SCHOOL OF COMPUTER SCIENCE

Weighted Sum Optimisation for Identifying Optimal Matchings in UK Kidney Exchanges

A Comparison to the Current Hierarchical Model

Lamb Chen

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering **worth 40CP**.

Monday 28th April, 2025

Abstract

Kidney transplantation is typically the preferred treatment for chronic kidney disease, offering a better quality of life and lower mortality compared to alternative treatments such as dialysis. Successful transplantations are dependent on blood-type and tissue-type compatibility between the donor and recipient patient. NHS Blood and Transplant (NHSBT) conducts matching runs every three months to identify compatible pairs from a registered pool of donors and patients and schedule transplants. Currently these matching runs utilise an integer linear programming model that hierarchically optimises a set of criteria decided by NHSBT. However, this hierarchical model forces the criteria to be optimised in a fixed order, and given the growing demand for donor kidneys, these order of priorities may change over time. This paper implements and explores the potential consequences of employing an alternative strategy, known as weighted sum optimisation, in place of hierarchical optimisation. We demonstrate the greater flexibility weighted sum holds and evaluate the trade-offs from prioritising different objectives.

Dedication and Acknowledgements

Thank you to my supervisor Christian Konrad for believing in me and supporting me throughout.

I would also like to express my gratitude to David Manlove, James Trimble, and William Pettersson, whose work provided the foundation for this dissertation. I am especially thankful for their valuable responses to my questions, which strengthened my understanding of their work and informed my contributions greatly.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others including AI methods, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

Lamb Chen, Monday 28th April, 2025

AI Declaration

I declare that any and all AI usage within the project has been recorded and noted within Appendix A or within the main body of the text itself. This includes (but is not limited to) usage of text generation methods incl. LLMs, text summarisation methods, or image generation methods.
I understand that failing to divulge use of AI within my work counts as contract cheating and can result in a zero mark for the dissertation or even requiring me to withdraw from the University.

Lamb Chen, Monday 28th April, 2025

Contents

List of Figures

List of Tables

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Christian Konrad.

Supporting Technologies

- Gurobi Optimizer for solving the mixed-integer linear programming models.
- Git for version control.
- Python 3.12.3 for implement both the hierarchical and weighted sum models.
- NetworkX 3.2.1 and PyVis 0.3.2 for creating the pool graph visualisation.
- Random kidney exchange instance dataset generator, accessed from https://wpettersson.github.io/kidney-webapp/#/.
- James Trimble's attempt at implementing the hierarchical model, accessed from https://github.com/jamestrimble/kep-hierarchical

Note: A ZIP file copy of my final code has been attached to the submission of this dissertation.

Chapter 1

Introduction

Every 65 minutes someone is diagnosed with kidney failure in the UK [1]. For patients with long-term kidney failure, they may choose to have a kidney transplant, where the recipient patient's kidney is replaced with a healthy kidney from a living or deceased donor [2]. As of 21 February 2025, there are 8000 on the UK transplant waiting list and only 3228 people on the waiting list have received a transplant since April 2024 [3]. There is an evident shortage of donor kidneys, with average waiting times for a kidney transplant being around 500 days, and significantly longer for Black and South Asian communities [1]. The limited availability of kidneys increases the importance of developing strategies to maximise the efficient utilisation of the existing supply and subsequently help reduce wait times.

Before 2006, the supply of donor kidneys was constrained because living donations in the UK were only allowed as direct exchanges between family members and friends [4]. This was significantly limiting if family members or friends were incompatible by blood group or human leukocyte antigens (HLA) tissue-type with the recipient [4]. The Human Tissue Act 2004 (Scotland, 2006) alleviated these limitations by enabling anonymous altruistic organ donation to be introduced into the donor sharing scheme, in addition to "paired/pooled" donations [5]. Consequently, in instances where the donor and recipient are incompatible, the pair now have the option to register into a pool of other incompatible donor-recipient pairs, within which they may exchange donor kidneys.

Figure 1.1 illustrates an example where Donor 1 is incompatible with their intended recipient, Patient 1, and Donor 2 is also incompatible with their intended recipient, Patient 2. Before the Human Tissue Act, if the pairs did not know each other no kidney transplants would have occurred. However, since the introduction of pooled donations, both pairs can decide to enter a pool. They would then be able to form a two-cycle paired kidney exchange (PKE) if Donor 1 was compatible with Patient 2, and Donor 2 was compatible with Patient 1.

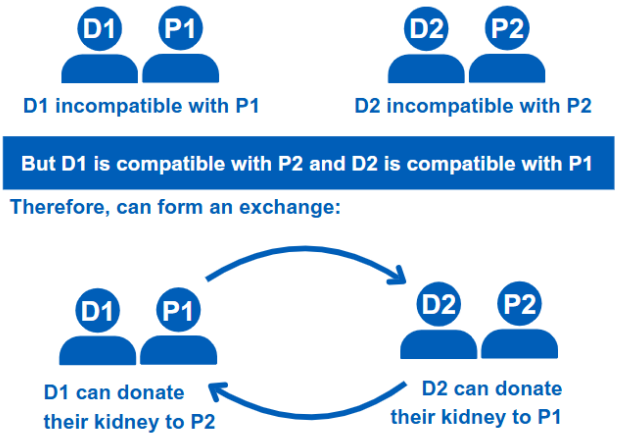

Figure 1.1: Explanation of a compatibility and a two-cycle exchange

Pooled donations only involve PKEs of up to length three due to time and logistical restrictions, such as required scheduling of double the number of operating theatres as pairs [6]. This is because all operations are typically scheduled on the same day to avoid legal complications and ensure that no pair is disadvantaged if a donor's circumstances were to change [7] [8]. As a result, six operating theatres would be required for a PKE of length three, also known as a three-cycle. Figure 1.2 illustrates an example of a three-cycle.

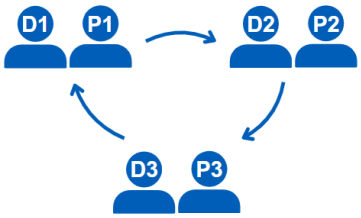

Figure 1.2: Example of a three-cycle

Altruistic donors donate kidneys without an associated patient, which triggers domino paired donation (DPD) chains illustrated in Figure 1.3. In DPD chains, the altruistic donor donates to an incompatible donor-recipient pair, then the donor from that pair donates either to another incompatible donor-recipient pair, or to the Deceased Donor Waiting List (DDWL) [9]. Restrictions on the length of PKEs are also placed on DPD chains. NHSBT defines a DPD chain as short if it contains one incompatible donor–recipient pair, and long if two [9]. The term exchange can refer to cycles or chains or both cycles and chains.

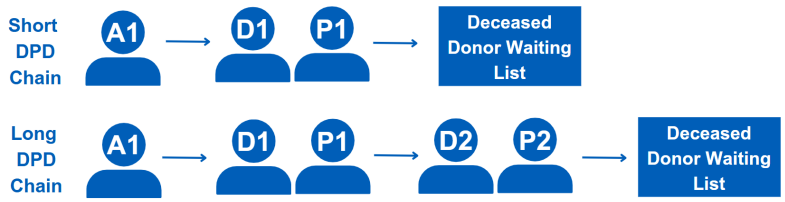

Figure 1.3: Difference between short DPD chains and long DPD chains

In April 2007, the NHS Blood and Transplant (NHSBT) introduced the UK's Living Kidney Sharing Scheme (UKLKSS), which helps incompatible donor-recipient pairs identify PKEs in which they can participate [10]. They perform quarterly matching runs of an algorithm that finds the optimal set of PKEs and DPD chains, within a given pool [10].

Matching scores play a critical role in determining which exchanges are to be included in the final optimal set. These scores are calculated based on various factors, such as immunological compatibility between donors and recipients, and are discussed further in the Optimal Exchanges Criteria section. In the pool graph, each edge is weighted according to its corresponding matching score, with higher scores indicating a greater level of compatibility between the donors and potential recipients. Consequently, one key objective for determining the optimal exchange set is the maximisation of the total score of the selected edges. Additional optimisation criteria, which are also detailed within the Optimal Exchanges Criteria section, include maximising the number of effective two-cycles, the overall number of transplants, and minimising the number of three-cycles.

Figure 1.4 presents an example pool graph instance with annotated edge weights. The directed blue edges indicate the compatibility between pairs, such that the source pair's donor would be able to donate to the target pair, which would be valued at the edge score. Highlighted in red is the optimal exchange set for the given instance, which comprises of one two-cycle, one three-cycle, and one DPD chain.

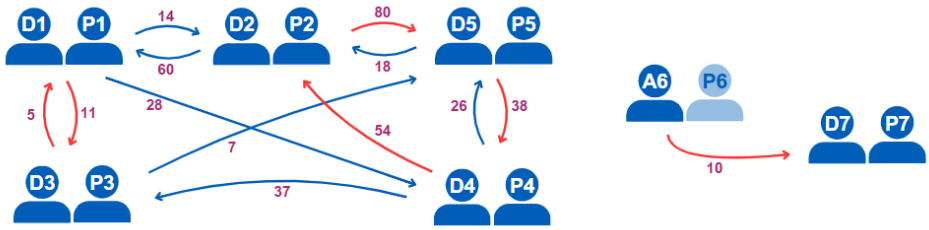

Figure 1.4: Example pool graph with the hierarchical optimal solution identified in red

The current version of the matching runs algorithm was developed by Manlove and O'Malley, and utilises integer linear programming (ILP) [11]. ILP solves optimisation problems by maximising, or minimising objective functions subject to constraints, where some or all the variables are restricted to integer values. In the UKLKSS context, the objective functions consist of a set of criteria defined by NHSBT, which are optimised lexicographically using ILP. This means each criterion is maximised in a fixed hierarchical order, with each one considered only after the preceding criteria have been satisfied as fully as possible.

However, as discussed by Manlove et al., clinical requirements may change over time, which could mean that the objective functions should no longer be optimised lexicographically [9]. An alternative technique mentioned in the paper's future work section was a weighted-sum approach, where each objective function is assigned a different weight [9]. These weights can be adjusted based on priorities for final exchanges, such as increasing the overall number of three cycles. Being able to make these adjustments means analysis can be performed on the trade-offs between objectives, as what constitutes as optimal may evolve with time. This has the potential to lead to more suitable solutions by providing greater flexibility in generating solutions based on ever-changing priorities.

