Spring 2014

Graphical Model for Three-Way Living Donor Kidney Exchange

Carmen Beatriz Rodriguez
Bard College, cr9889@bard.edu

Graphical Model for Three-Way Living Donor Kidney Exchange

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Carmen Beatriz Rodriguez Cabrera

Annandale-on-Hudson, New York
April, 2014

Abstract

Kidney transplantation is the treatment of choice for patients with end-stage renal disease (ESRD). There are three possible organ sources for these transplants; cadaver, living, and good samaritan donors. The living donors are usually friends or relatives of the patient. The benefits of living donors in kidney exchanges are to increase the patients' chance of receiving an organ sooner than patients waiting for cadaver donors, as well as providing them with a higher graft survival rate. In cases where a living donor is incompatible with their loved-one in need of a transplant, kidney paired exchanges are possible. Kidney paired exchanges involve two donor-recipient pairs where each donor cannot give a kidney to the intended recipient because of immunological incompatibility, but each recipient can receive a kidney from the other donor. This type of exchange offers a lifesaving alternative to waiting for a kidney from a deceased-donor waiting list. We explore how three-way exchanges can expand the opportunity for incompatible pairs to find compatible donors for their recipients and also how it can ease the burden for reciprocal compatibility. In this project, we generate a simulated population of incompatible donor-recipient pairs using data from the U.S. general population and the Organ Procurement and Transplantation Network. We assign each individual in a pair a blood type. From these assignments, we create a directed graph, where nodes represent incompatible pairs and directed edges represent possible exchanges determined by blood type. In addition to blood type, the model includes other kidney allocation considerations, such as the age of the recipient, immunologic sensitization and the hospital or treatment location of incompatible pairs. We assign these factors as priorities or weights to the nodes and to the directed edges of the graph. We find all possible three-way exchanges in the graph and present an algorithm to identify maximum weighted kidney three-way exchanges from the simulated population of incompatible pairs.

Dedication

To my parents and my grandmother for all their support and unconditional love. A mis padres y a mi abuela porque sin su ayuda y amor no hubiese llegado tan lejos. Los amo! And to all those people who are waiting for or received a kidney transplant-My father.

Acknowledgments

I have so many wonderful people to thank. First of all, I would like to thank my advisor Csilla Szabo for her immense help and support throughout this process. I am extremely grateful to her for letting me explore a topic that was very personal and at the same time very new for both of us. I would also like to thank my professors at Bard for their incredible knowledge transfer and mentorship. I am grateful to two important Math professors and senior project board members, Greg Landweber and Sam Hsiao, for their help and incredible ideas. I would especially like to thank my professor and academic advisor, Lauren Rose for believing in me. Since freshmen year, Lauren highlighted my math capabilities and encouraged me to perform better.

I do not even know how to start thanking my BEOP family. I am here right now because of the help and support I received from them. I would like to thank Jane Duffstein for her support and for being there for me through every step of the way. Thank you for giving me the opportunity to be part of the Peer Mentors group. This experience has helped me grow as a human as well as a more responsible student. I also want to thank all the BEOP scholars for making my college experience at Bard a wonderful one.

I would not be here without the support of my family and friends. I want to thank my parents for being so special. Their love, understanding and support keep me going everyday. I want to thank my two younger brothers, Franyelis and Franyany, for bringing so much joy to my life. My grandmother, I do not even know where to begin thanking her. She is the most kind and understanding person I know. I want to thank her for receiving me with open arms when I first arrived to the U.S. and for being a mother and a friend. Last but not least, I would like to thank all my friends. They have seen every side of me. I thank God for allowing me to meet such wonderful people who are not merely my friends but they are my sisters. I want to thank Ismary Blanco for being such a unique person, a loving friend and an older sister who never gave up on me. Rosemary Ferreira for being such an inspiring friend. Ayda Gonzalez for being understanding, joyful and kind. Anam Nasim for being encouraging, caring and a role model. Anabel Cabrera for being a friend I can count on no matter what and for being so fabulous. Danilsa Fernandez for being my "brujis", and a friend I can always count on. Maria Hoz for her love and fun nature. Samantha Burke for her love and wonderful personality. And Nushrat Hoque. I would also like to thank my friends and Bard alumni Jose Mendez, Andres Medina, Katherine Garzon, and Anisha Ramnani for being part of this four year experience.

Contents

List of Figures

List of Tables

1
Introduction

1.1 What Are Kidneys?

The kidneys are two bean-shaped organs, each about the size of a fist. They are located near the middle back with one on each side of the spine. Kidneys have the essential job of filtering waste from our blood and excess water from our bodies, which they do through the formation of urine. All of our blood passes through the kidneys about 20 times per hour [10], where about one million tiny tubular units inside the kidneys, called nephrons, filter waste. If the kidneys do not function, the waste formed from, for example, food would build up in the blood and damage our bodies. Also, we would swell up with excess water because for our body to work properly, it must contain a specific amount of water. Most people are born with two kidneys; however, people can live normal and healthy lives with only one functioning kidney.

1.2 End-Stage Renal Disease

Kidneys can stop working properly, due to diseases like high blood pressure, diabetes, infections, reactions to medicaments, or in some cases genetic abnormalities. When this

occurs, it means that the nephrons are losing their ability to filter waste. At this stage of partial loss of function, the kidney disease is called chronic kidney disease (CKD) [14]. This loss of performance could happen slowly or very fast depending on the condition or disease that triggered the damage. If it happens slowly, the damage will become apparent after years or even decades, or right after both kidneys stop functioning simultaneously [14].

The disease where both kidneys lose complete function is called end stage renal disease (ESRD) [14]. This stage is due to progressive CKD. As more nephrons are damaged they shut down, and after a certain point, the nephrons that remain cannot filter blood. This kidney failure is permanent and may lead to death if not treated. The only treatments available for this disease are dialysis or kidney transplantation.

1.3 Dialysis

Dialysis is a process used to remove waste and excess water from the body. There are two types of dialysis: hemodialysis and peritoneal dialysis. In hemodialysis, the patient is connected to an artificial kidney machine, which filters the waste from the blood and removes as much excess water as possible. Hemodialysis uses a fistula, in order to assist in the removal of waste from the patient's body. To form this fistula or bigger blood vessel, a minor surgery is performed under the patient's skin of an arm or leg, where an artery and a vein are artificially connected. In peritoneal dialysis, the removal of waste from the blood is done inside the body with the help of a catheter placed into the patient's abdomen [15, 16].

These two aforementioned treatments extend the life of the patient while awaiting for kidney transplantation. In hemodialysis, the patient receives treatment three times a week. The length of this treatment depends on what the doctor prescribes the patient, which is usually three to four hours [16]. The patient must be willing to travel to a treatment

center three days a week, which adds more costs to their lives. In peritoneal dialysis, the patients are trained at a dialysis center so that the patients can do the treatment at home or wherever they may be. The patient can perform the treatment by putting a cleansing fluid, called a dialysate, into the catheter that has already been placed into the abdomen. This fluid pulls the waste and extra fluid from the patient's blood. Dialysate exchanges can be done manually or with a machine called a cycler[16].

Most of these patients receiving dialysis suffer from depression and anxiety. Aside from the burden of receiving dialysis treatment, they also have to change their diet to consume less salts and limit the amount of fluid they intake, since liquid overload can cause high blood pressure.

1.4 Kidney Transplantation

Transplantation is the only "cure" for kidney disease. Patients who receive a kidney transplant live up to 10 years longer than those patients who continue receiving dialysis [8]. In addition to that, transplantation is less expensive than dialysis in the long run [5]. Many patients die receiving dialysis because their bodies cannot handle the treatment, but also many die waiting to receive a transplant. The outcome of the transplantation cannot be totally predicted, since it varies from patient to patient. The patient must take immunosuppressant drugs for life to prevent the immune system from rejecting the donor's kidney. Both treatments are risky, but patients with this condition do not have any other option for treatment.

Today, kidney transplantation is the treatment of choice for ESRD. This brings about a very important question: Where do we find donors? Donor sources can be classified as living donors, cadaveric, and altruistic or samaritan. Even though these three types of donors exist, it does not mean everyone with kidney failure will receive a kidney. Organ allocation is complicated. According to the United Network for Organ Sharing (UNOS), as

of March 28, 2014, there are 107,311 patients on the waiting list for a kidney transplant in the United States [10]. Cadaveric donors, as the name implies, are deceased patients who made a life decision of becoming organ donors. Living donors are usually family members or friends that want to donate to a loved one. In addition, kidneys from living donors have better graft survival, 16 years or more compared to the 8.6 survival years of a kidney from a deceased donor [8]. Living altruistic donors, also called good samaritan donors, are people who simply volunteer to donate an organ as an act of kindness.

In 2006, around 18,016 patients in the United States received a kidney transplant while 3,916 died waiting for a kidney to become available [9]. From these kidney transplants, 85 were done through Kidney Paired Donation (KPD) [1], which we will explore in-depth in Section 1.4.2. As it can clearly be seen from the waiting list statistics, unfortunately, the need for kidney transplants exceeds the number of kidneys available from deceased donors, resulting in approximately 19 people dying every day in the U.S. waiting for a transplant [8]. When a kidney from a deceased donor is available it has to be fairly distributed. Kidney allocation takes into account waiting time, priority of the patient, and most importantly compatibility. The allocation of kidneys raises questions like: What if the patient who urgently needs the transplant is the last person on the wait list? What happens to those people at the top of the list who have been waiting for years? These issues could be addressed using other sources of organs such as living donors to increase the amount of kidneys available for transplants.

1.4.1 Compatibility in Kidney Transplantation

Compatibility of donors and recipients is a big issue that arises with kidney transplantation. The two major categories of incompatibilities are blood group and tissue. The blood groups are A, B, AB, and O (ABO system). When including Rh (Rhesus factor), the

universal donor is O^- and the universal receiver is AB^+. From the information regarding compatibility in the Stanford School of Medicine Blood Center [17], we created Table 1.4.1.

Recipient	Compatible with Donor's Blood Type:
O Rh positive	O^+ and O^-
O Rh negative	O^-
A Rh positive	O^+ , O^-, A^+, A^-
A Rh negative	O^- , A^-
B Rh positive	O^+ , O^-, B^+, B^-
B Rh negative	O^-, B^-
AB Rh positive	O^+ , O^-, A^+, A^- , B^+, B^-, AB^+, AB^-
AB Rh negative	O^- , A^- , B^-, AB^-

Table 1.4.1: Blood type and Rh compatibility[17].

Tissue incompatibility occurs when the recipient has pre-formed antibodies to some of the donor's HLA (human leukocyte antigen)[1], which are six molecule-codes found in the cells that are genetically inherited. These pre-formed antibodies are proteins found in the blood, they detect and destroy foreign invaders in the body. The production of antibodies is catalyzed by antigens, which causes an immune system response in the body. In simpler words, because of these antibodies, the recipient's immunological system would see the donor's kidney as an invader and thus, reject it. This is the reason why recipients are required to take immunosuppressive drugs after transplantation.

Finally, before the transplant, an additional blood crossmatch test is carried out to determine how the recipient's system will react to the kidney from the donor. If the crossmatch is negative, then the transplantation can proceed as planned since this means that the recipient will likely not reject the donor's kidney. Otherwise, the transplant cannot proceed and the patient returns to the waiting list. If the kidney was from a cadaver or samaritan donor, it will go to the next compatible person on the waiting list.

1.4.2 Kidney Exchange Programs

Many candidates on the waiting list have family members or friends who want to donate one of their kidneys as a gift of life to their loved ones; however, about one third of these willing donors are rejected due to blood type or tissue incompatibilities [6, 8, 9]. Kidney paired donation (KPD) offers a possible solution to this problem. It takes an incompatible donor-recipient pair and finds another pair in a similar situation. The donors of these pairs then must be reciprocally compatible with the recipients of each pair. Two transplants can be performed where the recipient of one pair receives a kidney from the donor of the other pair involved, and both pairs benefit from the exchange. This exchange is shown in Figure 1.4.1 (A). KPD gives an incentive to donors who might not otherwise have volunteered to donate an organ to help a total stranger.

The idea of KPD was first introduced in 1986 by Felix Rapaport, a surgeon, scientist, and one of the "giants" in the field of organ transplantation. Some believed that this program could only benefit a very small number of recipients with incompatible donors; however, other countries, such as South Korea and Switzerland, took advantage of this program and started performing kidney exchanges in the late 1990's. It was not until the year 2000 that the first paired donation transplant was performed in the U.S when a pilot testing program was initiated by the United Network for Organ Sharing [6, 8].

In the past few years, KPD has become a growing source of transplantable kidneys. KPD is not only growing but also changing over time. When the program started, donor-recipient pairs were matched using local or regional databases with patient information. These patients were matched using a scheme called "first-accept match" in which a compatible pair was searched and when found the pairs would proceed to transplantation. These pairs would then be removed from the database [5, 11]. However, as earlier studies have noted, this scheme does not take into consideration other compatible pairs [8]. First-accepted

match does not maximize the number of transplants, which has been the main goal of the program since it was first introduced by Rapaport in 1986.

Generalizations of the basic kidney paired donation program include three-way exchanges, list exchange, domino-paired donation, voluntary compatible pair participation, and Never Ending Altruistic Donor (NEAD) Chain. Figure 1.4.1 shows all possible kidney exchanges in a chronological order from the original proposal by Felix Rapaport as shown in *Kidney paired donation* [8]. An important detail to note about all these exchanges is that all the donor operations are started simultaneously. This ensures that each donor has the freedom to withdraw at any time until undergoing anesthesia, without the worry that some intended recipient in the exchange will be left unfairly without a donor [6]. We briefly discuss all of these exchanges:

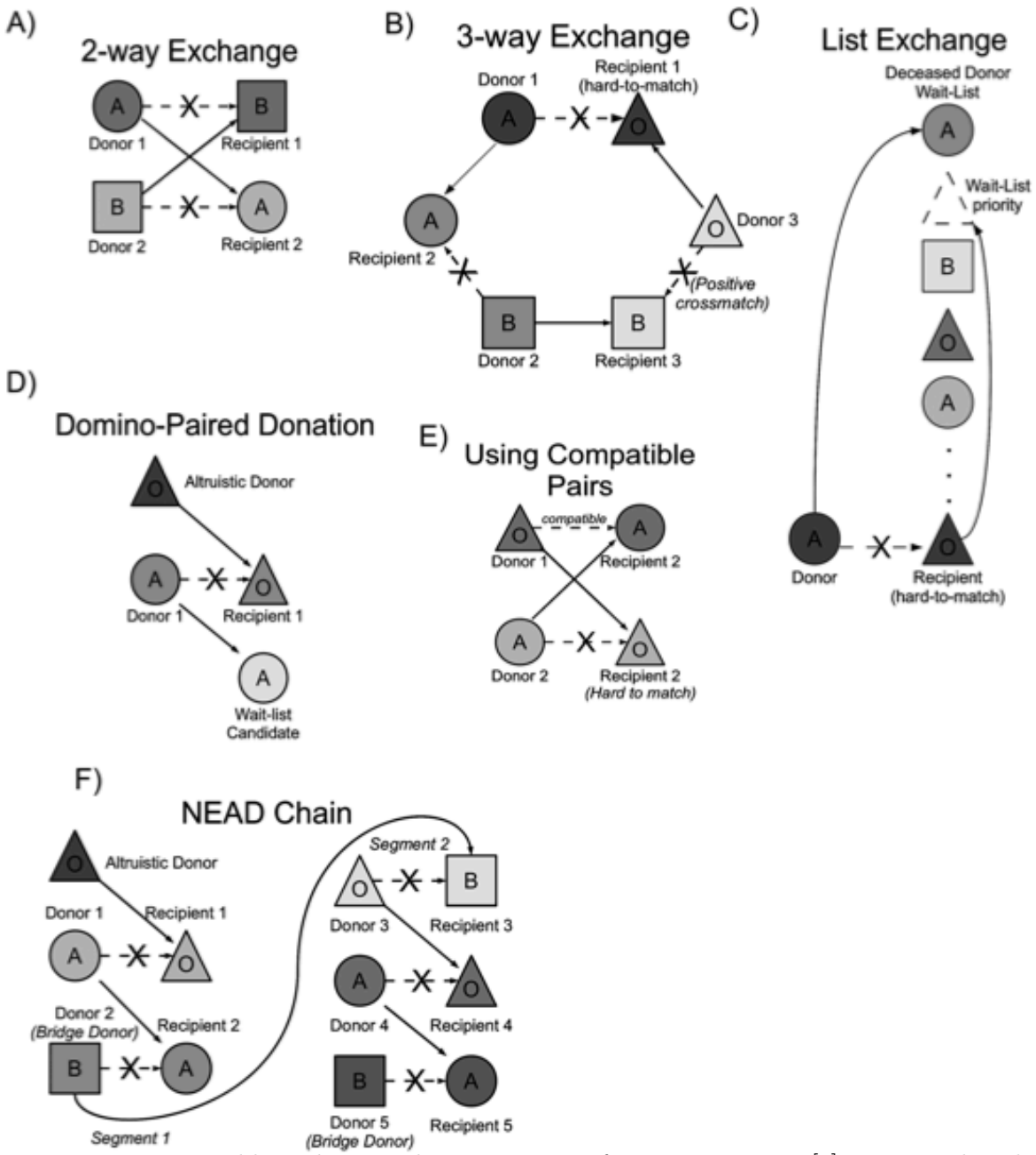


Figure 1.4.1: Possible Kidney Exchanges starting from 1986 to 2011[8]. Reprinted with permission of the Oxford University Press.

- **Three-Way Exchanges:** These exchanges work similarly to pair exchanges, except they include an additional incompatible pair as shown in Figure 1.4.1 (B). This type of exchange increases the opportunity of highly sensitized patients receiving a kidney transplant and eliminates the necessity of reciprocal compatibility for matching [1,8]. For example, in a two-way exchange, a blood type O recipient has not only the problem of needing a compatible donor, but also he or she needs to find a compatible donor whose incompatible recipient can receive a kidney from the blood type O recipient's incompatible donor [8]. This burden is eased with the inclusion of the third incompatible pair. This exchange and the paired exchange shown in figure 1.4.1 (A) only involve incompatible pairs.

- **List Exchange:** New additions to KPD programs include using organs from other sources. In a list exchange, the donor from an incompatible donor-recipient pair will donate to a candidate waiting for a deceased donor kidney and in return the recipient from the pair gets higher priority on the waiting list for a cadaveric donor as shown in Figure 1.4.1 (C).

- **Compatible Pairs:** Compatible donor-recipient pairs have already participated voluntarily in some paired donations [1]. As shown in Figure 1.4.1 (E), adding these pairs to the pool of exchanges not only increases the chance for the incompatible pairs to find a match, but it may also find the compatible pair a "younger or more immunologically favorable donor" [1]. This new approach is still under discussion since in some cases it may not benefit the compatible pair [2, 6, 8] because the recipient might end up with a worse donor. Also, compatible pairs could help solve the blood group O imbalance in incompatible donor-recipient pools. This is because of the compatible donor-recipient pairs about 65% have blood type O donors and only 45% have blood type O recipients [6].

- **Domino-Paired Donation:** In this exchange, as shown in Figure 1.4.1 (D), community donors also known as an altruistic donors are used. In the domino exchange, an altruistic donor can donate to a patient at the top of the waiting list, in which case the exchange or chain terminates. However, if the altruistic donor donates to the recipient of an incompatible pair, then the donor of the pair donates to someone on the waiting list. This latter exchange creates a domino effect since all the people in the waiting list benefit from it, moving up on the list [4]. It also helps patients who are disadvantaged by the current allocation system, for example, blood type O recipients who can only receive a kidney from a donor of the same blood type. This is because blood type O patients tend to have longer waiting periods.

