csv	230	7 545	2	59	552
	12	r23p236t0e8015time567.0.csv	236	7 968	2	66	697
	13	r23p241t0e8595time574.0.csv	241	8 536	2	61	605
	14	r23p246t0e8967time581.0.csv	246	8 908	2	62	612
	15	r23p248t0e9104time609.0.csv	248	9 054	2	59	482
	16	r23p254t0e9547time616.0.csv	254	9 497	2	61	503
	17	r27p274t0e10127time707.0.csv	274	10 038	2	60	539
	18	r27p285t0e9006time784.0.csv	285	9 006	1	40	330
	19	r27p289t0e9771time784.0.csv	289	9 771	1	45	448
	20	r34p230t0e6015time672.0.csv	230	5 996	1	98	537
	21	r34p231t0e6309time672.0.csv	231	6 289	1	107	533
	22	r34p232t0e6125time672.0.csv	232	6 107	1	111	469
	23	r34p232t0e6138time672.0.csv	232	6 119	1	99	475
	24	r34p233t0e6414time672.0.csv	233	6 396	1	102	567
	25	r34p304t0e12925time756.0.csv	304	12 852	2	37	156
	26	r34p310t0e13539time763.0.csv	310	13 466	2	39	173
	27	r34p334t0e18109time763.0.csv	334	18 013	1	237	9 311
	28	r35p305t0e10941time840.0.csv	305	10 809	1	101	791
	29	r35p312t0e14001time812.0.csv	312	13 846	2	57	417
	30	r35p317t0e13895time791.0.csv	317	13 815	2	73	874
	31	r35p317t0e14600time812.0.csv	317	14 484	2	60	492

Continued on next page

APPENDIX C. REDUCED INSTANCES FOR THE CYCLE VARIANT KEP FORMULATIONS 79

Data Set	Instance	File Name	PDPs	EdgesPDP	nSCC	PDP(p)	EdgesPDP(p)
	32	r35p326t0e16889time812.0.csv	326	16 752	2	85	1 098
	33	r36p300t0e8683time868.0.csv	300	8 647	1	100	811
	34	r39p285t0e8039time903.0.csv	285	8 021	2	28	124
	35	r39p291t0e8359time903.0.csv	291	8 342	1	150	974
	36	r39p367t0e18615time938.0.csv	367	18 488	2	97	1 493
	37	r43p364t0e14665time1015.0.csv	364	14 619	1	170	675
	38	r43p365t0e14772time1015.0.csv	365	14 728	1	162	582
	39	r45p342t0e11474time1050.0.csv	342	11 466	1	60	357
	40	r46p339t0e12528time1001.0.csv	339	12 528	1	44	340
	41	r46p341t0e12619time1001.0.csv	341	12 619	1	48	372
	42	r46p342t0e12932time1001.0.csv	342	12 932	1	53	538
	43	r46p343t0e12632time1001.0.csv	343	12 632	1	50	466
	44	r46p343t0e13072time1001.0.csv	343	13 072	1	54	530
	45	r46p345t0e12827time1001.0.csv	345	12 827	1	57	532
	46	r46p346t0e13498time1001.0.csv	346	13 498	1	55	584
	47	r46p346t0e13835time1001.0.csv	346	13 835	1	54	578
	48	r46p347t0e12682time1001.0.csv	347	12 682	1	51	472
	49	r46p347t0e13057time1001.0.csv	347	13 057	1	52	502
	50	r46p347t0e13209time1001.0.csv	347	13 209	1	53	522
	51	r46p348t0e13189time1001.0.csv	348	13 189	1	51	487
	52	r46p348t0e13201time1008.0.csv	348	13 200	1	49	391
	53	r46p348t0e13346time1008.0.csv	348	13 346	1	53	538
	54	r46p348t0e13484time1001.0.csv	348	13 484	1	54	527
	55	r46p348t0e13490time1001.0.csv	348	13 490	1	54	559
	56	r46p349t0e12545time1001.0.csv	349	12 545	1	48	392
	57	r46p349t0e13269time1001.0.csv	349	13 269	1	55	560
	58	r46p350t0e13112time1001.0.csv	350	13 112	1	53	532
	59	r46p350t0e13614time1001.0.csv	350	13 614	1	56	595
	60	r46p366t0e15573time1036.0.csv	366	15 565	2	53	249
	61	r47p379t0e15206time1071.0.csv	379	15 199	1	28	191
	62	r47p379t0e15246time1071.0.csv	379	15 239	1	67	676
	63	r48p365t0e14014time1092.0.csv	365	14 006	1	106	1 106
	64	r48p365t0e14133time1092.0.csv	365	14 124	1	107	1 103
	65	r48p366t0e14295time1092.0.csv	366	14 287	1	108	1 146
	66	r48p368t0e13651time1089.0.csv	368	13 651	1	53	446
	67	r48p374t0e15390time1092.0.csv	374	15 382	1	113	1 270
	68	r48p377t0e15918time1092.0.csv	377	15 903	1	116	1 481
	69	r49p362t0e13055time1113.0.csv	362	13 045	1	40	351
	70	r49p414t0e22616time1092.0.csv	414	22 394	2	54	407
	71	r49p434t0e27100time1092.0.csv	434	26 879	2	75	925
	72	r49p457t0e32295time1092.0.csv	457	32 086	1	330	17 055
	73	r50p371t0e13515time1127.0.csv	371	13 515	1	39	260
	74	r50p386t0e14856time1127.0.csv	386	14 856	1	43	329
	75	r50p387t0e15771time1134.0.csv	387	15 764	1	64	513