Consequently, the purpose of this paper is to address the open question of the relative advantages and disadvantages of adopting a weighted-sum optimisation approach, as opposed to hierarchical optimisation. To achieve this, we outline three key objectives. The first objective is to reimplement the current hierarchical model, in order to enable a means of comparison for our weighted sum model. Our second objective is to adapt the reimplementation of the current hierarchical model into a weighted-sum model. Once both models are successfully implemented, our third and final objective is to compare these two models through empirical evaluation using synthetically generated kidney exchange instances.

Our work aims to contribute to the literature through the concrete implementation and evaluation of the weighted sum model. Through a series of experiments, our evaluation demonstrates the benefits of weighted sum for obtaining new solutions that could be deemed more optimal in future circumstances.

Chapter 2

Background

NHSBT developed the first algorithm used for matching runs in early 2007. However, it was limited to processing datasets up to 100 potential transplants, and only identified PKEs involving two donor-recipient pairs [11]. In May 2007, Manlove and Biró utilised graph matching algorithms to create a version that handled three-transplant exchanges [11], and increased the capacity of the datasets to 3000 potential transplants. Their research prompted the NHSBT to allow PKEs to include three-transplant exchanges in April 2008, and this improved version was used for matching runs between July 2008 and October 2011 [11].

Identifying an optimal set of two-way exchanges was solvable with algorithms, such as maximum bipartite matching. By introducing three-transplant exchanges, finding an optimal set of PKEs transformed into an NP-hard problem, meaning it became computationally difficult to solve efficiently.

From 2010 to 2011, O'Malley collaborated with Manlove and Biró to use integer linear programming (ILP) to include altruistic donors in an improved version of the 2008 algorithm [11]. Since January 2012, the NHSBT has been using in-house software developed by Manlove and O'Malley to run Manlove and Biro's 2011 ILP model for the quarterly matching runs [11].

ILP solves optimisation problems defined by objective functions and constraints, where the variables are restricted to integer values. While ILP typically involves a single objective function, it becomes a multi-objective ILP once more than one objective function is introduced [12]. In the case of NHSBT matching runs, a multi-objective ILP approach is employed. The objective functions and constraints are based on a set of criteria decided by NHSBT, as detailed in the Optimal Exchanges Criteria section. A hierarchical optimisation strategy is used to solve the resulting ILP, where the objective functions are prioritised in a specific order. Optimisation begins with the highest-priority objective. Once the highest-priority objective is maximised, the next objective is maximised subject to the previous one, and so on down the hierarchy. Our contribution is to implement an alternative weighted sum approach, which can be formalised as the following:

Given a set of n objective functions $\{f_1(x), f_2(x), \ldots, f_n(x)\}$, apply a set of n weights $\{w_1, w_2, \ldots, w_n\}$ to each corresponding objective function and sum them together. This method reduces multiple objectives into a single scalar objective [12]:

$$\sum_{i=1}^{n} w_i f_i(x)$$

The goal is to maximise this weighted sum. Determining the values of the weights is subjective to the desired final solution [12]. For our weighted sum model, we implement the same objective functions and constraints as NHSBT to enable a fair comparison between our reimplementation of the original hierarchical model.

Roth et al. described identifying the maximum weight set of kidney exchanges, including pairwise and three-way exchanges, as two different integer programming formulations i.e., edge formulation and cycle formulation [13]. Edge formulation assigns decision variables to each edge in the graph, whereas cycle formulation assigns variables to each cycle in the graph. These decision variables represent the inclusion

or exclusion of each edge or cycle in the final solution. In the context of kidney exchange, the likelihood of there being more edges than cycles is greater than the likelihood of having more cycles than edges. Abraham et al. concluded that, due to scaling issues with edge formulation, cycle formulation is a more efficient method for identifying maximal sets [14]. Challenges in scaling using edge formulation result from variable proliferation and an exponential growth in constraints. For a directed graph with n nodes, the number of possible edges is bounded by $O(n^2)$, which means the number of decision variables for edge formulation would grow quadratically with the graph size. If the graph had m edges, edge formulation would require $O(m^3)$ constraints to ensure that the selected edges form valid cycles of length at most L, where L is the maximum length [14]. Comparatively, cycle formulation only requires $O(m^2)$ constraints to ensure each node participates in at most one cycle. This led Manlove and Biró to adopt cycle formulation for their implementation. Since our approach involves reimplementing their algorithm, we also employ a cycle-based formulation.

In order to associate the decision variables with each cycle within our cycle formulation, it is necessary to first identify all potential cycles. One common approach is Johnson's well-known algorithm for enumerating elementary cycles in directed graphs [15]. However, the need for continual recalculation of strongly connected component subgraphs in Johnson's algorithm made its implementation impractical, as it was incompatible with the design of our custom pool data structure. Furthermore, Johnson's algorithm is intended to identify cycles of any length, whereas our requirement is specifically for cycles of length k, where k is the current maximum cycle length set by NHSBT. To avoid the unnecessary identification of cycles of non-relevant lengths and the subsequent post-processing cost of removing unwanted cycles, we instead implement Gupta and Suzumura's algorithm, which efficiently finds all bounded-length simple cycles in a directed graph [16].

Recent work has focused on the development of hybrid models for hierarchical optimisation, which combine cycle formulation with position-indexed chain-edge formulation (PICEF) [17]. PICEFs are compact integer programming models that use arc positions to represent cycles and chains [18]. The arc position is the specific place a directed edge sits within a cycle or chain in the kidney exchange graph, such as being the first, second, or third donation in a sequence [18]. However, PICEF has been shown not to be able to model maximising backarcs, one of the NHSBT objective functions, and thus we decided to solely use cycle formulation for our implementation [17].

Chapter 3

Optimal Exchanges Criteria

At present, the definition of optimality for the UK's matching runs is determined by a set of criteria established by NHSBT. These criteria are fundamental to the original hierarchical implementation presented by Manlove et al., and our own models, as we utilise the same criteria to enable accurate comparisons and evaluation. The ILP model performs hierarchical optimisation by sequentially prioritising each criterion according to their predefined order [19]:

i. **Maximise the number of effective two-transplant exchanges, including three-transplant exchanges with embedded two-transplant exchanges.**

   Figure 3.1 depicts a three-transplant exchange with an embedded two-transplant exchange between (D1, P1) and (D2, P2), highlighted in red. As discussed, two-transplant exchanges are preferred over three-transplant exchanges due to their simpler logistics. Additionally, with fewer pairs involved, the overall risk is lower if a pair withdraws from the exchange, as fewer pairs are affected and the likelihood of such an occurrence is reduced. Maximising the number of effective two-transplant exchanges in the final optimal exchange set guarantees the introduction of three-transplant exchanges has no effect on the maximum number of pairwise exchanges that could possibly take place.

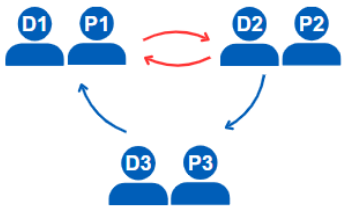

Figure 3.1: Example of three-cycle with embedded two-cycle highlighted in red

ii. **Subject to (i), maximise the total number of transplants.**

   A successful two-cycle would result in two successful transplants. For a short DPD chain, that would also equate to two transplants as the non-altruistic donor donates to the DDWL (refer back to Figure 1.3). Maximising the total number of transplants is equivalent to maximising the size of the optimal exchange set. This is important since the greater the number of people that receive transplants in each matching run, the greater the reduction in waiting lists and waiting times for patients.

iii. **Subject to (i) and (ii), minimise the number of three-transplant exchanges.**

   Although Criterion (i) ensures that a three-transplant exchange is not selected at the expense of a two-cycle, a two-cycle is still preferred over a three-cycle, even if the three-cycle includes an embedded two-cycle. Subsequently, Criterion (iii) is necessary to prioritise two-cycles over three-cycles when the choice involves the same donor-patient pairs. Figures 3.2 and 3.3 illustrate a specific example presented by Manlove et al. which involves deciding between the prioritisation between more two-transplant exchanges or more three-transplant exchanges. Due to Criterion (iii), the hierarchical model aims to minimise the number of three-cycles in the final exchange solution,

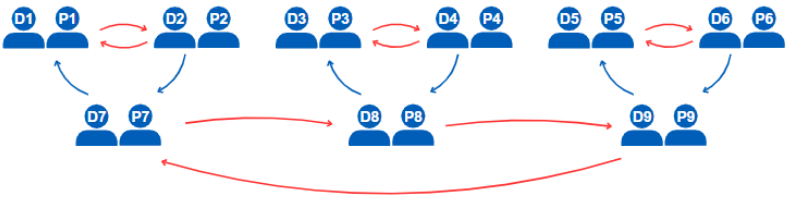

Figure 3.2: Expected hierarchical solution as maximum two-cycles

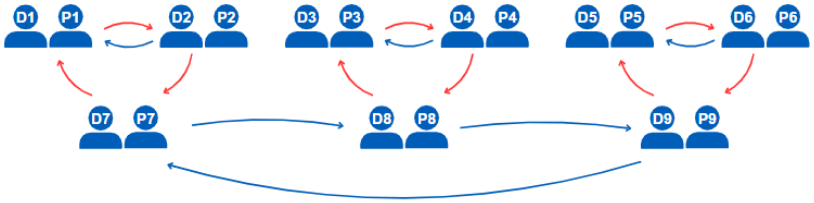

Figure 3.3: Not preferred solution as sacrificing the choice of two-cycles for three-cycles

and therefore should choose the red cycles highlighted in Figure 3.2, rather than the solution in Figure 3.3. In our evaluation of our weighted sum model, we examine this hierarchical example in relation to the trade-offs between Criterion (iii) and other criteria.

iv. **Subject to (i) to (iii), maximise the number of embedded 2-transplant exchanges in the 3-transplant exchanges.**

This is equivalent to maximising the number of backarcs in three-cycles. For a given cycle C with edges (u,v), (v, w), (w, u), then a backarc would be denoted by (v, u), (w, v), or (u, w). An example backarc is highlighted red in Figure 3.4 going from (D2, P2) to (D1, P1). Maximising backarcs is critical for providing fault-tolerance in a three-cycle, and lowers the risk of a three-cycle breaking down if a donor-patient pair were to withdraw. This may appear similar to Criterion (i) regarding embedded two-cycles, but the significance of both criteria is demonstrated in Figures 3.2 and 3.3, where the solution with pairwise exchanges is preferred over the one involving the backarcs.