- **NEAD Chain:** This is another type of KPD being implemented as illustrated in Figure 1.4.1 (F), which is similar to a domino exchange. This exchange also uses an altruistic donor and incompatible pairs. The altruistic donor donates to the recipient of an incompatible pair, then the donor of this pair donates to the recipient of another incompatible pair. This chain continues until the last donor who becomes a "bridge donor", waits for another chain of exchanges instead of donating to a candidate of the waiting list. This type of exchange can generate many exchanges, but it is risky because the last donor has the liberty to refuse to donate, which would not allow for the chain to continue [8].

2
Preliminaries

This chapter reviews some basic concepts of graph theory that will be needed for the development of the graphical model for three-way kidney exchange [12, 13].

2.1 Basic Graph Theory Definitions

In simple terms, a graph G, is a representation of a set of points, which are connected in pairs by lines. In mathematical terms,

Definition 2.1.1. A **graph** G consists of a non-empty finite set of elements $V(G)$ referred to as **vertices** or **nodes** and a finite family set $E(G)$, disjoint from $V(G)$, of **edges**. The word family in this definition allows for multiple edges, if not specified then the graph is **simple**. △

Definition 2.1.2. A **subgraph** of graph G is a graph that consists of subsets of $V(G)$ and $E(G)$. △

A graph contains an **incidence function**, which is used to associate each edge of G with an unordered pair of vertices of G. Therefore, if d is an edge and u and w are vertices,

then by the incidence function $d = \{u, w\}$ and so it is said to connect u and w. The edge d can also be written as uw. From this we can also say that u and w are **endpoints** of the edge d.

We say that two vertices are **adjacent** if they are connected by an edge, which also means that the two vertices are **incident** to such edge. We also say that two different edges are adjacent if they share a common vertex. As shown in Figure 2.1.1 z and y are adjacent vertices because they are connected, and zy and yw are adjacent edges since they share y as a common vertex.

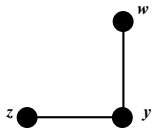


Figure 2.1.1: An illustration of adjacent edges.

The **degree** of a vertex v of G is the number of edges incident with v or the number of connections of v with other vertices in G or itself. This is represented as $deg(v)$. For example, in Figure 2.1.1, the $deg(y) = 2$.

Definition 2.1.3. A **walk** in G is a finite sequence of consecutive edges of the form $v_0v_1, v_1v_2, v_2v_3, \ldots, v_{n-1}v_n$. A walk is said to be closed if its first and last vertices are the same, and open if they are different. △

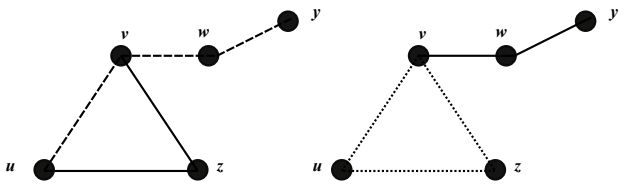


Figure 2.1.2: Example of an open walk u, v, w, y and a closed walk u, v, z, u.

Definition 2.1.4. A **path** in G is a walk in which the vertices are distinct. A path with n vertices is represented by P_n. △

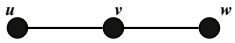


Figure 2.1.3: An illustration of P_3.

Definition 2.1.5. A **cycle** is a connected subgraph whose vertices are of degree two . It can also be defined as a closed walk. The length of a cycle is the number of its edges and so a cycle of length k is a $k-cycle$ and can be denoted by C_k. △

Definition 2.1.6. A **complete graph** K is a simple graph in which all vertices are adjacent and thus all vertices have the same degree. △

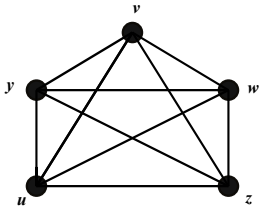


Figure 2.1.4: An illustration of a complete graph K_5.

Definition 2.1.7. A **directed edge** is an edge in which one vertex incident with it, is designated as the head vertex and the other vertex as the tail. A directed edge uv is said to be directed from its tail u to its head v. A **directed graph**, G is a collection of these directed edges. The **indegree** of a vertex v in $v \in V(G)$ counts the number of edges pointing towards v, edges for which v is the head. While the **outdegree** counts the number of edges pointing from v, edges for which v is the tail. △

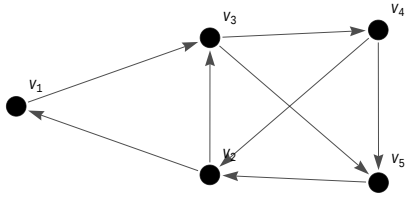


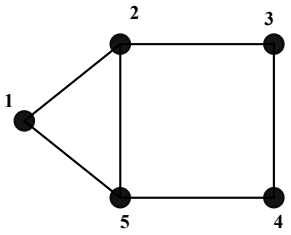
Figure 2.1.5: An illustration of a directed graph.

Note that it is very important to know if a graph is directed or undirected. This is because if graph G is directed and $uv \in E(G)$, then uv and vu are different directed edges.

Definition 2.1.8. The **density** of a graph G measures how many edges are in set $E(G)$ compared to the maximum possible number of edges between vertices in set $V(G)$. A directed graph that has no loops can have at most $|V(G)| \cdot (|V(G)| - 1)$ edges, and thus the density of a directed graph is the ratio $\frac{|E(G)|}{|V(G)| \cdot (|V(G)|-1)}$. △

A graph can be represented by a matrix or an array of numbers. This representation can be beneficial for storing large graphs in a computer. A **matrix** C, is an array of numbers arranged in rows and columns, where each item is called an entry, which can be represented as $C(i, j)$ where i corresponds to the row and j to the column location.

If G is a graph with vertices labelled as $\{1, 2, \ldots, n\}$, its **adjacency matrix A** is the $n \times n$ matrix whose ij-th entry is the number of edges connecting vertex i and j. The following figure shows an adjacency matrix and its corresponding graph.



(a) Simple graph

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

(b) Adjacency matrix

Figure 2.1.6: Example of Adjacency matrix and corresponding graph.

In this figure we see that the entry in the matrix is nonzero when vertex i and vertex j are joined, otherwise the entries $A(i, j)$ and $A(j, i)$ are zero. Another interesting fact about adjacency matrices is that the entry ij in the n-th power of an adjacency matrix gives the number of paths of length n from node i to node j.

2.2 Notes on Matching

Now that we have the basic definitions, we can enter the realm of matching, which is the main focus of this research project.

Definition 2.2.1. A **matching** M of graph G is a subgraph of G in which no two edges are adjacent. The vertices incident to the edges of a matching M are said to be **saturated**, while the remaining vertices are **unsaturated**. △

The **size** of a matching M is the number of non-adjacent edges. A **perfect matching** in graph G is a matching in which all vertices are saturated.

Definition 2.2.2. A **maximal matching** of graph G is a matching in which the addition of an edge, that is not part of the matching set, would vanish the matching. A **maximum matching** is the largest matching among all the possible ones in G. △

Given the matching M,

- an M**-alternating path** is a path that alternates between edges in M and edges not in M.
- an M**-augmenting path** is an M-alternating path in which the endpoints are unsaturated by the matching M.

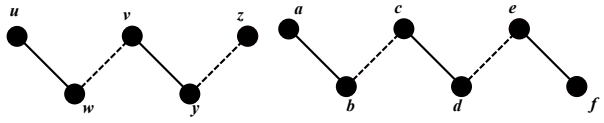


Figure 2.2.1: An illustration of a P_5 alternating path and a P_6 augmenting path.

Theorem 2.2.3. *(Berge Theorem 1957) A matching M in a graph G is a maximum matching in G if and only if G has no M-augmenting path*[13].

As will be discussed in the next chapter, the most recent algorithm for kidney paired donation uses Jack Edmonds' Blossom Algorithm from 1965. This algorithm uses Berge's Theorem 2.2.3. This is because we can find a maximum matching by simply searching for an augmenting path in graph G. Jack Edmonds presented this algorithm in his paper "Paths, trees, and flowers."

3

Existing Mathematical Models for Kidney Exchange

In Chapter 1, we introduced the current approach for KPD, "first-accept" match, which is meant to increase the opportunity of donor-recipient pairs to obtain a transplant. However, this first-accept scheme only finds one feasible match and does not take into account other possible matches. Many researchers such as Sommer E. Gentry, Alvin E. Roth, and Dorry L. Segev have worked on possible solutions to this problem by utilizing more advanced matching algorithms through computer simulations and graph theory.

3.1 Optimized Match Algorithm

The basic pairwise-exchange has been advanced to an optimized match by Dr. Dorry Segev, MD, a transplant surgeon at Johns Hopkins, Dr. Sommer Gentry, PhD, an applied mathematics professor at the United States Naval Academy, and their colleagues. They propose an optimized algorithm based on the Edmond's Algorithm from graph theory [5]. This optimization measures the benefit of decisions from limited resources and then gives the best decision [11]. In the context of paired donation, the limited resource is the pool

of incompatible donors who are willing to donate to their loved ones but cannot because of immunological incompatibilities.

In the optimized match model, a graph is created for KPD in which each node represents an incompatible donor-recipient pair and each undirected edge represents a reciprocally compatible foursome of paired donation. This means that the donor of the first incompatible pair is compatible to the recipient of the second pair and vice versa [1, 5, 11]. The edges of the graph determine which pairs are compatible based on information about blood type, tissue type, and other necessary tests. The algorithm attempts to make sure that every patients' needs are met. These needs are interpreted as factors in this new scheme and are translated into a number or a weight on each edge. Some of these factors are determined by the transplant community and include location of compatible pairs, highly sensitized patients, children, patients with other disease, and the age of patient [1, 5, 8]. After these priorities are set, the algorithm makes the optimal matches. The optimized match algorithm guarantees to provide us the maximum number of matches possible.

This algorithm operates on the created KPD graph and does not only look at one match like the "first-accept" match, but looks at all the possible matches. Once all the possible options are set, the best matches need to be made, which is when the optimized match algorithm is run. Recall from Section 2.2 that a matching on a graph is a subgraph in which no two edges are incident to the same node. In KPD terms, this means that one incompatible donor-recipient pair cannot be involved in more than one paired donation.

In order to study the efficiency of this algorithm, computer simulations have been executed by the creators in collaboration with other researches [5]. They found that with a pool of 1000 donor-recipients pairs, the optimized algorithm considers 10^{250} solutions before it picks one, unlike the "first-accept" match scheme. In a pool of 4000 potential recipients as stated in *Kidney Paired Donation and Optimizing the Use of Live Donor*

Organs, "nearly $750 million dollars would be saved by KPD compared with the cost of dialysis and deceased donor transplantation"[5].

This new approach, according to the authors, gives an even better result than merely extending the registry utilizing the "first-accept" match scheme. The implementation of this method would not only result in more transplants, but it also assures better matches and longer graft survival when compared to the "first-accept" scheme. Furthermore, this method allows sensitized patients, who are extremely difficult to match, to find a suitable donor. In the implementation of this optimized algorithm there is a need for a national database. Having a national database can expand the number of patients participating in exchanges, but it can also be challenging because it would add another level of complexity to the method. This is because, as discussed in *Kidney Paired Donation: Fundamentals, Limitations, and Expansions*, the transplantation community would have to make decisions about traveling and availability of a transplantation crew in the hospitals involved in the kidney exchanges [6].

3.2 Top Trading Cycles and Chain Algorithm

One of the first algorithms used for kidney living donation was the Top Trading Cycles and Chains Algorithm [9]. This algorithm, in comparison to the optimized match algorithm, uses cadaveric donors. It is assumed that D_i and R_i are incompatible donor-recipient pairs and w represents a possible kidney available from the cadaver wait list. The algorithm first creates a list with the preferred arrangements for each recipient. That is, it places all the preferred donors for each recipient, including cadaveric donors or the ones that are willing to trade their donors for a higher position on the waiting list. The algorithm consists of several steps and might lead to several rounds of exchanges until the available kidneys are transplanted. In the first step, the preferences of each recipient are modeled using a directed graph where vertices represent either a donor or a recipient and each directed edge

represents compatibility as well as preferences. The latter means that if an edge drawn from a vertex corresponding to a recipient points towards a node corresponding to a donor, that recipient is establishing a preference for that particular donor. If it is the donor's node that points towards the recipient's node, then it means that they are compatible. In the second step, cycles of any size in the graph are identified and the donor-recipients pairs that are in the cycle are removed; that is, the pairs proceed to transplantation. This step continues until all cycles are removed. The third step looks for chains that end with w. If edge R_iD_j is in the chosen chain, then recipient R_i receives a kidney from donor D_j, and R_j will receive priority on the wait list. The donor recipient-pair (R_i, D_j) is removed and what is left is a donor at the start of the chain that could have donated to R_i, but R_i had preferred D_j. This donor stays available for a future exchange.

One problem with this algorithm is that it creates big cycles, which are logistically impossible to carry out since they will need more coordination, more operation rooms, and more staff per person (donor or recipient), which makes it impractical. Also, some incompatibilities can only be determined after a crossmatch test and with bigger cycles there is a higher percentage of donor-recipient pairs that will exhibit such incompatibilities [9].

4
Model for Three-Way Kidney Exchanges

4.1 Objective

On average, nearly 2,500 patients are added to the waiting list for a kidney transplant each month in the United States. Using living donors reduces the waiting time for patients on the wait list. These living donors, as explained in Section 1.4, can be altruistic donors, compatible donors, or donors from incompatible donor-recipient pairs. The donors from incompatible pairs are usually family members or friends, and since they are incompatible, the recipients end up being placed on the waiting list. In our model, we look at three-way exchanges, increasing the number of transplantations with the addition of an incompatible donor-recipient pair to the KPD strategy. This means that instead of having a paired donation, we make exchanges where three patients receive a transplant instead of two. Not only does a three-way exchange allow patients to receive a kidney sooner than waiting for a cadaver donor, but kidneys from living donors generally have better graft survival rates than a deceased donor kidney [8].

For our three-way exchange model, we create a directed graph $G_N = (V, E)$ from N incompatible pairs, which contains the set $V(G_N)$ of vertices or nodes that represent incompatible donor-recipient pairs, together with a set $E(G_N)$ of directed edges which denote a match and possible exchange between two incompatible donor-recipient pairs. Weights are assigned to both the edges and nodes of the directed graph G_N. These weights are based on recipients' and exchange priorities, such as, the hospital location of incompatible pairs, blood type, and age of the recipient. Considering these factors, we create an algorithm for finding weighted three-cycles in the directed graph to maximize the number of transplants by looking at three-way kidney exchanges and paying attention to the complexities of the transplantation process in general. In this chapter, we present the graphical model.

4.2 Simulation of Incompatible Donor-Recipient Population

No direct data is available regarding incompatible donor-recipient pairs that would enter a national KPD program. We, therefore, simulate a population using distributions from Organ Procurement and Transplantation Network's (OPTN) national database and blood centers [10, 17]. Some of the characteristics of the population are drawn from distributions describing end-stage renal disease patients eligible for renal transplantation. The simulation was done in R, which is a software for statistical computing and graphics.

4.2.1 Blood Type of Incompatible Donor-Recipient Pairs

The simulated population consists of blood type, age, and an assigned hospital or treatment location for each pair. For ABO blood type and +/- Rh we use the distribution shown in Table 4.2.1 of the U.S. general population [17], assuming that blood type is independent from ethnicity. We also assume that the donors and recipients have the same blood type distribution.

Blood Type and Rh	Frequency
O Rh positive	37.4%
O Rh negative	6.6%
A Rh positive	35.7%
A Rh negative	6.3%
B Rh positive	8.5%
B Rh negative	1.5%
AB Rh positive	3.4%
AB Rh negative	0.6%

Table 4.2.1: Blood type and +/- Rh Distribution of the U.S. general population [17].

We determine incompatibility between donor-recipient pairs by looking at blood type. After the data is generated, we have a sample with two variables, one for the donor's blood type and the other for the recipient's blood type. We created a function in R with the purpose of ensuring that the blood type samples generated give us incompatible donor-recipient pairs. This R code can be found in Appendix A. These pairs are the nodes of our exchange directed graph. For these pairs we assume one donor per patient.

For blood type O^- donors in the simulated population, we assume the donor is incompatible with the recipient because of positive crossmatch. That is, we assume that the recipient has pre-formed antibodies against foreign tissues and thus the recipient is sensitized. We do not have specific data on crossmatch tests, but we want to ensure our simulated population has blood type O^- donors because they are the only ones that can donate to blood type O^- recipients.

4.2.2 Age of Recipient

The recipient's age for the simulated population is generated by sampling from the distribution of ages of transplant recipients in the United States from the Organ Procurement and Transplantation Network's (OPTN) national database [10]. This distribution is a good

representation of candidates waiting for a transplant. Table 4.2.2 shows this distribution divided into intervals, from January 1, 1988 to November 30, 2013 [10].

Age Category	Number
<1 Years	110
1-5 Years	3,660
6-10 Years	3,694
11-17 Years	11,425
18-34 Years	65,323
35-49 Years	109,386
50-64 Years	118,672
65-80 Years	37,529
Total	349,799

Table 4.2.2: Age distribution of kidney recipients.

We generate the age sample and assign each recipient an age interval. To give a specific age to each recipient, we generate an uniformly distributed random number between the endpoints of such intervals in MS Excel. In the case of the donor's age we assume they are of various ages, healthy and old enough to serve as donors.

4.2.3 Location of Incompatible Pairs

The treatment location of incompatible pairs is another important variable for prioritizing exchanges. When pairs are closer, the exchange is easier to carry out because donors would not have to separate from their loved ones. Other models such as the optimized match algorithm discussed in Section 3.1 consider location as a very important factor. We used information of hospitals in New York State that perform transplantations from OPTN [10]. Each pair is randomly assigned a hospital from Table 4.2.3. For these assignments, we assume that the donor and the recipient from an incompatible pair are at the same hospital. Also, we do not consider the number of transplants at each treatment location. In other words, we assume that all hospital are equally likely to perform a kidney transplant.

Hospital's Name	**City**
Albany Medical Center Hospital	Albany
New York-Prebysterian/Columbia	New York City
Montefiore Medical Center	New York City
New York University Medical center	New York City
SUNY Downstate Medical Center	New York City
Buffalo General Hospital/ Children Hospital	Buffalo
Strong Memorial Hospital	Rochester
Westchester Medical Center	Valhalla
North Shore University hospital	Manhasset
Erie County Medical Center	Buffalo
Mount Sinai Medical Center	New York City
SUNY Upstate Medical Center	Syracuse

Table 4.2.3: Possible Locations of Incompatible Pairs.

4.2.4 Small Example of a Generated Incompatible Pairs Population

In this subsection we show an example of a complete population of six incompatible donor-recipient pairs. For this example we do not use blood type and +/− Rh.

Donor's Blood Type	Recipient's Blood Type	Recipient's Age(interval)	Recipient's Age(specific)	Location of Pair
A	O	18-34	31	Montefiore Medical Center
O	A	65-80	78	Erie County Medical Center
O	B	65-80	72	North Shore University hospital
AB	A	11-17	13	Montefiore Medical Center
B	A	50-64	64	New York University Medical center
A	O	35-49	38	Albany Medical Center Hospital

Table 4.2.4: Sample population characteristics example.

4.3 Directed Graph and Finding Edges

The need for direction makes the creation of this graph slightly more complex. To be able to identify the donating pair we need directed edges. In this graph, the node designated as the tail of the directed edge represents the donor and the head of the edge corresponds to the compatible recipient. A switch between the node designated as the tail and the node appointed as the head gives a completely different edge, and thus a different match.