Continued on next page

Data Set	Instance	File Name	PDPs	EdgesPDP	nSCC	PDP(p)	EdgesPDP(p)
	76	r50p474t0e34772time1127.0.csv	474	34 537	2	161	5 315

Table C.1: Instances for the cycle variant: Reduction on the input size.

## APPENDIX D

# COMPATIBILITY EVALUATION

---

To find out if a patient and a potential donor are a kidney match, there are three main blood tests that are typically applied. These are blood typing, HLA typing, and cross-matching. The following information was taken from [10, 32, 44, 45, 46].

### D.1 BLOOD TYPING (ABO COMPATIBILITY)

Blood typing is the first blood test that determines if the potential donor's blood is a compatible match with the patient's blood. This test measures blood antibodies that react with different blood groups. There are 4 different blood types. The most common blood type in the population is type O. The next most common is blood type A, then B, and the rarest is blood type AB. The blood type of the donor must be compatible with the recipient. The rules for blood type in transplantation are the same as they are for blood transfusion. Some blood types can give to others and some may not. Blood type O is considered the universal donor. People with blood type O can give to any other blood type. Blood type AB is called the universal recipient because they can receive an organ or blood from people with any blood type. The Rh factor (+ or -) of blood does not matter. [Table D.1](#) depicts which blood types are compatible.

Recipient's Blood Type	Donor's Blood Type
O	O
A	A or O
B	B or O
AB	A, B, AB or O

Table D.1: Blood type compatibility chart: O is the universal donor and AB is the universal recipient.

If the donor's blood type works with the patient blood type, the donor will take the next blood test (tissue typing).

## D.2 HLA TYPING

HLA typing is also called "tissue typing". HLA stands for human leukocyte antigen. The first blood test is to determine the tissue (HLA) type of the patient and the potential donor to see how well they match. This test identifies certain proteins in a patient's blood called antigens. Antigens are markers on the cells in people body, which help the body, tell the difference between self and non-self. This allows the body to protect itself by recognizing and attacking something that does not belong to it such as bacteria or viruses.

The body also sees antigens on a transplanted organ that are different from its own and it sends white blood cells to attack the organ. When the body attacks the new organ, it is rejecting it. Although there are many different antigens, there are six, which have been identified as having an important role in transplantation. They are the A, B, and DR antigens. There are two antigens for each letter and they are identified by numbers. Thus, a HLA type might look something like this:

A2, A30 B8, B70 DR3, DR8

People inherit these from their parents, three (A, B, and DR) from their mother

and three (A, B, and DR) from their father. Children born of the same parents may inherit the same combination or a different combination of antigens. If someone has brothers or sisters, there is a 25% chance that she will have inherited the same six antigens as one of them, a 50% chance of having three of the same antigens and a 25% chance of having none of the same antigens. Except for identical twins and some brothers and sisters, it is very rare to get an exact match between two people, especially if they are unrelated. The chance of finding an exact match with an unrelated donor is about one in 100,000. Although transplantation centers try to match antigens as much as possible for kidney and pancreas recipients, they do transplant organs into recipients who have no antigens in common, and these patients do very well.