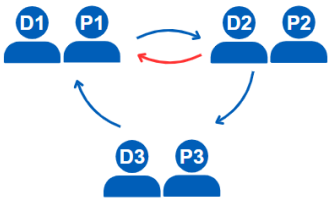

Figure 3.4: Example of three-cycle with a backarc highlighted in red

v. **Subject to (i) to (iv), the overall match score is maximised i.e. the sum of scores calculated for each potential transplant in all exchanges in the result.**

The matching scores for each edge weight are calculated by NHSBT based on [19]:

1. The number of quarterly matching runs in which the recipient has previously participated, multiplied by 50. Recipients who have historically been unable to be matched are given greater priority, and thus higher scores.
2. Points based on the degree of sensitisation, ranging from 0 to 50 points depending on the recipient's calculated reaction frequency (cRF), which varies from 0% to 100%. Sensitisation occurs when a recipient has developed antibodies against certain human leukocyte antigens

(HLA) present in potential donors [20]. Antigens are foreign substances that trigger the production of antibodies by the immune system [21]. The cRF score is also known as a calculated Panel Reactive Antibody (cPRA) score, and estimates the proportion of donors in the kidney exchange program who will be tissue-type incompatible with that recipient [17]. The greater the score, the more highly sensitised the recipient is, meaning they are more likely to reject a kidney from a donor because they have high levels of anti-HLA antibodies [20]. A higher score indicates that the recipient has a higher level of antibodies, which makes finding a compatible donor more difficult.

3. HLA mismatch points, giving 15, 10, 5 or 0 points for mismatch levels 1, 2, 3 or 4, respectively. The NHS determines these HLA levels predominately by comparing donor and recipient HLA types, and counting the number of unique donor HLA antigens not present in the recipient [22]. Mismatches in HLA between the recipient and donor can lead to rejection of the transplanted organ because the immune system recognises foreign antigens as threats [23]. Higher HLA matches reduce the risk of organ rejection, and therefore are more likely to lead to better transplant outcomes. Although this may seem to conflict with (2), the overall matching score is designed to balance the trade-off between the difficulty of finding a compatible donor and the likelihood of achieving long-term graft survival.
4. Donor-donor age difference points, where 3 points are awarded if the donor-donor age difference of less than 20 years

Manlove et al. derive constraints and objective functions from the set of criteria, in order to ensure that each preceding criterion is upheld within the ILP model. These objective functions are created and solved using CBC, an open-source mixed-integer programming (MIP) solver implemented in C++ [9]. CBC was chosen for its open-source licensing, which enables simpler deployment within the NHS.

An earlier effort to implement the hierarchical ILP model presented in the Manlove et al. paper was carried out by James Trimble, a former doctoral student under Manlove's supervision [9]. Trimble's approach of transforming the input into a custom pool data structure offered greater flexibility compared to forcing the input into a predefined graph structure. Designing a custom data structure simplifies the implementation by enabling precise control over how relationships between objects are represented, as well as the operations associated with each node, edge, and the overall graph structure. Our work builds upon the foundational contributions of Trimble, though the key distinctions between our implementations will be highlighted throughout the Methodology. The evaluation of our own hierarchical implementation, in comparison to our weighted sum version, utilises the same benchmarks Manlove et al. used to assess their original ILP implementation.

Chapter 4

Project Execution

4.1 Modelling the pool graph

Our hierarchical model is a reimplementation of the ILP model currently used by NHSBT, and based on the solution methodology presented in Manlove's Paired Kidney Exchange paper, which will be referred to as the original paper **manlove˙how˙2018-1**.

The pool of incompatible donor-recipient pairs and altruistic donors can be modelled as a directed graph, where each donor-recipient pair and altruist is represented by a node. An edge exists between node A and node B if A is compatible with, and able to donate to B. Associated with each edge is a match score, which is assumed to be provided within our input and precomputed by NHSBT using the calculation outlined in the Literature Review. To avoid confusion with the weighted sum weightings, the score will always be referred to as a score, despite being an edge weight. Additionally, we represent non-altruist donor-patient pairs as tuples, i.e., (donor, patient). For altruist donors, we either refer to them as a single entity (altruist) or associate them with a dummy-patient (altruist, dummy_patient).

4.2 Dataset generation

Information about the pool, including donors, recipients, compatibilities between them, and scores, is provided by a dataset generator that creates random static kidney exchange instances. This particular dataset generator was chosen for its improvements of the Saidman generator, which is used for kidney exchange instances in literature [24][25]. These improvements were based on statistical evidence from the UKLKSS, making it more representative of real-world UK scenarios and therefore better suited to our implementation. One significant alteration was removing the Saidman generator's assumption that the blood groups of donors and recipients and the cPRA levels of the recipients are independent [24]. From a clinical perspective, if a patient's donor(s) were both blood-type compatible and tissue-type compatible, they would be considered fully compatible and could proceed with a transplant directly, without needing to participate in a kidney exchange program [24]. This makes the Saidman generator's assumption unrealistic, as it is more likely that a patient who is blood-type compatible with their donor(s) would be tissue-type incompatible with their donor(s). Since cPRA reflects the level of a recipient's sensitisation, assuming it is independent of blood type ignores the fact that highly sensitised patients are generally harder to match. As a result, the generated datasets would likely overestimate the number of fully compatible donor–recipient pairs. The improved instance generator corrects this by altering the cPRA distribution used through incorporating observed dependencies between blood type and cPRA from the UKLKSS data, and thus producing more realistic kidney exchange instances.

The NHSBT matching runs involve approximately 300 recipients at a time, we generate our own datasets of a similar scale, with each instance consisting of 300 recipients [17]. Since the start of the UKLKSS in 2012, altruistic donors have accounted for approximately 6% to 10% of all annual living donor kidney donations [26]. To reflect this real-world distribution we set the proportion of altruistic donors in the generated dataset to 0.1, as all altruistic donors will be included in the final solution, either by being matched or by donating to the DDWL. To ensure our generated dataset closely reflects real-world instances, we utilise the recipient blood group and donor blood-group distributions determined by the recipient blood

group from the improved instance generation paper [24]. We assume that each patient has exactly one incompatible donor willing to donate on their behalf in exchange for a kidney from another donor, since this is the majority of real-world cases.

```
{ "data" :
    {
    "1" : { "sources" : [1],
            "dage" : 32,
            "matches" : [ { "recipient" : 2, "score" : 32}]},
    "2" : { "sources" : [2],
            "dage" : 43,
            "matches" : [ { "recipient" : 1, "score" : 51},
                          { "recipient" : 3, "score" : 71} ]},

    "3" : { "sources" : [3],
            "dage" : 65,
            "matches" : [ { "recipient" : 1, "score" : 19} ]},
    "5" : { "altruistic": true,
            "dage" : 50,
            "matches" : [ { "recipient" : 3, "score" :  23} ]}
    },
    "recipients": {
        "1": {
          "cPRA": 0.20,
          "bloodtype": "A",
          "hasBloodCompatibleDonor": false
        },
        "2": {
          "cPRA": 0.14,
          "bloodtype": "B",
          "hasBloodCompatibleDonor": false
        },
        "3": {
          "cPRA": 0.89,
          "bloodtype": "AB",
          "hasBloodCompatibleDonor": true
        }
    }
}
```

Figure 4.1: Example JSON dataset input file

These generated kidney exchange instances are utilised to construct synthetic pools for assessing the performance of the weighted-sum program and for comparing it with the hierarchical version. Figure 4.1 provides an example of the dataset output, which is a JSON file containing two top-level keys, “data” and “recipients”. The desired graph instance is illustrated in Figure 4.2 to supplement the understanding of how the JSON file translates to a directed graph, and highlighted in red is the expected optimal solution.

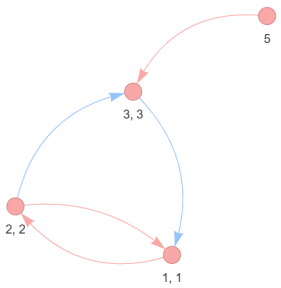

Figure 4.2: Visualisation of graph created from the JSON file in Figure 4.1

The “data” key maps to a dictionary where each key is a donor’s ID. Each donor is associated with a dictionary containing the IDs of their original intended recipients who they are incompatible with (“sources”), and the IDs of the patients in the pool who they are compatible with (“matches”). Ev-

ery donor's dictionary also includes information about each donor's age ("dage"), and whether they are marked as an altruistic donor. Altruistic donors have no associated ids of any intended recipients ("sources"), since they do not have specific patient(s) who they wish to donate to [27].

The "recipients" key contains a dictionary where each key is a patient's ID. Each patient is associated with a dictionary that provides their cPRA score ("cPRA"), blood-type ("bloodtype"), and whether their original donor is blood-type compatible with them. The cPRA scores are used for evaluating the number of highly sensitised patients that have been selected as part of the final optimal exchange set by our implementation. Patients with cPRA scores greater than or equal to 85% are classified as highly sensitised [28]. This metric is valuable for our evaluation of the models, as highly sensitised patients are harder to match, and thus, an increase in the number of highly sensitised patients included in the final optimal exchange set represents a more positive outcome.

The JSON output generated by the dataset serves as the input format for our program. Pool information is read from the JSON file using Python's built-in JSON package, then stored in our custom pool data structure.

Class	Purpose	Attributes
Pool	- Contains all the elements of the pool graph, including the identified cycles.	- patients (dictionary) - donor_patient_nodes (list) - altruists (list) - all_cycles (list)
Patient	- Stores information about a specific patient (NHS id, cPRA score, blood-type) - Used in DonorPatientNode to represent the patient in a donor-patient pair - Contains list of Gurobi MIP binary variables for solving ILP.	- id (int) - mip_vars (list) - cpra (float) - blood_type (string)
Altruist	- Stores information about a specific altruistic donor (NHS id and age) - Used in DonorPatientNode to represent the donor in a donor-patient pair - Has a list of recipient patients who the altruistic donor is able to donate to - Contains a list of Gurobi MIP binary variables for solving ILP - Contains its own associated Gurobi MIP binary variable to denote whether the altruist has been matched in a cycle in the final optimal solution	- id (int) - age (int) - recipient_patients (list) - mip_vars (list) - mip_unmatched (Gurobi binary variable)
Donor	- Stores information about a specific non-altruistic donor (NHS id and age) - Used in DonorPatientNode to represent the donor in a donor-patient pair - Contains list of Gurobi MIP binary variables for solving ILP	- id (int) - dage (int) - mip_vars (list)
DonorPatientNode	- Represents a donor-patient pair. - Contains a donor and patient who are incompatible with each other. - Includes altruists and dummy patients. - Boolean logs if the donor is altruist. - Has a list of recipient patients who the donor is instead able to donate to. - Out edges of this node are tracked in a list of DonorPatientEdge objects - Gets assigned an index.	- donor (Donor) - patient (Patient) - recipient_patients (list) - out_edges (list) - is_altruist (boolean) - index (int)
DonorPatientEdge	- Represents an edge between two DonorPatientNodes - Gets stored within a DonorPatientNode out_edges list - That DonorPatientNode is the source, and the target node is the donor_recipient_node stored in this DonorPatientEdge - Stores the edge score	- donor_recipient_node (DonorPatientNode) - score (int)
Cycle	- Contains a list of DonorPatientNodes which are in a cycle - Contains its own associated Gurobi MIP binary variable to denote whether the cycle has been chosen to be included in the final optimal solution. - Stores length of the cycle - Boolean logs if the cycle is in fact a pseudo-cycle, and therefore a chain - Gets assigned an index.	- donor_patient_nodes (list) - length (int) - index (int) - is_chain (boolean) - mip_var (Gurobi binary variable)