Figure 4.3.1 makes these designations for head and tail of a directed edge more clear.

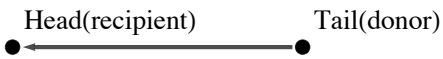


Figure 4.3.1: Example of a directed edge.

The directed edges of the graph are created using the blood type information of the incompatible donor-recipient pairs generated. We assume that incompatible pairs are related, but that donor-recipient exchanges happen between unrelated participants. Also, these generated pairs comprise the nodes of the graph.

Since our main goal is to look at three-way exchanges, we do not need reciprocal compatibility, this means we do not require two directed edges between two ordered pair of nodes indicating that the exchange can be done both ways. That is, the donor of pair A can be compatible with the recipient of pair B, but the donor of pair B does not need to be compatible with the recipient of pair A.

Furthermore, in our graph we do not allow for self-loops because this would indicate that the pair is compatible and thus it should not be part of the population. The out-degree of a node, as defined in Section 2.1, is the number of recipients the donor in the tail node can donate to. The in-degree of a node, also defined in Section 2.1, is the number of compatible donors a recipient has. Note that a node from blood type O^- will have a large number of edges pointing from it since blood type O^- is the universal blood type donor and donors with this blood type will be compatible with recipients of every other blood type. The nodes representing recipients with blood type O^- will have the least in-degree, since they can only receive from their same blood type.

To construct the directed graph, we create a function in MATLAB that takes as input the incompatible donor-recipient pairs by blood type and outputs the adjacency matrix of the digraph. The code for this function is given in Appendix B. Figure 4.3.2 shows an

example of an exchange graph obtained from the population of six incompatible donor-recipient pairs sample from Table 4.2.4.

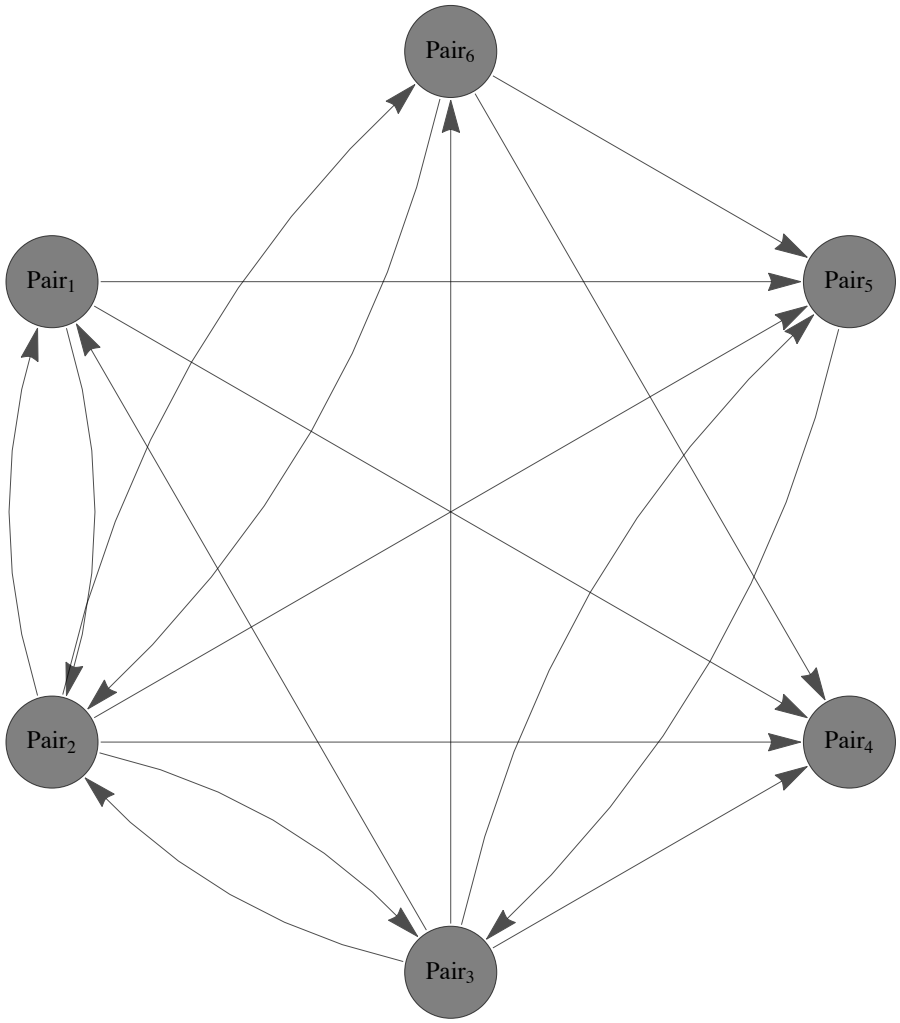


Figure 4.3.2: Example of graph from the population in Table 4.2.4

4.4 Assigning Node and Edge Weights

A matching on the exchange graph represents a decision about which donors and recipients should proceed with a kidney exchange. This does not refer to the mathematical definition for matching. The first objective is to maximize the number of possible transplants by matching incompatible donor-recipient pairs in three-way exchanges. This matching must

reflect the complexities of the transplantation procedure and system in general, meaning that it should take into account information concerning the compatibility factors, which in our case only include blood type of donors and recipients.

We created a kidney allocation formula based on the recipient and matching priorities. This formula prioritizes recipients and matches by assigning weights to the nodes and edges of the directed exchange graph. These weights are numbers associated with the particular factor. The exchanges that will be suggested are the ones scoring highest according to the allocation weight system that we create.

The following Table 4.4.1 serves as a reference for the definitions of all variables used in this section:

Variable Name	Definition
NW_i	Weight assigned to node i.
$EW_{i,j}$	Weight assigned to the directed edge from node i to node j.
W_{ra}	Weight assigned to the recipient of an incompatible pair because of the age factor.
W_{bt}	Weight assigned to the incompatible pair because the pair includes a blood type O^- recipient or sensitized recipients which we assume are the recipients with blood type O^- donors.

Table 4.4.1: Definition of Variables of the Weight System.

4.4.1 Nodes Weight: Age, Blood Type and Immunologic Sensitization

Recall that the nodes represent incompatible donor-recipient pairs. The node weight, NW_i, is the weight assigned to node i as a way to prioritize the recipients that should get a transplant immediately or in a first round (once a compatible donor appears). Because the recipients are the only ones benefited from the transplanted kidney, the weight system and thus the allocation formula has to be fair for every group. As Dr. Sommer Gentry expresses in Optimization in Medicine and Biology, "the organ allocation decision must

balance utility with equity" [1]. Therefore, our weight system tries to emphasize equity.

Patients are prioritized by looking at factors that can change how they are viewed by the transplantation community. These factors include treatment location of the incompatible pairs, age of the recipients, age difference between donors and recipients, immunologic sensitization, blood type of the recipient, O^- blood type recipients, waiting time if the recipient has been placed in the waiting list, and many more. In this graphical model we will consider the factors: recipient's age, blood type of recipient, immunologic sensitization of recipients, and hospital location of the incompatible pairs in New York State.

All nodes start with 25 points. This is because every incompatible pair is equally valuable. The chart in Figure 4.4.1 shows an outline of the node weight scheme.

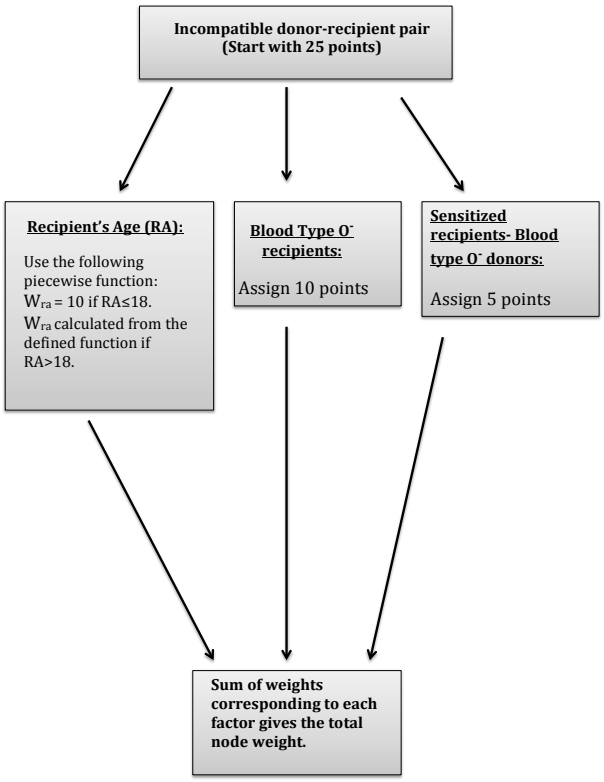


Figure 4.4.1: Node Weight Assignment Scheme.

Recipient's age: Age is a debatable factor to consider in a kidney or any organ allocation formula. According to Robert M. Veatch in Transplantation Ethics, there are two sides in considering age as a factor. One side argues that more is owed to young people since they have not had many years of life, while others argue that if that were the case then old people would be treated unfairly in the weight system [18]. This is because they would be judged for simply living their lives. A question that arises is: From what age should a person be considered old? Considering these arguments, we decide that older recipients should receive fewer points in the weight system because they could be more likely to develop or already have other health issues during the time waiting and thus transplanting the organ to them might be not be as convenient, even more so if the patient rejects it. In our model, we try our best to keep the system fair and efficient.

The weights assigned to the recipients because of age range from 0-10 points and are assigned by using a piecewise defined function shown in Figure 4.4.2. Pediatric patients ($\leq$ 18 years old) are prioritized in every exchange and are given the maximum number of points, which is 10 points. From eighteen years on, we use a linear function to calculate the age weights. From this plot, we can see that the older the recipient is, the lower the weight gets.

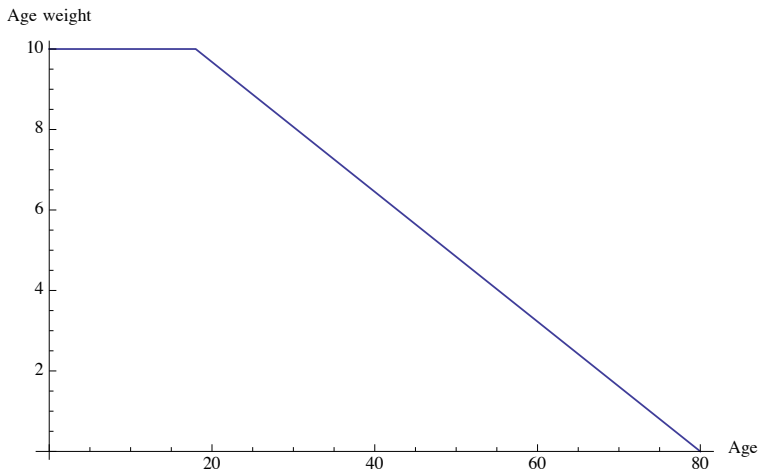


Figure 4.4.2: Piecewise Function for Age Weight Assignment.

Other possible functions for this weight assignment could be exponential functions, but after considering allocation and studying the arguments about age priority, we decided to use this piecewise function for our model because it was a better depiction of our opinions on the subject.

Blood Type: In this weight system, we prioritize incompatible pairs with blood type O^- recipients. This is because blood type O^- is the universal blood group of the general population of the United States, and thus there might be a lot of blood type O^- donors as well as a lot of blood type O^- recipients. The problem that arises is that blood type O^- donors can donate to every blood group, but blood type O^- recipients can only receive organs from the same blood type. If we do not give higher priority to blood type O^- recipients, all these donors might end up donating to recipients of other blood groups. This could cause the blood type O^- recipients to get longer waiting times for a kidney transplant. In our population of incompatible pairs, there will never be enough type O^- donors to allow all pairs having a type O^- recipient to match. Therefore, we assign a high weight to incompatible pairs containing a blood type O^- recipient. We assign them 10 points.

To ensure we had blood type O^- donors in our incompatible pairs population, we generated the blood type O^- donors and assumed they were incompatible with their respective recipient because of a positive crossmatch and that the recipient was sensitized as explained in Section 4.2.1. A sensitized recipient has difficulty finding a match and so when a possible match is found, this recipient should be highly considered for the exchange. Therefore, we assigned a weight of 5 points to these recipients. Also, these recipients, because they are incompatible with their blood type O^- donor, are allowing other exchanges.

We summarize the node weight scheme:

- First assign 25 points to all nodes.
- Recipient's Age-Weights or W_{ra}: Create a piecewise defined function that if the age of the recipient of node i is less than or 18 years old then assigns 10 points to the overall weight. If the age of the recipient of node i is more than 18 years old, then apply the continuous linear function defined above and find the weight corresponding to his/her age.
- Weights that are related to blood type or W_{bt}: If the recipient of node i has blood type O^-, then assign a weight of 10 points. If the donor of node i has blood type O^- then the recipient is sensitized and so add 5 points to its weight. Overall weight of node i is equal to the sum $W_{ra} + W_{bt}$.

All the functions to assign these weights to our incompatible pairs population were created in MATLAB. The codes can be found in Appendix C.

4.4.2 Weight on Edges: Location of Incompatible Pairs

The directed edges represent a match between two nodes, that is, two compatible incompatible donor-recipient pairs by blood type. We assign weights $EW_{i,j}$ to the edges of our directed graph. The weight is assigned to the edge connecting node i with node j. In our model, we assign weights to the edges based on the hospital location of incompatible pairs and the distance between these locations.

First of all, all edges start with 25 points. This is because every transplant is equally valuable. If the incompatible pairs involved in the exchange are in the same hospital, then we assume the hospital has the necessary doctors, staff, and operating rooms available. However, when they are treated in different hospitals, the donors of the pairs will usually travel to the hospital where the compatible recipient is located. This may be expensive

depending on the economic situation of the donor and might also result in a difficult family separation since the donor is a relative of a recipient receiving a kidney in another hospital.

As mentioned, we look at the distance between hospitals in New York State in miles as shown in Table 12.0.1 in Appendix D. We assign each directed edge a weight based on Figure 4.4.3. Because we are looking at hospitals in New York State only, some hospitals might be fairly close, other might be more than six hours away and so we use the following scheme:

- If the pairs involved in an exchange are in the same hospital, then assign the edge a weight of 10 points.
- If the distance between location of pair i and pair j is less than 20 miles, then assign the edge 5 points.
- If the hospitals in which pair i and pair j are located are more than or 20 miles apart but less than 150 miles from each other, assign 4 points.
- If the hospitals are more than or 150 miles apart but less than 255 miles from each other, assign 3 points.
- If the hospitals are more than or 255 miles apart but less than 350 miles from each other, assign 2 points.
- If the hospitals are more than or 350 miles apart from each other, assign 1 point.

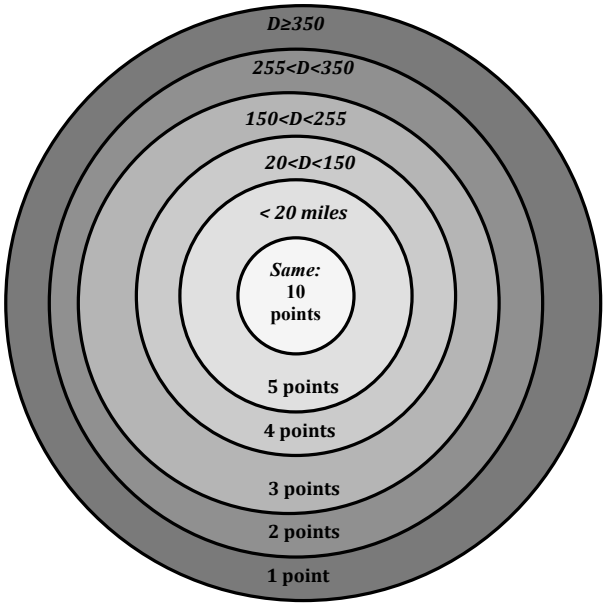


Figure 4.4.3: Edge Weight Assignment Scheme.

In this weight system we assumed that the farther apart the hospitals are, the harder it is for the donor to travel to the compatible recipient and so we assign less points. This is also discussed by Sommer Gentry et al. in *Kidney Paired Donation: Fundamentals, Limitations, and Expansions* [6]. They mention how in a cohort study, it was found that recipients of live donor kidney transplants with 2-8 hours of cold ischemia, the restriction in blood supply to tissues of organ, time did not have worse transplant function, or increased rates of acute rejection compared with transplants with less than 2 hours of cold ischemia time[6]. From this information, we infer that if the donor did not travel, but rather the kidney is transported to the other hospital, the organ would be viable and functioning if the hospitals are not very far away, that is, they are not more than 8 hours away as suggested by the study results.

All the functions used to assign weights to the directed edges were created in MATLAB. This code can be found in Appendix D.

4.5 Directed Three-Cycles: Three-Way Exchanges

In this model, we look at incompatibility mostly by blood type, and in a very few cases by positive crossmatch. As was expressed in the objective, our main emphasis is to maximize the number of kidney exchanges. We do this by looking at three-way exchanges where the only preference of recipients is compatibility. A three-way kidney exchange, as shown in Figure 4.5.1, involves three incompatible donor-recipient pairs i, j, k such that the donor of pair i is compatible with the recipient of pair j, the donor of pair j is compatible with the recipient of pair k, and the donor of pair k is compatible with the recipient of pair i. This could also work in the reverse direction. In these exchanges, the donors are not willing to donate unless their recipients can receive transplants. The exchange has to be a win-win situation for everyone involved.

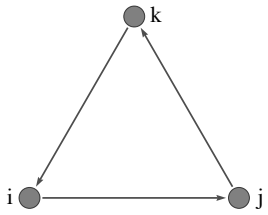
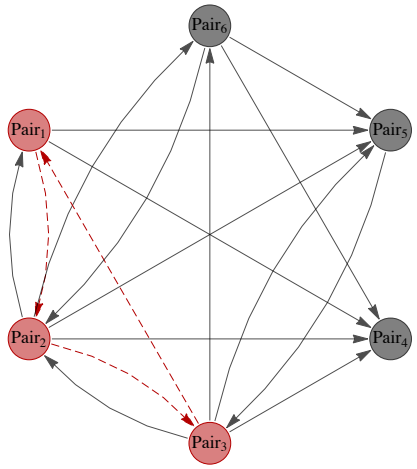


Figure 4.5.1: Directed Three-cycle.

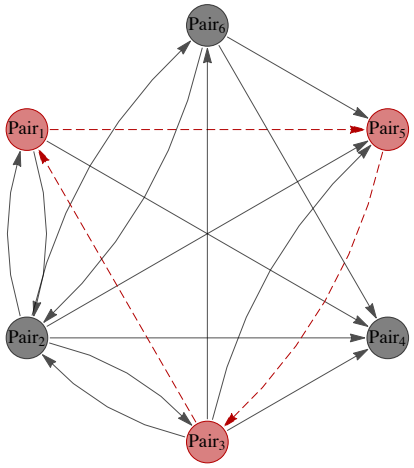
In our model, the matches between pairs are represented by directed edges. These directed edges indicate which donor is donating to which recipient. That is, an incoming edge means that the donor of the source pair is compatible with the recipient of the target pair. We find all possible directed three-cycles on the directed graph we create from the incompatible pairs blood type information. In these directed three-cycles we do not need reciprocal compatibility between the pairs involved; however, we might encounter some instances where they occur.

Figure 4.5.2 shows an example of directed three-cycles found in the exchange graph shown in Figure 4.3.2, which was obtained from the population sample from Table 4.2.4.

