

A Policy Agnostic Framework for Post Hoc Analysis of Organ Allocation Policies



Agathe de Vulpian

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy
in Machine Learning and Machine Intelligence

Declaration

I, Agathe de Vulpian of St Edmund's College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

The code used in this thesis does not rely on any previously written software other than the standard Python packages.

This dissertation contains 14,958 words excluding bibliography, photographs and diagrams but including tables, figure captions, footnotes and appendices.

Agathe de Vulpian
August 2021

Acknowledgements

Foremost, I would like to thank my supervisors Prof. Mihaela van der Schaar and Jeroen Berrevoets for their guidance and support throughout this project.

I am beyond grateful for my family and friends for their constant encouragement throughout this dissertation and my year at Cambridge. A particular thank you to Jules Baudet for sharing his ideas and being so generous with his time and to Loic Lannelongue for his valuable support in the last few weeks.

I would also like to thank Dr Brent Ershoff, for sharing his expertise on the topic of organ transplantation. Particularly, how the US transplant system works and how the UNOS dataset is organised.

Abstract

The amount of patients that need a donor organ far surpasses the number of donors. This shortage results in many patients dying or being removed from the transplant waitlist before ever receiving a transplant, leading us to question whether allocation policies are assigning organs in an equitable, effective, and efficient manner. Using historical data, researchers simulate how a novel allocation policy would offer organs across the patient-population before it is actually implemented in practice. However, to our knowledge, no framework exists to provide a post hoc analysis of a policy after it has been implemented. Post hoc analysis is necessary because the implemented policy may differ wildly from its original proposal. Such variation can be the result of (among others) clinical preference across transplant centers, patients rejecting organ offers, or changing clinical practice. Further complicating matters are conflicting definitions of optimal allocation. For example, the UK takes an altruistic perspective on allocation by maximising a population-wide survival, whereas the US and central EU take an individualistic perspective by selecting the patient that is most in need. To account for all these sources of variation, an accurate system for post hoc analysis should learn and model the implemented policy from data. In addition, to support clinicians in assessing any past or future implemented policy in any jurisdiction a policy agnostic framework is required.

We propose a policy agnostic framework based on priority queues, able to learn any implemented policy from data to effectively model patient priority, health deterioration, and variations in allocation practices. Our framework is tested using the United States' liver transplantation waitlist data, provided by the United Network for Organ Sharing (UNOS). With a case study on the MELD-Na policy for liver transplantation, we show our framework can accurately model an implemented policy, with which we can conduct meaningful analyses. Specifically, we confirm that MELD-Na places emphasis on patients with hyponatremia when allocating organs. Moreover, health professionals have clinically verified the applicability and correctness of the results of our framework. The work in this dissertation is the first step towards analysing any implemented policy. It can be used to verify whether policies meet their intended effect when implemented, or to conduct further analyses to refine current policies in the aim of guiding strategic policy improvements.

Table of contents

1	Introduction	1
1.1	Core Challenges	3
1.2	Contributions	3
1.3	Outline of the Dissertation	5
2	Background	7
2.1	Organ Allocation Policies	7
2.2	Learning to Rank (LTR)	9
2.3	Clustering	12
2.4	Queuing Systems For Organ Allocation	14
3	Problem Formulation	17
4	Organ Queue	19
4.1	Organ Types	20
4.2	The Timing Model	21
4.3	The Ranking Model	23
5	Experiments on Real Data	25
5.1	The Data	25
5.2	The MELD-Na Allocation Policy	26
5.3	Evaluation of Organ Queue	27
5.3.1	Training and Testing Data	27
5.3.2	Experimental Method	28
5.3.3	Evaluation Metrics	29
5.3.4	Baselines	29
5.4	Results	30
5.4.1	Main Results	30
5.4.2	Additional Analyses	31

5.5	Clinical Application	33
6	Further Analysis: How it Works	35
6.1	Evaluation of the Timing Model	35
6.1.1	Evaluation Metric	35
6.1.2	Baseline	36
6.1.3	Results	36
6.1.4	Clinical Interpretation of Results	38
6.2	Evaluation of the Ranking Model	39
6.2.1	Evaluation Metrics	39
6.2.2	Baseline	41
6.2.3	Training and Testing Data	41
6.2.4	Results	42
6.2.5	Clinical Interpretation of Results	43
7	Conclusion & Future Work	45
7.1	Conclusion	45
7.2	Future Work	46
	References	49
	Appendix A Supplementary Material	55
A.1	Original Timing Model	55
A.2	Additional Results	57
A.3	Evaluation Metrics	58

Chapter 1

Introduction

Organ transplantation is the only effective treatment available for irreversible organ failure. However, due to the limited availability of donors, over 100,000 Americans remain on the transplant waitlist, with patients often having to wait months before an organ becomes available (Colvin et al., 2021). Despite the increase in the number of donors, an average of 17 people die each day in the U.S. while waiting for a transplant (OPTN, 2021) with many more removed from the transplant list without having received an organ because they have become too sick for transplant (Kwong et al., 2021). In studying such tragic outcomes, appalling conclusions can be made. Notably, the distributions of organs amongst different patient groups such as blood type groups, patients listed for repeat transplantation, patients with rare anti genes, or patients with particular medical conditions is unequal (Sanfilippo et al., 1992) causing discrepancies in waiting times, and ultimately, in survival. Faced with these discrepancies, new allocation policies are being researched and developed with the aim of reconciling disparities while still improving outcomes. For instance, the introduction of the MELD-Na (Kim et al., 2008) policy which replaced the Model for End-Stage Liver Disease (MELD) policy (Malinchoc et al., 2000) in the U.S. was motivated by the belief that MELD-Na could prevent 7% of the deaths that occurred under the MELD policy by better allocating organs to patients with hyponatremia whose health condition was more severe. Hyponatremia means the sodium level which your body needs for fluid balance, blood pressure control, as well as for the nerves and muscles is below normal. However, this assertion, much like all other statements which convince clinicians for the implementation of a new policy, is verified using historical data in a simulation setting, and may thus not account for variations that happen in a real-life setting once the policy is implemented. With the aim of providing the fairest and most effective allocation policy, it is essential to provide a framework to evaluate such assertions after the implementation of new policies.

In theory, a transplant system entails three components that interact and evolve in real-time: the patients joining and leaving the waitlist, the organs arriving for transplant, and an allocation policy that dictates to which patient an organ should be offered. This policy evolves over time to incorporate novel medical knowledge, it differs in each country and may have different objectives. Some policies aim at offering the organ to the sickest patient first (Kim et al. (2008), Malinchoc et al. (2000)), while others aim at maximising post-transplant survival (Neuberger et al., 2008). In recent years, this complex problem has captured the interest of machine learning experts, who have proposed new allocation policies through data-driven approaches to learning the compatibility between donors and recipients (Berrevoets et al. (2021), Yoon et al. (2017), Xu et al. (2021)). These approaches, for example, do not only consider potential outcomes, but also organ scarcity (Berrevoets et al., 2020).

In practice, however, many other factors influence the transplant system, and consequently which patient receives an organ. There are two main variations which happen before and after offers are made. The first, pre-offer is due to the specification of donor criteria. For each patient, their clinician specifies criteria the donor must meet (e.g. age, weight, blood type, distance), if an incoming organ does not meet these criteria, the patient will not receive an offer for it even if following the policy the patient was eligible to receive an offer. The post-offer variations encompass the clinician's decision to accept/reject and offer. A clinician may choose to reject an organ offer for medical reasons (e.g. the patient is too ill, multiple organ transplants are required) or logistics reasons (e.g the operating room or the surgeon is unavailable) in which case the offer is made to another patient. An organ is offered to patients sequentially until one of them accepts. This ordered list of patients constitutes the match run which is essentially the priority ordering of patients in the waitlist for a given organ under a given allocation policy influence by real-life variations.

Due to these variations, the match run given by the allocation policy in a simulation setting called the **theoretical policy** and the match run given by the allocation policy in a real-life setting, called **implemented policy** may differ. Then in complement of the analyses conducted on historical and simulated data when proposing a policy, it is necessary to perform post hoc analyses of the implemented policy to evaluate if it is performing as it was intended. To achieve this the implemented policy must be learned from data. Previous works have focused on developing dynamic systems to provide the information needed before a policy change is put into effect (Thompson et al., 2004) however, to the best of our knowledge no model exists to achieve post hoc analysis.

The goal of this dissertation is thus to develop a framework in which it will be possible to learn and model any *implemented* policy with the logic of being able to assess certain aspects of its performance. As the allocation policies used are constantly evolving, to provide support in the long run, it is necessary to develop a framework that will be policy agnostic, meaning it will be suited to evaluate the post hoc performance of any theoretical allocation policy for which we have data.

1.1 Core Challenges

The core challenges of modelling the implemented policy of any theoretical allocation policy are (a) Organs and patients are unique and high dimensional with a mix of continuous and categorical features making organ-patient interaction highly complex (b) The match runs produced by the implemented policy are affected by pre-offer variations which influence which patients are included in the match run and by post-offer variations which affect the length of the match run. (c) The match run provided in the dataset is incomplete, as it is truncated as soon as an offer is accepted and thus we never have access to a full ranking of all the patients on the waitlist (d) All offers received by a patient are dependent on all other patients on the waitlist, making the covariate space when deciding which patient should get the offer even larger. (e) Different policies may function differently. For instance, the current theoretical policy in the US (Kim et al., 2008) uses only patient covariates to allocate organs while the current policy in the UK (Neuberger et al., 2008) uses both patient and organ covariates. We need a framework that can represent the corresponding implemented policy for both cases. (f) Patients' health may deteriorate (or improve), causing their priority with respect to the policy in place to evolve.

1.2 Contributions

In this dissertation, we do not attempt to create a new allocation policy as has been done in many recent publications. Instead, we focus on creating a framework to learn and model the real-life implementation of an allocation policy, from data collected under the policy of interest.

We aim to model the real-life transplant system in a policy agnostic way by incorporating aspects of operations research and machine learning to “translate” an organ transplant system into a queuing system in which the parameters are learned from data. We use queuing theory as it provides a stable setting in which it is possible to assess the different performance

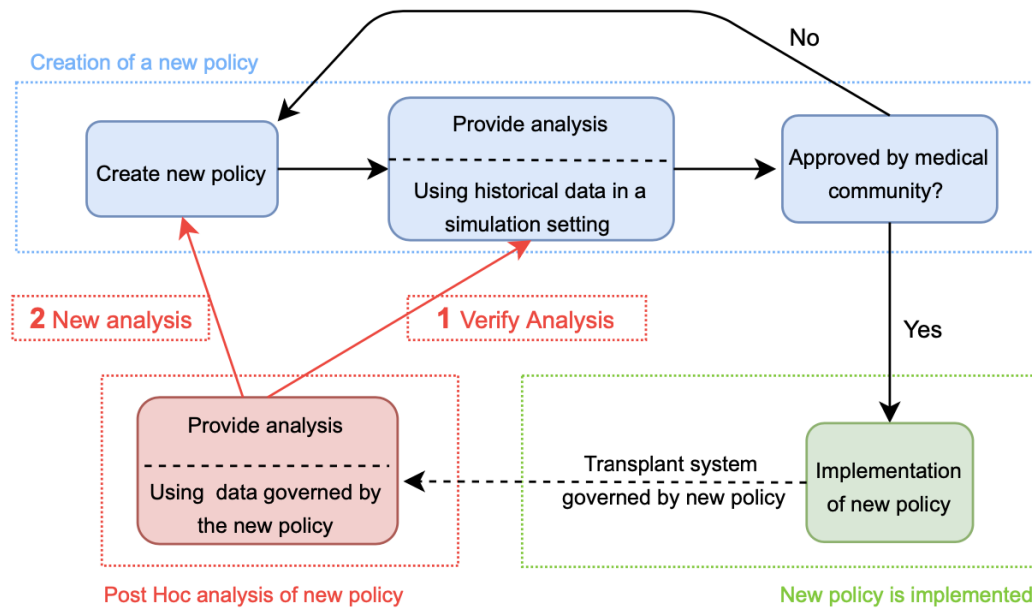


Fig. 1.1 **How we envisage the usage of our framework in medical practice.** When a new policy is created, it undergoes analysis using historical data to seek approval (blue). The policies which are approved are then implemented (green). Using our framework (red) it will be possible to **1** verify that the intent of the new policy is met, and **2** conduct further analyses to refine the current policy and guide strategic policy improvements.

aspects of the implemented policy. With the logic that such analyses could answer the ongoing controversy surrounding the observed discrepancies between different groups of patients (Barone et al., 2008; Sanfilippo et al., 1992).