Table 4.1: Custom data structure classes to represent the pool

4.3 Representing chains as pseudo-cycles

Table 4.1 lists the various Python classes created for the pool data structure, and their associated attributes. Notably in Table 4.1, no chain class is defined, which is one key distinction between this and Trimble's implementation. Trimble treats cycles and altruist chains as distinct, whereas the original paper reinterprets altruist chains as pseudo-cycles. This is achieved by introducing a dummy recipient for each altruistic donor, who serves as a theoretical patient compatible with any donor in the pool. As illustrated in Figure 4.3, the altruistic donor A1 is associated with a dummy intended recipient P1, with whom they are deemed incompatible. Donors D2, D3, and D4 are modelled as compatible to P1, and thus able to donate to P1. A1 is considered capable of donating to any patient in the pool, namely P2, P3, and P4. Given that D2 is compatible with P3, the resulting pseudo-cycle represents the underlying DPD chain, as demonstrated in Figure 4.4.

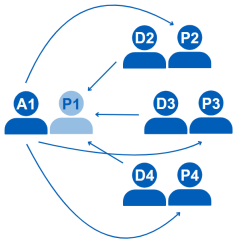

Figure 4.3: Altruist A1 can donate to any patient P2, P3, or P4.
Dummy patient P1 can receive from any donor D2, D3, D4.

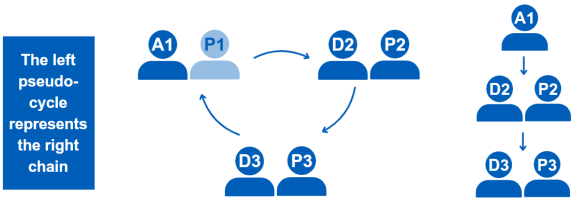

Figure 4.4: How pseudo-cycles model chains.

Pseudo-cycles simplify the integer programming model by enabling cycles and chains to be represented within a single set of constraints and objective functions, rather than two separate sets. Additionally, Trimble was required to define a search function for finding both cycles and chains, whereas our implementation only requires one single cycle-finding function. Storing cycles and chains separately increases the complexity of the ILP model, since more variable types are required to ensure the constraints are enforced properly. The term donors will begin to be used inclusively to also refer to altruistic donors, and the term cycles will be used to refer to both ordinary cycles and pseudo-cycles (chains).

4.4 Cycle-finding

A naive brute-force DFS approach was used to initially find all these potential cycles, which explores all possible paths starting from a donor-patient node, incrementally increasing the depth until either the maximum allowed cycle length is reached, or there are no more unvisited outgoing edges, at which point it backtracks. The runtime of this implementation of naive DFS is $O\left(n \cdot \left(d^k \cdot k\right)\right)$, where n is the number of donor-patient nodes (including altruist nodes), k is the maximum cycle length, and d is the maximum branching factor, defined as the maximum out-degree for a node n in the pool [29]. This approach was chosen for its simplicity, as the primary objective was to first understand how to formulate and solve the ILP. For smaller datasets containing approximately 300 instances, naive DFS performed effectively. However, scaling up to larger datasets with around 500 instances, the program experienced timeouts due to the repeated exploration of equivalent cycles, as duplicate removal was deferred to a post-processing stage.

To avoid the redundant exploration, we instead assign each node with an index before performing DFS. Within the DFS, a constraint is applied to only continue with the DFS if the index of the current node is less than the node it is about to visit. This prevented the cycle-finding from timing out.

To further reduce the runtime for cycle finding in large datasets, Gupta and Suzumura's algorithm for finding all bounded-length simple cycles in directed graphs was implemented in place of the original DFS approach [16]. Although the theoretical runtime is bounded by $O\left((c+n)(k-1)d^k\right)$, which is equivalent to the naive DFS bound $O\left(n \cdot \left(d^k \cdot k\right)\right)$, Gupta and Suzumura's algorithm is significantly faster in practice. This is because they follow a similar approach to avoiding redundant searches as Johnson's well-known cycle enumeration algorithm, but rather than relying on strongly connected component subgraphs, their method constructs subgraphs during each node iteration using Breadth-First-Search (BFS).

Gupta and Suzumura's algorithm begins with an outer loop that iterates over each node u in our original pool graph G. For every *u*, it constructs a subgraph H induced by the vertices {*u, u+1,...n*}, where n is the total number of vertices in G. The subgraph H is subsequently used to construct another subgraph G' by performing a BFS starting from node *u*, traversing up to *k-1* levels, where *k* is the maximum cycle length. If a node *v* in H is unreachable from node *u*, then it is impossible for a k-length cycle including u and v to exist. This is why the BFS is limited to *k-1* levels, and demonstrates how creating the subgraph G' helps reduce the search space for length *k* cycles.

After identifying G', the algorithm uses a function called CYCLE_SEARCH to recursively perform a modified version of DFS to enumerate all the simple cycles of length at most *k* that start and end at a given node *u* in the subgraph G'. The modified DFS avoids redundant exploration by utilising several key data structures:

- A stack to track the current path during the DFS traversal.
- Forward and backward length counters to track path distances for all nodes. Forward length is the current depth in the DFS tree, which equates to the forward path length from the start node, to the current node. Backward length is the minimum distance from the current node *v* which has invoked CYCLE_SEARCH, to the start node *u*. The backward length represents the shortest known distance back to the start node via a cycle. Initially set to infinity, the backward length is only updated when a cycle is found.
- Lock dictionary and a blocking list dictionary called Blist to prune unnecessary branches and prevent revisiting nodes that have previously failed to lead to a valid cycle. Lock stores an integer value for each node. If node *u* has been visited at least once and is not on the stack, Lock(v) stores the maximum possible length of a path from node *v* to the start node *u* that has the possibility of reaching back to the start node with at most *k* total edges in it, plus one. The Blist dictionary remembers dead-end paths by tracking the nodes that have been unsuccessful in leading to a cycle back to start node *u*.

The CYCLE_SEARCH function returns the backward length of the node that initiated its call. It is first invoked by the start node *u*, which is the node used to construct G' through BFS. Each time CYCLE_SEARCH is executed on a node *v*, *v* is added to the stack, its backward length is set to infinity, and its lock value is set to the forward length of *v*. After these updates, the function iterates over each of *v*'s neighbours. If a neighbour node *w* is the start node, then a cycle has been detected and can be added to the list of found cycles. In this case, the backward length of *v* is set to 1, as *v* is one edge away from the start node *u*. Otherwise, if the forward length plus one is less than the maximum cycle length *k*, and is also smaller than the neighbour's lock value, CYCLE_SEARCH is recursively called on the neighbour node and the process repeats. After iterating through all the node's neighbours, if a cycle including v has been found, then the backward length would have been updated to the minimum distance from the node *v* which has invoked CYCLE_SEARCH, to the start node *u*. This triggers the RELAX_LOCKS function, which relaxes the lock constraints on nodes that had previously been pruned due to failed paths, enabling them to be reconsidered for inclusion in valid cycles. Specifically, it sets the new lock value for node *v* to *k - backward_length + 1*, representing the maximum allowed forward path length from that node to still form a cycle of length at most *k*. However, if no cycle is found including node *v*, then it is added to the Blist of its neighbours to prevent future searches through the same dead-end.

4.5 Defining constraints

Once all potential cycles have been found, cycle formulation can be applied by associating a binary decision variable for each identified cycle. Every donor and patient stores a list that contains the binary variables associated with the cycles in which they could potentially participate. For altruist donors, this list also contains an "unmatched" binary variable, which has value 1 if the altruist remains unmatched, indicating they have not been selected in any of the final optimal exchanges. These cycle variables can subsequently be used for imposing the following constraints on the final solution:

1. Each donor and each patient can only participate in at most one cycle. No donor can donate to multiple patients, and no patient can receive multiple donor kidneys.
2. If an altruist donor does not participate in the final optimal exchange set, it will be "unmatched". Unmatched altruists still contribute to the final total number of transplants as they are considered to have donated to the deceased donor kidney waitlist (DDWL).

Enforcing Constraint (1) means ensuring that the sum of the binary variables in each list does not exceed 1, therefore guaranteeing each donor and each patient will only participate in at most one cycle. However, for altruistic donors, the constraint is required to be that the sum of the binary variables in each list must equal 1. This is because Constraint (2) is implemented by introducing the additional "unmatched" binary variable for each altruistic donor. Recall that this unmatched binary variable is also included within each altruist's list of associated binary variables, along with the cycle variables. Requiring the sum of this list to equal 1 ensures each altruist must either be included within a pseudo-cycle or remain unmatched.

4.6 Defining objective functions

After the constraints have been applied, we can proceed to define the objective functions for the model. As mentioned in the Optimal Exchanges Criteria section, the original hierarchical ILP model was solved using CBC. However, due to the absence of licensing restrictions for this project, coupled with limited familiarity with C++, Gurobi was selected as the solver for our implementation. Python was subsequently selected as the programming language, due to the integration of Gurobi's optimisation engine with Python. We closely follow their original methodology for formulating each criterion outlined in the Optimal Exchanges Criteria section as objective functions in our implementation using Gurobi, with our modifications noted where applicable:

Criterion (i): Maximise the number of effective 2-transplant exchanges

Given a directed graph G, the original model constructs an undirected graph G' = (V, E) using the same set of vertices. Each edge in G' represents a two-cycle in G, meaning an edge (u, v) exists in G' if and only if (u, v) and (v, u) exist in the directed graph G. The Micali-Vazirani implementation of Edmonds's algorithm is then applied to compute N, the size of a maximum cardinality matching in G' [9]. To enforce Criterion (i), the constraint that the number of two-cycles and three-cycles selected in the final solution must be greater than or equal to N.