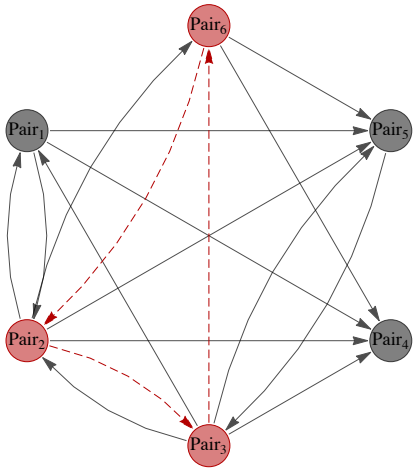
All the functions used to find the directed three-cycles were created in MATLAB. The code can be found in Appendix E. From this function we found five possible three-way exchanges between the incompatible pairs in the population described in Table 4.2.4. The first exchange is between incompatible pairs 1, 2 and 3, shown in Figure 4.5.2a. Other possible three-way exchanges are between pairs 1, 5 and 3, shown in Figure 4.5.2b, pairs 2, 3, and 6, shown in Figure 4.5.2c, pairs 2, 5, and 3, illustrated in Figure 4.5.2d, and finally between pairs 3, 6, and 5, shown in Figure 4.5.2e. From these possible exchanges we then have to decide which exchanges maximize the weights of the nodes as well as the weights of the directed edges. Therefore, we have created an algorithm, which will be discussed in the next chapter, that makes this decision.



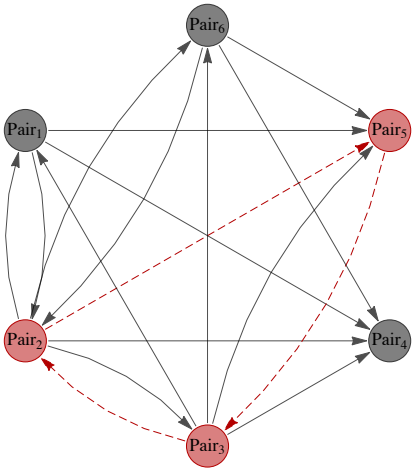
(a) Three-cycle $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$.



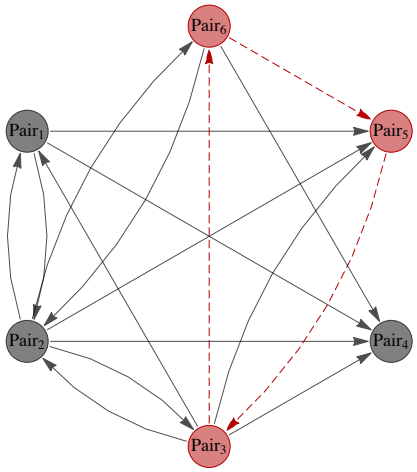
(b) Three-cycle $1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$.



(c) Three-cycle $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$.



(d) Three-cycle $2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$.



(e) Three-cycle $3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$.

Figure 4.5.2: Directed three-cycles in the graph from the population in Table 4.2.4

5
Algorithm for Three-Way Kidney Exchanges

In this chapter, we provide a complete description of our algorithm for selecting not only three-cycles with the highest weights, but also with the largest number of patients. This algorithm is created to solve the problem of overlapping three-cycles. This means that one or two incompatible donor-recipient pairs can be involved in several three-way exchanges. Note that if all the nodes involved in the three-cycle are not part of any other exchanges then they are automatically selected. The algorithm finds three-way exchanges by looking at weights assigned to the incompatible pairs represented by nodes, as well as weights assigned to the matches between two incompatible donor-recipient pairs represented by directed edges. With this algorithm, we can decide which three-cycles maximize the total weights of the three-cycle in the graph and suggest the best transplantation arrangement for patients.

5.1 No Three-Cycles Overlaps

When there is no overlapping of nodes, it means that all the incompatible donor-recipient pairs involved in the three-way exchanges have no pairs in common. For this scenario we

would not need an algorithm to select the best transplantation arrangements since they can all be possible because they are not subsets. That is, all the recipients and donors involved in these non-overlapping three-cycles may have unique characteristics.

5.2 Three-Cycles Overlaps

In the algorithm we have to consider a few general overlapping cases:

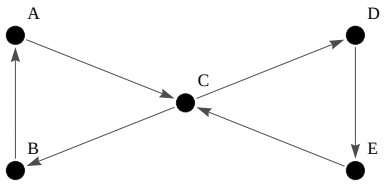


Figure 5.2.1: Overlapping Case 1.

- *Case 1:* The first general case as shown in Figure 5.2.1 consists of two three-cycles ACB and CDE sharing only one node in common, namely node C. First, we find the total edge weight and node weight for each three-cycle. Then, we compare these sums and if

$$\sum W_{ACB}^{nodes} > \sum W_{CDE}^{nodes},$$

and

$$\sum W_{ACB}^{edges} > \sum W_{CDE}^{edges},$$

then we choose the three-cycle ABC as a candidate for possible three-way exchanges. If we have a case such as,

$$\sum W_{ACB}^{nodes} > \sum W_{CDE}^{nodes}$$

and

$$\sum W_{ACB}^{edges} < \sum W_{CDE}^{edges},$$

then we consider only the weights of the nodes that are either donating to the common incompatible pair or receiving from the common incompatible pair. We do not consider edge weights for this decision, unless we need a tie breaker, because these weights are only based on the hospital location of the pairs in New York State. Also, node weight considers very important factors such as blood type O^- recipients and sensitized recipients, which are things that are highly taken into consideration in the transplantation process. When we compare the node weights for the final decision, we add the node weights of A and B, $W^{nodes}_{A,B}$ from three-cycle ACB, and node weights of D and E, $W^{nodes}_{D,E}$ from three-cycle CDE and compare these sums. The cycle with the highest sum will be chosen as a candidate for exchanges . The way we interpret this final choice is by observing that the chosen three-cycle has higher priority recipients than the other and so it should use the common incompatible pair for the exchanges.

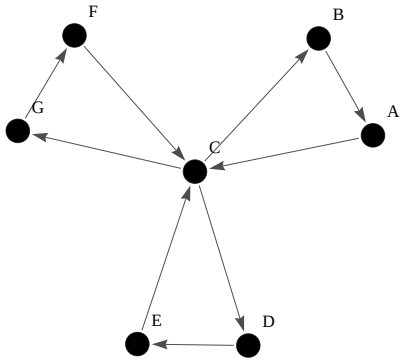


Figure 5.2.2: Overlapping Case 2.

- *Case 2:* The overlapping of Case 1 can also occur with three or more three-cycles, that is three or more cycles share a common node or incompatible pair. Figure 5.2.2 shows three-cycles CBA, CDE, CGF, where the common node is C. In this case, we follow the same procedure as in Case 1. However, the final comparison would be

done between the two three-cycles that have the highest total node weight and total edge weights.

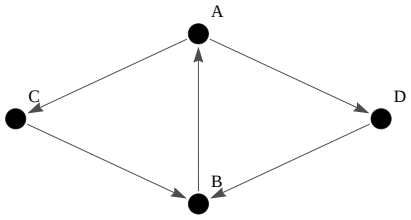


Figure 5.2.3: Overlapping Case 3.

- *Case 3:* In this case, two three-cycles share two nodes. In Figure 5.2.3 we see three-cycles ADB and ACB, where the two common nodes are A and B. We compare W_D^{nodes} and W_C^{nodes} to see which of the recipients have higher priority for a transplant based on these weights. The cycle containing the node with the highest weight will be chosen as a candidate for exchanges.

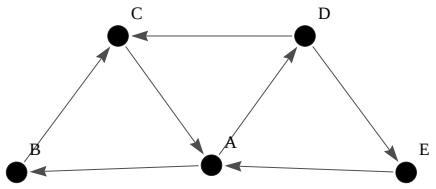


Figure 5.2.4: Overlapping Case 4.

- *Case 4:* In this last case, we have a combination of Cases 2 and 3. We give Figure 5.2.4 as an example. Here we see three-cycles BCA, CAD, and ADE. We see that some three-cycles of the overlap share only one node in common, namely three-cycles BCA and ADE, which share node A. For these three-cycles we apply the procedure for Case 1 or Case 2 and decide which of the two should use the incompatible pair A by comparing the sum of W_B^{nodes} and W_C^{nodes} from three-cycle BCA, and the sum of W_D^{nodes} and W_E^{nodes} from three-cycle ADE. Once we make this decision, we move

to the overlap of two common nodes, namely three-cycles BCA and CAD and also three-cycles CAD and ADE. From the decision of one node overlap we will also know which two node overlaps we should compare. For example, if from the one node overlap case we decide that three-cycle BCA should use the incompatible pair A because

$$W_B^{nodes} + W_C^{nodes} > W_D^{nodes} + W_E^{nodes},$$

that is, B and C have higher priority recipients, then we know that for the two nodes overlap we can only compare BCA and CAD because we have already established that node A should not be involved in three-cycle ADE. We compare BCA and CAD using the procedure from Case 3.

5.3 Algorithm Overview

In this section, we describe the steps of the algorithm to find which three-cycles maximize the total weights in the graph and include the most incompatible pairs.

Preliminary steps:

1. Generate a matrix containing incompatible pairs' blood type information, the recipient's age, the location of each incompatible pair given by the hospital where the patient receives care, and the distance between each hospital. This information was discussed in Section 4.2.
2. A directed graph G_N is created using the blood type information of the N incompatible pairs. This graph is represented by an adjacency matrix A. As shown in Section 4.3 with MATLAB code given in Appendix B.
3. The nodes and edges of G_N are assigned a weight using the weight system established in Section 4.4. Appendix C and D have the functions used for this.

4. Find all possible directed three-cycles in G_N using the adjacency matrix from Step 1 (Section 4.5 and Appendix E). In kidney exchange terms, in this step we find all possible three-way exchanges in the population of incompatible donor-recipient pairs. Check if there are any reciprocally compatible cycles, if so only leave one possibility.
5. Find the sum of edge weights and node weights for each directed three-cycle (Appendix F, Function 5).
6. Find the directed three-cycles that have one or two nodes in common (Appendix F, Function 1).
7. Find the number of times each node appears in an exchange (Appendix F, Function 2). This will helpful for the following step.
8. Find the node that is involved in the maximum number of exchanges and identify which exchanges it belongs to (Appendix F, function 4). This node in most cases will have a blood type O^- donor since this is the blood type of universal donors. We call this node n_{max} (Appendix F, function 3).

Solution Steps:

9. Find the total node weight and total edge weight for each directed three-cycle that involves n_{max} and compare these sums in order to find the three-cycles that have the maximum total weight (Appendix F, Function 6 and Function 7).
10. If you get only one cycle in Step 9 for this iteration it means you found only one cycle that maximizes the weights and thus skip this step and go to Step 11. Otherwise, in this part of the algorithm, we identify the cases.

    - Check if the three-cycles obtained from the previous step, apart from n_{max}, have another node in common. If they do, you have encountered a Case 3 overlap as defined in Section 5.2. Otherwise, follow the procedure for Case 1 or Case 2.

- If during this check you find that the nodes that distinguish the three-cycles obtained in Step 9 are in another three-cycle together, include the new three-cycle for the comparisons. This will give you an overlapping Case 4 and thus use the procedure for Case 4 to make the decision. If you do not encounter this situation, then follow the procedure for overlapping Case 3. You should keep track of every step for each iteration so that you do not have to repeat comparisons.

11. Store the resulting three-cycle for the iteration and repeat steps $8 - 11$, where the n_{max} becomes the node with the next highest cycle count.
12. Once n_{max} reaches zero, the remaining nodes do not participate in any possible exchanges because they were not found to be part of any three-cycle. Return to Step 4 where you found all three-cycles in the population, and for each of the three-cycles you stored as candidates in Step 11, eliminate all three-cycles from the list of three-cycles that contain the stored cycle's participants. If you find another non-overlapping three-cycle, store it with all the possible exchanges along with the three-cycle from which it was found. If you do not find any other three-cycles after this deletion continue to Step 13.
13. If you get additional non-overlapping three-cycles from Step 12, pick the largest group. Otherwise, look at the stored three-cycles from Step 11 and if they do not share any nodes, that is they have no overlaps, choose all of them as possible exchanges. If they overlap, see in which case these overlaps fall and do as explained for this overlapping case at the beginning of the chapter. From this final step, we get the maximum weighted three-cycles possible that include the highest amount of patients from the population of incompatible pairs.

6

Algorithm Implementation on Populations of $N = 6$ and $N = 20$

In this chapter, we implement the algorithm using two different samples of incompatible donor-recipient pairs.

6.1 Small Example: $N = 6$

We apply the algorithm to the example population from Table 4.2.4. We use a small population of incompatible pairs for a better illustration of how the algorithm works. For this implementation of the algorithm we only use blood type and do not consider Rh. Therefore, we use the distribution of blood type ABO.

Step 1: In this step, we look at the characteristics of our incompatible pair population. For this example the characteristics of a random population of six incompatible donor-recipient pairs($N = 6$) can again be found in Table 6.1.1.

Step 2: We create a directed graph G_6 as a representation of our incompatible pair population. The nodes of the graph serve as representation of the six incompatible donor-recipient pairs. The directed edges show the matches between the pairs. The following is

Donor's Blood Type	Recipient's Blood Type	Recipient's Age	Location of Pair
A	O	31	Montefiore Medical Center
O	A	78	Erie County Medical Center
O	B	72	North Shore University hospital
AB	A	13	Montefiore Medical Center
B	A	64	New York University Medical center
A	O	38	Albany Medical Center Hospital

Table 6.1.1: Population Characteristics for example of N=6.

the adjacency matrix of the graph of this population:

$$A_G = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

In this matrix the ones represent a match between incompatible pairs. Figure 6.1.1 contains the graph for this population. We also calculate the density of the directed using the formula given in definition 2.1.8. We get that our graph has 17 directed edges out of the 30 maximum possible edges between the six nodes.

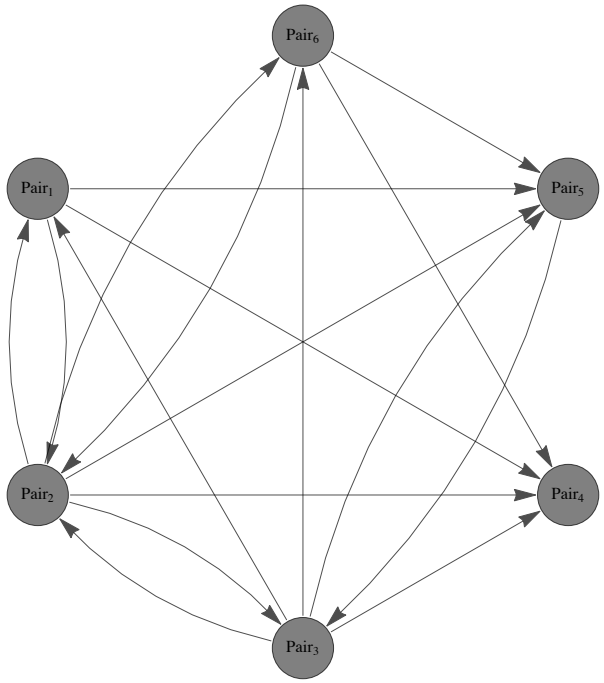


Figure 6.1.1: Example of graph from the population in Table 6.1.1

Step 3: The nodes and directed edges of graph G_6 are assigned a weight using the schemes developed in Sections 4.4.1 and 4.4.2. We used the functions created in MATLAB and defined in Appendices C and D. These functions use the information for weight assignment shown in Figures 4.4.1 and 4.4.3. The following matrices in Table 6.1.2 contain these results.

$Pairs$	$Weight$
1	43
2	30
3	31
4	35
5	28
6	42

(a) Node Weights.

$Tail$	$Head$	$Weight$
1	2	26
1	4	35
1	5	30
2	1	26
2	3	26
2	4	26
2	5	26
2	6	27
3	1	30
3	2	26
3	4	30
3	5	30
3	6	28
5	3	30
6	2	27
6	4	29
6	5	29

(b) Edge Weights.

Table 6.1.2: Weight assignments to graph G_6.

From the matrix in 6.1.2a, we see that node or pair one has a weight of 43 which means the pair has a high priority patient. In Table 6.1.2b, we see that pair one can be matched with pairs 2, 4 or 5. However, we see that it has a higher edge weight, that is the treatment locations are closer, when matched with pair 4.

Step 4: For this population of six incompatible pairs there are five possible directed three-cycles or three-way kidney exchanges. These are discussed in Section 4.5 and Figure 4.5.2. These directed three-cycles are :

$$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1,$$

$$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1,$$

$$2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2,$$

$$2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2,$$

and three-cycle

$$3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3.$$

Step 5: In this step, we find the sum of of the node weights and edge weights of each three-cycle. Table 6.1.3 has the results. We use Function 5 in Appendix F.

Three-cycles	$EW_{ij} + EW_{jk} + EW_{ki}$	$NW_i + NW_j + NW_k$
$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$	82	125
$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$	90	120
$2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$	81	125
$2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$	82	115
$3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$	87	120

Table 6.1.3: Sum of Node weights and Edge Weights for all possible three-cycles when $N = 6$.

Step 6: In this step, we find the three-cycles that share nodes in common. We use the MATLAB Function 1 defined in Appendix F. Table 6.1.4 summarizes the results.

In Table 6.1.4, C_{node1} corresponds to having one node in common and C_{node2} to having two. When C_{node2} is zero, it simply means that the three-cycles only overlap through one node.

Step 7: In this step, we find the number of three-way exchanges in which a single incompatible pair could participate in. We used Function 2 from Appendix F. Table 6.1.5 has the information.

Three cycle A	Three cycle B	C_{node1}	C_{node2}
$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$	$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$	1	3
$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$	$2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$	2	3
$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$	$2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$	2	3
$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$	$3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$	3	0
$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$	$2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$	3	0
$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$	$2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$	5	3
$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$	$3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$	5	3
$2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$	$2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$	2	3
$2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$	$3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$	3	6
$2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$	$3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$	5	3

Table 6.1.4: Common nodes in three cycles for $N = 6$.

Pair	Three-cycle-count
1	2
2	3
3	5
4	0
5	3
6	2

Table 6.1.5: Node Count in Three-cycles when $N = 6$.

ITERATION 1:

- **Step 8:** We find that $n_{max} = 3$ for this iteration of the algorithm using Function 3 of Appendix F. From the results of Step 7, we note that this n_{max} node has the maximum possible three-way exchanges. This is also the result of applying Function 4 from Appendix F.

- **Step 9:** Since n_{max} (or node 3) is included in all exchanges, the sums of node and directed edge weights of all three-cycles with n_{max} is the same as shown in Table 6.1.3 in Step 5. Using these sums, we found that the three-cycle with maximum total node and edge weight when $n_{max} = 3$ is $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$.

- **Step 10:** Since we only found one three-cycle that maximized the total weights, we skip this step and move on to Step 11. Otherwise, we would have to look for other

common nodes in the three-cycles aside from the n_{max} node and then follow the procedure for the overlapping case that applies.

- **Step 11:** We store three-cycle $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$ as a candidate for a possible three-way kidney exchange from this population of incompatible pairs. We save the exchange in the following matrix, to which we will add all the candidates we find in all iterations.

$$Exchanges_{possible} = \begin{bmatrix} 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1 \end{bmatrix}$$

Now we move on to the pair with the next highest cycle count.