The second blood test measures antibodies to HLA; this test is done for the patient only and is repeated frequently (sometimes monthly but less often dependent upon the transplant program policy). HLA antibodies can be harmful to the transplanted organ and they can increase or decrease over time so they must be measured while waiting for a transplant, immediately before a transplant surgery, and sometimes following transplantation. If a patient has HLA antibodies in their blood, they are considered HLA “sensitized” and it is best to find a donor with HLA types that avoid the HLA antibodies in the patient’s blood. Importantly, HLA antibody levels can change following events such as blood transfusions, miscarriages, minor surgeries (including dental work or fistula replacement) or severe infections.

In order to determine whether or not a patient already has any specific HLA antibodies, a lab specialist will test a patient’s blood (serum) against lymphocytes (white blood cells) obtained from a panel of about 100 blood donors. These 100 donors represent the potential HLA makeup for a donor from that area. Percent PRA (%PRA) is the number of reactions within that panel. PRA stands for Panel Reactive Antibodies. If a candidate’s serum does not react with any of the donor samples, the candidate is not sensitized and has a PRA of 0. If a candidate’s serum reacts in 80 out of 100 samples, the patient has a PRA of 80%. Theoretically, that

means that if a donor becomes available from that donor pool, the recipient would experience acute rejection 8 out of 10 times. That patient might have to wait a very long time until a compatible donor becomes available. As mentioned above, HLA antibodies can vary over time and so the %PRA can also change.

### D.3 CROSS-MATCHING

A serum cross-match is a blood test applied several times to both a patient and the donor, including right before the transplant surgery. To do the test, cells from the donor are mixed with patient's serum. If patient's serum has antibodies against the donor's cells, the antibodies will bind the donor cells and be detected using a fluorescent detection method. If these antibodies are at high levels, the donor cells will be destroyed. This is called a positive cross-match and it means that the transplant cannot take place. To do so would result in immediate rejection of the transplanted kidney.

# BIBLIOGRAPHY

---

- [1] D. J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 295–304, San Diego, USA, June 2007.
- [2] F. Alvelos. Maximizing expected number of transplants in kidney exchange programs. *Electronic Notes in Discrete Mathematics*, 52:269–276, 2016.
- [3] R. Anderson, I. Ashlagi, D. Gamarnik, M. Rees, A. E. Roth, T. Sönmez, and M. Utku Ünver. Kidney exchange and the alliance for paired donation: Operations research changes the way kidneys are transplanted. *Interfaces*, 45(1):26–42, 2015.
- [4] R. Anderson, I. Ashlagi, D. Gamarnik, and A. E. Roth. Finding long chains in kidney exchange using the traveling salesman problem. *Proceedings of the National Academy of Sciences*, 112(3):663–668, 2015.
- [5] R. M. Anderson. *Stochastic Models and Data Driven Simulations for Healthcare Operations*. PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 2014.
- [6] I. Ashlagi, D. S. Gilchrist, A. E. Roth, and M. A. Rees. Nonsimultaneous chains and dominos in kidney-paired donation – Revisited. *American Journal of Transplantation*, 11(5):984–994, 2011.