Given that our implementation has already enumerated all feasible cycles, we instead enforce Criterion (i) from the literature review by formulating a maximisation objective function (MaxTwoCycles), rather than incorporating it as a constraint. Each potential cycle is associated with a binary decision variable that indicates whether it is selected in the final solution. The objective function assigns a coefficient of 1 to variables corresponding to two-cycles or to three-cycles that contain an embedded two-cycle. All other cycles are assigned a coefficient of 0. A binary decision variable is also associated with each altruistic donor to signify whether the donor remains unmatched. As unmatched altruistic donors do not contribute to two-transplant exchanges, their corresponding variables are assigned a coefficient of 0. This ensures that the objective function prioritises the inclusion of two-transplant exchanges.

Criterion (ii): Subject to (i), maximise the total number of transplants

Both our hierarchical models create a maximising objective function (MaxSize) where the cycle variables are multiplied by coefficients corresponding to the lengths of their respective cycles. Specifically, a coefficient of 2 for two-cycles and 3 for three-cycles, to reflect the contribution of each cycle to the overall objective function. Additionally, a binary decision variable is introduced for each altruistic donor to indicate whether the donor remains unmatched. In their original implementation, these variables are assigned a coefficient of 1 in the objective function to discourage leaving altruistic donors unmatched unless there is no possibility for them to be involved in a cycle that yields higher weights (2 or 3).

Criterion (iii): Subject to (i) and (ii), minimise the number of 3-transplant exchanges

Define a minimising objective function, such that all cycle variables associated with three-cycles are multiplied by a coefficient of 3 whilst all other cycles, and altruistic donor variables are assigned a coefficient of 0. The inclusion of three-cycles increases the overall objective value, and given the minimisation objective, this discourages the selection of three-way transplant exchanges.

The Gurobi Mixed-Integer Programming (MIP) solver is limited to optimising all objectives in the same direction, either maximising, or minimising each of them, and does not support a combination of maximisation and minimisation objectives. As a result, the minimising objective function for Criterion (iii) was converted into a maximisation objective function (MinThreeCycles) by negating the quicksum of the cycle and altruist variables. For the remaining criteria, we created our objective functions as defined in the original hierarchical model, with the sole difference being the use of Gurobi as opposed to CBC.

Criterion (iv): Subject to (i) to (iii), maximise the number of embedded 2-transplant exchanges in the 3-transplant exchanges

As mentioned in the Optimal Exchanges Criteria section, this criterion is equivalent to maximising the number of backarcs. Thus, another maximising objective function (MaxBackarcs) is defined where all three-cycle variables are multiplied by a coefficient equal to the number of backarcs. Two-cycle and altruistic donor variables are multiplied by 0 as they are not relevant to this criterion.

Criterion (v): Subject to (i) to (iv), the overall match score is maximised

A final maximising objective function (MaxOverallScore) is applied, where each cycle variable is multiplied by a coefficient equal to the sum of the matching scores of the edges in the cycle. Altruist donor variables are assigned a coefficient of 0 because if they are left unmatched they donate to the DDWL, where the recipient is unknown and therefore a matching score cannot be calculated.

Once the constraints and objective function have been defined within Gurobi, the ILP model becomes fully defined and ready for optimisation. We used Gurobi's setObjectiveN method to set these multiple objective functions in order of priority. Trimble's approach used the setObjective method, and included an additional constraint to enforce each previous objective in between each optimisation for each criterion.

4.7 Implementing the weighted sum model

When transforming the hierarchical model to the weighted sum model, the constraints and objective functions remained the same as we are evaluating the models using the same NHSBT criteria. However, the priorities within the objective functions had to be altered from having an increasing order of priorities for each criterion, to all being set to zero. This is because in the weighted sum model, there is no predetermined priority order, but rather the weights inform a priority order implicitly. The weighted sum approach produces a less definitive priority order compared to hierarchical optimisation, as it considers trade-offs between objectives rather than pursuing absolute maximisation of each one.

The weight list entailing the weights for each objective function is obtained through the program command line flag "weights". We carry out numerous experiments to determine the ideal selection of these

weights as part of our evaluation of the weighted sum model.

In the resulting optimal hierarchical and weighted sum solutions, Gurobi assigns values greater than 0.5 to decision variables that are included as part of the optimal solution. To extract the final set of optimal exchanges, all cycle variables are iterated over, and those with values exceeding 0.5 are identified as part of the solution.

4.8 Output

Our program generates three text files and two HTML files. Gurobi MIP solver finds one proven optimal solution for the model by default, but stores sub-optimal solutions in a Solution Pool. Once the optimal solution has been determined, it is documented in the first output text file, along with details regarding the pool and characteristics of the final solution for further evaluation. In addition, the sub-optimal solutions are retrieved and recorded in a separate second output text file. This is to provide insight into the rationale behind Gurobi's selection of the optimal solution over alternative, sub-optimal solutions. For example, one may observe that the optimal solution includes a greater number of two-cycles, therefore contributing to its eventual selection. The third output text file lists the final objective values for each solution's objective functions, which was used for the comparative analysis of Trimble's hierarchical model with our own.

The last two output files are HTML files generated from a visualisation tool that we developed to illustrate the solutions in the pool graph instances. One output file is a visualisation of the entire original pool graph, with the nodes and edges that are included within the optimal solution highlighted in red. The other is a visualisation of only the selected nodes and edges. We transform the original pool into a NetworkX DiGraph, and then convert NetworkX DiGraph into a graph network visualisation through PyVis.

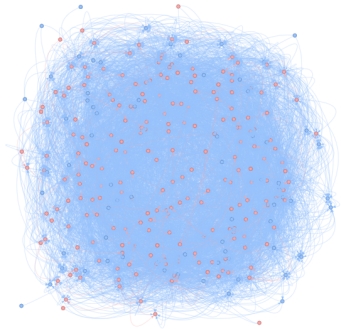

Figure 4.5: Visualisation of a 300 patient pool graph. Highlighted in red are the nodes and edges selected in the optimal exchange returned by the hierarchical model.

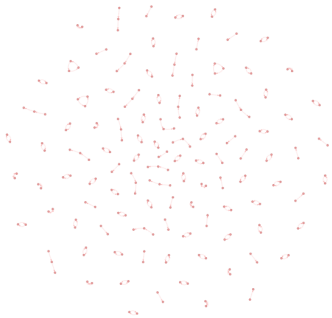

Figure 4.6: Visualisation of a 300 patient pool graph with only the nodes and edges selected in the optimal exchange shown.

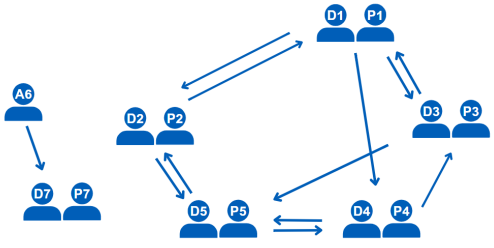

Figure 4.7: Example pool graph

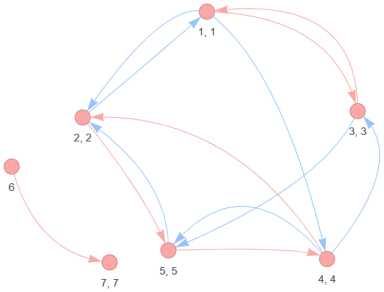

Figure 4.8: Visualisation of the hierarchical optimial solution of Figure 4.7

Figure 4.5 is an example of a visualisation of the full generated graph instance for a dataset containing 300 patients. It demonstrates the complexity of kidney exchange problems, thereby emphasising the necessity to develop kidney exchange solvers. Meanwhile, Figure 4.6 provides an example of a visualisation of only the final nodes and edges selected for the optimal solution. In addition, a concrete example of a pool graph and the resulting visualisation is provided with Figures 4.7 and 4.8. The source code for the project is hosted in a GitHub repository [30].

Chapter 5

Evaluation

5.1 Testing the hierarchical reimplementation

To verify the correctness of our implementation of Gupta's algorithm, we first ran simple instances to confirm the number of cycles returned was as expected. Then we compared the number of cycles detected by our implementation to those found using Trimble's implementation across multiple random datasets to ensure the number was the same.

For checking the implementation of the objective functions, we evaluated the individual objective functions using specifically constructed JSON instances with known expected outputs. This allowed us to verify the solution characteristics against the predefined expectations. The only objective that could not be tested in isolation was the minimisation of three-cycles. When this criterion is specified without any accompanying objective functions, the solution trivially selects no cycles at all, and therefore had to be tested in tandem with other objectives, such as maximising two cycles.

We further validated our model by verifying that the final results from running our ILP model on multiple generated datasets matched the number of donors, patients, altruists, two-cycles, three-cycles, and chains in Trimble's identified optimal exchange set.

After proving the correctness of our objective function implementation, we compare the final objective values obtained by Trimble's hierarchical model and our own to analyse the difference between our implementations. The only variance observed was in the last two objective values for Criterion (iv) maximising the number of backarcs, and Criterion (v) maximising the overall score objective functions.

The reason for our differing maximising backarcs objective values is our pseudo-cycle modelling which means we consider backarcs differently to Trimble. In a two-length DPD chain where A1 is an altruists, and (D2, P2) is compatible with (D1, P1), our implementation would consider the red arcs in Figure 5.2 as backarcs, whilst Trimble would treat the green arcs in Figure 5.1 as backarcs. The edges from (D2, P2) to A1 and from (D1, P1) to A1, as depicted in Figure 5.2, exist due to the dummy patient associated with A1 who is modelled to be able to receive kidneys from all donors. Similarly, the edge A1 to (D2, P2) exists because all altruist donors are modelled as able to donate to all patients. As Trimble does not utilise pseudo-cycle modelling, the green edge from A1 to (D2, P2) in Figure 5.1 does not actually exist within Trimble's pool graph. Nevertheless, the edge is included as a backarc for the maximising backarc objective function due to the altruist's ability to donate to (D2, P2). If (D1, P1) withdrew an exchange would still be able to occur, meaning the fault-tolerance of the exchange increases.