ITERATION 2:

- **Step 8:** From Step 7, we see that there are two incompatible pairs with the same "next highest cycle count." We pick one at random. We chose $n_{max} = 2$ for this iteration of the algorithm, meaning that node 2 is part of the most three-cycles. Applying Function 4 from Appendix F, we find that this n_{max} is involved in 3 three-way exchanges. Specifically, $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$, $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$, and $2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$.

- **Step 9:** The sums of node and directed edge weights of all three-cycles with n_{max} are shown in Table 6.1.6 below. Using these sums, we found that the three-cycle with maximum total node and edge weight when $n_{max} = 2$ is $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$.

Three-cycles	$EW_{ij} + EW_{jk} + EW_{ki}$	$NW_i + NW_j + NW_k$
$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$	82	125
$2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$	81	125
$2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$	82	115

Table 6.1.6: Sum of Node weights and Edge Weights for all possible three-cycles when $n_{max} = 2$.

- **Step 10:** Since we only found one three-cycle that maximized the total weights, we skip this step and move on to Step 11.

- **Step 11:** Since three-cycle $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$ is already stored as a possible three-way kidney exchange, we move on to the pair with the next highest cycle count.

ITERATION 3:

- **Step 8:** In the previous iteration we noticed that there were two incompatible pairs with the same "next highest cycle count," namely pair 2 and pair 5. We looked at pair 2 and thus now we let $n_{max} = 5$ for this iteration of the algorithm. Applying Function 4 from Appendix F, we find that this n_{max} is involved in 3 three-way exchanges. Namely, $1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$, $2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$, and $3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$.

- **Step 9:** Since $n_{max} = 5$, the sums of node and directed edge weights of all three-cycles with n_{max} are shown in Table 6.1.7 below. Using these sums, we found that the three-cycle with maximum total node and edge weight when $n_{max} = 5$ is $1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$.

Three-cycles	$EW_{ij} + EW_{jk} + EW_{ki}$	$NW_i + NW_j + NW_k$
$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$	90	120
$2 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2$	82	115
$3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$	87	120

Table 6.1.7: Sum of Node weights and Edge Weights for all possible three-cycles when $n_{max} = 5$.

- **Step 10:** Since we only found one three-cycle that maximized the total weights, we skip this step and move on to Step 11.

- **Step 11:** We store three-cycle $1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$ as a possible three-way kidney exchange. We save the exchange in the following matrix together with the result

from past iterations.

$$Exchanges_{possible} = \begin{bmatrix} 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1 \\ 1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1 \end{bmatrix}$$

Now we move on to the pair with the next highest cycle count.

ITERATION 4:

- **Step 8:** From Step 7 we see that there are two incompatible pairs with the same "next highest cycle count," namely pair 1 and pair 6. For this iteration, we let $n_{max} = 1$. Applying Function 4 from Appendix F, we find that this n_{max} is involved in 2 three-way exchanges. Namely,

$$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1,$$

and

$$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1.$$

- **Step 9:** The sums of node and directed edge weights of all three-cycles with n_{max} are shown in Table 6.1.8 below. Using these sums, we found that the three-cycle with maximum total node and edge weight when $n_{max} = 1$ is $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$.

Three-cycles	$EW_{ij} + EW_{jk} + EW_{ki}$	$NW_i + NW_j + NW_k$
$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$	82	125
$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$	90	120

Table 6.1.8: Sum of Node weights and Edge Weights for all possible three-cycles when $n_{max} = 1$.

- **Step 10:** Since we only found one three-cycle that maximized the total weights, we skip this step and move on to Step 11.

- **Step 11:** Since three-cycle $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$ is already stored as a possible three-way kidney exchange, we move on to pair 6 which has the same cycle count as this n_{max}.

ITERATION 5:

- **Step 8:** For this iteration, we let $n_{max} = 6$. Applying Function 4 from Appendix F, we find that this n_{max} is involved in 2 three-way exchanges. These are: $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$, and $3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$.

- **Step 9:** The sums of node and directed edge weights of all three-cycles with n_{max} are shown in Table 6.1.9 below. Using these sums, we found that the three-cycle with maximum total node and edge weight when $n_{max} = 6$ is $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$.

Three-cycles	$EW_{ij} + EW_{jk} + EW_{ki}$	$NW_i + NW_j + NW_k$
$2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$	81	125
$3 \longrightarrow 6 \longrightarrow 5 \longrightarrow 3$	87	120

Table 6.1.9: Sum of Node weights and Edge Weights for all possible three-cycles when $n_{max} = 6$.

- **Step 10:** Since we only found one three-cycle that maximized the total weights, we skip this step and move on to Step 11.

- **Step 11:** We store three-cycle $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$ as a candidate for a possible three-way kidney exchange. We save the exchange in the following matrix together with the result from past iterations.

$$Exchanges_{possible} = \begin{bmatrix} 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1 \\ 1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1 \\ 2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2 \end{bmatrix}$$

  Now we move on to the pair with the next highest cycle count.

ITERATION 6:

- **Step 8:** We find that there are no more n_{max} nodes. This means that the remaining node is not involved in any three-cycle. We stop the iterations here.

END

Step 12: We now look at the stored possible three-cycles shown in the following matrix.

$$Exchanges_{possible} = \begin{bmatrix} 1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \\ 1 \rightarrow 5 \rightarrow 3 \rightarrow 1 \\ 2 \rightarrow 3 \rightarrow 6 \rightarrow 2 \end{bmatrix}$$

We first delete all the three-cycles from the list of three-cycles in Step 4, that involve pairs 1, 2, and 3, and get no other possible cycles, indicating that all the cycle's patients are present in all other exchanges. We do the same for pairs 1, 5, and 3 and then for pairs 2, 3, and 6. We find that for each cycle we end up deleting all the exchanges because pair 3 is part of all of them. We proceed to step 13.

Step 13: Since we did not find any other non-overlapping cycles, we need to decide which one of the stored exchanges should be selected. We know that pair 3 is in all of the final exchanges, which we also discussed in the first iteration of the algorithm and in the previous step. Therefore, from these three possible exchanges only one could be picked since a node cannot be involved in more than one match.

We identify the overlapping cases in which these three-cycles fall under. These three-cycles give us an example of a Case 4 overlap. We use the information of common nodes obtained in Step 6 of this implementation. The graph in Figure 6.1.2 shows how these three-cycles overlap in a Case 4.

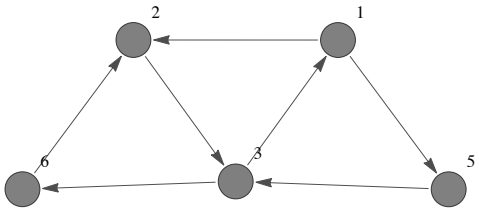


Figure 6.1.2: Case 4.

As explained in Section 5.2, this case is a combination of cases 1 or 2 and 3. This can be seen in the graph. We break this graph into these smaller cases to help us further simplify the problem. We first look at the possible three-cycles that fall under overlap-

ping Case 1, namely three-cycles $1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$ and $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$. These are shown in Figure 6.1.3. We follow the procedure for Case 1. That is, we compare the

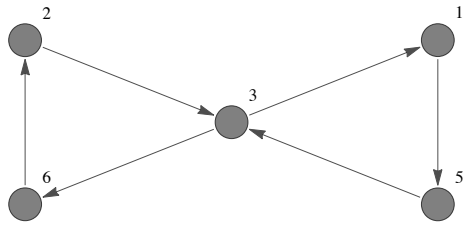


Figure 6.1.3: Case 1:
$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$ and $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$.

$\sum W_{2,6}^{nodes}$ to $\sum W_{1,5}^{nodes}$. These weights are obtained from Step 3. For this case, we get that $\sum W_{2,6}^{nodes} > \sum W_{1,5}^{nodes}$. This means that the recipients of pairs 2 and 6 are higher priority patients than those of pairs 1 and 5. And so, from this case we select three-cycle $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$. Now we look at those three-cycles that overlap like Case 3.

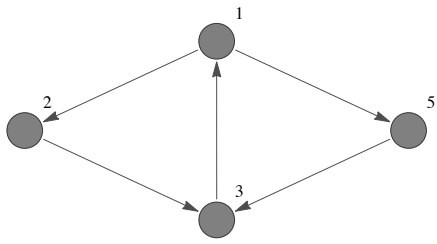


Figure 6.1.4: Case 2:
$1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$ and $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$.

Figure 6.1.4 shows the overlap between $1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$ and $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$. These three-way exchanges share two nodes. Following the procedure for Case 3, we compare W_2^{nodes} and W_5^{nodes} and find that $W_2^{nodes} > W_5^{nodes}$. Therefore, we choose three-cycle $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$. Note that we do not worry about the edge weights unless we need a tie breaker. This is because in our model we are only looking at hospitals in New York State, and thus we do not consider location as the major determining factor. In the procedure for Case 4, once we find the maximum three-cycle from Case 1 or 2, we only do comparisons

between the three-cycles that contain it. This means that we would have done the next comparison right after the Case 1 procedure.

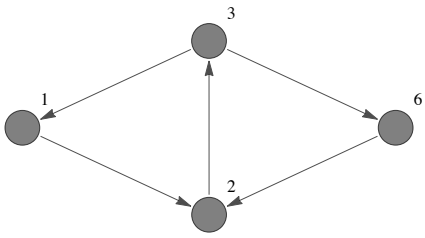


Figure 6.1.5: Case 2:
$2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$ and $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$.

As shown in Figure 6.1.5 above, three-cycles $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$ and $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$ have two common nodes. Applying the procedure for Case 3, we find that $W_1^{nodes} > W_6^{nodes}$. And therefore, we choose three-cycle $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$ as the result for this case.

Now that we have compared all possible exchanges in their corresponding overlapping case, we get three-cycle $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$.

In terms of three-way kidney exchanges, from this population only the exchange between the donor of pair 1 and the recipient of pair 2, the donor of pair 2 and recipient of pair 3, and the donor of pair 3 and recipient of pair 1, can occur. Also, if we look at the node weights in Table 6.1.2a of Step 3, which represent the priority of the pair, we see that these three pairs have high weights. We also see that pair 1 is the one with the greatest need.

In this example, all the three-cycles were overlapping and thus implementing the algorithm gave us only one possible three-way exchange. This might not have been the best example in terms of giving us various exchanges; however, this was a great example to illustrate all the general overlapping cases that this model studies.

6.2 Example with a Population of 20 Incompatible Pairs

In this example we generate a more realistic size population with 20 incompatible pairs using blood type and Rh. The following is the implementation of the algorithm presented:

Step 1: In this step we look at the characteristics of our incompatible pair population which are shown in Table 6.2.1.

Donor	Recipient	Recipient's age	Location of Pair
A^+	O^+	34	5
A^+	O^+	59	3
B^+	A^+	58	2
A^+	O^+	77	1
B^+	O^+	63	10
O^-	B^+	31	10
A^-	O^-	58	2
O^+	O^-	36	3
A^+	O^+	52	10
B^+	O^+	61	11
O^-	A^-	53	5
AB^+	O^+	36	11
O^+	A^-	31	11
O^+	O^-	35	10
A^+	O^+	52	10
A^+	A^-	51	8
B^+	O^-	60	1
A^+	O^+	48	4
A^+	O^-	27	2
O^+	O^-	12	3

Table 6.2.1: Population Characteristics for example of N=20.

Step 2: We create a directed graph G_{20} as a representation of our incompatible pair population. The nodes of the graph serve as representation of the twenty incompatible donor-recipient pairs. The directed edges show the matches between the pairs. Figure 6.2.1 shows the graph for this population. As it can be seen from the graph, as the sample size is increased, the directed graph becomes more complex, and so it is harder to identify three-cycles. This is because the density of this graph is 118/380. This means that out

of the 380 possible directed edges between the 20 pairs, this graph only contains 118 of them.

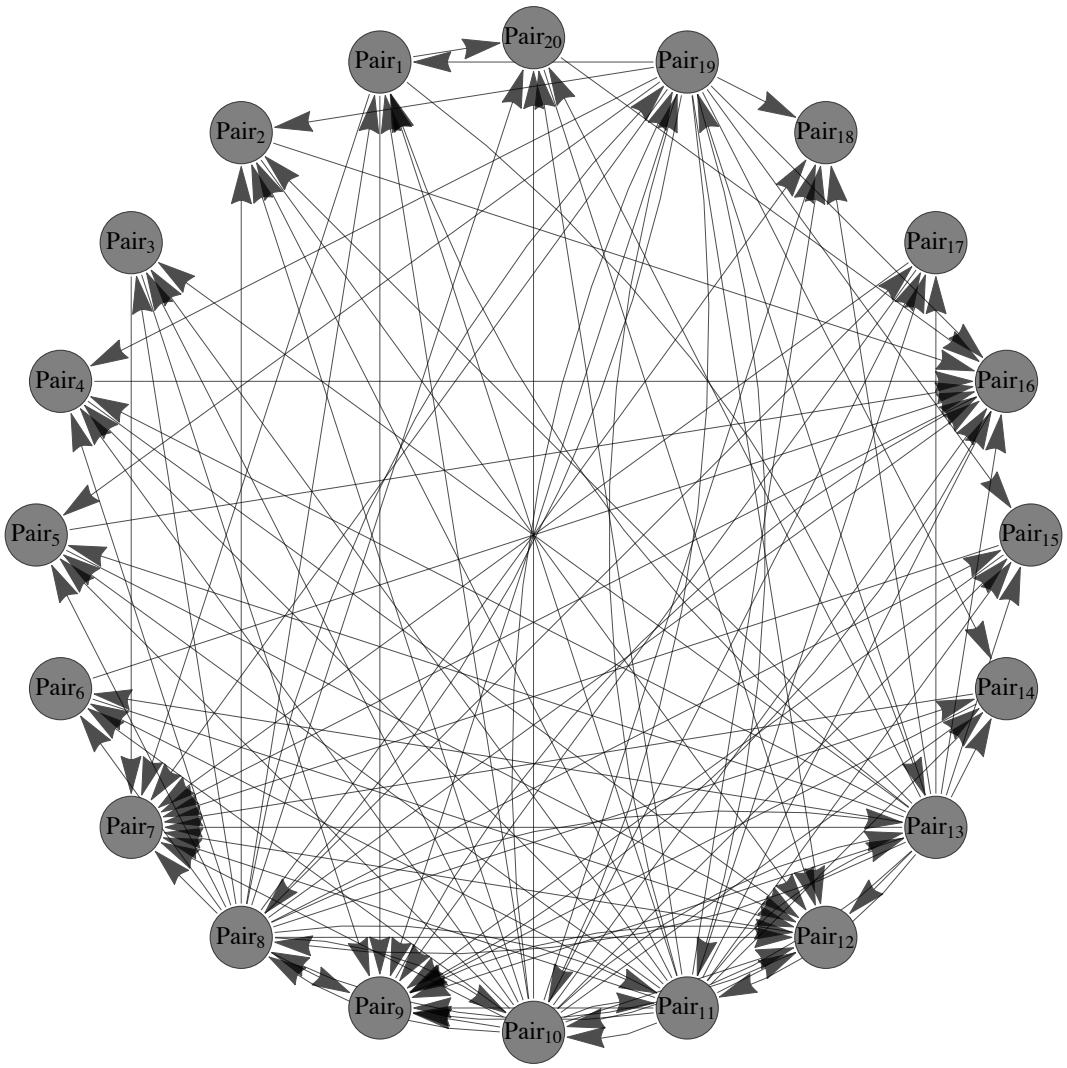


Figure 6.2.1: Exchange Directed Graph for N=20

Step 3: The nodes and directed edges of graph G_{20} are assigned a weight using the schemes developed in Sections 4.4.1 and 4.4.2. We do not show these weights here because there are 118 matches or edges between pairs to consider, but they can be easily calculated using the functions created in MATLAB and defined in Appendices C and D.

Step 4: For this population of 20 incompatible pairs there are 21 possible three-cycles or three-way kidney exchanges. The possible three-cycles are shown in Table 6.2.2. Note that there are some reciprocally compatible cycles; that is, within one three-cycle there are two possible exchanges, for instance $8 \longrightarrow 10 \longrightarrow 13 \longrightarrow 8$ and $8 \longrightarrow 13 \longrightarrow 10 \longrightarrow 8$ are two possible three-way exchanges between pairs 8, 10 and 13. In this model we can choose either one. See discussion in Section 6.3 for a more detailed explanation.

Three-cycles
$1 \longrightarrow 13 \longrightarrow 8 \longrightarrow 1$
$1 \longrightarrow 13 \longrightarrow 10 \longrightarrow 1$
$1 \longrightarrow 13 \longrightarrow 11 \longrightarrow 1$
$1 \longrightarrow 13 \longrightarrow 19 \longrightarrow 1$
$8 \longrightarrow 10 \longrightarrow 11 \longrightarrow 8$
$8 \longrightarrow 10 \longrightarrow 13 \longrightarrow 8$
$8 \longrightarrow 10 \longrightarrow 19 \longrightarrow 8$
$8 \longrightarrow 11 \longrightarrow 10 \longrightarrow 8$
$8 \longrightarrow 11 \longrightarrow 13 \longrightarrow 8$
$8 \longrightarrow 11 \longrightarrow 19 \longrightarrow 8$
$8 \longrightarrow 13 \longrightarrow 10 \longrightarrow 8$
$8 \longrightarrow 13 \longrightarrow 11 \longrightarrow 8$
$8 \longrightarrow 13 \longrightarrow 19 \longrightarrow 8$
$8 \longrightarrow 19 \longrightarrow 10 \longrightarrow 8$
$8 \longrightarrow 19 \longrightarrow 11 \longrightarrow 8$
$10 \longrightarrow 11 \longrightarrow 13 \longrightarrow 10$
$10 \longrightarrow 11 \longrightarrow 19 \longrightarrow 10$
$10 \longrightarrow 13 \longrightarrow 11 \longrightarrow 10$
$10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10$
$10 \longrightarrow 19 \longrightarrow 11 \longrightarrow 10$
$11 \longrightarrow 13 \longrightarrow 19 \longrightarrow 11$

Table 6.2.2: All Three-cycles in a Population of N=20.

Step 5: In this step, we find the sum of the node weights and edge weights of each three-cycle. These can be found using Function 5 in Appendix F.

Step 6: In this step, we find the three-cycles that share nodes. We do not show the results, but they can be found using the MATLAB Function 1 defined in Appendix F.

Step 7: In this step, we find the number of times a single incompatible pair participates in a three-way exchange. We used Function 2 from Appendix F. The results are summarized in Table 6.2.3.

Pair	Three-cycle-count
1	4
8	12
10	12
11	12
13	13
19	10

Table 6.2.3: Node Count in Three-cycles when $N = 20$.

The remaining 14 incompatible pairs in the population were unmatchable because there were no donors compatible with the pairs to form a three-cycle.

ITERATION 1:

- **Step 8:** We find that $n_{max} = 13$ using Function 3 of Appendix F. We can easily find the three-cycles in which this n_{max} is involved using Function 4 in Appendix F.

- **Step 9 and Step 10:** The result from these steps is that the three-cycle with maximum total weight is three-cycle $10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10$. Since we got more than one cycle with maximum total weight, we went through the cases and followed the procedure.

- **Step 11:** We store three-cycle $1 \longrightarrow 13 \longrightarrow 19 \longrightarrow 1$ as a possible three-way kidney exchange. We save the exchange in the following matrix together with the result from past iterations.

$$Exchanges_{possible} = \big[10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10\big]$$

Now we move on to the pair with the next highest cycle count.