- 
- [7] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows*. Wiley, New York, USA, 2009.
- [8] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Mathematical Programming*, 59(1):413–420, 1993.
- [9] P. Biro, D. F. Manlove, and R. Rizzi. Maximum weight cycle packing in directed graphs, with application to kidney exchange programs. *Discrete Mathematics Algorithms and Applications*, 1(4):499–517, 2009.
- [10] J. M. Blumberg, H. A. Gritsch, E. F. Reed, J. M. Cecka, G. S. Lipshutz, G. M. Danovitch, S. McGuire, D. W. Gjertson, and J. L. Veale. Kidney paired donation in the presence of donor-specific antibodies. *Kidney International*, 84(5):1009–1016, 2013.
- [11] M. Constantino, X. Klimentova, A. Viana, and A. Rais. New insights on integer-programming models for the kidney exchange problem. *European Journal of Operational Research*, 231(1):57–68, 2013.
- [12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, USA, third edition, 2009.
- [13] M. Desrochers. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1991.
- [14] J. P. Dickerson, D. F. Manlove, B. Plaut, T. Sandholm, and J. Trimble. Position-indexed formulations for kidney exchange. In *Proceedings of the 2016 ACM Conference on Economics and Computation (EC’16)*, pages 25–42, ACM, New York, USA, 2016.
- [15] J. P. Dickerson, A. D. Procaccia, and T. Sandholm. Failure-aware kidney exchange. In *Proceedings of the 14th ACM Conference on Electronic Commerce (EC’13)*, pages 323–340, ACM, New York, USA, 2013.

- 
- [16] J.P. Dickerson, A. D. Procaccia, and T. Sandholm. Optimizing kidney exchange with transplant chains: Theory and reality. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12), Volume 2*, pages 711–718, International Foundation for Autonomous Agents and Multiagent Systems, Richland, USA, 2012.
- [17] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
- [18] H. N. Gabow. Path-based depth-first search for strong and biconnected components. *Information Processing Letters*, 74(3–4):107–114, 2000.
- [19] S. E. Gentry, R. A. Montgomery, B. J. Swihart, and D. L. Segev. The roles of dominos and nonsimultaneous chains in kidney paired donation. *American Journal of Transplantation*, 9(6):1330–1336, 2009.
- [20] K. M. Glorie, J. J. van de Klundert, and A. P. M. Wagelmans. Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manufacturing & Service Operations Management*, 16(4):498–512, 2014.
- [21] M. X. Goemans. Combining approximation algorithms for the prize-collecting TSP. arXiv:0910.0553 [cs.DS], October 2009.
- [22] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.
- [23] IBM. Concert technology version 12.1 C++ API reference manual. URL: <ftp://ftp.software.ibm.com/software/websphere/ilog/docs/optimization/cplex/refcpcplex.pdf>, 2009.
- [24] D. B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.

- [25] X. Klimentova, F. Alvelos, and A. Viana. A new branch-and-price approach for the kidney exchange problem. In B. Murgante, S. Misra, A. M. A. C. Rocha, C. Torre, J. G. Rocha, M. I. Falcão, D. Taniar, B. O. Apduhan, and O. Gervasi, editors, *Computational Science and Its Applications – ICCSA 2014, Part II*, volume 8580 of *Lecture Notes in Computer Science*, pages 237–252, Cham, Switzerland, 2014. Springer.
- [26] G. Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992.
- [27] R. Leishman, D. Stewart, A. Kucheryavaya, L. Robbins, T. Sandholm, and M. Aeder. Reasons for match offer refusals and efforts to reduce them in the OPTN/UNOS kidney paired donation pilot program (KPDPP). Presented at the American Transplant Congress, Philadelphia, USA, May 2015.
- [28] 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.
- [29] S. Malik and E. Cole. Foundations and principles of the Canadian living donor paired exchange program. *Canadian Journal of Kidney Health and Disease*, 1(6):1–7, 2014.
- [30] D. F. Manlove and G. O’Malley. Paired and altruistic kidney donation in the UK: Algorithms and experimentation. *ACM Journal of Experimental Algorithms*, 19(2):2.6:1–2.6:21, 2014.
- [31] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960.
- [32] National Kidney Foundation. Blood tests for transplant. URL: <https://www.kidney.org/atoz/content/BloodTests-for-Transplant>. Last visited: February 2017.