A second reason for differing objective value is because Trimble's implementation of the maximising score objective function incorporates a supplementary score function that increases the cycle variable coefficient by an additional donor age difference score. This age difference score is calculated using the absolute difference between the ages of the two donors in a given edge. If this absolute difference is less than or equal to 20, this entails a small age difference, resulting in an additional 3 points added to the original score. The absolute difference is then subtracted from a set maximum age of 70, and then the answer is squared and divided by 100,000 to create a tie breaker score, which is also added to the original score. The smaller the age difference, the greater the tie breaker score. For two potential exchanges

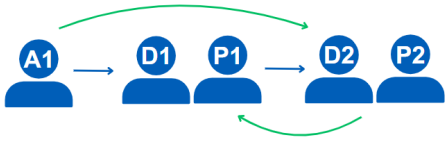

Figure 5.1: Edges considered as backarcs in Trimble's implementation

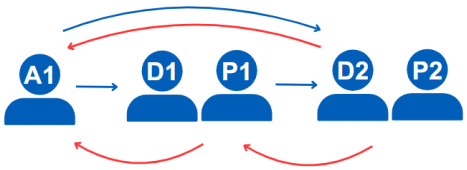

Figure 5.2: Edges considered as backarcs in our implementation

with the same score, the supplementary score function creates a preference for the exchange with a lower age difference. However, as described in the literature review, the NHSBT calculation for the matching score already considers the donor-donor age difference points. Consequently, our implementation does not include this supplementary score function, and therefore we do not get the same final maximising overall score objective value.

5.2 Weighted sum evaluation

The performance of the original hierarchical ILP model was assessed by evaluating the impact of each individual objective, in addition to characteristics of the pool compared to identified solutions. Each individual objective is tested in our reimplementation of their hierarchical model to verify its correctness. However, using this as an evaluation method would not be feasible for comparing the hierarchical and weighted sum approaches, as the weighted sum requires all constraints to be considered collectively. Instead, we compare the final exchange set characteristics to determine the differences between the models. These characteristics are the total score, transplants, non-altruist nodes, two-cycles, three-cycles, altruists, two-length chains, and three-length chains. A majority of the characteristics we chose for evaluation were used for the evaluation of the original ILP model [9].

As mentioned in the Project Execution section, the parameters for the dataset generator include setting the proportion of altruistic donors to 0.1, with the recipient and donor blood group distributions based on the recipient blood group from the improved instance generation paper [24]. We assume that each patient has a single incompatible donor willing to donate on their behalf in exchange for a kidney from another donor, and specify the number of patients included in the dataset during the explanation of each experiment, as this varies.

Based on the criteria, we identified the most notable differences in prioritisation includes attempts to maximise the overall score, number of transplants, and number of three cycles selected in the final optimal solution. We also experimented with weights to identify a set that would allow the weighted sum model to generate the same solutions as the hierarchical model.

5.2.1 Weight choices and normalisation

Inevitably, the weighted sum model will result in a series of Pareto-optimal solutions, meaning no objective can be improved without making at least one other objective worse [12]. Through a variety of experiments, we demonstrate the greater flexibility the weighted sum provides in comparison to the hierarchical model by exploring the obtainable Pareto-optimal solutions, and interrelationships between the criteria. This increased versatility is a direct result of being able to assign specific weights to each objective function. A key challenge lies in determining the appropriate values for these weights. In our

evaluation, we follow the usual practice of assigning weights such that their sum equals one [12]. This strategy is further justified by the fact that any set of arbitrary weights can be normalised to sum to one by dividing each weight by the total sum of the set of weights. Additionally, this approach aids interpretability since each weight represents the relative importance of its corresponding objective function as a proportion of the whole.

The choice of weights depends on the relative importance of each objective for a specific situation, as well as an appropriate scaling factor [12]. Different objectives naturally have different magnitudes, and the scaling effect is when the optimisation is distorted as certain objectives overpower others simply because of its larger inherent value, not because it is more important. Scaling factors are used to counteract the scaling effect by rescaling the objectives, making them comparable. To investigate the scaling effects in our weighted sum model, we apply it to generated datasets with 100, 300, and 600 patients, then observe the resulting objective values listed in Table 5.1. We run the weighted sum model by assigning a weight of 0.2 to each objective function. The value 0.2 was chosen to apply equal weighting to all five objectives, and ensure sum of the weights equals 1.

Objective	**MaxTwoCycles**	**MaxSize**	**MinThreeCycles**	**MaxBackarcs**	**MaxOverallScore**
100 Patient Objective Value	9	25.5	-7	11	1,245
300 Patient Objective Value	52	221	-67	53	12,988
600 Patient Objective Value	87	321.2	-99	97	18,424

Table 5.1: Objective values for each objective function from running 100, 300, and 600 patient generated datasets

Table 5.1 shows that MaxOverallScore and MaxSize are on a different scale than the other objectives, with the gap widening as the size of the dataset increases. This scaling effect can be avoided by normalising the objective functions [12]. A standard method of normalisation for weighted sum models is using min-max normalisation, given by the formula:

$$x_{\text{norm}} = \frac{x - \min(X)}{\max(X) - \min(X)}$$

To apply normalisation, we chose a minimum objective value of 0 for all objective functions as it could be the case that the final optimal exchange set is empty. Then, to obtain the maximum objective value for each objective we run a separate optimisation model on each objective individually. For example, we solely run the objective function MaxOverallScore on the pool to use the returned value as the maximum. This method proved effective for smaller datasets with a size of 100 patients, but running on larger datasets with about 300 patients was too slow. Subsequently, we instead ran the model twice to manually calculate a scaling factor. The first run assigns a weight of 0.2 to each objective to determine the objective values without introducing preference or bias. These objective values are used to help guide the magnitude difference between the new assigned weights to ensure they correctly represent the desired relative importance.

5.2.2 Prioritising the overall score

The example presented by Manlove et al. discussed in the literature review is suited for demonstrating how the weighted sum model can be leveraged. We run both our hierarchical model and weighted sum model on the pool graph instance shown in Figures 5.3 and 5.4, where we set the matching scores of all the edges to 1, except for edges unique to the 3 three-cycles within the graph i.e. (6,6) to (2,2), (2,2) to (7, 7), (1,1) to (5,5), (4, 4) to (1, 1), (8,8) to (3,3), and (3,3) to (9, 9). As a result, the total score of the optimal solution identified in red in Figure 5.3 by our hierarchical approach would be 9. In contrast, Figure 5.4 shows the optimal solution identified by our weighted sum model using the weights listed in

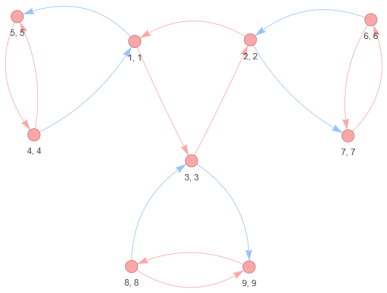

Figure 5.3: Hierarchical solution with total matching score 9.

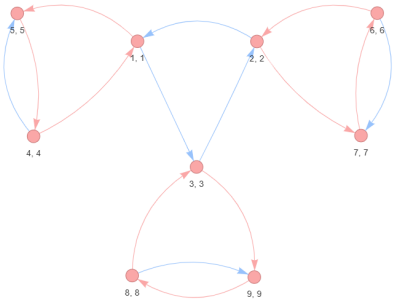

Figure 5.4: Weighted sum solution with total matching score 15.

Table 5.2, which has a total score of 15. This illustrates how the hierarchical model's preference for two-cycles can result in optimal solutions with a significantly lower overall score compared to alternative solutions. Consistently favouring solutions with fewer three-cycle exchanges may therefore risk foregoing transplants that would yield higher score matchings with better long-term outcomes. In such cases, our weighted sum model offers greater flexibility, as the adjustment of weights enables the generation of a range of solutions, therefore facilitating easy adaptation to evolving priorities over time.

Objective	**MaxTwoCycles**	**MaxSize**	**MinThreeCycles**	**MaxBackarcs**	**MaxOverallScore**
Weight	0.1	0.01	0.1	0.1	0.69

Table 5.2: Weights used to obtain solutions in Figures 5.3 and 5.4

The weights were chosen by initially setting all the other objectives to 0.1 except MaxOverallScore, which was set to 0.6. We then adjust the weight of MaxSize to account for the approximately 10 times larger magnitude of the MaxSize compared to MaxTwoCycles, MinThreeCycles, and MaxBackarcs.

5.2.3 Prioritising the number of transplants

Prioritisation for effective two-cycles in the hierarchical model can also result in a non-maximal number of transplants in the optimal solution. Countries like Spain and the Netherlands have chosen maximising the number of transplants to be their primary objective function, a strategy that NHSBT may consider adopting in future [17]. To further highlight the advantage of the weighted model's ability to adapt to changing priorities, we manually construct another pool graph instance, as shown in Figure 5.5 and 5.6, where the matching scores for all the edges are set to one. Figure 5.5 is the optimal solution returned by our hierarchical model, and results in fewer total transplants compared to Figure 5.6, which is the optimal solution identified by our weighted sum model. Table 5.3 lists the weights used to achieve the weighted sum solution. To decide these weights, we followed the same scaling factor approach as when we chose the weights in Table 5.2.

Objective	**MaxTwoCycles**	**MaxSize**	**MinThreeCycles**	**MaxBackarcs**	**MaxOverallScore**
Weight	0.1	0.699	0.1	0.1	0.001

Table 5.3: Weights used to obtain solutions in Figures 5.5 and 5.6

5.2.4 Trade-offs from prioritising

Equally, more transplants does not necessarily equate to a higher total score. This is evident in Table 5.4, which presents results from an experiment investigating the trade-offs of using a weighted sum model to prioritise a higher overall score on 300 and 600 patient datasets, compared to hierarchical solutions. The number of transplants is higher in both hierarchical solutions, yet the overall score is higher in the

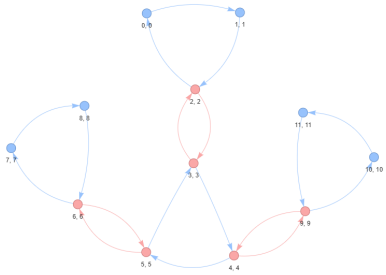

Figure 5.5: Hierarchical solution with 6 total transplants.

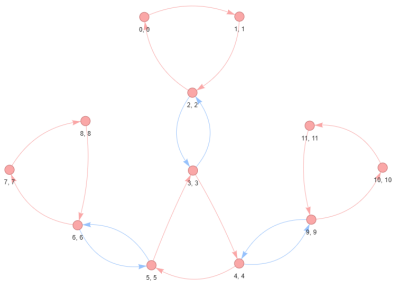

Figure 5.6: Weighted sum solution with 12 total transplants.

weighted sum solutions, showing that a smaller size solution can still lead to a greater total score.

The dataset size of 300 patient was chosen to reflect real-world scenarios, while the 600 patient dataset was intended to explore the potential impacts as the UKLKSS expands in the future. We applied the same weights in Table 5.2 for the weighted sum models here. The weighted sum returned a higher score than the hierarchical model for both the 300 and 600 patient datasets but at the expense of the number of total transplants, which is lower in the weighted sum results. Since we set the weight of the MaxSize objective to 0.01, this is an expected consequence of increasing the overall score.