6. ALGORITHM IMPLEMENTATION ON POPULATIONS OF $N = 6$ AND $N = 2068$

ITERATION 2:

- **Step 8:** We find that $n_{max} = 8$ using Function 3 of Appendix F. We can easily find the three-cycles in which this n_{max} is involved using Function 4 in Appendix F.
- **Step 9 and Step 10:** The result from these steps is that the three-cycle with maximum total weight is three-cycle $8 \longrightarrow 13 \longrightarrow 19 \longrightarrow 8$. We do the same as the previous iteration.
- **Step 11:** We store three-cycle $8 \longrightarrow 13 \longrightarrow 19 \longrightarrow 8$ as a possible three-way kidney exchange. We save the exchange in the following matrix together with the result from past iterations.

$$Exchanges_{possible} = \begin{bmatrix} 10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10 \\ 8 \longrightarrow 13 \longrightarrow 19 \longrightarrow 8 \end{bmatrix}$$

Now we move on to the pair with the next highest cycle count.

ITERATION 3:

- **Step 8:** We find that $n_{max} = 10$ using Function 3 of Appendix F. We can easily find the three-cycles in which this n_{max} is involved using Function 4 in Appendix F.
- **Step 9 and Step 10:** The result from these steps is that the three-cycle with the maximum total weight is three-cycle $10 \longrightarrow 13 \longrightarrow 10 \longrightarrow 10$. We do the same as the previous iteration.
- **Step 11:** Three-cycle $10 \longrightarrow 13 \longrightarrow 10 \longrightarrow 10$ is already stored as a possible three-way kidney exchange.

We move on to the pair with the next highest cycle count.

ITERATION 4:

- **Step 8:** We find that $n_{max} = 11$ using Function 3 of Appendix F. We can easily find the three-cycles in which this n_{max} is involved using Function 4 in Appendix F.
- **Step 9 and Step 10:** The result from these steps is that the three-cycle with maximum total weight is three-cycle $11 \longrightarrow 13 \longrightarrow 19 \longrightarrow 11$. We do the same as the previous iteration.
- **Step 11:** We store three-cycle $11 \longrightarrow 13 \longrightarrow 19 \longrightarrow 11$ as a possible three-way kidney exchange.

$$Exchanges_{possible} = \begin{bmatrix} 10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10 \\ 8 \longrightarrow 13 \longrightarrow 19 \longrightarrow 8 \\ 11 \longrightarrow 13 \longrightarrow 19 \longrightarrow 11 \end{bmatrix}$$

Now we move on to the pair with the next highest cycle count.

ITERATION 5:

- **Step 8:** We find that $n_{max} = 19$ using Function 3 of Appendix F. We can easily find the three-cycles in which this n_{max} is involved using Function 4 in Appendix F.
- **Step 9 and Step 10:** The result from these steps is that the three-cycle with maximum total weight is three-cycle $10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10$. We do the same as the previous iteration,that is, we look at the cases.
- **Step 11:** Three-cycle $10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10$ is already in the group of possible three-way kidney exchanges. We move on to the pair with the next highest cycle count.

ITERATION 6:

- **Step 8:** We find that $n_{max} = 1$ using Function 3 of Appendix F. We can easily find the three-cycles in which this n_{max} is involved using Function 4 in Appendix F.

- **Step 9 and Step 10:** The result from these steps is that the three-cycle with maximum total weight is three-cycle $1 \longrightarrow 13 \longrightarrow 19 \longrightarrow 1$. We did the same as the previous iteration.

- **Step 11:** We store three-cycle $1 \longrightarrow 13 \longrightarrow 19 \longrightarrow 1$ as a possible three-way kidney exchange.

$$Exchanges_{possible} = \begin{bmatrix} 10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10 \\ 8 \longrightarrow 13 \longrightarrow 19 \longrightarrow 8 \\ 11 \longrightarrow 13 \longrightarrow 19 \longrightarrow 11 \\ 1 \longrightarrow 13 \longrightarrow 19 \longrightarrow 1 \end{bmatrix}$$

END

Step 12: After deleting all the three-cycles from the list of three-cycles in Step 4 with pairs 10, 13, and 19, we are left with no three-cycles. When we delete all of those with pairs 8, 13, and 19, we are also left with no other cycles. We get the same result with pairs 11, 13, and 19. When we follow this procedure with pairs 1, 13, and 19, we get three-cycle $8 \longrightarrow 10 \longrightarrow 11 \longrightarrow 8$ or $8 \longrightarrow 11 \longrightarrow 10 \longrightarrow 8$, which are reciprocal cycles. We store this in the following matrix.

$$Exchanges_{possible} = \begin{bmatrix} 10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10 & \\ 8 \longrightarrow 13 \longrightarrow 19 \longrightarrow 8 & \\ 11 \longrightarrow 13 \longrightarrow 19 \longrightarrow 11 & \\ 1 \longrightarrow 13 \longrightarrow 19 \longrightarrow 1 & 8 \leftrightarrow 10 \leftrightarrow 11 \leftrightarrow 8 \end{bmatrix}$$

Step 13: From the possible exchanges we see that exchange $1 \longrightarrow 13 \longrightarrow 19 \longrightarrow 1$, unlike the other possible exchanges, gives two possible non-overlapping exchanges. Note that all the stored possible three-cycles share pairs 13 and 19, which gives us an overlapping Case 3. We compare the node weights of pairs 1, 8, 10 and 11. If we were to only look at the three-cycle with maximum node and edge weight for this case, we would choose three-cycle $10 \longrightarrow 13 \longrightarrow 19 \longrightarrow 10$; however, three-cycle $1 \longrightarrow 13 \longrightarrow 19 \longrightarrow 1$ gives us the possibility of performing six exchanges and involves patients with high node weights. Therefore, we

pick this three-cycle and its derivative, that is three-cycle $8 \longrightarrow 10 \longrightarrow 11 \longrightarrow 8$ or $8 \longrightarrow 11 \longrightarrow 10 \longrightarrow 8$.

6.3 Discussion of the Results

As we can see from the algorithm executions, there was a lot overlapping between matches and thus overlapping between three-cycles. These overlaps do not allow the algorithm to give us many maximum three-cycles from the simulated population. We also see this on the graph for each sample population, where the directed graph corresponding to the 20 incompatible pairs is so dense that you can barely see which pairs are getting matched. This was also shown by the density calculation where we found that the graph of this population only has 31% of the maximum possible number of directed edges between the 20 nodes. Therefore, if the graph was to include all of them it will get convoluted.

From the population of six incompatible donor-recipient pairs, we found that out of the five three-way exchanges existing in the population, only one exchange is possible, $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$. This is the only exchange possible from this population because, as mentioned in the algorithm implementation, incompatible pair 3 is present in all of the candidate three-cycles. This three-way exchange may not be the most convenient in terms of treatment location since other exchanges have higher total edge weight, but it happens to involve the patient with the highest priority, namely the recipient from incompatible pair 1, which is very important. Because of this high weight, the recipient of pair 1 should receive a transplant earlier. Also, in Table 6.1.1, we can see that this higher weight is due to the fact that the recipient of pair 1 has blood type O.

In exchange terms, the end result of the algorithm's execution in this population is that the donor of pair 1 donates to the recipient of pair 2, the donor of pair 2 donates to the recipient of pair 3, and the donor of pair 3 donates to the recipient of pair 1. This results in three transplantations and six surgical procedures, two for each donor-recipient exchange.

The donor's nephrectomy (removal of kidney) is performed at the same time to prevent the possibility of one donor withdrawing his or her commitment after the other donor has undergone the nephrectomy as discussed in Section 1.4.2. Because of the six surgical procedures, this exchange will need more coordination than a two-way exchange since more operating rooms and surgical teams would be required. In this particular exchange, the donor of pair 1 will have to travel to Erie County Medical Center, which is the hospital we assigned to pair 2. The donor of pair 2 will travel to North Shore University hospital and the donor of pair 3 will travel to Montefiore Medical Center where its compatible recipient is being treated.

From the population of 20 incompatible donor-recipient pairs, only two exchanges are possible from the 21 three-way exchanges found in the population because of extensive three-cycles overlaps. These are $1 \longrightarrow 13 \longrightarrow 19 \longrightarrow 1$ and $8 \longrightarrow 10 \longrightarrow 11 \longrightarrow 8$ or $8 \longrightarrow 11 \longrightarrow 10 \longrightarrow 8$. In the results we learned that three-cycles $8 \longrightarrow 10 \longrightarrow 11 \longrightarrow 8$ and $8 \longrightarrow 11 \longrightarrow 10 \longrightarrow 8$ are reciprocal cycles; that is, these three pairs can exchange donors in two possible ways. In this model, we would not be able to decide which of the two is better because we get the same total weights and so we can pick either one. However, for this decision the best test would be a tissue typing test because it helps to determine which match is stronger, resulting in a better outcome. That is, it helps decide if it is better for the donor of pair 8 to donate to the recipient of pair 10 or if the donor of pair 10 should donate to the recipient of pair 8. We discuss this as a future addition to the model in Section 8.2.3.

In terms of exchanges in this population, in the first three-way living donor exchange, the donor of pair 1 donates to the recipient of pair 13, the donor of pair 13 donates to the recipient of pair 19, and the donor of pair 19 donates to the recipient of pair 1. In the second one, the donor of pair 8 donates to the recipient of pair 10, the donor of pair 10 donates to the recipient of pair 11 and the donor of pair 11 donates to the recipient of

pair 8. We obtain a total of six transplants and twelve surgical procedures at six different hospitals, which as explained for the previous population, require a lot more coordination and collaboration from the transplantation team.

In the population of 20 incompatible pairs, we also studied the possibility of doing two-way exchanges. We found that if we were to do two-way exchanges where pairs must be reciprocally compatible, there are twelve possible exchanges, but they all share nodes. We applied our node weight strategy as a determining factor and found that if we did two-way exchanges in this population, we would get only three exchanges. We get an exchange between pairs 10 and 19, pairs 13 and 8, and pairs 12 and 9. This shows that we, at least in this simulated population of incompatible pairs, do get more transplants when we do three-way exchanges, we obtain six exchanges, if we do not count the reciprocality of three-cycle $8 \longrightarrow 10 \longrightarrow 11 \longrightarrow 8$.

In addition, we also examined the possibility of combining pairwise donations with three-way donations and found that doing a combination achieves even more transplants. From a combination of the two exchanges we would get eight possible transplants. This is because from the two-way exchanges, the exchange between pairs 12 and 9 is unique, meaning it is the only one that does not overlap with the results of the three-way exchanges. This way we give the opportunity to two other recipients to receive a transplant that otherwise would have stayed waiting for a compatible living donor or would have been added to the waiting list for a cadaver donor.

Furthermore, we generated a different population of 20 incompatible pairs with unique characteristics and found the same results. That is, we found only two possible three-way exchanges from the 41 three-cycles in the population, and thus six transplantations. We again encountered a lot overlaps between the three-cycles where only 3 pairs were not involved in any three-cycles. We also found that if only two-way exchanges were performed, there would be a total of six transplantations, which means three reciprocally compati-

ble pairs. Thus, using two-way and three-way exchanges separately in this population of incompatible pairs give us the same number of possible exchanges, but the three-way exchanges involve higher priority patients than the two-way. Finally, we observed that if we allow both exchanges, we get a combination of two possible three-cycles and one two-way exchange and thus, eight patients have the opportunity of receiving a kidney transplant.

7
Sensitivity Analysis

In this chapter, we test our model's sensitivity to parameter changes. The parameter values and assumptions made during the creation of a model are subjected to change or error in estimation. This is because they are decided by the modeler and thus, they are based on assumptions made on their importance. Sensitivity analysis is defined as the exploration of these potential changes and their effect on the conclusions derived from the original model. We defined the decision variable as the variable of which we have control to change. The strategy, in sensitivity analysis, refers to the set of values for all the decision variables of a model. An optimal strategy, that is an optimal set of values, is the strategy that optimizes the designed model [19, 20].

In our model, we let the decision variable be the weight assigned to incompatible pairs because of the recipients age and blood type O (with and without Rh) recipients and donors. We define the base case strategy as the strategy used for the results shown in Chapter 6. In this analysis, we tried different strategies and analyzed the changes in the results.

7.1 Sensitivity Analysis for the Sample $N = 6$

We perform sensitivity analysis on the sample with six incompatible donor-recipient pairs. As mentioned, the decision variable is the node weight. We defined the following equation for the weight:

$$NW = a(W_{ra}) + b(BTR) + c(BTD),$$

where BTD is the weight assigned to recipients with blood type O (with and without Rh) donors or sensitized recipients, BTR is the weight assigned to recipients with blood type O, and W_{ra} is the weight assigned based on recipient's age. In this equation, a, b, and c represent percentages such that $a + b + c = 100\%$. We vary these percentages. All the parameter levels that are used in the sensitivity analysis have equal sized intervals. That is, $BTR = 10$ and $BTD = 5$, which fall in the same interval as the age weight which ranges from 0-10.

We start with a, b, and c being the same percentages which means all factors are weighted equally. After running the algorithm with these changes in the node weight, we get the same results. That is,

$$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1$$

is the three-cycle with the maximum total node and edge weight. We also changed the percentages so that BTR and BTD are weighted evenly. We obtained the same results.

Furthermore, we changed the parameter values so that BTR gets a 65% and BTD a 25%, and then switched the percentages so that BTD gets weighted 65% of the total node weight. For this latter analysis, we got the same result;

$$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 1,$$

however, in the third iteration of the algorithm when $n_{max} = 5$, we get a different result; we get $2 \longrightarrow 5 \longrightarrow 3 \longrightarrow$ instead of $1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 1$. Note that W_{ra} gets weighted only 10%

in the NW in both strategies. This is because we do not want the age of the recipient to be a major determining factor because its use is still debatable as we discussed in Section 4.4.1.

7.2 Sensitivity Analysis for the Sample $N = 20$

We use the same equation for NW as in the sensitivity analysis for $N = 6$. We also do the same parameter changes and obtain the same result for all the iterations. As the previous analysis, when we assign BTD a higher percentage, we get completely different results in every iteration and thus different final exchanges. The end result of the algorithm is that three-cycles

$$1 \longrightarrow 13 \longrightarrow 10 \longrightarrow 1$$

and

$$8 \longrightarrow 11 \longrightarrow 19 \longrightarrow 8,$$

or

$$8 \longrightarrow 19 \longrightarrow 11 \longrightarrow 8,$$

are the possible exchanges in the population of 20 incompatible pairs. These exchanges involve the same pairs as the three-cycles acquired from using the node weight scheme constructed for the model. Also, both results contain a convenient exchange in terms of treatment locations. That is, both results contain an exchange that has high edge weights and another that has moderately high edge weights.

From this sensitivity analysis we see how the way in which we assign the weights is very important. We assign these weights so that we can know which patients should be prioritized when a kidney from a living donor is available. We prioritize patients because they can be hard to match, they can receive only from specific blood types, or because of their ages. There is absolutely human bias in how we assigned the weights in this

model and this is one of the reasons why we see these changes in outcome once parameter values are changed. We emphasize that in this application people's lives are affected. Thus the weight assignments should be done carefully. In Section 8.2.4 of the next chapter we suggest the use of an unbiased process.

8
Conclusions and Future Work

8.1 Conclusions

Any adult who is medically healthy, psychosocially suitable, and willing to donate can be a live kidney donor. As we have discussed, the difficulty for a recipient is to find a donor that is compatible by blood type and crossmatch reactivity. In this model, we present an algorithm for finding and choosing three-way exchanges that have high priority patients in populations of incompatible donor-recipient pairs. The proposed algorithm uses a weighted directed graph to accomplish this objective. The ability to perform three-way exchanges or more has been demonstrated to increase the number of possible exchanges that can be identified [3]. Applying the algorithm to our simulated populations, which were created by using the blood type distribution of the general population of the U.S. and data from distributions describing end-stage renal disease patients eligible for renal transplantation, gave us this same result. We have shown that in the case of the 20 incompatible pairs, six exchanges are possible if we use three-way exchanges in contrast to the three possible two-way exchanges. We also examined how using a combination of both three-way and

pairwise exchanges can expand the opportunity for other patients to receive a kidney transplantation.

As we discussed at the beginning of the project, as of March 28, 2014, there are 107,311 patients on a waiting list for a kidney transplant. We believe that KPD and other exchanges, such as three-way or higher, should be implemented. These exchanges give patients a higher chance at receiving a kidney transplant. There are many logistical difficulties, but with good coordination between the doctors and the hospitals involved, we think it is possible to use this strategy. One recommendation to ease logistical issues could be to perform all the exchanges in one hospital where the transplantation crews from the other hospitals gather. This way the other hospital's resources are utilized guaranteeing that the process is fairly done.

Furthermore, according to Sommer Gentry in Optimization in Medicine and Biology, doing three-way exchanges gives the same outcome as doing unrestricted n-way exchanges, where $n \geq 4$. These larger exchanges add algorithmic complexity and logistical difficulties in practice, but they are still possible [1]. For example, a four-way exchange can give the opportunity to four patients to receive a kidney transplant, but these transplantations would require eight surgical procedures and thus eight surgical teams. And if the patients are not in the same hospital, it would involve four hospitals.

Using these exchanges could be lifesaving for many of these patients waiting for years for a compatible donor. These patients have their lives on hold waiting for a donor while also receiving dialysis. And, ironically, the patients on the waiting list depend on getting much sicker to advance to the top of the list.

8.2 Future Work

There are many possible directions for future research to take for this graphical model for living donor three-way exchanges. For future work, we suggest the translation of the

algorithm created in Chapter 5 to a MATLAB program, and also its modification to include a combination of two-way and three-way exchanges. We also consider using additional factors for the edge and node weights of the exchange directed graph. And finally, we explain the analytic hierarchy process as an unbiased way to assign node and edge weights. One important thing to note is that the design of this model is open to different kinds of opinions and ideas. The model created is the representation of an opinion and assumptions on the topic. It could vary from person to person.

8.2.1 Algorithm Implementation

The next step for this project is to create a computer program, possibly in MATLAB, that will run the final algorithm. This is crucial for working with larger sample sizes. In an e-mail discussion with Sommer Gentry, a mathematician involved in the advancement of Optimized Match discussed in Section 3.1, she discussed the complexities of using more than two incompatible pairs. For these larger exchanges, one solution would be to write an integer program that describes the constraints and objective of the algorithm. One possible integer programming solver is IBM's OPL/CPLEX system [24].

8.2.2 Using a Combination of Two-Way and Three-Way Exchanges

As we discovered in the example with 20 pairs, using a combination of pairwise and three-way exchanges gave a greater number of possible exchanges than when considering pairwise or three-way exchanges individually. One addition to the algorithm is to simultaneously search for combinations of two-way and three-way exchanges. After finding three-cycles in the current algorithm, there may be pairs involved in pairwise reciprocally compatible exchanges, but cannot be arranged into a three-cycle. As discussed in Section 6.3, with a combination of these programs we can get even more exchanges.

8.2.3 Additional Weight Factors for Nodes and Edges

From the sensitivity analysis we saw that how we assign weights can drastically change the outcome of the exchanges being suggested. By including more factors for assigning weights, we are able to make better choices as to how much each factor should be weighted. First of all, additional data about incompatible donor-recipient pairs would be useful for the weight assignments. One possibility might be to contact the Organ Procurement and Transplantation Network and ask to see if they can grant access to data sets with incompatible pairs information, if they exist. Having more information about the incompatible pairs allows us to add more factors into the model for determining the node and edge weights of the created exchange directed graph.

- We can include waiting times on the cadaver wait list by blood type of recipient for node weight assignment. This will help decide which patients should be prioritized based on their blood type. In our model, we prioritized blood type O^- patients; however, it would be good to include other patients as well. Also, including the length of time on the list for each recipient can be very helpful because in practice, this is one of the factors considered when a kidney becomes available.