In addition to confirming analyses made with the theoretical policy, such a framework also provides avenues to conduct new analyses to guide strategic policy improvements (Figure 1.1). This may be necessary as the period between the development and the implementation of a new policy is so lengthy due to regulations, the features recorded for each patient and the profiles of the patients may have evolved. For instance, MELD-Na was published in 2008 but was only implemented in 2016. In this time, the Patient Protection and Affordable Care Act (ACA) extended Medicaid eligibility and access to insurance coverage to millions of Americans, notably low-income adults (Goyal and Weiner, 2018), which caused an increase in the proportion of patients listed with Medicaid across all organs (Trivedi et al., 2018).

Lastly, this framework may also have an impact on organ donation. Indeed, any interventions enhancing transparency and perceived fairness of organ allocation may improve willingness to donate (Boulware et al., 2007) as in the United States, 71% of the population believe the allocation system is “unfair”, causing them to be less willing to donate. From the perspective of the patient and clinician having a better overview of the transplant system through these queues could aid in making important medical decisions such as how to proceed when an organ offer is received.

In the course of this work we make several contributions which we highlight below:

Reviewing the literature on queueing in organ allocation. In particular, we highlight where existing queueing systems fall short when modelling realistic allocation policies.

Our framework, Organ Queue. We introduce Organ Queue, a novel method for modelling the implemented allocation policy aimed at fitting the high dimensional organ to recipient matching problem into a queueing system.

A case study on MELD-Na. We demonstrate Organ Queue’s potential by using our framework to model real data on liver transplantation under the MELD-Na policy for organ allocation.

1.3 Outline of the Dissertation

In the next chapter, we present useful background on the topic of organ transplantation and the machine learning methods we use. We review previous attempts of modelling the transplant waiting list with a queueing system. Chapter 3 describes the problem formalism. In the subsequent chapter, we motivate our choice of organ types for the queueing model and describe the two main components of our framework. Chapter 5 presents the experimental methods and results of our framework on a case study of real liver transplantation. Chapter 6 dives into the performance and medical interpretation of both components of our framework separately. The dissertation ends with a summary and reflections on future works.

Our study primarily focuses on liver transplants in the United States as we use the American transplant dataset: the United Network for Organ Sharing (UNOS) data. (Cecka, 2000). Throughout this dissertation we had the privilege of working closely with doctors and researchers in Los Angeles notably affiliated with UCLA, they provided valuable insights both on the dataset and challenges currently faced by the organ transplant system. All clinical findings and assertions in this dissertation have been discussed and verified by them.

Chapter 2

Background

In this chapter, we give the necessary background for a strong understanding of our methods. We first review organ allocation policies and why so many different ones have been developed. We then present the learning to rank (LTR) task and the most popular approaches to solving it. LTR will be a core part of this dissertation as our framework ranks patients for organs. We then review clustering methods as our framework makes use of clustering to discretise the organ space and create queues. Lastly, we review previous attempts of modeling the transplant system with a queuing system and show in what ways our proposed method is unique.

2.1 Organ Allocation Policies

In the United States, new waiting list registrations continue to increase for heart ([Colvin et al., 2021](#)), liver ([Kwong et al., 2021](#)), kidney ([Hart et al., 2021](#)) and pancreas ([Kandaswamy et al., 2021](#)) transplants. However, despite the increase in the number of organ donations, which reached an all-time high in 2019 for livers and kidneys, there is a severe shortage of available donors organs across all organs and all jurisdictions. This shortage caused 2,406 patients to be removed from the liver transplant waitlist in 2019 in the US without receiving an organ due to death or being too sick for transplant ([Kwong et al., 2021](#)). As organs are a scarce and perishable resource, allocating them is a complex challenge that impacts the lives of thousands of patients because the award of an organ to one candidate denies it to others. New knowledge on patient to organ compatibility, and the emergence of data-driven studies have spurred the development of many new allocation policies which all have one common goal: to better allocate organs to improve the outcome of transplanted patients and reduce the number of patients removed from the waitlist without an organ.

A myriad of policies for the allocation of deceased donor organs has been proposed, in an attempt to incorporate new medical knowledge and provide equal consideration for both the needs of the sickest patients and the efficient use of organs. For instance, in the case of livers transplantation, the United States has adopted a sickest first policy with priority primarily based on the Model for End-Stage Liver Disease (MELD) (Malinchoc et al., 2000) and MELD-Na (Kim et al., 2008). The MELD-Na score, which is a function of four routinely measured laboratory values, has been shown to be predictive of waitlist mortality. The policy allows for exception points whereby patients without a high laboratory MELD score can achieve higher priority and allows patients to get highest priority if they have certain conditions such as acute liver failure. This policy is different from the current policy in the U.K. that has adopted the Transplant Benefit Score (TBS) (Neuberger et al., 2008), which offers livers nationally to patients predicted to gain the most survival benefit from receiving the particular liver.

The goal of the system we propose is to be able to model any allocation policy. A simple policy, for example, would consider each patient independently of all others when placing them into a queue. Thus, the placement on the queue would only depend on the patient's compatibility with each organ type and the patient's priority for each of those types. Here, the patient would receive the first compatible organ for which they had the highest priority.

However, this is not how more sophisticated policies work, in Table 2.1 we provide a simple illustrative example. Assume there exist two organ types A and B, which have different arrival rates and added life years for patients. When patient X arrives on the waitlist, a policy that maximises the added life years for the entire population would not assign an incoming organ of type B to patient X as long as patient X can afford to continue to wait for an organ of type A. As such, X should not be placed on the queue for organs of type B when they arrive in the system, as they should only be placed on queue B when their survival time is lower than the expected arrival time for organs of type A. To model policies that maximise the benefit for the entire population, our system must be able to model such behavior.

Table 2.1 **Example of an allocation policy.** Illustrative example presenting 2 organ types (A, B), the added life years if transplanted into patient X (ALY-X) or into any other patients (ALY-other) and their arrival rates

Organ Type	ALY-X	ALY-other	Arrival Rate
A	50	0.25	every 2 months
B	5	5	every week

2.2 Learning to Rank (LTR)

Machine-Learned Ranking (MLR) or Learning to Rank (LTR) is a class of algorithmic techniques that apply supervised machine learning to solve ranking problems. Training data consists of lists of items with some partial order specified between items in each list. This order is typically induced by giving a numerical or ordinal score or a binary judgment (e.g. "relevant" or "not relevant") for each item. The most common application of LTR is in information retrieval (IR) systems for instance for document retrieval. For the document retrieval task, the system maintains a collection of documents d_1, d_2, d_3, \dots , then, given a query q , the system ranks the documents from the collection and returns the top ranked documents. The ranking task is performed by using a ranking model to sort the documents. The ranking model can be expressed as learning the function \mathcal{R} such that:

$$\mathcal{R}(\{d_i, d_k \dots d_j\}, q) \longrightarrow [d_j \triangleright d_k \triangleright \dots d_i]_q \quad (2.1)$$

Where, the notation $[d_i \triangleright d_j]_q$ indicates d_i is ranked higher than d_j for query q . In the setting of this dissertation the documents will be patients and the query organs, the "patient ranking system" is illustrated in Figure 2.1.

The ranking algorithms are evaluated using information retrieval measures, such as the Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen, 2017) which takes into account element relevance and positional information and Mean Average Precision (MAP) (Yue et al., 2007) used to evaluate a ranking when there are only two levels of relevance: relevant and irrelevant. Both metrics are detailed and used in later chapters.

Liu (2011) categorised the existing LTR algorithms into three groups which differ by the number of documents that are combined in the cost during training: the pointwise, the pairwise and the listwise approaches.

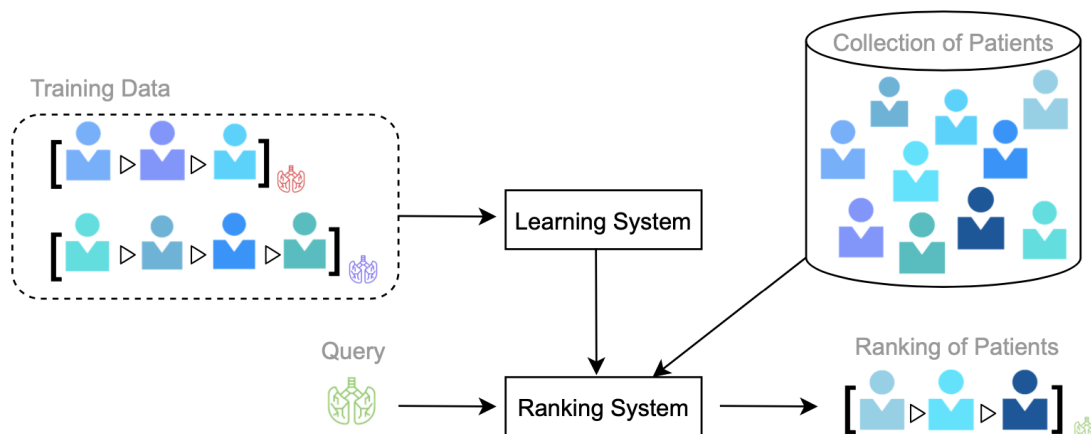


Fig. 2.1 **Learn to rank for patient retrieval**, in the setting of this dissertation, the queries are organs and the documents are patients awaiting transplant. The notation $[A \triangleright B]_q$ indicates patient A is ranked higher than patient B for query/organ q .

The **pointwise** approach looks at a single document at a time. It assumes each query-document pair in the training data has a numerical or ordinal score, in this case the learning-to-rank problem can be approximated by a regression problem which aims at learning a function predicting the score of the document to discover the best ranking for individual results. These algorithms minimize a loss function over the training set which assigns a loss to each individual feature vector. Examples include SubSet Rank (Cossock and Zhang, 2006), McRank (Li et al., 2007) and PRank (Crammer et al., 2001) These approaches model ranks as intervals on the real line and as a result the final ranking is highly dependant on the choice of the rank boundaries.

The **pairwise** approach looks at two documents at once and assumes that each query-document pair in the training data has a numerical or ordinal score. In this case, the learning-to-rank problem is approximated by a binary classification problem that predicts which document is better in a given pair. Then the loss function is calculated over every pair of feature vectors pertaining to the same query. The founding methods of this approach are RankSVM (Herbrich et al., 2000) which shows that the ranking problem can be solved with Linear SVM classification, RankBoost (Freund et al., 2003) which iteratively creates and aggregates a collection of “weak rankers” to build an effective ranking procedure and RankNet (Burges et al., 2005) which puts forward a natural probabilistic cost function on pairs of examples. These three methods have been the basis for many other approaches

Finally, the **listwise** approach decides on the optimal ordering using the entire list of documents. The group structure of ranking is maintained and ranking evaluation measures can be more directly incorporated into the loss functions in learning. Indeed, in this case the loss function treats each query (and its associated documents) as a single instance and accordingly computes a single loss for all the documents pertaining to the same query. There are two main sub-techniques for listwise LTR: The first, directly optimises the information retrieval measures such as the NDCG or the MAP, this is the case of SoftRank (Taylor et al., 2008) and AdaRank (Qin et al., 2010). The second defines a list wise loss function as an indirect way to optimise the IR evaluation metric, examples include ListNet (Cao et al., 2007) which adopts the KL divergence for loss function by defining a probabilistic distribution in the space of permutation for learning to rank and ListMLE (Lan et al., 2014) which employs the likelihood loss as the surrogate for the IR evaluation metrics.