Additionally, the weighted sum solution seems to generally result in fewer two-cycles because, unlike the hierarchical model, it does not continually enforce the maximization of two-cycles throughout the optimisation.

Total	**Original Pool (300)**	**Hierarchical (300)**	**Weighted Sum (300)**	**Original Pool (600)**	**Hierarchical (600)**	**Weighted Sum (600)**
Edges	11,241	-	-	43,430	-	-
Score	-	6,928	8,017	-	15,793	18,888
Transplants	-	153	145	-	357	348
Non-altruists	300	120	112	600	290	281
2-cycles	130	9	5	6,280	24	5
3-cycles	1,043	15	15	405	45	51
Altruists	33	30	33	67	66	63
2-chain	1,903	27	3	5,045	25	8
3-chain	25,251	3	27	122,330	41	55

Table 5.4: Results from running hierarchical model and weighted sum model, with weights from Table 5.2

We include in Table 5.4 the number of edges to further support our decision to use cycle formulation over edge formulation. For the 300 size dataset, the total number of possible cycles is 1,173, whilst the total number of edges is 11,241. Had we used edge formulation, we would have had to create binary variables for all the edges. Mixed integer programming solvers, like Guorbi, perform faster and better with fewer binary variables, hence opting for cycle formulation is preferred as there is a far smaller total number of cycles than edges.

Objective	**MaxTwoCycles**	**MaxSize**	**MinThreeCycles**	**MaxBackarcs**	**MaxOverallScore**
Weights 1	0.25	0.25	0.25	0.25	0
Weights 2	0	0.25	0.25	0.25	0.25
Weights 3	0	0.01	0.45	0.539	0.001
Weights 4	0.1999	0.6	0.1	0.1	0.01
Weights 5	0.01	0.49799	0.01	0.4	0.0001
Weights 6	0.01	0.49799	0.01	0.4	0.01

Table 5.5: List of different weight sets used for running the solutions in Table 5.6

Table 5.5 lists results for a sample of the experiments we ran on a 300 patient dataset using the weights in Table 5.6. Some weights, particularly in MaxSize, at first seen arbitrary, but the precision of the values is to ensure the sum of the weights does equate to 1. In Table 5.5, HSPs refers to the number of highly sensitised patients.

One key observation relates to our investigation on the impact of assigning a weights of value 0. We first set the MaxOverallScore weight to 0, as shown in the Weights 1 values in Table 5.5, which resulted in a score of 8,677. In contrast, the resulting score using Weights 3 was 6,335. This initially seems counter-intuitive, as one might expect the score to be minimised when the MaxOverallScore objective is no longer being considered. However, the MaxOverallScore objective can still be partially satisifed indirectly because the objectives are not completely separable. Actions that optimise one objective may simultaneously benefit others, meaning the objectives are highly interdependent.

Since the hierarchical model already maximises the number of two-cycles selected, it is reasonable for us to attempt to maximise the number of three-cycles using the weighted sum model. Some objectives have greater influence on the choice between three-cycles and two-cycles than others. Since we are hoping to increase the number of three cycles, not minimise them, we set MaxTwoCycles, and MinThreeCycles to lower relative weights. This is because MaxTwoCycles prioritises two-cycles over three-cycles with no backarcs. Using this rationale, we arrived at the values in Weights 5 and Weights 6, which increased the number of 3-cycles in the both the hierarchical solution, and the rest of the weighted sum solutions.

Total	**Score**	**Transplants**	**Non-altruist**	**2-cycles**	**3-cycles**	**2-chain**	**3-chain**	**HSPs**
Original Pool	-	-	300	154	1668	1361	16486	187
Hierarchical Solution	10,419	229	196	26	29	9	24	166
Weighted Sum (Weights 1)	8.677	224	191	10	35	0	33	162
Weighted Sum (Weights 2)	13,269	227	194	7	39	3	30	167
Weighted Sum (Weights 3)	6,335	149	116	1	16	0	33	91
Weighted Sum (Weights 4)	13,122	232	199	10	38	1	32	170
Weighted Sum (Weights 5)	9,841	228	195	0	43	0	33	165
Weighted Sum (Weights 6)	10,556	231	198	0	44	0	33	164

Table 5.6: Pool characteristics of the hierarchical solution, and weighted sum solutions obtained from running the weights listed in Table 5.5

Originally, we hoped to determine weights that would specifically increase the number of selected highly sensitised patients in weighted sum solutions. We successfully demonstrated that it is indeed possible to achieve solutions with more highly sensitised patients than with a hierarchical model, as evidenced by the results from Weights 2 and Weights 4 in Table 5.6. However, identifying weights that consistently increase the number of such patients is not feasible. This is because highly sensitised patients are not directly related to any objective. Although they are considered in the calculation of the matching scores, other factors also influence the matching score. As a result, assigning weights to the MaxOverallScore objective cannot act as a reliable lever to adjust the number of selected highly sensitised patients. The only objective that would have a predictable impact on the number of highly sensitised patients is MaxSize. This is trivially because a larger pool size would increase the probability of having more highly sensitised patients. If increasing highly sensitised patients were to become a priority, NHSBT would likely opt for adding a new criterion to the existing set instead.

One general observation from running multiple experiments was how the number of three-length altruist chains selected would increase proportionally with the weight of MaxBackarcs. Given the way we model our chains as pseudo-chains, this is expected as the pseudo-edges associated with the dummy patient are considered backarcs (refer back to Figure 5.2). Furthermore, we noticed that setting high weights for MaxOverallScore has a negligible impact due to the scaling effect, as its objective value is significantly larger than that of any other objectives.

As the weighted sum model has to contend with many competing trade-offs, the hierarchical model is comparatively simpler for NHSBT to implement and execute, as it requires less time spent determining which exact weights to use. Having a clear solution and objective in mind makes optimising and improving models much easier than when dealing with ambiguous goals.

5.2.5 Weighted sum as a hierarchical model

For maximum flexibility and adaptability, the weighted sum model should be able to reproduce the same solution as the hierarchical model. Here, we show that it is possible to use the weighted sum model to approximate hierarchical solutions, or obtain identical hierarchical solutions, by implicitly encoding priority levels into the weight set.

Total	Score	Transplants	Non-altruist	2-cycles	3-cycles	Altruists	2-chain	3-chain
Hierarchical Solution (300)	6,524	147	114	9	14	33	12	21

Table 5.7: Results from running hierarchical model on a 300 patient dataset

Objective	MaxTwoCycles	MaxSize	MinThreeCycles	MaxBackarcs	MaxOverallScore
Weight	0.399	0.3	0.2	0.1	0.01

Table 5.8: Weights used to obtain solution in Table 5.9

Total	Score	Transplants	Non-altruist	2-cycles	3-cycles	Altruists	2-chain	3-chain
Weighted Sum (300)	7,060	143	110	6	14	33	10	23

Table 5.9: Initial attempt using weights in Table 5.8 to reproduce the hierarchical solution in Table 5.7

After generating a new dataset of 300 patients, we apply the hierarchical model to obtain the solution presented in Table 5.7, which became our target output for the weighted sum model. Initially, we assigned weights that increased in order of the hierarchical optimisation order. These weights are listed in Table 5.8 and subsequently gave the results in Table 5.9.

From observing the solution in Table 5.9 and the final objective values, we readjusted the weights to focus on reducing the importance of the MaxOverallScore while increasing the priority assigned to MaxTwoCycles. This iterative process of evaluating the outcomes and adjusting the weights ultimately led to the final set of weights, as shown in Table 5.10, which produced the solution presented in Table 5.7.

Objective	MaxTwoCycles	MaxSize	MinThreeCycles	MaxBackarcs	MaxOverallScore
Weight	0.44	0.2599	0.15	0.1	0.001

Table 5.10: Weights used for weighted sum model to reproduce hierarchical solution in Table 5.7

However, it is likely that NHSBT would prefer a consistent set of weights to ensure an equivalent hierarchical solution, rather than fine-tuning the weight set each time a matching run is conducted. To evaluate the reliability of the weight set in Table 5.10, we apply them to a newly generated dataset of 300 patients, as well as to a new dataset of 600 patients. This generated the results in Table 5.11.

Although the resulting weighted sum solutions do not precisely match the hierarchical solution, the selections regarding the number of transplants, cycles, and chains are almost identical. The only significant variation is the difference in total scores, which seems to increase with the size of the dataset. Additionally, the weighted sum solution generally results in fewer two-cycles because, unlike the hierarchical model, it does not continually enforce the maximization of two-cycles throughout the optimisation.

Total	Score	Transplants	Non-altruists	2-cycles	3-cycles	Altruists	2-chain	3-chain
Hierarchical Solution (300)	7,125	165	132	16	16	33	10	21
Weighted Sum (300)	7,272	165	132	13	17	31	11	22
Hierarchical (600)	15,793	357	290	24	45	66	25	41
Weighted Sum (600)	16,562	358	291	21	46	67	23	44

Table 5.11: Solutions from running the weights listed in Table 5.10

Overall, the most weight sensitive objectives are MaxTwoCycles, MaxBackarcs, and MinThreeCycles. However, due to the interrelationships between these objectives, this is highly dependent on the weight set for MaxSize and MaxOverallScore, which would dominate the result if set at a relatively higher weight. This is a result of the scaling effect, and future work could explore additional normalisation methods to reduce this interdependence.

Chapter 6

Conclusion

As the demand for kidney donors continues to increase, so too does the need to improve the existing system for matching donors and patients in order to make the most of the available supply. Our work has implemented and explored an alternative model to the existing system, in order to address the open question of the benefits it may hold. After successfully implementing both the current hierarchical and new weighted sum models, we compared them based on potentially desirable future outcomes, such as further maximising the number of transplants. From our experiments, we demonstrated that the weighted sum approach is capable of producing new solutions that may prove more optimal as circumstances evolve, offering greater flexibility compared to the current hierarchical model. For example, the ability to increase the overall score of the final optimal exchange set.

However, this flexibility comes at a cost in the form of trade-offs and an unbounded set of potential solutions. Stringent NHS regulations would make it challenging for a weighted sum model to be adopted, as reaching agreement on the precise weights to assign each objective function could be difficult. There are no pre-determined ideal weights due to the subjectivity of what an optimal solution entails. Importantly, we do not claim that our proposed weighted sum model is concretely better than the existing hierarchical model, but do illustrate the advantages the weighted sum model can yield. The weighted model proves particularly valuable for exploring the full range of Pareto-optimal solutions and for deepening our understanding of how the objectives interact with one another. Subsequently, it could also be used as a useful tool to aid decision-making for choosing future priorities.