- We can also add panel reactive antibody (PRA) information. The PRA is a test performed on the blood of the patient to determine whether or not the patient has any specific HLA antibodies. A lab specialist tests the blood of the patient against white blood cells obtained from a panel of about 100 blood donors. Percent PRA is a percentage between 0 and 99, representing the proportion of the population to which the patient being tested will react via pre-existing antibodies. If the patient's blood does not react with any of the donor samples, then the patient is not sensitized and thus has a PRA of 0. A high PRA usually means that the individual is prone to immunologically react against a large proportion of the donor population. Using

PRA as a factor will replace the immunologic sensitization that we assumed for our model in the node weights scheme [10].

- For the edge weights the only factor we used in this model is location of the incompatible pair in New York State, in terms of hospitals. The location radius can be expanded to incorporate hospitals from other states. We can also add tissue typing test to the edge weight. This test, performed on all patients, matches the number of antigens or HLA that the donor and the recipient share. These antigens can recognize the difference between two people's body tissue. Adding this factor could help with deciding the strength of compatibilities between two incompatible pairs. We also discussed tissue typing in Section 1.4.1.

8.2.4 Analytic Hierarchy Process (AHP)

In Section 4.4, we discussed the procedure for assigning weights to the nodes and directed edges of the exchange directed graph. These weight values assigned are based on assumptions we made about the importance of different factors after learning about organ allocation. One way to assign weights without having any strong opinions over them is by using the analytic hierarchy process (AHP). The AHP is a multiple criteria decision-making tool. Broad areas in which AHP has been applied, include resource allocation, business process re-engineering, public policy decisions, healthcare, and many more [21–23].

AHP is an eigenvalue approach to the pairwise comparisons [21]. This process provides the means of breaking the problem into a hierarchy of sub problems, which can be more easily evaluated. These evaluations are converted into numerical values and processed to rank each alternative on a numerical scale.

In our model, the problem is the allocation of living donors in the simulated population of incompatible donor-recipient pairs. The main objective for us, the decision makers, is the participant's satisfaction [22]. That is, we want recipients to find compatible living

donors in the simulated population by participating in three-way kidney exchanges. Using this process we can make an unbiased decision regarding the node weight and the directed edge weight. In our model, we use a limited amount of factors for the node weight, but as discussed in the previous section, other factors can be included for the weights of the nodes.

9
Appendix A : Population Simulation in R

Simulation of donor-recipient incompatible pairs characteristics.

```
Blood Distribution probability of the general population in the United States
not including Rh. Here blood type O is the universal donor:
P(X=A)= 42
P(X=B)= 10
P(X=AB) =4
P(X=O)= 44
# The following functions are used to generate the incompatible donor-recipient pairs:

#This function checks if two blood types are incompatible
#and if they are it returns TRUE.

Incompatible<-function (d, r){
  #checks if d= Donor and r=Recipient are incompatible by blood type.
  # A=1, B=2, AB=3, O=4
  if (d==r){
    return('FALSE')
  }else if ( d==1 & r==3){
    return ('FALSE')
  }else if (d==2 & r==3){
    return ('FALSE')
  }else if (d==4){
    return ("TRUE")
  }else{
    return ('TRUE')
  }
}
```

```
# This part of the program makes sure that we get incompatible pairs.
#We generate a  sample N of donor-recipient pairs.
Probability<-c (.44,.42,.10,.04)
### A=1, B=2, AB=3, O=4##
BT<-c (4,1,2,3)
Library (xlsx)
#Matrix of the resulting incompatible pairs:
incpairs<-matrix(ncol=2,nrow=N)
#Stores these pairs in an excel file:
write.xlsx (x = incpairs, file = "incp.xlsx",
           sheetName = "Sprojsheet", row.names = FALSE)
i<-1
while (i<=N) {
 #We can vary the size of n which represents the number of donors,
 and recipients from which we generate incompatible pairs:
  D<-sample(BT,n, prob=Probability,replace=TRUE)
  R<-sample(BT,n, prob=Probability,replace=TRUE)
  for (j in length(D)){
    for (k in length(R)){
      if (incompatible(D[j],R[k])){

        incpairs[i,1]<-D[j]
        incpairs[i,2]<-R[k]
        i<- i+1

      }
    }
  }

}
```

When we were applying the algorithm we encountered the problem of having too many three-cycles for comparisons and with one or two common nodes. This was a problem because we were doing the implementations by hand, and so we wanted to find manageable numbers. To ease this problem, we added Rh information to the blood types. The other was simpler, but this is more realistic.

We modified the functions defined above to generate the incompatible pairs using blood type and Rh:

```
probability<-c(0.066,0.374,0.015,0.085,0.063,0.357,0.006,0.034)
### 1= O-, 2=O+,3=B-,4=B+,5=A-, 6=A+, 7=AB-, 8=AB+,
BT<-c(1,2,3,4,5,6,7,8)

incompatible2<-function(x,y){
  #checks if x= DONOR,y=RECIPIENT are incompatible
  ### 1= O-, 2=O+,3=B-,4=B+,5=A-, 6=A+, 7=AB-, 8=AB+,
  if (x==1){
    return('TRUE')
  }else if ( x==8 & (y==1 | y==2 | y==3 |y==4| y==5| y==6 |y==7)){
    return ('TRUE')
  }else if (x==2 & (y==7 | y==5 | y==3 | y==1 )){
    return ('TRUE')
  }else if (x==3 & (y==1 | y==2 | y==5 | y==6 )){
    return ("TRUE")
  }else if (x==4 & (y==1 | y==2 | y==3 | y==5 | y==6 |y==7)){
    return ("TRUE")
  }else if (x==5 & (y==1 | y==2 | y==3 | y==4 )){
    return ("TRUE")
  }else if (x==6 & (y==1 | y==2 | y==3 |y==4| y==5|y==7 )){
    return ("TRUE")
  }else if (x==7  &(y==1 | y==2 | y==3 |y==4| y==5| y==6)){
    return ("TRUE")

  }else{
    return ('FALSE')
  }
}
library(xlsx)
library(rJava)
library(xlsxjars)
incpairsRH<-matrix(ncol=2,nrow=N)
write.xlsx(x = incpairsRH, file = "incpsrhN.xlsx",
           sheetName = "Sprojsheet2", row.names = FALSE)
i<-1
#N= number of incompatible donor-recipients pairs we want.
while (i<=N) {
  D<-sample(BT,n, prob=probability,replace=TRUE)
  R<-sample(BT,n, prob=probability,replace=TRUE)
  for (j in length(D)){
         if (incompatible2(D[j],R[j])){
```

```
          incpairsRH[i,1]<-D[j]
          incpairsRH[i,2]<-R[j]
           i<- i+1

      }
    }
  }
```

Age Data of Recipients:

```
#We used data from the OPTN website
#This sample gives the age in intervals and so we use these intervals and then generate
#an uniformly distributed number in the interval for each recipient in  MS Excel.

Probability_recage<-c(0.0003144636,0.01046306,0.01056026,0.03266705,
0.1867428,0.3127083, 0.3392576,0.1072864)
intv = c('<1','1-5','6-10','11-17', '18-34','35-49','50-64','65+')
##N= number of incompatible donor-recipients pairs we want.
Age<-sample(intv,N,prob=probl, replace=TRUE)
Age_Rc<-data.frame(age)
barplot(table(age))
library(xlsx)
write.xlsx(x = Age_Rc, file = "age.xlsx",
           sheetName = "Sprojsheet", row.names = FALSE)
```

Location Data:

```
#We assigned a random hospital to each donor-recipient pair
from the following list:
# 1= Albany Medical Center Hospital
#2=New York-Presbyterian/Columbia
#3=Montefiore Medical Center
#4=New York University Medical center
#5=SUNY Downstate Medical Center
#6=Buffalo General Hospital/ Children Hospital
#7= Strong Memorial Hospital
#8=Westchester Medical Center
#9=North Shore University hospital
#10=Erie County Medical Center
#11=Mount Sinai Medical Center
#12=SUNY Upstate Medical Center

##N= number of incompatible donor-recipients pairs we want.
location<-sample(1:12,N,replace=TRUE)
Location_drpair<-data.frame(location)
write.xlsx(x = Location_pair, file = "location.xlsx",
           sheetName = "Sprojsheet", row.names = FALSE)
```

10
Appendix B: Making the Graph in MATLAB

```
function [result] = Compatible (D,R)

##Usage:  [result] = Compatible (D,R).
##Inputs:
     # D = donors blood type.
     # R= recipients blood type.
##Outputs:  result= True or False.
##Operation:  Determines if a donor and recipient are compatible by blood type.
#  Uses information about blood compatibility discussed in the first chapter.
#Assignments:
# A=1,
#B=2,
#AB=3,
# O=4

##If the donor of and the recipient have the same blood type,
#then they are compatible. Also, if the donor is blood type O, they are compatible.
If D==R || D==4;
result= 'T';
#If the donor is blood type A and the recipient is blood type AB,
# then they are compatible.

Else if D==1 && R==3;
result= 'T';

#If the donor is blood type B and the recipient is blood type AB,
# then they are compatible.

Else if D==2 && R==3;
result= 'T';
```

```
#If none of the above is true then the blood types of donor and
#the recipient inputted in the function are incompatible.
Else
result= 'F';
    end
    end
end
end
```

We modified the function *Compatible(D,R)* so that we can find the graph for the pairs when use blood type and Rh. It does the same as the previous function. The only difference is that it considers eight different blood types instead of the general four.

```
function [result] = Compatible2(D,R)
##Assignment:
 # 1= O-, 2=O+,3=B-,4=B+,5=A-, 6=A+, 7=AB-, 8=AB+

if  D==1;
        result= 'T';

else if D==2 && ( R== 2 || R== 4 || R== 6 || R== 8) ;
       result= 'T';

else if D==3 && ( R== 3 || R== 4 || R== 7 || R== 8);
        result= 'T';
else if D==4 && ( R== 4 || R== 8);
        result= 'T';

else if D==5 && ( R== 5 || R== 6 || R== 7 || R== 8);
        result= 'T';

else if D==6 && (R== 6 || R== 8);

        result= 'T';

else if D==7 && (R== 7 || R== 8);
        result= 'T';

else if D==8 &&   R== 8;
        result= 'T';

    else
       result= 'F';
    end
    end
    end
    end
    end
    end
    end
end
end
```

```
function [A] = Graph (BT)
# Creates the adjacency matrix of a directed graph.
#Usage:  [A] = Graph (BT).

#Inputs:
     #BT= matrix of incompatible donor-recipient pairs by blood type.
#Outputs:  A= an adjacency matrix used to represent the created directed graph.
#Nodes- represent incompatible donor-recipient pairs.
#Edges- represent possible matches between two pairs/ edges by blood type.
#The ones in the matrix represent a match.
#Because the graph is directed, this matrix is not symmetrical.

#Operation:  Use the incompatible pairs data and within incompatible pairs find
#possible matches for each pairs by looking at  the donors and recipients blood type.


#The first thing we do is to calculate the length of the Blood Type matrix.
L = size (BT, 1);
#Initialize the adjacency matrix.
A= zeros (L, L);

 #Loop through the donors blood type.
For i= 1:L;
#Loop through the recipients blood type.
For j=1:L;
#Check if the donor and recipient are compatible using the Compatible function.
#Use this if Rh is not considered.
# if it is then use Compatible2(BT (i, 1), BT (j, 2)) in this step.

        Compatible_pairs= Compatible (BT (i, 1), BT (j, 2));

#If the donor-recipient are in the same row, that is they are incompatible pairs,
and if the donor is blood type O, then report this as a zero. This means not a match.
#If we consider Rh, we change BT (i, 1)==4 to BT (i, 1)==1.

        If i==j && BT (i, 1)==4;
     A(i,j)=0;

#Else if the pairs are compatible record this as a 1 in the adjacency matrix.
        Else if compatible_pairs=='T';
                       A (i, j)= 1;

                end
               end
     end
end
end
```

11
Appendix C: Node Weights in Matlab

```
Function [Node_weight_age]=Age_weight (Age)

#Usage:  [Node_weight_age] = Age_weight (Age)

#Inputs:  Age= matrix with age of  recipients.

#Outputs:  Node_weight_age= a matrix containing the weights assigned
#to each node-recipient- because of the age factor.

#Operation:
 #The first thing we do is to find the length of Age.
L=size (Age);
#Initialize the matrix.
Node_weight_age= zeros (L,1);
#Loop through the Age matrix
For i= 1:L;

#If the recipient is pediatric then assign the maximum weight.
if  Age(i)<= 18;
           Node_weight_age (i)=10;
#If the recipient is between these ages then applied the following continuous function
#to calculate the age weight.

        elseif  Age(i)>18 && Age(i)<= 80;
             age_w = ((-5 * Age(i))/ 31) + (400/31);
                 Node_weight_age (i)=abs (round (age_w));
        end
end
```

```
Function [Node_w]= Node_weight (BT, Age)
# Assigns weights to the nodes of the directed graph.

##Usage:  [Node_w] = Node_weight(BT, Age)

##Inputs:
        #BT= matrix of incompatible donor-recipient pairs by blood type.
        #Age:  matrix of recipients age

##Outputs:  Node_w= a matrix containing the weights assigned to each
 node-recipient- because of factors like age, blood type O
 and immunologic sensitization.

##Operation:  All recipients, nodes, get 25 points because all patients are
 equally important. If the recipient has any of the prioritizing factors then
 a weight should be assigned that recipient/ node, so that the recipient is
  considered a high priority patient and so can receive a transplant once
   we find a compatible donor.

#The first thing we do is to calculate the length of the Blood Type matrix.
L=size (BT, 1);

# Initialize the Node_w matrix.
Node_w= zeros (L, 1);

#Age weights obtained from the Age_weight function.
Age_w=Age_weight (Age);

#Loop through the entire BT matrix
For i= 1:L;

#If the recipient is blood type O then the recipient should get a few extra points-
#10 points.

  If BT (i, 2)==4;
#Weight assigned to the recipient because of his/her age.
   Aw=Age_w (i);

#Assign the sum of all weights to the i-th entry of Node_w matrix.
  Node_w (i) = Aw + 10 + 25;

# If the blood type of the donor of an incompatible donor-recipient pair
is O do the following.
#We assume that the recipient is highly sensitized.
#This means that the recipient has pre-formed antibodies
#that makes it hard to find a donor.

#If Rh is included BT (i, 1)==4 should be replaced with
BT (i, 1)==1, because 1=O-  and the same for
BT(i,2)~=4 --> BT(i,2)~=1.
```

```
else if BT (i,1)==4 && BT(i,2)~=4;
        Aw=Age_w (i);
         Node_w (i)= Aw + 5 + 25;


#If none of these factors are present then just add the 25 free
 points to the age weight the recipient received.
  else
             Aw=Age_w (i);
             Node_w (i)= Aw + 25;
     end

    end
end
end


function[Node_wt]=Node_weight_Triplet(BT, Age)
#This function finds the weight of the nodes involved
in three-cycles using the function defined above Node_weight(BT,Age):
#Adjacency matrix of graph
DiGraph= Graph(BT);

# Three-cycles in population
DTriad= Threecycle_directed(DiGraph);

L=size(DTriad,1);
 # Node weight of individual nodes.
Single_nodew=Node_weight(BT,Age);

Node_wt= zeros(L,3);
for i=1:L;
    Node_wt(i,1)=Single_nodew(DTriad(i,1));
    Node_wt(i,2)=Single_nodew(DTriad(i,2));
    Node_wt(i,3)=Single_nodew(DTriad(i,3));


end
end
```

12
Appendix D: Edge Weights in Matlab

We assign a variable to each location of the incompatible donor-recipient pairs and provide a table with the distance between in each hospital in miles.

- AMCH=Albany Medical Center Hospital
- NYPC=New York-Prebysterian/Columbia
- MMC=Montefiore Medical Center
- NYUMC= New York University Medical Center
- SDMC=SUNY Downstate Medical Center
- BGH= Buffalo General Hospital/ Children Hospital
- SMH=Strong Memorial Hospital
- WMC= Westchester Medical Center
- NSUH=North Shore University Hospital
- ECMC= Erie County Medical Center
- MSMC= Mount Sinai Medical Center
- SUMC= SUNY Upstate Medical Center

Hospital	AMCH	NYPC	MMC	NYUMC	SDMC	BGH	SMH	WMC	NSUH	ECMC	MSMC	SUMC
AMCH	0	140	146	148	155	286	225	123	156	282	144	145
NYPC	140	0	239	9.8	16	398	337	20.4	17.1	394	5.2	249
MMC	146	239	0	13.2	22.5	403	342	17.4	16.1	399	8.6	254
NYUMC	148	9.8	13.2	0	8.1	398	337	27.9	16.3	394	4	249
SDMC	155	16	22.5	8.1	0	392	340	37.2	18.4	397	11.4	252
BGH	286	398	402	398	392	0	71.5	375	392	3.7	383	149
SMH	225	337	342	337	340	71.5	0	314	333	69.9	321	88
WMC	123	20.4	17.4	27.9	37.2	375	314	0	32.6	389	24.1	244
NSUH	156	17.1	16.1	16.3	18.4	392	333	32.6	0	408	19.1	265
ECMC	282	394	399	394	397	3.7	69.9	389	408	0	379	146
MSMC	144	5.2	8.6	4	11.4	383	321	24.1	19.1	379	0	253
SUMC	145	249	254	249	252	149	88	244	265	146	253	0

Table 12.0.1: Distance between hospitals in miles.

```
Function [Edge_w]= Edge_weight (A, location, distances)

##Usage:  [Edge_w] = Edge_weight (A, location, distances)

##Inputs:
#A= adjacency matrix of the directed graph created from incompatible
# donor-recipient pairs information.
# Location:  location of incompatible pairs. All hospitals are in New York State.
#Distances= 12 x 12 matrix containing the distance between
  #the twelve selected hospitals in miles.

##Outputs:  Edge_w= a matrix containing the weights assigned to each edge-match-
#because of the location factor in the three cycles.

#Operation:  All matches receive 25 points because
#all transplants are equally important.

#If the pairs involved in the three-way exchange are in the same hospital
# then their edges-matches-should receive the maximum weight.
#Otherwise, the donor thats compatible
#to the recipient of another pair would have to travel to hospital.
#We use the distance information and decide on the weights
#depending how far, driving, the hospitals are.

#The first thing we do is to find all possible three-cycles in the directed graph.
DTriad= Threecycle_directed(A);

# Calculate the length of the DTriad matrix.
L=size(DTriad,1);

#Initialize the Edge_w matrix. Same number of rows as the DTriad matrix.
#Each column will contain the weight assigned to each of the edges- 3 edges total.
Edge_w=zeros (L,3);

#Initialize the distancebetween matrix.
#This would be a matrix containing the distance between the specific hospitals
```

```
#of the pairs involved in the exchange.
distancebtween= zeros(L,3);

# Loop through the rows of DTriad.
for i= 1: L;

#Find the location of each pair.
#They get updated as the value of  i changes
            loc_a=location(DTriad(i,1));
           loc_b= location(DTriad(i,2));
           loc_c= location(DTriad(i,3));

#Find the distance between each location and store it in the distancebetween matrix.
#These distances would be updated as the value of   i changes.

            distancebetween(i,1)= distances(loc_a,loc_b);
            distancebetween(i,2) = distances(loc_b,loc_c);
            distancebetween(i,3)= distances(loc_c,loc_a);

#Loop through the columns of DTriad.
For j=1:3;

        If distancebetween (i, j)==0;

               Edge_w (i, j)= 10 + 25;

       elseif  distancebetween(i,j)> 0 && distancebetween(i,j)< 20;

                 Edge_w (i, j)= 5+ 25;

    elseif distancebetween(i,j) >=20 && distancebetween(i,j)<150;

                  Edge_w (i, j)= 4+ 25;

         elseif distancebetween(i,j) >=150 && distancebetween(i,j)<255;

                   Edge_w (i, j)= 3+ 25;

          elseif distancebetween(i,j) >=255 && distancebetween(i,j)<350;

                    Edge_w (i, j)= 2+ 25;
          else
                     Edge_w (i, j)= 1+ 25;

end
    end
end
end
```