We choose to focus on pairwise methods. From the point of view of complexity, the pointwise methods seems to be preferable over pair-based methods since the latter must learn using $O(m^2)$ pairs rather than m examples. However, pairwise approaches work better in practice than pointwise approaches because predicting relative order is closer to the nature of ranking than predicting class label or relevance score. From the point of view of accuracy, listwise algorithms have been shown empirically (Li, 2011) to outperform pointwise algorithms. Some publications (most prominently Liu (2011)) suggest that listwise approaches are fundamentally superior to pairwise ones. However, Köppel et al. (2019) shows, this is not necessarily always the case. Moreover, listwise algorithms are usually computationally more demanding and lack conceptual simplicity. Also, they are comparatively more prone to overfit as conjectured by Tan et al. (2013). Then pointwise methods seem to be the best compromise between complexity and accuracy.

We work with pairwise methods, the most popular of which are those developed by Christopher J.C. Burges and his colleagues at Microsoft, RankNet (Burges et al., 2005), LambdaRank (Burges et al., 2006), LambdaMART (Wu et al., 2010) which have proven to be very successful algorithms for solving real world ranking problems. LambdaMART is the boosted tree version of LambdaRank, which is based on RankNet.

RankNet employs a probabilistic cost function which uses a pair of sample items to learn how to rank them. This function essentially tries to minimize the number of swaps required to correct an incorrect ordering of chosen items. The paper further suggest to explore this approach by implementing this cost function through a neural network, due to their flexibility

and their rapidity in test phase compared to competing kernel methods, the network is then optimized through gradient descent. Burgess et. al. then found that during RankNet training procedure, the costs are not required, only the gradients (λ) of the cost with respect to the model score are. They discovered that scaling the gradients by the change in NDCG found through swapping each pair of documents gave good results. The core idea of LambdaRank is thus to use this new cost function for training a RankNet. Going further, LambdaMART combines the idea of LambdaRank and Multiple Additive Regression Trees MART (Friedman, 2001) by treating the ranking problem as a multiclass classification problem, with one class for each of the levels of relevance identified in the training set. LambdaMART uses gradient boosted decision trees using a cost function derived from LambdaRank for solving a ranking task.

LambdaRank and LambdaMART are computationally more demanding to train compared to the pairwise optimization of RankNet. Their key advantage over RankNet is the increase in training speed. Then since our problem does not deal with huge amounts of data we choose to favour simplicity and decide to employ the RankNet method.

For the problem of ranking patients for given organs, using a ranking algorithm is not enough due to the high dimensionality of the organ space. To address this challenge we employ clustering.

2.3 Clustering

Clustering is a natural approach for identifying phenotypic characterizations by grouping “similar” instances into distinct clusters. In this dissertation we cluster organs to create organ types. Clustering algorithms can be divided into two categories: unsupervised and supervised clustering. Unsupervised clustering algorithms, utilize the feature space solely to learn the partitions that maximize some given objective. Supervised clustering algorithms utilize both the feature space and the label space for constructing clusters (Eick et al. (2004), Finley and Joachims (2005)). In the setting of this dissertation we wish to cluster organs, however there is no label associated to this task thus we focus on unsupervised clustering methods.

Amongst unsupervised clustering algorithms two categories arise: the hierarchical and non-hierarchical methods. Hierarchical clustering methods (Johnson, 1967) produce a classification in which small clusters are nested within larger clusters. Kaufman and Rousseeuw (2009) further divides hierarchical clustering methods in two subgroups. Hierarchical *ag-*

glomerative methods also known as AGNES (Agglomerative Nesting) which generate a classification in a bottom-up manner, by a series of agglomerations in which small clusters, initially containing individual elements, are fused together to form progressively larger clusters. And *divisive* hierarchical methods, also known as DIANA (Divisive ANALysis Clustering Algorithm) methods which generate a classification in a top-down manner, by progressively sub-dividing the single cluster which represents an entire dataset.

Non-hierarchical clustering methods (or partitioning clustering), on the other hand, generate a classification by partitioning a dataset into a set of (generally) non-overlapping groups. These methods are generally much less demanding of computational resources than the hierarchical methods. There are three main categories of such methods. Single-pass methods which produce clusters that are dependent upon the order in which the compounds are processed. An example is the Leader method (Hartigan, 1975) which requires the user to specify the approximate radius of a cluster. Relocation methods, such as k-means (Lloyd, 1982) and expectation-maximisation (EM) algorithm (Dempster et al., 1977), which assign compounds to a user-defined number of seed clusters and then iteratively reassign compounds to see if better clusters result. k-means uses centroids whereas EM uses statistical distributions. And lastly, nearest neighbour methods, such as the Density-based spatial clustering of applications with noise algorithm (DBSCAN) (Ester et al., 1996), which assign compounds to the same cluster as some number of their nearest neighbours.

In view of complexity and practicality, we choose to work with the unsupervised k-means algorithm as have done multiple papers focusing on organ transplantation (Berrevoets et al. (2020), Berrevoets et al. (2021), Xu et al. (2021)). K-means clustering has had extensive use in healthcare research due to the algorithm's simplicity and capacity to scale to large and complex data. Indeed, Liao et al. (2016) showed k-means is the best clustering method to cluster end stage renal disease patients who initiated hemodialysis. Such clustering is also used by Tosto et al. (2016) to cluster patients with Alzheimer's Disease to examine subgroups of patients and by Haldar et al. (2008) to classify patients with asthma.

Ranking and clustering are not enough to address the problem at hand of modelling the organ transplant system. Indeed in the transplant system one of the most important components, then to incorporate this aspect of the transplant system we will also employ queues to construct our framework.

2.4 Queuing Systems For Organ Allocation

Green (2006) demonstrates the growing use of queuing models in healthcare for instance to analyse the impact of various admissions policies to ICU facilities or to model the occupancy level of hospital beds. Queuing models have, in fact, already been applied to the organ transplant problem. We now review past approaches.

A first model is the one developed by Zenios (1999) which assume K classes of organs and patients where candidates of the same class are allocated organs on a first come first transplant (FCFT) basis. One limitation of this model is that these classes of patients and organs have no medical meaning. Using the same approach, but with medically meaningful classed, Stanford et al. (2014) designed four classes based on blood types (ABO types). It is worth noting however, that limiting to only four types means the model does not taking into account other features relevant to determine compatibility such as age, weight, size, or location to create the types.

Alternative approaches have also been explored, such as Boxma et al. (2011) which proposed the use of double-matched queues; queues for the patients as well as queues for the organs and Abellán et al. (2006) which does not consider patient or organ types and simply models the system with a single queue of patients served following FCFT service discipline. These alternative have the benefit of being simple enough to conduct further analysis on the system in a straight forward manner using well known queuing models such as M/M/1 (Hall, 1991) (M/M/1 represents a system having a single server, where arrivals are determined by a Poisson process and job service times have an exponential distribution). However these simplification preclude wide application as for instance since organs are a scarce and perishable we may assume none are accumulating in a queue.

To improve upon Stanford et al. (2014) and Zenios (1999), approaches which cannot model patient health deterioration, Drekić et al. (2015) proposed a two-level dependent quasi birth and death process, where for each queue, patients can move from lower to higher priority levels to account for the degradation in patient health. With the higher priority queue being pre-emptive, meaning if an organ of type k arrives it will be offered to the patient on the high priority level for type k before being offered to those on the low priority level of type k . This model is the most comprehensive to date, however gaps to model a real life allocation policy still remain. Indeed, in real life sicker patients gain access to transplants quicker than healthy ones. However patients in Drekić et al. (2015) are still served following FCFT discipline within the 2 priority levels meaning a patient can never have access to an

Table 2.2 **Comparison with related queuing systems.** Organ Queue satisfies 5 key aspects to model a transplant system: (1) It uses single matched queues as organs are scarce and perishable resources, (2) it uses multiple queues for the patients to model organ-recipient compatibility (2.b) specifies how many queues, 4 corresponds to ABO types queues, (3) it is able to model the health degradation of patients over time, (4) it avoids using the over-simplifies FCFT policy to model different patient priorities, (5) it can model any allocation policy

REFERENCE	(1)	(2)	(2.b)	(3)	(4)	(5)
Boxma et al. (2011)	×	×	-	×	×	×
Abellán et al. (2006)	✓	×	-	×	×	×
Zenios (1999)	✓	✓	K	×	×	×
Stanford et al. (2014)	✓	✓	4	×	×	×
Drekic et al. (2015)	✓	✓	4	✓	×	×
Ours	✓	✓	$4 \times K$	✓	✓	✓

organ before a patient who was added to the high priority queue before them, even if his health condition is worse.

With this work we propose to combine the ideas of previous works. We use $4 \times K$ queues to make use of blood groups and other features to model the patient-organ compatibility. We include a mechanism to model patient’s health deterioration by allowing patients to be added to multiple queues at differed times. We also take into account the limitation of Drekic et al. (2015) and propose to model continuous priority by allowing patient to be added at any position in the queue not only at the back of one at the moment of their arrival.

Lastly, all previous queuing systems model specific allocation policies, Zenios (1999) considers only open-loop randomized policies entirely described by the fraction of organs from a certain class that are allocated to patients of given class, Stanford et al. (2014) only considers ABO-identical transplantation policy and Drekic et al. (2015) models the CanWAIT (Canadian wait-listing algorithm in transplantation) policy. While the goal of our model is to be able to learn and model any allocation policy from the data, even more complex ones due to the model’s capacity to hold out on when to add a patient to the queues. Then since no other model is able to do this, we do not benchmark our model against other existing model but rather benchmark it against a rule based version of the allocation policy in place corresponding to the theoretical policy. Table 2.2 summarizes this comparison of existing queuing models for organ transplantation.

Chapter 3

Problem Formulation

In practice, the allocation policy suffers from variations that affect the allocation of organs pre and post-offer. These variations cause the allocations made by the implemented policy to differ from those made by the theoretical policy. Because of these variations, the offers made may not comply with the theoretical policy and the clinical analyses made pre-implementation may not be valid in real life. Having this discrepancy creates the need for a framework to learn and model the implemented policy and which can be used to analyse it.

Notation. Let $\mathcal{X} \subset \mathbb{R}^d$ denote the space of all possible patients, and let $\mathcal{O} \subset \mathbb{R}^e$ denote the space of all possible organs. Let $\mathcal{X}(t) \in \mathcal{P}(\mathcal{X})$ denote the set of patients in the waitlist at time t . Let $\mathbf{X} \in \mathcal{X}$ denote the feature vector of a patient and $\mathbf{O} \in \mathcal{O}$ the feature vector of an organ. Denote t_j the arrival time of \mathbf{O}_j . We assume we have an observational dataset containing N patients, $\mathbf{X}_1, \dots, \mathbf{X}_N$ and M organs $\mathbf{O}_1, \dots, \mathbf{O}_M$. We assume K organ types, denote \mathbf{O}^k an organ of type k .

Let π be the theoretical allocation policy in place, and Γ_π its implemented policy (i.e the policy encompassing all variations). Given an organ and the set of patients in the waitlist at time of the organ's arrival, Γ_π returns an ordered priority list for that particular organ.

$$\Gamma_\pi(\mathbf{O}_j) = \Gamma_\pi(\mathcal{X}(t_j), \mathbf{O}_j) \rightarrow [\mathbf{X}_i \triangleright \mathbf{X}_n \triangleright \mathbf{X}_m \dots]_{\mathbf{O}_j} \quad (3.1)$$

with $\mathbf{X}_i, \mathbf{X}_n, \mathbf{X}_m, \dots \in \mathcal{X}(t_j)$. Where, the notation $[A \triangleright B]_{\mathbf{O}_j}$ indicates A is ranked higher than B for organ \mathbf{O}_j in the ordered list meaning A will receive an offer for the organ and if A refuses, then B will receive an offer. \triangleright is a quasiorder on the feature space \mathcal{X} that satisfies reflexivity ($\mathbf{X}_i \triangleright \mathbf{X}_i$), antisymmetry ($\mathbf{X}_j \not\triangleright \mathbf{X}_i \Rightarrow \mathbf{X}_i \triangleright \mathbf{X}_j$) and transitivity ($\mathbf{X}_i \triangleright \mathbf{X}_j \triangleright \mathbf{X}_m \Rightarrow \mathbf{X}_i \triangleright \mathbf{X}_m$), for all $\mathbf{X}_i, \mathbf{X}_j \dots \mathbf{X}_m \in \mathcal{X}$. Then, the ranking is said to be consistent.