By solving the weighted sum model under a range of weight configurations, we were able to deeply examine the exact impacts and trade-offs resulting from prioritising different objectives. In addition, we showed that it is possible to assign weights in such a way that the weighted sum model reproduces the optimal solution obtained by the hierarchical model, therefore achieving all the objectives we initially stated.

Future work could involve further optimising the performance of the weighted sum model and experimenting with additional normalisation methods to support continued investigations into weight selection.

Bibliography

[1] Kidney Care UK, *Kidney transplant waiting list action required*, Jul. 2024. [Online]. Available: https://kidneycareuk.org/news-from-kidney-care-uk/kidney-transplant-waiting-list-action-required/ (cit. on p. 1).

[2] NHS Blood and Transplant, *Kidney*. [Online]. Available: https://www.nhsbt.nhs.uk/organ-transplantation/kidney/ (cit. on p. 1).

[3] NHS Organ Donation, *Statistics about organ donation*. [Online]. Available: https://www.organdonation.nhs.uk/helping-you-to-decide/about-organ-donation/statistics-about-organ-donation/ (cit. on p. 1).

[4] NHS Organ Donation, *500 people have now donated a kidney to a stranger*. [Online]. Available: https://www.organdonation.nhs.uk/get-involved/news/500-people-have-now-donated-a-kidney-to-a-stranger/ (cit. on p. 1).

[5] C. Dyer, "Paired kidney transplants to start in the United Kingdom," *BMJ : British Medical Journal*, vol. 332, no. 7548, p. 989, Apr. 2006, ISSN: 0959-8138. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1450034/ (cit. on p. 1).

[6] BBC News, "Strangers allowed to give organs," Apr. 2006. [Online]. Available: http://news.bbc.co.uk/1/hi/health/4942732.stm (cit. on p. 2).

[7] NHS Blood and Transplant, "UK Living Kidney Sharing Schemes: Your questions answered," Tech. Rep. [Online]. Available: https://nhsbtdbe.blob.core.windows.net/umbraco-assets-corp/12181/29190_033mv-living-kidney-sharing_print_olc2172.pdf (cit. on p. 2).

[8] The Renal Association, British Transplantation Society, "Guidelines for Living Donor Kidney Transplantation," Tech. Rep. Fourth Edition. [Online]. Available: https://bts.org.uk/wp-content/uploads/2018/07/FINAL_LDKT-guidelines_June-2018.pdf (cit. on p. 2).

[9] D. F. Manlove and G. O'malley, "Paired and Altruistic Kidney Donation in the UK: Algorithms and Experimentation," *ACM J. Exp. Algorithmics*, vol. 19, 2.6:1–2.6:21, Jan. 2015, ISSN: 1084-6654. DOI: 10.1145/2670129. [Online]. Available: https://dl.acm.org/doi/10.1145/2670129 (cit. on pp. 2, 3, 8, 14, 19).

[10] NHS Organ Donation, *UK's living kidney sharing scheme hits 1000th transplant milestone*, Mar. 2019. [Online]. Available: https://www.organdonation.nhs.uk/get-involved/news/uk-s-living-kidney-sharing-scheme-hits-1000th-transplant-milestone/ (cit. on p. 2).

[11] D. Manlove, "How Operational Research Helps Kidney Patients in the UK," *Impact*, vol. 2018, no. 1, pp. 16–19, Jan. 2018, Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/2058802X.2018.1435455, ISSN: 2058-802X. DOI: 10.1080/2058802X.2018.1435455. [Online]. Available: https://doi.org/10.1080/2058802X.2018.1435455 (cit. on pp. 3, 4).

[12] Kalyanmoy Deb, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. 2014, ISBN: 978-1-4614-6939-1 978-1-4614-6940-7. DOI: 10.1007/978-1-4614-6940-7. [Online]. Available: https://link.springer.com/10.1007/978-1-4614-6940-7 (cit. on pp. 4, 19, 20).

[13] A. E. Roth, T. Sönmez, and M. U. Ünver, "Efficient Kidney Exchange: Coincidence of Wants in Markets with Compatibility-Based Preferences," *American Economic Review*, vol. 97, no. 3, pp. 828–851, Jun. 2007, ISSN: 0002-8282. DOI: 10.1257/aer.97.3.828. [Online]. Available: https://www.aeaweb.org/articles?id=10.1257/aer.97.3.828 (cit. on p. 4).

[14] D. J. Abraham, A. Blum, and T. Sandholm, "Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges," in *Proceedings of the 8th ACM conference on Electronic commerce*, San Diego California USA: ACM, Jun. 2007, pp. 295–304, ISBN: 978-1-59593-653-0. DOI: 10.1145/1250910.1250954. [Online]. Available: https://dl.acm.org/doi/10.1145/1250910.1250954 (cit. on p. 5).

[15] D. B. Johnson, "Finding All the Elementary Circuits of a Directed Graph," *SIAM Journal on Computing*, vol. 4, no. 1, pp. 77–84, Mar. 1975, ISSN: 0097-5397, 1095-7111. DOI: 10.1137/0204007. [Online]. Available: http://epubs.siam.org/doi/10.1137/0204007 (cit. on p. 5).

[16] A. Gupta and T. Suzumura, *Finding All Bounded-Length Simple Cycles in a Directed Graph*, arXiv:2105.10094 [cs], May 2021. DOI: 10.48550/arXiv.2105.10094. [Online]. Available: http://arxiv.org/abs/2105.10094 (cit. on pp. 5, 13).

[17] M. Delorme, S. García, J. Gondzio, J. Kalcsics, D. Manlove, and W. Pettersson, "New Algorithms for Hierarchical Optimization in Kidney Exchange Programs," *Operations Research*, vol. 72, no. 4, pp. 1654–1673, Jan. 2023, Publisher: INFORMS, ISSN: 0030-364X. DOI: 10.1287/opre.2022.2374. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/opre.2022.2374 (cit. on pp. 5, 8, 9, 21).

[18] J. P. Dickerson, D. F. Manlove, B. Plaut, T. Sandholm, and J. Trimble, *Position-Indexed Formulations for Kidney Exchange*, arXiv:1606.01623 [cs], Jun. 2016. DOI: 10.48550/arXiv.1606.01623. [Online]. Available: http://arxiv.org/abs/1606.01623 (cit. on p. 5).

[19] NHS Blood and Transplant, "POL274/11 – Living Donor Kidney Transplantation," Tech. Rep., Oct. 2024. [Online]. Available: https://nhsbtdbe.blob.core.windows.net/umbraco-assets-corp/34788/pol274.pdf (cit. on pp. 6, 7).

[20] *Kidney Transplant for Highly Sensitized Patients - UChicago Medicine*. [Online]. Available: https://www.uchicagomedicine.org/conditions-services/transplant/kidney-transplant/kidney-transplant-for-highly-sensitized-patients (cit. on p. 8).

[21] "In brief: The innate and adaptive immune systems," in *InformedHealth.org [Internet]*, Institute for Quality and Efficiency in Health Care (IQWiG), Aug. 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK279396/ (cit. on p. 8).

[22] NHS Blood and Transplant, "POL186/11 – Kidney Transplantation: Deceased Donor Organ Allocation," Tech. Rep., Feb. 2021. [Online]. Available: https://nhsbtdbe.blob.core.windows.net/umbraco-assets-corp/22127/pol186.pdf (cit. on p. 8).

[23] Matt Ronin, *What is cPRA and What Role Does It Play in Kidney Transplants?* [Online]. Available: https://www.immunofree.com/why-immunofree/blog/what-is-cpra-and-what-role-does-it-play-in-kidney-transplants/ (cit. on p. 8).

[24] M. Delorme, S. García, J. Gondzio, J. Kalcsics, D. Manlove, W. Pettersson, and J. Trimble, "Improved instance generation for kidney exchange programmes," *Computers & Operations Research*, vol. 141, p. 105 707, May 2022, ISSN: 0305-0548. DOI: 10.1016/j.cor.2022.105707. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054822000107 (cit. on pp. 9, 10, 19).

[25] S. L. Saidman, A. E. Roth, T. Sönmez, M. U. Unver, and F. L. Delmonico, "Increasing the opportunity of live kidney donation by matching for two- and three-way exchanges," *Transplantation*, vol. 81, no. 5, pp. 773–782, Mar. 2006, ISSN: 0041-1337. DOI: 10.1097/01.tp.0000195775.77081.25 (cit. on p. 9).

[26] BJGP Life, *What GPs need to know about non-directed altruistic kidney donation – BJGP Life*, Aug. 2022. [Online]. Available: https://bjgplife.com/what-gps-need-to-know-about-non-directed-altruistic-kidney-donation/ (cit. on p. 9).

[27] William Pettersson and James Trimble, *Kidney.optimalmatching.com/api/input_format*. [Online]. Available: https://kidney.optimalmatching.com/api/input_format (cit. on p. 11).

[28] *(PDF) Improving Transplant Opportunities for Patients who are Sensitized (ITOPS): Protocol for a Feasibility, Randomized, Controlled, Phase III Clinical Trial*. [Online]. Available: https://www.researchgate.net/publication/347609198_Improving_Transplant_Opportunities_for_Patients_who_are_Sensitized_ITOPS_Protocol_for_a_Feasibility_Randomized_Controlled_Phase_III_Clinical_Trial (cit. on p. 11).

[29] *Branching Factor - an overview — ScienceDirect Topics.* [Online]. Available: https://www.sciencedirect.com/topics/computer-science/branching-factor (cit. on p. 12).

[30] Lamb Chen, *Lamb-chen/kidney-exchange-solver: A kidney exchange program solver! University of Bristol dissertation.* [Online]. Available: https://github.com/lamb-chen/kidney-exchange-solver (cit. on p. 17).

Appendix A

Appendix A: AI Prompts/Tools

I used Perplexity AI https://www.perplexity.ai/ for sourcing NHS statistics that were difficult to manually find. An example prompt would be: "How many annual altruistic donors are there in the UK's Living Kidney Sharing Scheme?".

I used ChatGPT to aid my theoretical understanding of the way Gupta and Suzumura's algorithm for Finding All Bounded-Length Simple Cycles in a Directed Graph by running through examples of how the algorithm is expected to work. An example prompt would be: "Please provide a specific example running through the Gupta and Suzumura's algorithm for Finding All Bounded-Length Simple Cycles in a Directed Graph to help me understand the locking mechanism."