This functions acts just as the function above. The only difference is that it outputs the edge weight of all individual directed edges not only the ones involved in the three-cycles. It uses the function that finds pairs to first find all possible matches between incompatible pairs.

```
function[Pairs]=Find_pairs(BT)
#Input: Blood types of incompatible donor-recipient pairs.
#Outputs: Possible two way matches between pairs.
L= size(BT,1);
A= Graph(BT); #Adjacency Matrix
rows_p= L * (L-1);
Pairs=zeros(rows_p, 2);
count=1;
for i=1:L
    for j=1:L
        if i==j;
            Pairs(count,1)= 0;
            Pairs(count,2)= 0;
        elseif A(i,j)== 1;
            Pairs(count,1)= i;
            Pairs(count,2)= j;
            count= count + 1;
        end
    end

end
  n=all(Pairs==0,2);
  Pairs(n,:)=[];
end

function[Edge_wa]= Edge_weight_all(BT,location, distances)
#Inputs:
#BT=  incompatible donor-recipient pairs blood type information.
# Location:  location of incompatible pairs. All hospitals are in New York State.
#Distances= 12 x 12 matrix containing the distance between
#the twelve selected hospitals in miles.

#Outputs:  Edge_wa= a matrix containing the weights assigned to each edge-match-
#because of the location factor .


Pairs= Find_pairs(BT);
L=size(Pairs,1);
Edge_wa=zeros(L,3);

for i= 1: L;

           loc_a=location(Pairs(i,1));

            loc_b= location(Pairs(i,2));
```

```
            Edge_wa(i,1)=Pairs(i,1);
            Edge_wa(i,2)= Pairs(i,2);

            distancebetween= distances(loc_a,loc_b);



           if distancebetween==0;

                Edge_wa(i,3)= 10 + 25;

            elseif  distancebetween > 0 && distancebetween< 20  ;

                Edge_wa(i,3)= 5 + 25;


              elseif distancebetween >=20 && distancebetween<150;

                  Edge_wa(i,3)= 4 + 25;

              elseif distancebetween >=150 && distancebetween <255;

                  Edge_wa(i,3)= 3 + 25;

              elseif distancebetween >=255 && distancebetween <350;

                  Edge_wa(i,3)=  2 + 25;
              else
                  Edge_wa(i,3)= 1 + 25;
           end
end
end
```

13
Appendix E: Directed Three Cycles in MATLAB

The following function finds all possible three cycles in the adjacency matrix of a given graph.

Pseudo Code

```
function[ DTriad]= Threecycle_directed(A)
Find length,n,  of the nxn adjacency matrix;
Find number of rows by calculating the different combinations
of three nodes where order matters,
that can be made from n nodes ;
Initialize DTriad. Set DTriad equal to a matrix of zeros with three columns and
how many rows you found;
Initialize counter to one;
Loop/Iterate through all the possible combination of triples from $n$ nodes.
         If triple forms a cycle/triangle:
                  Add this triple to the DTriad matrix
                  Add one to the counter and proceed to the next triple.
```

Actual function with comments.

```
function[DTriad] = Threecycle_directed(A)

#Usage:  [DTriad] = Threecycle_directed(A);

#Inputs: A = an adjacency matrix created from a directed graph where nodes
 #represent incompatible donor-recipient pairs and the edges represent
# a possible match between two pairs/ edges.
#The ones in the matrix represent a match.

#Outputs: DTriad= a matrix containing all the possible  directed three cycles in the
```

```
Adjacency matrix.

#Operation: If the three nodes being compared are connected in a way a
triangle/ 3 cycle is made, then we add the three nodes to the DTriad matrix.

#The first thing we do is to calculate the length of the adjacency matrix A.
L = length(A);

#In this step we generate the number of rows the DTriad matrix should have
# by counting all possible  3 cycles  from the total
#number of nodes, which is the same as the length of A.
rows_n= nchoosek(L,3)* factorial(3);

#Initialize the Triad matrix by making a matrix of zeros with dimensions of
#rows_n by 3(three nodes that make up a three cycle).
DTriad=zeros(rows_n,3);


#Initialize the count.This will help us move across rows of DTriad later in
#the function.
count= 1;

#Loop through  the different combinations of nodes.
for i= 1:L;
   for j=1:L;
       for k= 1:L;
            if i==j || i==k || j==k;
                DTriad(countt,1)=0;
                DTriad(countt,2)=0;
                DTriad(countt,3)=0;
       # This condition is necessary to determine if the three nodes are
       #connected in a cycle.If these condition are true then nodes  i,j,k
       #form a three cycle and so the function proceeds to the next step
       #which is to store these nodes in the Triad matrix .
            elseif A(i,j)==1 && A(k,i)==1 && A(j,k)==1 && i<j && i<k;
                DTriad(countt,1)=i;
                DTriad(countt,2)=j;
                DTriad(countt,3)=k;
      # As mentioned before count help us navigate through the rows of DTriad.
      #In this step, count is reset to the next row of DTriad where a new directed
      #three cycle is stored.
                 count=count +1;
             end
     end
  end
end
end
#Used to delete zeroes from the DTriad matrix so that we only get the 3-cycles.
n=all(DTriad==0,2);
DTriad(n, :)=[ ];
```

14
Appendix F: Functions used in the Algorithm

Function 1: Find common nodes.

```
function[Common_node]=Find_common_nodes(DTriad,BT, Age, location, distances)

#Usage:[Common_node] = Find_common_nodes(DTriad,BT, Age, location, distances).

##Inputs:
#DTriad = a matrix containing all the possible  directed three cycles in the
Adjacency matrix.
#BT= matrix of incompatible donor-recipient pairs by blood type.
#Age:  matrix of recipients age
# Location:  location of incompatible pairs. All hospitals are in New York State.
#Distances= 12 x 12 matrix containing the distance between
#the twelve selected hospitals in miles.

##Outputs: Common_node= a matrix containing information about
the rows of the three cycles sharing one or two common nodes,
and also contains these common nodes.

##Operation: Compare all three cycles and if they share one or two
nodes in common store them.

#Weight assigned because of age of recipient.
Recipients_Age_weight=Age_weight(Age);

# Single Node weight based on blood type and age.
Nodeweight=Node_weight(BT, Recipients_Age_weight);

#Node weight of nodes involved in  a three-cycle.
Nodeweight_triad=Node_weight_Triplet(BT, Recipients_Age_weight);
```

```
#Edge weight of directed edges involved in a three-cycle.
Edgeweight_triad=Edge_weight(Graph(BT),location,distances);

L=size(DTriad,1);
L2=size(BT,1);

#Matrix containing three-cycles and their corresponding node and edge weights.
All_info=[DTriad  Edgeweight_triad  Nodeweight_triad];

#Initialize the Common_node matrix.
Common_node=zeros(L2,4);

count=1;
for m=1:L;
    for n=m+1:L;
        count= count +1 ;
        matchcount=3;
        for y=1:3;
            for x=1:3;
                if All_info(m,y)==All_info(n,x);

                    Common_node(count,1)=m;
                    Common_node(count,2)=n;

                    Common_node(count,matchcount)= All_info(m,y);
                    matchcount=matchcount + 1;


                end
            end
        end
    end
end
#Used to delete any remaining zeroes from the Common_node matrix.
n =all(Common_node==0,2);
Common_node(n,:)=[];

end
```

Function 2: Find cycle count of nodes.

```
function[Cyclecount]= Count_nodes(BT)

##Usage: [Cyclecount]= Count_nodes(BT).

##Inputs:  BT= matrix of incompatible donor-recipient pairs by blood type.

##Outputs: Cyclecount= a matrix containing information about the number
of  three-cycles  in which each node appears.

nodesnumber=size(BT,1);
```

```
#Adjacency matrix of graph.
A=Graph(BT);

#Three-cycles in the population of N incompatible pairs.
DTriad=Threecycle_directed(A);
L=size(DTriad,1);

#Initialize matrix
Cyclecount=zeros(nodesnumber,1);
for z=1:L;
    for w=1:3;
        Cyclecount(DTriad(z,w))= Cyclecount(DTriad(z,w)) + 1;
    end
end
end
```

Function 3: Find node with the maximum cycle count.

```
function[max_node_num]=Find_node_maxcount(BT)

##Usage: [max_node_num]=Find_node_maxcount(BT).

##Inputs:  BT= matrix of incompatible donor-recipient pairs by blood type.

##Outputs: max_node_num = node that is involved in the most three cycles.

#Number of times a node appears in a three-cycle.
Cyclecount= Count_nodes(BT);
S=size(BT,1);
#Largest cycle count
maxn=max(Cyclecount);
for i=1:S;
    if Cyclecount(i)== maxn;
        #row location of  the node that is involved in the most three-cycles
        max_node_num= i;

    end
end
```

Function 4: Find the cycles that have the node with maximum cycle count in common.

```
function[Rows_of_Triad]=Find_row_ofcn(max_node_num, DTriad,BT, Age, location, distances)

##Usage: [Rows_of_Triad]=Find_row_ofcn(DTriad,BT, Age, location, distances).

##Inputs:
#max_node_num= the node with the highest cycle count.
#DTriad = a matrix containing all the possible  directed three-cycles in the
Adjacency matrix.
#BT= matrix of incompatible donor-recipient pairs by blood type.
#Age:  matrix of recipients age
# Location:  location of incompatible pairs. All hospitals are in New York State.
```

```
#Distances= 12 x 12 matrix containing the distance between
#the twelve selected hospitals in miles.

##Outputs: Rows_of_Triad  = the rows of DTriad where the common node max_node_num is
#located

# Finds the common node as well as the row location of the three-cycles that contain it.
Common_nodes=Find_common_nodes(DTriad,BT, Age, location, distances);
Cyclecount= Count_nodes(BT);
CN=size(Common_nodes,1);

Rows_of_Triad=zeros(CN,1);
for i= 1: CN;
    C_N1= Common_nodes(i,3); #Common node 1
    C_N2=Common_nodes(i,4);#Common node 2

    if  C_N1==max_node_num || C_N2==max_node_num;
        Rows_of_Triad(i)= Common_nodes(i,1);
        Rows_of_Triad(i+1)= Common_nodes(i,2);

    end
end
Rows_of_Triad=unique(Rows_of_Triad, 'rows'); #Do not repeat rows
n=all(Rows_of_Triad==0,2);
Rows_of_Triad(n, :)=[ ];
end
```

Function 5: Find total edge weights and node weights of all three-cycles in the population of incompatible pairs.

```
function [ SUMS] = TotalENweightsAll(BT, Age, location, distances)

##Usage: [ SUMS] = TotalENweightsAll(BT, Age, location, distances).

##Inputs:
#BT= matrix of incompatible donor-recipient pairs by blood type.
#Age:  matrix of recipients age
# Location:  location of incompatible pairs. All hospitals are in New York State.
#Distances= 12 x 12 matrix containing the distance between
#the twelve selected hospitals in miles.

##Outputs: SUMS = Matrix containing  all the three cycles
and their corresponding total edge weight and total node weight.

A= Graph(BT);
DTriad= Threecycle_directed(A);
Edgeweight_triad= Edge_weight(A,location,distances);
Recipients_Age_weight=Age_weight(Age);
Nodeweight_triad=Node_weight_Triplet(BT, Recipients_Age_weight);
All_info= [DTriad Edgeweight_triad Nodeweight_triad];
```

```
L=size(DTriad,1);
SUMS= zeros(L,5);

for i=1:L;

    SUMS(i,1:3)= DTriad(i,1:3);
    SUMS(i,4)= round(sum(All_info(i,4:6),2)); #total edge weight of cycle
    SUMS(i,5)=round(sum(All_info(i,7:9),2)); #total node weight of cycle

end
end
```

Function 6: Find total edge weights and node weights of each three cycle that have the n_{max} in common.

```
function [SUM] = TotalENweights(max_node_num, BT, Age, location, distances)

##Usage: [SUM] = TotalENweights(max_node_num, BT, Age, location, distances).

##Inputs:
#max_node_num= the node with the highest cycle count.
#BT= matrix of incompatible donor-recipient pairs by blood type.
#Age:  matrix of recipients age
# Location:  location of incompatible pairs. All hospitals are in New York State.
#Distances= 12 x 12 matrix containing the distance between
#the twelve selected hospitals in miles.

##Outputs: SUMS = Matrix containing the three cycles with n_max
and their corresponding total edge weight and total node weight.

A= Graph(BT);
DTriad= Threecycle_directed(A);
Edgeweight_triad= Edge_weight(A,location,distances);
Recipients_Age_weight=Age_weight(Age); # weight assigned to recipients because of their
Nodeweight_triad=Node_weight_Triplet(BT, Recipients_Age_weight);
RowsofT=Find_row_ofcn(max_node_num, DTriad,BT, Age, location, distances);
All_info= [DTriad Edgeweight_triad Nodeweight_triad];

L=size(RowsofT(i),1);
SUMS= zeros(L,5);
for i=1:L;
    SUMS(i,1:3)= DTriad(RowsofT(i),1:3);
    SUMS(i,4)= sum(All_info(RowsofT(i),4:6),2); #total edge weight of cycle
    SUMS(i,5)=sum(All_info(RowsofT(i),7:9),2); #total node weight of cycle

end
end
```

Function 7: Find the cycles, from the group that have the node with maximum cycle count in common, that have the maximum total weights.

```
function [ MAXSUMCYCLES ] = Find_Cycles_maxts(SUM)
```

```
##Usage: [ MAXSUMCYCLES ] = Find_Cycles_maxts(SUM).

#Inputs:
SUM = Matrix containing the three-cycles  with n_max
and their corresponding total edge weight and total node weight.

#Outputs: MAXSUMCYCLES = Matrix containing the three-cycles
that have maximum sums.

 #Maximum node weight sum in the total weight sums
 of the three cycles that have n_max in common.
maxnode_wt_sum= max(SUMS(:,5));

#Find all the three-cycles with this  maximum total
node weight.
rowfind=find(SUMS(:,5)== maxnode_wt_sum);

S=size(rowfind,1);
MAXSUMCYCLES2=zeros(S,3);
count=1;
 #if there is only one three-cycle with n_max, then save it.
if S==1;

   MAXSUMCYCLES2(count,:)= SUMS(rowfind(1),1:3);

#Otherwise, check which one has the highest total edge weight.
 else
    for i=1:S;
        for j=i+1:S;

            if SUMS(rowfind(i),4)> SUMS(rowfind(j),4);
                MAXSUMCYCLES2(count,:)= SUMS(rowfind(i),1:3);

            elseif SUMS(j,4)> SUMS(rowfind(i),4);
                MAXSUMCYCLES2(count,:)= SUMS(rowfind(j),1:3);
            else
                MAXSUMCYCLES2(count,:)= SUMS(rowfind(i),1:3);
                count= count + 1;
                MAXSUMCYCLES2(count,:)= SUMS(rowfind(j),1:3);
            end
       count=count + 1;
        end
    end
end
#deletes extra zeroes.
n=all(MAXSUMCYCLES2==0,2);
MAXSUMCYCLES2(n,:)=[];
#deletes repeated cycles.
MAXSUMCYCLES2=unique(MAXSUMCYCLES2, 'rows');
end
```

Bibliography

[1] Gino J. Lim Lee, *Optimization in Medicine and Biology*, Auerbach Publications, Taylor and Francis Group, 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL 33487-2742, 2008.

[2] A.E. Roth, T. Sonmez, and M. Utku Unver, *Kidney Exchange*, Quarterly Journal of Economics **May** (2004), 457–488.

[3] S.L. Saidman, A.E. Roth, T. Sonmez, M. Utku Unver, and F. Delmonico, *Increasing the opportunity of live kidney donation by matching for two and three way exchanges*, Transplantation **81(5)** (2006), 773–782.

[4] R.A. Montgomery, S.E. Gentry, W.A. Marks, Daniel S. Warren, Janet Hitler, Julie Houp, Andrea A. Zachary, J.K. Melancon, W.R. Maley, H. Rabb, C. Simpkins, and Dorry L. Seger, *Domino paired kidney donation: a strategy to make best use of live non-directed donation*, Lancet **368(9533)** (2006), 419–421.

[5] Dorry Segev, Sommer E. Gentry, Daniel S. Warren, Brigitte. Reeb, and R.A. Montgomery, *Kidney Paired Donation and Optimizing the Use of Live Donor Organs*, American Medical Association **293, No.15** (April 20, 2005), 1883–1890.

[6] Dorry L. Segev, Sommer E. Gentry, and Robert A. Montgomery, *Kidney Paired Donation: Fundamentals, Limitations, and Expansions*, American Journal of Kidney Disease **57(1)** (2011), 144–151.

[7] Dorry L. Segev, Sommer E. Gentry, and Robert A. Montgomery, *Association Between Waiting Times for Kidney Transplantation and Rates of Live Donation*, American Journal of Transplantation **7** (2007), 2406–2413.

[8] C.Bradley Wallis, Kannan P. Samy, Alvin E. Roth, and Michael A. Rees, *Kidney paired donation*, Nephrol Dial Transplant **26** (2011), 2091–2099.

[9] Olivia M. Carducci, *The Mathematics behind Lifesaving Kidney Exchange Programs*, Math Horizons **18(1)** (September, 2010), 26–29.

[10] Organ Procurement and Transplantation Network(OPTN), *optn.transplant.hrsa.gov* (Accessed October 13, 2013).

[11] Optimized Match, *OptimizedMatch.com* (Accessed October 13, 2013).

[12] Robin J. Wilson, *Introduction to Graph Theory*, Longman Group Ltd, 1996.

[13] Douglas B.West, *Introduction to Graph Theory*, Prentia-Hall, Inc, 2001.

[14] MedlinePlus, *http://www.nlm.nih.gov/medlineplus/dialysis.html* (Accessed March 31, 2014).

[15] WebMD, *http://www.webmd.com* (Accessed March 31, 2014).

[16] Davita, *http://www.davita.com* (Accessed March 31, 2014).

[17] Stanford School of Medicine: Blood Center, *http://bloodcenter.stanford.edu* (Accessed March 31, 2014).

[18] Robert M. Veatch, *Transplantation Ethics*, Georgetown University Press, Washington D.C, 2000.

[19] John C. Helton, *Uncertainty and Sensitivity Analysis for Models of Complex Systems*, Department of Mathematics and Statistics Arizona State University.

[20] David J. Pannel, *Sensitivity analysis: strategies, methods, concepts, examples*, School of Agricultural and Resource Economics, University of Western Australia (Accessed April 8th, 2014), 1-19.

[21] S. Vaidya Omkarprasad and Sushil Kumar, *Analytic hierarchy process: An overview of applications*, European Journal of Operational Research **169** (15 July 2004), 1-29.

[22] Thomas L. Saaty, *How to make a decision: The Analytic hierarchy process*, European Journal of Operational Research **48** (1990), 9-26.

[23] Navneet Bhushan and Kanwal Rai, *Strategic Decision Making: Applying the Analytic Hierarchy Process* (2004), 11-21.

[24] Sommer Gentry, *Personal Communication* (October 10, 2013).