Motivation. The need for our framework is created by the fact that

$$\Gamma_{\pi}(\mathcal{X}(t_j), \mathbf{O}_j) \neq \pi(\mathcal{X}(t_j), \mathbf{O}_j) \quad (3.2)$$

Training Data. We assume access to a dataset of past organ offers comprising the features of each organ offered and the features of the patients to whom the offers were made ordered according to the order the offers were made in. Then we have access to

$$\mathcal{D} = \{(t_m, \mathcal{X}(t_m), \mathbf{O}_m, W_m = [\mathbf{X}_i \triangleright \mathbf{X}_l \triangleright \dots] \mathbf{O}_m)\}_{m=1}^M \quad (3.3)$$

with W_m being the match run for organ \mathbf{O}_m , $W_m \subseteq \mathcal{X}(t_m)$ and t_m the date at which the organ arrived and the offers where made. As the problem is policy specific, we will train our model on one policy at a time, then we split \mathcal{D} into policy specific data denoted $\mathcal{D}_{\pi_1}, \mathcal{D}_{\pi_2}, \dots$. Moreover, for each patient \mathbf{X}_n the data contains when they were added to the waitlist and when they were removed.

Addressing Core Challenges. To address the core challenges (stated in Chapter 1) we use a queuing-theoretic framework with unsupervised learning to cluster the organs into “organ types”, and then construct priority queues (associated with each organ type) wherein incoming patients are assigned. Challenge (a) concerning complex interactions between patients and organs due to their high dimensional feature spaces is addressed by using organ types. Challenge (b) regarding post and pre offer variations is addressed by building our model post hoc of the introduction of a new policy and using the data governed by the policy to learn the implemented policy. Challenge (c) dealing with incomplete match runs is addresses by adding randomly sampled patients to the end of the match runs to be able to learn which patients are not prioritised for a given organ. Challenge (d) concerning the dependence of the match runs on the patients in the waitlist is tackled by attempting to learn functions which are a function of both the patient \mathbf{X}_n , and the waitlist $\mathcal{X}(t)$. Challenges (e) and (f) regarding the specific functioning of policies and the health deterioration of patients are addressed by allowing patients to be added to multiple queues at differed times.

Correctly fitting the organ allocation problem into a queuing system is a challenging task as in order to adequately learn the allocation policy in place from the data, the model must (i) be able to account for health deterioration of patients (ii) be able to model patients that are more critically ill than other (iii) learn which organ types each patient is compatible with under the policy in place and the clinician’s criteria.

Chapter 4

Organ Queue

Having presented the necessary background and notations in the previous chapters, this chapter presents Organ Queue, the framework developed. We include a discussion of the organ types which allow us to split the problem into K queues and provide a detailed description of the two components of our system: the ranking model and the timing model.

We propose a framework to model the implemented policy able to correctly predict which patients will receive organ offers. To this end, we choose to model the transplant system using a queuing system, however, simply placing each patient at the end of queues when they arrive only allows to model first come first transplant policy (FCFT). Instead we propose a more sophisticated system able to model any allocation policy and model degradation in patient's health. Our system, Organ Queue, introduces K queues, Q_1, \dots, Q_K that correspond to K clusters of organs. In choosing the value of K , we introduce a trade-off between having very specific organ types and having enough data of each type to be able to train our model. In particular, as we take K to infinity, each queue corresponds to a specific organ. To fit the transplant system into these queues it is necessary to learn if and when a patient should be added to each queue and where the patient should be added within the queue. This requires to learn the ranking function for each organ types and the patient-organ type compatibility to only add patient to queues for which they will receive offers under the implemented policy. All these elements are characteristic of the policy in place and will be learnt from data. Once these queues are correctly filled, the framework can be used to predict which patients will receive organ offers under the implemented policy and to conduct further analysis on the implemented policy. Indeed, upon the arrival of an organ, its type is determined and the queue corresponding to this type is given as the predicted match run.

More specifically, we assume K organ types from which are formed K priority queues. Each priority queue is an ordered lists, we denote $Q_k = [\mathbf{X}_i \triangleright \mathbf{X}_j \triangleright \mathbf{X}_l]_k$ the priority queue assigned to type k which contains three patients: $\mathbf{X}_i, \mathbf{X}_j, \mathbf{X}_l$. To be able to correctly model all types of allocation policies we will build two models. The **timing model** which given a patient \mathbf{X}_n predicts when (if ever) they should be added to each of the K queues. The **ranking model** which given a set of patients and an organ type, returns an ordered priority list of these patients for that organ. These models will be used back-to-back to correctly place each patient \mathbf{X}_n in each priority queue Q_k at the accurate times. Both these models are policy agnostic and will thus be able to model any policy π for which we have data \mathcal{D}_π . They are designed to address the core challenges of the problem at hand in the best possible way as will be described in further sections of this chapter.

In order to use our framework to perform various analysis on the implemented policy, we choose to make it as stationary as possible and impose that patients not be re ordered once placed on queues. This enforces the relative order on a queue to remain unchanged and puts even more emphasis on placing patients on the queues at the correct times. Then a patient's health degradation may not be witnessed on a single queue but will be in the queuing system as a whole as the health degradation will trigger the patient to be added to other queues. We assume there is no limit on the number of patients that can be waiting on a queue.

4.1 Organ Types

Modelling the arrival of the high-dimensional organs is challenging, to address this difficulty we introduce K organ types. To create the organ types, we first use the k-means clustering algorithm, then split the donors of each cluster into four subgroups corresponding to the blood types (A, B, O, AB) resulting in $4 \times k$ organ types. Splitting the clusters into blood types allows them to have medical significance and restricts access to certain queues due to the rules of blood type compatibility detailed in Table 4.1.

In the dataset, there also exists blood group subtypes A1, A2, A1B and A2B. In general, 80% of blood group A and AB persons are subtype A1 and A1B, respectively ([Procurment and Network, 2011](#)). However, this is not observed in the data, this is because the majority of clinicians record the patients or donors as A without specifying the subtype. For this reason we decide not to use subtypes, even if they would allow to model more detailed blood type compatibility rules. Indeed, since 2002 the OPTN allows for transplantation of livers from

Table 4.1 **Blood group compatibility rules** and proportion of donors in each blood group in the UNOS dataset for liver transplantats.

Donor Blood Group	Can be transplanted to	Proportion
O	A, B, O, AB	48%
A	A, AB	37.5%
B	B, AB	11.5 %
AB	AB	3%

donors that are blood group “A, non-A1” into O candidates ([Procurement and Network, 2011](#)). We will explore the effect of having a different number of organ types.

4.2 The Timing Model

Having to model complex policies and patient health deterioration, requires to add patients to different queues at different times. Indeed some organs may be less suited for a particular patient, however when the patient comes close to dying, we must consider these organs. Hence, when a patient’s condition deteriorates, we begin placing them on queues for less suitable organs. Predicting when and with which queues to do this is why we need the timing model. For each organ we have access to the match run i.e. the ordered list of patients who received an organ up until the patient who accepted it. And for each patient we have the list of offers they received including the date of each offer and the type of the organ that was offered. Denote T_n^k , the time in days after patient \mathbf{X}_n ’s arrival on the waitlist after which they should be added to queue k . If \mathbf{X}_n is never added to queue k , then $T_n^k = \text{NaN}$.

Predicting when to add a patient to a queue is not an easy task. The main reason for this is that, when a patient accepts an offer (or dies or is removed from the list before receiving one), we can no longer observe how the timing of queues would have unfolded. In other words, a patient’s list of offers is almost never complete. In addition, the match run for each organ is incomplete since as soon as a patient accepts the organ no other offers are made, we thus do not have access to which other patients would have received an offer for this organ and thus would have been on the queue for that organ.

Then, the only information we have regarding if a patient is on a given queue is when a patient receives an offer for an organ of this queue’s type. From this information we can build a target dataset indicating that at least from that point on the patient was on the queue for this organ type. However, knowing exactly when the patient joined the queues is impossible since

the match run is incomplete and knowing which other queues the patient is compatible with is not possible since the list of offers is incomplete. We decide to use the first time the patient receives an offer for an organ of a given type (in number of days since the patient was added to the waitlist) as an indicator of when the patient must be added to a given queue. However, patients only receive offers from few organ types. Then all organ types for which the patient never received an offer will have missing time until offer. For some organ types, 90% of entries are missing. Training a model to predict T_n^k is thus extremely challenging. To overcome this difficulty, we must develop a model that uses features both to predict the probability of a response and to predict the value of a response if it occurs.

Ideally, we have to take into account other patients when predicting allocation, indeed patients only receive an offer when all patients with higher priority or higher utility for an organ have received and rejected the offer. Then the logical method would be to include the waitlist $\mathcal{X}(t)$ as an input along side the patient covariates \mathbf{X}_n . We thus constructed a neural network following a siamese architecture which takes in these two inputs (details in section A.1). However, we found such a network to perform worse than models based solely on \mathbf{X}_n (see Table A.2). We argue that the lack of data - having only minimally observed match runs and list of offers - heavily impacts training difficulty. Indeed having such large amounts of “unobserved” data develops noise for this model making the learning task much harder.

Instead, we explored the use of Hurdle Models (Cragg, 1971) to predict T_n^k using only \mathbf{X}_n . A hurdle model is a two-part model that specifies one process for zero counts and another process for positive counts which occur once a hurdle is cleared. In our case, clearing the hurdle corresponds to being added to a queue. Then one process predicts if the patient should be added to a queue and the other predicts when they should be added. To place the hurdle at zero, all missing entries in the dataset (i.e. queues patient were not added to) are filled with zero and +1 is added to all other values. Then a binomial probability model governs the binary outcome of whether a count variable has a zero or a positive value (i.e. if a patient is added to a queue or not). If the value is positive, the “hurdle is crossed,” and the conditional distribution of the positive values (i.e. when to add a patient to a queue) is governed by a zero-truncated count model. Hurdle models differ from count models and zero inflated models as they assume zeros and non-zeros (positives) to come from different data-generating process whereas count models assume they come from the same and zero inflated models assume a mixture of distributions. Hurdle models have been used in medicine due to their ability to handle excess zeros and over dispersion. For instance for

Vaccine Adverse Event (Rose et al., 2006) or HIV-risk reduction intervention (Hu et al., 2011).

4.3 The Ranking Model

The ranking model's goal is to learn a function \mathcal{R} which ranks patients for an organ as the implemented policy would. Given a set of patients and the type of an organ (called the query), the ranking model must return the ordered priority list of these patients for that query

$$\mathcal{R}(\{\mathbf{X}_i, \mathbf{X}_j \dots \mathbf{X}_l\}, k) \longrightarrow [\mathbf{X}_i \triangleright \mathbf{X}_j \triangleright \dots \mathbf{X}_l]_k \quad (4.1)$$

As the patients have no absolute priority but rather a priority which is dependent on the organ and the allocation policy we train a different model for each allocation policy, and learn \mathcal{R} as a function of the organ type k .

We use the RankNet approach (Burges et al., 2005) to ranking which solves the preference learning problem directly by training on pairs of examples. Denote \bar{P}_{ij}^k , where $i, j = 1, \dots, N$ and $k = 1, \dots, K$ the desired target values with $\bar{P}_{ij}^k = 1$ and $\bar{P}_{ji}^k = 0$ if $[\mathbf{X}_i \triangleright \mathbf{X}_j]_k$ and $\bar{P}_{ij}^k = 0.5$ when no information is available as to the relative rank of two patients for an organ of type k .