- [33] J. P. Pedroso. Maximizing expectation on vertex-disjoint cycle packing. In B. Murgante, S. Misra, A. M. A. C. Rocha, C. Torre, J. G. Rocha, M. I. Falcão, D. Taniar, B. O. Apduhan, and O. Gervasi, editors, *Computational Science and Its Applications – ICCSA 2014, Part II*, volume 8580 of *Lecture Notes in Computer Science*, pages 32–46, Cham, Switzerland, 2014. Springer.
- [34] B. Plaut, J. P. Dickerson, and T. Sandholm. Fast optimal clearing of capped-chain barter exchanges. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 601–607, AAAI Press, Phoenix, USA, 2016.
- [35] A. E. Roth, T. Sönmez, M. U. Ünver, F. L. Delmonico, and S. L. Saidman. Utilizing list exchange and nondirected donation through ‘chain’ paired kidney donations. *American Journal of Transplantation*, 6(11):2694–2705, 2006.
- [36] A. E. Roth, T. Sönmez, and M. Utku Ünver. Kidney Exchange. *The Quarterly Journal of Economics*, 119(2):457–488, 2004.
- [37] A. E. Roth, T. Sönmez, and M. Utku Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151–188, 2005.
- [38] A. E. Roth, T. 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.
- [39] S. L. Saidman, A. E. Roth, T. Sönmez, M. U. Ünver, and F. L. Delmonico. Increasing the opportunity of live kidney donation by matching for two- and three-way exchanges. *Transplantation*, 81(5):773–782, 2006.
- [40] Secretaría de Salud, Centro Nacional de Transplantes. Estadísticas. URL: [http://www.cenatra.salud.gob.mx/interior/trasplante\\_estadisticas.html](http://www.cenatra.salud.gob.mx/interior/trasplante_estadisticas.html). Last visited: February 2017 (in Spanish).
- [41] R. Sedgewick and K. Wayne. *Algorithms*. Pearson Education, Upper Saddle River, USA, fourth edition, 2011.

- 
- [42] M. Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981.
- [43] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
- [44] UC Davis Transplant Center. Matching and compatibility. URL: [http://www.ucdmc.ucdavis.edu/transplant/livingdonation/donor\\_compatible.html](http://www.ucdmc.ucdavis.edu/transplant/livingdonation/donor_compatible.html). Last visited: February 2017.
- [45] United Network for Organ Sharing (UNOS). Living donation: Information you need to know. Document 107 R4.16, Richmond, USA, 2016. URL: [http://www.unos.org/wp-content/uploads/unos/Living\\_Donation.pdf](http://www.unos.org/wp-content/uploads/unos/Living_Donation.pdf).
- [46] U.S. Department of Health and Human Services. About CPRA. Organ Procurement and Transplantation Network, URL: <https://optn.transplant.hrsa.gov/resources/allocation-calculators/about-cpra/>. Last visited: February 2017.

# AUTOBIOGRAPHY

---

Lizeth Carolina Riascos Alvarez

Candidata para obtener el grado de  
Maestro en Ciencias en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León  
Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

## FORMULATIONS AND ALGORITHMS FOR THE KIDNEY EXCHANGE PROBLEM

Nací en Planeta Rica, municipio del departamento de Córdoba, ubicado a 45 minutos de su capital Montería. Mis padres son Simón Riascos Abdala y Licel Alvarez Herazo. A los 16 años ingresé a la Universidad Nacional de Colombia Sede Medellín al programa de Ingeniería Industrial. La tesis que desarrollé fue premiada a Mejor Trabajo de Grado en Ingeniería Industrial, haciéndome acreedora de una beca para un posgrado de mi preferencia en dicha universidad. Comencé a trabajar desde que era estudiante, como monitora del curso Estadística II en la universidad, trabajo que sostuve durante año y medio. Mientras cursaba mi último semestre ingresé como practicante de logística en la fábrica de motocicletas AUTEKO S.A en Itagüí, Colombia. Al graduarme entré en la firma de análisis de datos IDATA S.A.S en Medellín hasta mi inscripción en este programa de posgrado. Durante

---

mis estudios de posgrado en la UANL, llevé a cabo una estancia de investigación de cinco meses en el Programa de Posgrado de Investigación de Operaciones e Ingeniería Industrial de la Universidad de Texas en Austin, EUA.