The RankNet developed is detailed in Figure 4.1. It consists of a neural network which given input x_i^k outputs $f(x_i^k)$. The input x_i^k is the concatenation of the features of patient \mathbf{X}_i and the one hot encoding of organ \mathbf{O}_m 's type, denoted ϕ_k . To rank two patient \mathbf{X}_i and \mathbf{X}_j for organ \mathbf{O}_m of type k we input separately into the same network x_i^k and x_j^k . We consider models $f: \mathcal{R}^{d+K} \mapsto \mathcal{R}$ such that the rank order of a set of test samples is specified by the real values that f takes, specifically, $f(x_i^k) > f(x_j^k)$ is taken to mean that the model asserts that $[\mathbf{X}_i \triangleright \mathbf{X}_j]_k$ (Burges et al. (2005)). Since we are interested in predicting the posterior $\mathbb{P}([\mathbf{X}_i \triangleright \mathbf{X}_j]_k) = P_{ij}^k$, the network uses a logistic function on the difference of the outputs.

$$P_{ij}^k = \frac{e^{o_{ij}^k}}{1 + e^{o_{ij}^k}} = 1 - P_{ji}^k \quad \text{with} \quad o_{ij}^k = f(x_i^k) - f(x_j^k) \quad (4.2)$$

Then RankNet uses a binary cross entropy cost function on the priorities of the patients for a given organ type:

$$C_{ij}^k = -\bar{P}_{ij}^k \log P_{ij}^k - (1 - \bar{P}_{ij}^k) \log (1 - P_{ij}^k) \quad (4.3)$$

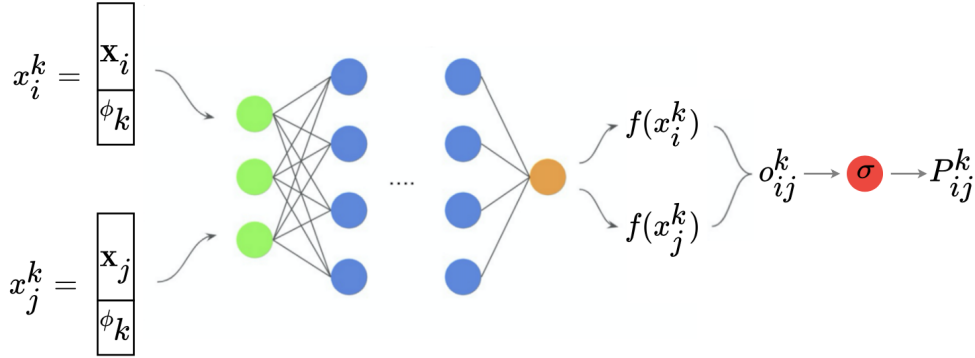


Fig. 4.1 **Our RankNet Architecture**, ϕ_k is the one hot encoding for an organ of type k and σ represents the logistic function

Using a cost function on the difference of the outputs reflects the underlying goal of learning to rank which does not focus on the exact score that each item gets, but cares about the relative ordering among all the items.

During training we feed the network pairs of patients who are within the same match run. The ultimate goal of our ranking model is to be able to output the ordered offer list for an organ given the waitlist. However, challenge (c) concerned with the fact that the match run only contains a subset of the patients, impacts the training of the RankNet. Indeed since the match runs do not contain the entire set of patients in the waitlist but only those on the top of the priority list for a given organ, if the RankNet were trained only using these match runs it would not be able to learn compatibility criteria between organ types and patients as all the patients it would be trained on would be already filtered to be compatible for the given organ. To address this challenge, when training the ranking model, we add to the match run W_m associated to each organ \mathbf{O}_m a few randomly sampled patients who did not receive an offer for that organ, we call this the extended match run. Denote the set of added patients S_m , these patients are all given the lowest priority i.e. if \mathbf{O}_m is of type k , $P_{ij}^k = 0 \quad \forall \mathbf{X}_i \in S_m, \mathbf{X}_j \in W_m$ and $P_{ij}^k = 0.5 \quad \forall \mathbf{X}_i, \mathbf{X}_j \in S_m$. This enable the model to be more robust by learning which patients would be irrelevant to an organ type.

In this chapter we have presented the main components of our model and motivated our design choices. These models will be trained and tested using real life organ transplant data in the following chapters.

Chapter 5

Experiments on Real Data

In this chapter, we present the data we use to train and test our framework. Focusing on the MELD-Na policy for liver transplants, we provide results on the performance of our general framework Organ Queue and perform various ablations studies to show the utility of each of its components. We also provide an example of how our framework can be used to analyse an implemented policy by assessing if the MELD-NA policy achieves its intended effect.

5.1 The Data

We have access to the United Network for Organ Sharing (UNOS) data (Cecka, 2000) since 1996 for liver, pancreas, intestine, kidney, thoracic and transplants. The data includes all patients who were placed on the waitlists for these organs, including those who received a transplant and those who were removed without receiving one. For all of these patients, we have access to a large number of features which have evolved through time, reflecting clinicians' growing understanding of organ-recipient compatibility and post transplant outcomes. These features range from clinical (e.g. serum sodium concentration) to socio-economic ones (e.g. education level, income). The data contains a similar range of features for donors.

We choose to focus on liver transplantation for which we have 318,049 patients. We limit the data used to post 2005 (112,775 patients) for consistency as the features recorded were different before that. For example, before 2005, the concentration in serum bilirubin and creatinine were not documented whereas now a days these variables are deemed essential to assess a patient's condition. We restrict the study to non-pediatric patients (103,821 patients) as pediatric transplants do not follow the same allocation due to the complexity of finding matches notably due to restrictions on organ size and immunosuppressive drugs use.

We do not consider patients who underwent multiple organ transplantation (e.g simultaneous liver kidney transplantation) as the goal is to correctly model the allocation policy governing the allocation of livers specifically. Lastly, we only consider deceased-donor organs as most often living donor donations are directed (meaning the donor specifies the recipient) thus do not follow the implemented allocation policy which our model aims to learn.

For the medical features, the dataset comprises initial entries recorded when patients were added to the waitlist and final entries recorded upon removal from the waitlist (due to transplant, death or abandonment). As we attempt to develop a prediction model, we use only the initial features, and remove all of the data concerned with the transplant and the post-transplant outcomes. As a pre-processing step, we convert all categorical variable with fewer than five options into indicator variables using dummy variable encoding and ordinal encode the other categorical variables. We drop features which have an excessive proportion of missing values and use a simple imputer to account for the missing values, using the most frequent strategy for categorical features and the mean strategy for numerical features.

To evaluate Organ Queue’s performance, we choose the MELD-Na allocation policy (Kim et al., 2008) which came into effect on January 11th 2016. (Kalra and Biggins, 2018) after UNOS proposed that MELD-Na score — an extension of the Model for End-Stage Liver Disease (MELD) which incorporates the serum sodium — was able to better rank candidates based on their risk of pre-transplant mortality. In December 2017, UNOS updated the (exception point) scoring system, we consider this as the start of a new policy and only work on the data between January 11th 2016 and December 1st 2017. In this time frame, 15,186 deceased donor livers were donated, and 21,523 new patients were added to the waitlist, in addition to the 13,557 patients already on the waitlist at the start of the time frame.

5.2 The MELD-Na Allocation Policy

The MELD-Na Allocation Policy offers organs to patients following their MELD-Na score. As soon as 2004 (Heuman et al. (2004), Biggins et al. (2005)), Serum sodium concentration (Na) was suggested as a useful predictor of mortality in patients with end-stage liver disease awaiting liver transplantation. Various studies were conducted to identify the exact link between sodium concentrations and mortality (Biggins et al. (2006), Kim et al. (2008), Leise et al. (2011)) in an attempt to find the optimal way of defining a new score, the MELD-Na score to prioritise patients for transplant . This score is based off of the previously used Model

for End-Stage Liver Disease (MELD) score which is computed on the basis of the serum bilirubin concentration and serum creatinine concentration, and the international normalized ratio for the prothrombin time. The equation found to compute the MELD-Na is as follows:

$$MELDNa = MELD + 1.32 \times (137 - \bar{Na}) - (0.03 \times MELD \times (137 - \bar{Na})) \quad (5.1)$$

with

$$\bar{Na} = \begin{cases} 125 & \text{if } Na < 125 \text{ mmol/L} \\ 137 & \text{if } Na > 137 \text{ mmol/L} \\ Na & \text{otherwise} \end{cases} \quad (5.2)$$

This equation places emphasis on serum sodium concentrations between 125 and 137 mmol/L as this is the range where the most meaningful differential effect of serum sodium concentration on mortality appears (Kim et al., 2008).

5.3 Evaluation of Organ Queue

We evaluate Organ Queue's performance by using the timing and the ranking model, back to back to assess if these are able to correctly model the implemented allocation policy by offering the incoming organs to the correct patients. For each organ, we compare the match run predicted by Organ Queue, denoted \mathcal{Q} , to the match run in the data denoted \mathcal{R} . \mathcal{Q} is simply the entire queue corresponding to the incoming organ's type.

5.3.1 Training and Testing Data

We consider the data governed by the Meld-Na policy and use the data from 11/01/16 to 31/10/17 to train the models and the data from 01/11/17 to 30/11/17 to test it. Details concerning the training of the ranking and the timing model are given in the following chapter. The test is conducted in two phases. The first X days are used to fill the queues with patients. For each incoming patient, the timing model predicts if and when they should be added to each queues and the ranking model places them on those queues at the corresponding day. Then, starting on day $X + 1$, once the queues contain patients, the system starts to consider incoming organs and produces a match run for them while continuing to add incoming patients to the queues.

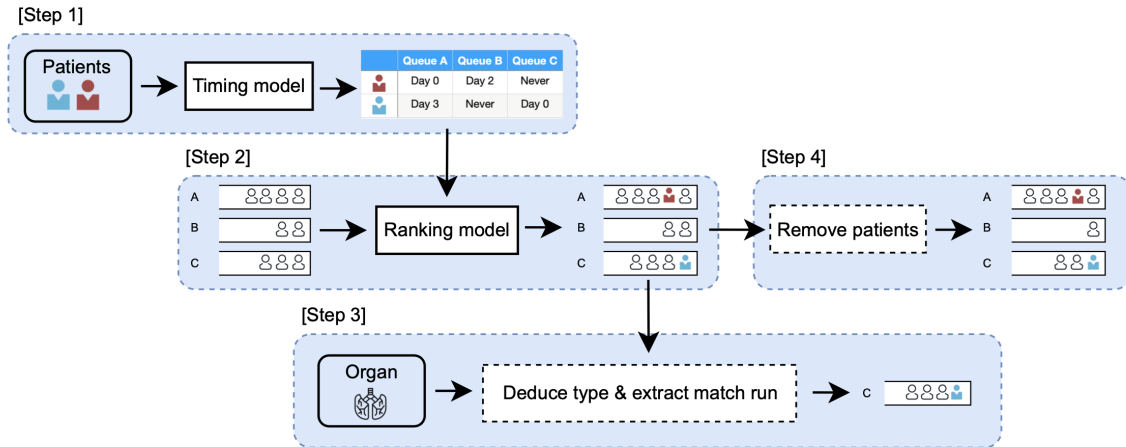


Fig. 5.1 **Diagram of the experimental method for Organ Queue.** Each day of the test 4 steps are performed: *[Step 1]* For each incoming patient that day use the timing model to predict if and when each patient should be added to each queues. *[Step 2]* For each patient that should be added to a queue that day, use the ranking model with the previous day's queues to place the patient within the queues. *[Step 3]* (In the second phase only) For each incoming organ, deduce the type of the organ, and return the queue corresponding to that type as the match run for that organ. *[Step 4]* Any patient in the testing system which was removed from the transplant system that day in the real data is removed from the queues it was a part of. (these updated queues are used for *[Step 2]* of the next day)

5.3.2 Experimental Method

In the evaluation process, two particularities must be taking into account. First, the system we evaluate on only contains a subset of the patients on the waitlist: those whose arrival date is after 01/11/17 (this subset is denoted \mathcal{P}). In order to be able to compare \mathcal{R} and \mathcal{Q} we must reduce \mathcal{R} to contain only patients who are in the test system. Denote $\tilde{\mathcal{R}} = \mathcal{R} \cap \mathcal{P}$. Secondly, patients leave the transplant system. However, Organ Queue does not predict which patient will accept the offer and thus be removed. Then the subset \mathcal{P} of patients in the system will be updated using the real dataset to remove from it those who are no longer in the transplant system. Denote $\tilde{\mathcal{Q}} = \mathcal{Q} \cap \mathcal{P}$ the match run predicted by Organ Queue reduced to patients in the test system which are still in the transplant system. Then for each day of the test we have 4 steps (see Figure 5.1).

5.3.3 Evaluation Metrics

To evaluate the similarity between $\bar{\mathcal{R}}$ and $\bar{\mathcal{Q}}$, we use two metrics. First, the **Jaccard Index** (*Jac*) which treats $\bar{\mathcal{R}}$ and $\bar{\mathcal{Q}}$ as sets is used to evaluate if the match runs contain the same patients. We use this metric on sets as finding which patients to include in the match run is in itself a challenging task. The Jaccard Index produces a percentage of similarity between two sets by returning a value between 0 and 1, with 1 signifying the sets are the same.

$$Jac(\bar{\mathcal{R}}, \bar{\mathcal{Q}}) = \frac{|\bar{\mathcal{R}} \cap \bar{\mathcal{Q}}|}{|\bar{\mathcal{R}} \cup \bar{\mathcal{Q}}|}. \quad (5.3)$$

Second, we use an edit distance, which handles $\bar{\mathcal{R}}$ and $\bar{\mathcal{Q}}$ as ordered lists and returns the minimum number of operations required to transform one list into the other. We choose to work with the **Levenshtein distance** (Levenshtein et al., 1966) which allows as operations insertions, deletions and substitutions. The value of the Levenshtein distance is between 0 and $\max(|\bar{\mathcal{R}}|, |\bar{\mathcal{Q}}|)$ and is thus extremely dependent on the length of the lists. To make the edit distances comparable across organ offers it is given as a ratio by dividing the distance by the maximum length between the length of $\bar{\mathcal{R}}$ and $\bar{\mathcal{Q}}$. As this ratio (between 0 and 1) represents a distance the lower the value of the ratio the better the performance.

$$Lev_ratio(\bar{\mathcal{R}}, \bar{\mathcal{Q}}) = \frac{Lev_dist(\bar{\mathcal{R}}, \bar{\mathcal{Q}})}{\max(|\bar{\mathcal{R}}|, |\bar{\mathcal{Q}}|)} \quad (5.4)$$

The length of the match runs depends on the clinician's decision to accept or reject an organ which we do not model in our system. Thus to evaluate the similarity of the match runs we consider all possible lengths for both match runs. Denote $M = \max(|\bar{\mathcal{R}}|, |\bar{\mathcal{Q}}|)$ and $\bar{\mathcal{R}}_i$ to mean the match run $\bar{\mathcal{R}}$ cut at the i^{th} patient or if $|\bar{\mathcal{R}}| < i$ simply $\bar{\mathcal{R}}$ in its entirety. Then, we report the Jaccard Index and Levenshtein Ratio as follows

$$\mathbf{Jac}(\bar{\mathcal{R}}, \bar{\mathcal{Q}}) = \frac{1}{M} \sum_{i=1}^M Jac(\bar{\mathcal{R}}_i, \bar{\mathcal{Q}}_i) \quad (5.5)$$

$$\mathbf{Lev}(\bar{\mathcal{R}}, \bar{\mathcal{Q}}) = \frac{1}{M} \sum_{i=1}^M Lev_ratio(\bar{\mathcal{R}}_i, \bar{\mathcal{Q}}_i) \quad (5.6)$$

5.3.4 Baselines

We use two baselines both of which do not consider any queues or organ types. First, one based on the laboratory MELD-Na score of patients. To produce a match run \mathcal{B} from this baseline, we take all the patients in the test system who are blood type compatible with the

Table 5.1 **Results of organ allocation with Organ Queue.** We report the Jaccard index (between 0 and 1, higher is better \uparrow) and the Levenstein distance ratio (between 0 and 1, lower is better \downarrow) between the real match run \mathcal{R} and the predicted match run \mathcal{Q} obtained using (1) Our timing and ranking model (2) Our ranking model and an ablation timing model which adds a patient to all its blood type compatible queues upon arrival (3) Our timing model and an ablation ranking model which orders the patients in the queue only using decreasing MELD-Na score (4) two baselines which use no queues. We consider 20 and 32 clusters. The test is conducted on 20 days with the first phase lasting 10 days, results are averaged over 118 match runs (standard deviation in brackets)

Metric	Our model		Timing Model Ablation		Ranking Model Ablation		Baselines	
	20 clusters	32 clusters	20 clusters	32 clusters	20 clusters	32 clusters	MELD-Na	ML
Jac \uparrow	0.0482 (0.073)	0.071 (0.097)	0.035 (0.067)	0.033 (0.069)	0.040 (0.079)	0.062 (0.087)	0.040 (0.093)	0.021 (0.026)
Lev \downarrow	0.963 (0.061)	0.941 (0.088)	0.987 (0.023)	0.981 (0.032)	0.972 (0.044)	0.946 (0.081)	0.975 (0.03)	0.985 (0.041)

incoming organ and order them in decreasing MELD-Na score to recover the ordered list \mathcal{B} . Second, we use a machine learning approach which produces a ranking of patients for specific organs. This model is based on the RankNet described in section 4.3. However, instead of concatenating the one hot encoding of the donor type to the patient covariates, we concatenate the donor covariates. This model is trained with the same data as our ranking model. Then to produce a match run \mathcal{M} for a given organ \mathbf{O}_m using this model, we feed the covariates of every pair of patients in \mathcal{P} concatenated with the organ’s \mathbf{O}_m covariates into our model and produce from the set of ordered pairs of patients an ordered list of patients.

5.4 Results

5.4.1 Main Results

In table 5.1 we report the performance of our model (with 20 and 32 clusters) in comparison to the baselines and conduct an ablation study to assess the utility of the timing and ranking models. The first columns is the performance when using the timing and ranking model as described in sections 4.2 and 4.3. The second columns use an ablation of the timing model where each patients is added to all the queue it is blood type compatible with at the time of the patient’s arrival in the waitlist, patients are then placed within the list using the ranking model of section 4.3. In the third columns our timing model is used however instead of using the ranking model to order patients on each queue, the patients are ordered using the decreasing Meld-Na score. Finally, the last columns present the results given by the two baselines. (Additional results in Table A.1).

Table 5.2 Results of organ allocation with Organ Queue for different organ clustering methods and blood groups. We report the Jaccard index and the Levenstein distance ratio between the real match run \mathcal{R} and the predicted match run \mathcal{Q} for 32 queues. The 32 organ types are created using (i) k-means then further splitting into blood groups (ii) k-means and (iii) blood groups further split using k-means. We also report the performance for each blood group using *k-means + blood* clustering method. The test is conducted on 20 days with the first phase lasting 10 days.

	Organ Clustering Method			Organ Blood Type			
	<i>k-means + blood</i>	<i>k-means</i>	<i>blood + k-means</i>	A	B	O	AB
Jac \uparrow	0.071 (0.097)	0.037 (0.055)	0.033(0.565)	0.126 (0.213)	0	0.056 (0.055)	0
Lev \downarrow	0.941 (0.088)	0.970 (0.048)	0.977 (0.035)	0.878 (0.215)	1	0.966 (0.029)	1

We find that our model outperforms the baselines. In addition our models performs better then the timing and ranking ablations proving each component of the model is necessary. Essentially the ranking model ablation and the MELD-Na baseline rank the patient in the same way: using the decreasing MELD-Na score however, they don’t rank the same patients. The fact that the ranking model ablation outperforms the baseline by so much (for 32 clusters) proves the utility of the queues and the timing model to narrow down which patients to rank. The difference in the performance of our model and the timing model ablation shows the impact of delaying when to add patients to queues as well as choosing which queues to add them to. The low performance of the *ML* baseline is due to challenge (a) concerning the high dimentionality of the organ to patient compatibility space, it shows the utility of reducing the high dimensional problem to a lower dimensional problem by using K organ types.

5.4.2 Additional Analyses

Clustering method. We report an ablation study on the method of clustering organs into types (Table 5.2). Our chosen method (used for Table 5.1) which first clusters using k-means then splits each cluster into the 4 blood groups (denoted *k-means + blood*) yields the best performance.

Blood types. There are large disparities in the number of donors per blood groups (Table 4.1) which translates in discrepancies in the number of donors per queues and thus the number of patients added to these queues. As it could be expected, this affects the performance of our model (Table 5.2). Indeed the timing model is unable to correctly model when to add patients to the queues corresponding to blood type B and AB donors as there are very

Table 5.3 **Results of organ allocation with Organ Queue with the *mixed* method for MELD-Na policy and a new policy.** We report the Jaccard index and the Levenstein distance ratio between the real match run \mathcal{R} and the predicted match run \mathcal{Q} for 32 queues. The *normal* method uses our timing model, the *mixed* method uses the timing ablation for queues corresponding to blood group B and AB and our timing model for queues O and A, both use our ranking model. In addition, the performance is given for each blood group using the *mixed* method these can be compared to the results in Table 5.2. Lastly we also report the performance for a new policy. All tests are conducted on 20 days with the first phase lasting 10 days, the results are averages over the number of donors for each column.

	All Organs		Organ Blood Type (<i>mixed</i> method)				New Policy
	<i>normal</i>	<i>mixed</i>	A	B	O	AB	<i>mixed</i>
donors	119	119	41	14	60	4	127
Jac \uparrow	0.071 (0.097)	0.085 (0.143)	0.126 (0.213)	0.060 (0.049)	0.056 (0.055)	0.186 (0.1959)	0.082 (0.147)
Lev \downarrow	0.941 (0.088)	0.932 (0.137)	0.878 (0.215)	0.960 (0.020)	0.966 (0.029)	0.873 (0.116)	0.939 (0.149)

few patients which are added to these queues. The timing model then simply never adds patients to these queues causing the match run predicted for the organs associated to those queues to be empty and the performance to be at its lowest. Not being able to generalise models to certain blood types is a common problem, indeed in Drekić et al. (2015) due to the disparate relative frequencies of blood types, no results for blood type AB are presented as the corresponding parameter estimates were deemed not to be reliable. To remedy this and to obtain an accurate model for all blood groups we decide to use the timing ablation for the queues corresponding to donors of blood group B and AB. To recall the timing ablation adds each patient to all its blood type compatible queues upon arrival on the waitlist. Only patients of blood group B or AB are compatible with donors of blood group B and AB then we apply the timing ablation only for these patients for these queues, this is called the *mixed* method. The *mixed* method improves considerably the performance of our model (Table 5.3).

Another Policy. Lastly, we show our model can learn any allocation policy from data by showing it can achieve similar performance on a different allocation policy. We consider the data from 18/06/2013 to 01/06/2015 when the Share 35 distribution policy was implemented in complement of the MELD score and before exception scores were updated (Kalra and Biggins, 2018). Allocation results after training our ranking and timing model on this data are given in Table 5.3 column **New Policy**.

5.5 Clinical Application

Verifying the intended effect of MELD-Na. The MELD-Na score was developed so as to give more points, and thus higher priority, to patients with hyponatremia (defined as serum sodium concentration <137 mmol/L) as the the MELD score does not accurately reflect the risk of death of this patient group. Then the MELD-Na policy was implemented with the belief that for these patients, the difference between the MELD score and the MELD-Na score would be large enough to make a significant difference in the probability of receiving a liver transplant and averting death. We now use our framework to verify that the MELD-Na policy achieves its intended effect.

We use our model to learn the implemented policy of the MELD policy and the implemented policy of the MELD-Na policy (using different data governed by each of these policies). Then we run a test using patients amongst which patients with hyponatremia and observe if these patients receive more offers and are at the head of more queues under the MELD-NA implemented policy than the MELD implemented policy.

We consider 19 patients whose serum sodium concentration is between 125 and 137 mmol/L as this is the range where the most meaningful differential effect of hyponatremia on mortality appears (Kim et al., 2008). Under the MELD implemented policy, these patient are added on average to 3.74 of the 32 queues, while under the MELD-NA implemented policy the same patients are added to on average 4.52 queues. Proving these patients' health is deemed worse under the MELD-Na policy. In addition, given the same queue, we observe where in the queue each of the implemented policies places a patient with hyponatremia. This test is conducted with many different queues and patients with hyponatremia always reaching the same conclusion (illustrated in Table 5.4): the MELD-Na implemented policy gives the patient with hyponatremia higher priority in the queue. These results prove that the implemented policy achieves the intended effect of the theoretical policy.

In this chapter we have showed our framework is able to truthfully model an implemented policy. Using a case study focused on the MELD-Na policy we have shown our framework outperforms a baseline based on the theoretical policy and one which uses ML without any consideration of queues. Through ablation studies we have exhibited the utility of each of our design choices. In addition, we have used our framework to confirm that the MELD-Na implemented policy reaches the desired effects of the MELD-Na theoretical policy by prioritising patients with hyponatremia. This proves our framework is able to fulfill its goal

Table 5.4 **Illustrative example placing patient 1368548 with hyponatremia into a queue** using the ranking model for the MELD and MELD-Na implemented policy. Patients are represented by their waitlist ID.

Policy	Resulting Queue
MELD	[1366757 ▷ 1365761 ▷ 1365811 ▷ 1365372 ▷ 1366491 ▷ 1365548 ▷ 1365851 ▷ 1366298 ▷ 1365924 ▷ 1365742 ▷ 1366478 ▷ 1365357 ▷ 1368548 ▷ 1366358 ▷ 1365951 ▷ 1366110 ▷ 1366342 ▷ 1366459]
MELD-Na	[1366757 ▷ 1368548 ▷ 1365761 ▷ 1365811 ▷ 1365372 ▷ 1366491 ▷ 1365548 ▷ 1365851 ▷ 1366298 ▷ 1365924 ▷ 1365742 ▷ 1366478 ▷ 1365357 ▷ 1366358 ▷ 1365951 ▷ 1366110 ▷ 1366342 ▷ 1366459]

of providing analyses on the implemented policy. In the following chapter we investigate the performance of each of the components of Organ Queue separately.

Chapter 6

Further Analysis: How it Works

In this chapter, we present how each of the components of Organ Queues is trained and evaluated. We also give clinical interpretation of their output. To ease comprehension Figures 6.1 and 6.5 help re-situate each component within the broad Organ Queue system.

6.1 Evaluation of the Timing Model

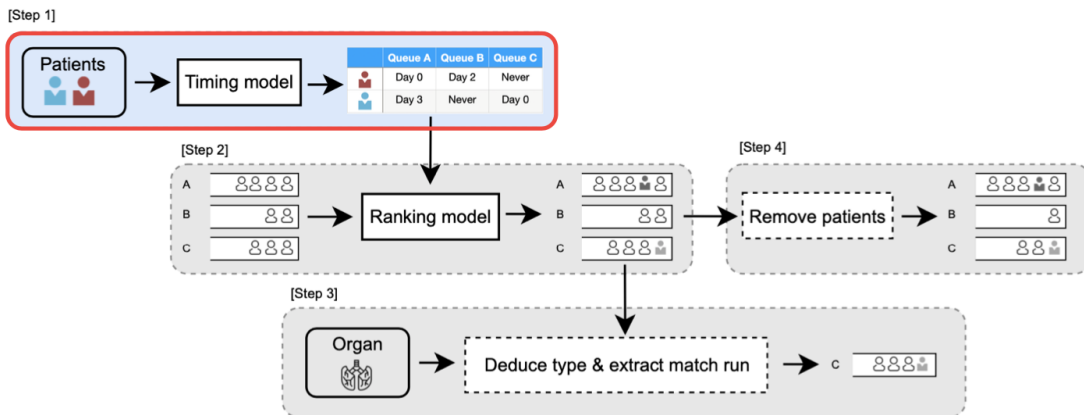


Fig. 6.1 Re-situating the timing model within the Organ Queue system. Recall the timing model predicts if and when a patient should be added to each queue.

6.1.1 Evaluation Metric

We construct K hurdle models which predict the time until first offer (if any) for each organ type separately. Denote \hat{T}_n^k the target value, then we define the leaky RMSE for queue k as:




	Queue A	Queue B	Queue C
	0	2	NaN
	3	NaN	0
	2	1	1

Fig. 6.2 **Computation of the baseline for the regressor**, the table indicates the number of days after which the patients should be added to each queue, and NaN if they are never added. Then baseline for queue A is $(0 + 3 + 2) / 3$ and for B is $(2 + 1) / 2$

$$\text{LeakyRMSE}_k = \sqrt{\frac{\sum_{i=1}^N \left[\max(0.2 * (T_n^k - \hat{T}_n^k), T_n^k - \hat{T}_n^k) \right]^2}{N}} \quad (6.1)$$

This loss takes into account that the time until a patient was added on the queue may be less than the target value which is when the patient receives an offer for the queue. Then this loss penalises less predicting a smaller value for T_n^k than a larger one.

6.1.2 Baseline

We use two baselines. The baseline for the regressor predicts a constant value specific to each queue computed by taking the mean of the non missing entries for each queue (Figure 6.2). The baseline for the classifier simply predicts 1 for all the queues the patient is blood type compatible with and 0 otherwise.

6.1.3 Results

We consider 12,604 patients who were added to the waitlist after January 11th 2016 and who were removed from the list before December 1st 2017 and the offers they received. We explore and compare different configurations of hurdle models, with different classification and regression models such as logistic and linear regression, support vector machine (SVM), random forest (RF) and gradient boosting machine (GBM) (Results are given in Table A.2). The best performance comes from using a Random Forest Classifier (RFC) followed by a Random Forest Regressor (RFR).

The Random Forest Classifier has 100 estimators, uses the entropy criterion, balanced sub-sampled class weights to account for the unbalanced data (particularly for queues of blood type B and AB), and requires a minimum number of 3 samples to split. It is trained on the entire dataset which is binarised (1 if the patient received an offer for the queue, 0

Table 6.1 **Hurdle model performance for different number of clusters.** The mean leaky RMSE (lower is better) averaged over all clusters (standard deviation in brackets) for different number of clusters

		Our Timing Model			
		8 Clusters	20 Clusters	32 Clusters	48 Clusters
Leaky RMSE		30.58 (15.56)	32.96 (13.51)	32.98(14.75)	33.46 (16.83)

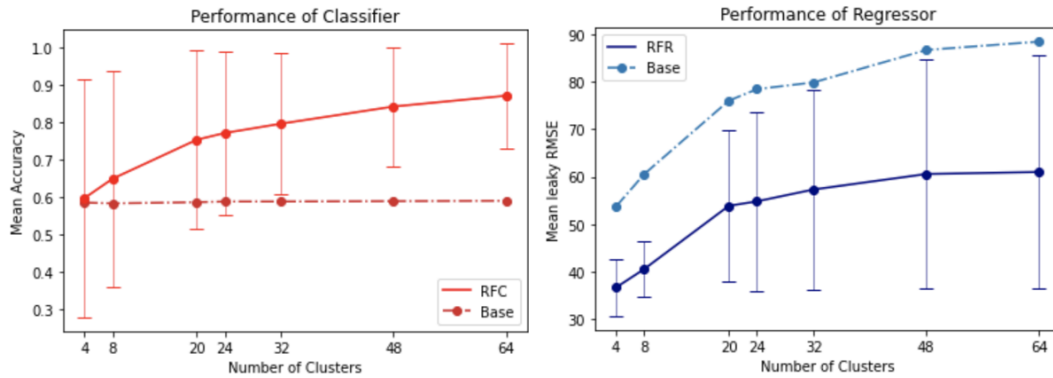


Fig. 6.3 **Performance of the random forest classifier (RFC) and random forest regressor (RFR) of the hurdle model for different number of clusters** The performance is averaged over the number of clusters. For the classifier the accuracy (higher is better), and for the regressor the leaky RMSE (lower is better), the error bars represent the variability in performance over the clusters. The dotted lines show the baseline's performance for each task. (Baselines described in section 6.1.2)

otherwise). The Random Forest Regressor uses 100 estimators, the minimum number of samples at a leaf node is 4, it considers a maximum 3 features when searching for the best split, requires a minimum decrease in impurity for a split of 0.5 and has a maximum depth of 10. It is only trained on data where the patient received an offer. The performance of the hurdle model for different number of organ types is given in Table 6.1.

We investigate the performance of the classifier and the regressor separately into more detail to understand the effect of the number of clusters. Figure 6.3 clearly illustrates that a trade off will have to be made. Indeed, as the number of clusters increases, the accuracy of the classifier increases and the variability in performance amongst the classifier decreases. Conversely, as the number of clusters increases the performance of the regressor worsens (leaky RMSE increases) and the variability in performances increases. To address this trade off we work with 32 clusters which offers a middle ground.

Table 6.2 **Example of outputs from the timing model with 32 queues.** The timing model only adds patients to queues they are blood type compatible with. - indicates the patient was never added to the corresponding queue while an integer value specifies how long after the patient’s arrival they should be added to the queue.

Queue		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
Patient	Type	A	B	O	AB	A	B	O	AB	A	B	O	AB	A	B	O	AB	A	B	O	AB	A	B	O	AB	A	B	O	AB	A	B	O	AB		
1348223	O	-	-	1	-	-	-	-	-	-	-	12	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-		
1254238	A	2	-	-	-	-	-	6	-	-	-	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	-	-	-

6.1.4 Clinical Interpretation of Results

Classifier. We verify that the classifier of the hurdle model is able to capture and learn simple patient to organ compatibility criteria such as blood type compatibility. To assess this we run the timing model for a set of test patients and recover the timing predictions (i.e. T_n^k for $k = 1..K$ for each patient in our test group). Then observing which queues the timing model adds the patient to, we compute which percentage of these the patient is blood type compatible with. Recall O patients can receive a transplant only from O donors, A patients from A and O donors, B patients from B and O donors and AB patients can receive an organ from a donor of any blood group.

We use 32 queues and test on a set of 159 patients, on average 99.27% of the queues the patients are added to respect the blood type compatibility criteria. (see Table 6.2 for an example) This proves our timing model can capture medical criteria of compatibility to accurately model the transplant system.

Regressor. To interpret what the Random Forest Regressor is carrying out we focus on feature importance. We consider 32 organ types, record feature importance for each of the 32 regressions and compute the average importance for each feature. The top five most important features on average along with the value of the average feature importance are presented in Figure 6.4. The clinician we work with confirmed that all 5 of these are important factors when deciding which patient will receive an offer under the MELD-Na policy. Meaning that under MELD-Na these are all important indicators when predicting when a patient should be added to the queues. This prove that our model is correctly learning the implemented policy from the data.

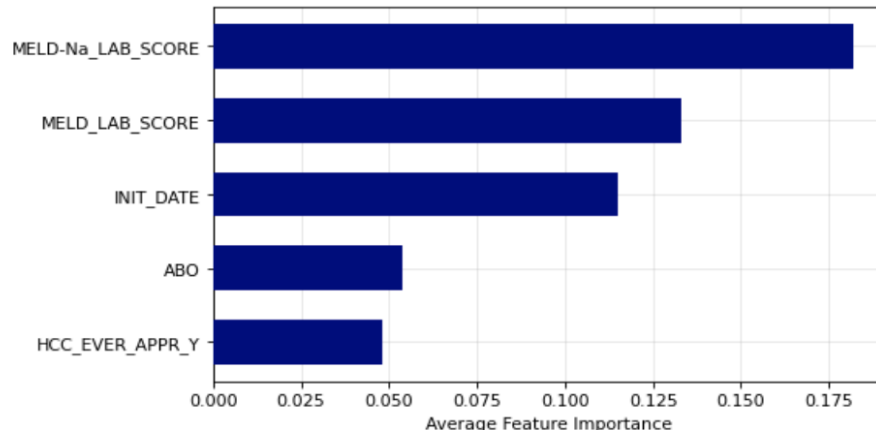


Fig. 6.4 **Top 5 highest average feature importance averaged over 32 the Random Forest Regressors** . The "MELD-Na LAB SCORE" and "MELD LAB SCORE" are the laboratory MELD-Na and MELD score when the patient joined the waitlist, the "INIT DATE" is a proleptic gregorian ordinal of the date at which the patient joined the waitlist, "ABO" is the patient's blood group and "HCC EVER APPR Y" is a binary variable indicating if the patient ever had an approved Hepatocellular Carcinoma (HCC) exception.

6.2 Evaluation of the Ranking Model

6.2.1 Evaluation Metrics

The NDCG Metric

There exists many statistical metrics to evaluate distances between rankings such as Spearman's ρ and Kendall's τ . However, these fail to take into account element relevance and positional information (Kumar and Vassilvitskii, 2010) concepts which are crucial to evaluating a result set in information retrieval (See Table A.3). Thus, the ranking accuracy is given using a Normalised Discounted Cumulative Gains measure of top-r documents retrieved (NDCG @ r) (Järvelin and Kekäläinen, 2017) which takes into account element relevance and positional information by giving each patient a relevance score $rel()$ for a given organ. The first patient to receive an offer has highest score, scores decrease in the order of the offers, any patient who did not receive an offer has score 0. The DCG (Eq.(6.2)) places a strong emphasis on retrieving relevant documents by penalizing highly relevant documents that appear lower in the search by reducing the graded relevance value logarithmically proportionally to the position of the result. Normalising the DCG allows to compare queries for which search results list vary in length.

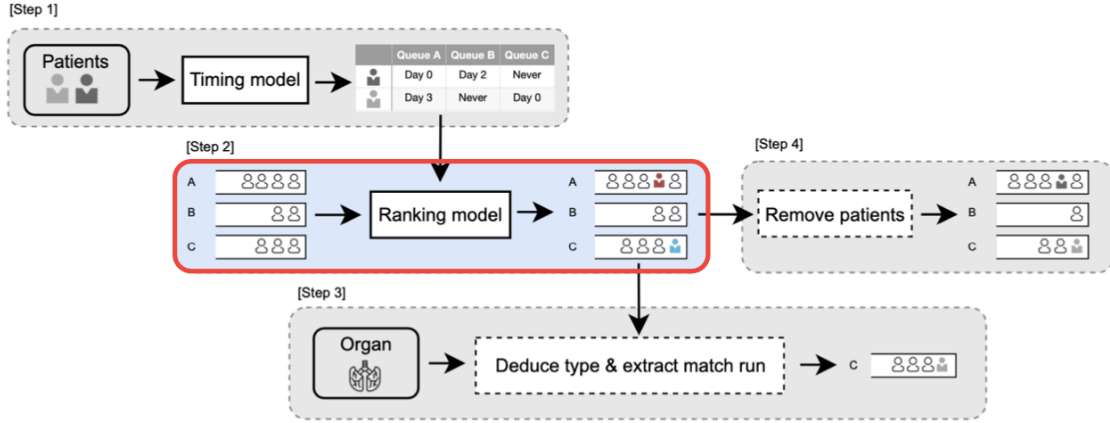


Fig. 6.5 **Re-situating the ranking model within the Organ Queue system.** Recall the ranking models learns a ranking function for each organ type to place patients within queues.

The NDCG computed at rank r is given by

$$NDCG@r = \frac{DCG@r}{iDCG@r} \quad DCG@r = \sum_{i=1}^r \frac{rel(i)}{\log_2(i+1)} \quad (6.2)$$

with $rel(i)$ the relevance score of the i^{th} patient and $iDCG$ being the DCG of the ideal order, so that a perfect ordering has $NDCG = 1$. For queries with fewer than r offers the NDCG is computed for all returned patients. $0 \leq NDCG@r \leq 1$, with a higher value being better. An alternative to the DCG in Eq.(6.2) given in Eq.(6.3), it puts a stronger emphasis on the relevance score and thus gives a relatively too big importance to the first few patients compared to the following ones. Seeing as for most organs the first offers are not accepted it is important to place a strong emphasis on the entire ranking, then we choose to work with the DCG in Eq.(6.2).

$$DCG@r = \sum_{i=1}^r \frac{2^{rel(i)} - 1}{\log_2(i+1)} \quad (6.3)$$

The mAP Metric

In addition to the NDCG metric we use the Mean Average Precision (mAP) metric to evaluate the performance of our model to distinguish and rank relevant/irrelevant patients. Indeed we need our ranking model not only to correctly order the patient who received an offer but also to correctly predict those who will be irrelevant for a particular organ type and who will thus be ranked lowest for it. For this metric, the precision for the top- k documents for query q is

introduced. Since this only makes sense for binary classes, we binarize the relevance score so that $rel(i) = 1$ indicates the i^{th} patient is relevant i.e. received an offer for that query/organ, and $rel(i) = 0$ indicates the i^{th} patient is irrelevant i.e. did not receive an offer. Then the average precision over all relevant patients, where n is the number of patients for query q is

$$AvgP_q = \frac{1}{n \cdot P@n} \sum_{r=1}^n rel(r) P_q@r \quad P_q@r = \frac{1}{r} \sum_{i=1}^r rel(i) \quad (6.4)$$

Finally the mAP (between 0 and 1) for a dataset with Q queries is given by:

$$mAP = \frac{1}{Q} \sum_{q=1}^Q AvgP_q \quad (6.5)$$

6.2.2 Baseline

We obtain a ranking using the MELD-Na score, it is constructed by ordering the patients in the extended match run in decreasing MELD-Na score order taking into account blood type compatibility. Meaning that in this ranking, the patients which are blood type compatible with the organ are ranked with higher priority than those which are not compatible even if their MELD-Na is lower.

6.2.3 Training and Testing Data

Organ Procurement Organisations (OPOs) are non-profit organizations responsible for the evaluation and procurement of deceased-donor organs for organ transplantation. They have the ability to explicitly mark a record on the match run as a ‘bypass’ to indicate that the candidate/center never actually received an offer. Candidates are ‘bypassed’ for a variety of reasons for instance in the case of a directed donation or natural disaster. Even if these candidates never receive the offer, the bypassed offers give us an insight on which patients the policy would have made an offer to. Thus we decide to leave these offers in the dataset used for training and testing. This causes the number of offers to sometimes explode giving rise to an excessive amount of training pairs, thus we apply a cut off and choose to only work with the first 2053 offers for an organ, which corresponds to the 75th percentile on the number of offers. Then the median number of offers (including bypassed offers) becomes 407, this will be used as the rank for the NDCG.

All the patients in the match run in the data are ‘relevant’ to the organ type considered, in order to make our ranking network more robust we add to each match run a random number

of randomly selected patients, this is called the extended match run. The goal is to permit our model to encounter patients which are ‘irrelevant’ to a particular organ type for instance patients that are blood type incompatible so that it learns to give them lowest priority. The ranker is tested on 10 sets of offers containing 10 extended match runs. The performance is evaluated using the NDCG and mAP.

6.2.4 Results

We train on 10,000 offers and add between 1 and 100 randomly selected patients to each match run. This yields more than 3 million training pairs. We train for 50 epochs with a batch size of 256. The RankNet used has 3 hidden layers and a dropout layer. The data is normalised and we use an Adam optimizer with learning rate 0.01. We average the test results on 10 sets of 10 match runs, adding to each of the match runs between 1 and 100 random patients. In Table 6.3 we report the mean NDCG (at rank 407) for two rankings: the ranking obtained using our RankNet and the ranking obtained using the baseline.

Table 6.3 **Performance of our ranking model for different number of clusters.** NDCG@407 (higher is better) of the rankings given by the RankNet and by decreasing MELD-Na score averaged over 10 * (10 extended match runs) (standard deviation in brackets). A random number of patients between 1 and 100 are added to each match run.

	Our Ranking Model				Baseline
	8 Clusters	20 Clusters	32 Clusters	48 Clusters	
NCDG	0.695 (0.095)	0.725(0.093)	0.736 (0.07)	0.753 (0.091)	0.445 (0.102)

Our ranking model must be able to adequately differentiate relevant and irrelevant patients. Indeed, when the model will be used on the transplant system there will be a mix of relevant and irrelevant patients. We wish to further investigate the effect of extending the match runs. To do so we train one model with a random number between 1 and 200 of ‘irrelevant’ added patients to each match run (denoted Q +200) and a second model with no added ‘irrelevant’ patients added (denoted Q +0). Then, we test both these models on data with 10, 100, 200 and and 500 randomly added patients to each match run in the test data. The mAP is used to evaluate the models’ capacity to sort relevant and irrelevant patients. Results in Table 6.4 show performance decreases when adding more irrelevant patients to each match run of the test set. However, the performance is better when the model is trained with irrelevant patients (right column of Table 6.4), proving the model has learnt to differentiate irrelevant patients.

Table 6.4 **Effect of adding randomly sampled ‘irrelevant’ patients to the match runs when training the ranking model**, mAP (higher is better) of two RankNets: one trained with with no added patients to the match runs (Train Q+ 0) and the other trained with a random number between 0 and 200 of randomly added irrelevant patients to each match run (Train Q + 200). Then tested with a random number between 1 and X of randomly added patients to each match run of the test set for X = 10, 100, 200, 500. Using 32 clusters.

	Train Q +0	Train Q +200
Test Q +10	0.784	0.929
Test Q +100	0.437	0.723
Test Q +200	0.378	0.602
Test Q +500	0.195	0.476

6.2.5 Clinical Interpretation of Results

The theoretical MELD-Na policy uses the patient’s probability of dying to rank patients. This only depends on \mathbf{X} , the covariates of the patient and not on $\mathcal{X}(t)$ the other patients in the waitlist or on the organ \mathbf{O}_j . Denote \mathcal{M} the ranking function of the theoretical MELD-Na policy, we have that

$$\mathcal{M}(\{\mathbf{X}_i, \mathbf{X}_m, \mathbf{X}_l\}, k_1) = \mathcal{M}(\{\mathbf{X}_i, \mathbf{X}_m, \mathbf{X}_l\}, k_2) \quad (6.6)$$

However, when analysing the results of the ranking model, we notice this does not hold for the ranking function \mathcal{R} that is learnt. Indeed, the same set of patients will not be ranking in the same order for two organs of different type.

$$\mathcal{R}(\{\mathbf{X}_i, \mathbf{X}_m, \mathbf{X}_l\}, k_1) \neq \mathcal{R}(\{\mathbf{X}_i, \mathbf{X}_m, \mathbf{X}_l\}, k_2) \quad (6.7)$$

This shows that the theoretical policy is affected by the variations such as the donor criteria specified by the clinician and thus that the resulting implemented policy may not allocate organs in the same manner.

In this chapter, we introduced evaluation metrics and baselines for the timing and ranking models. We found that for the timing model, a smaller amount of clusters gives a better performance (Table 6.1) whereas for the ranking model a larger number gives higher NDCG (Table 6.3). The clinical interpretation of the outputs of the timing model shows the classifier can capture elements of patient to donor compatibility while the regressor can capture specific aspects of the allocation policy studied. Concerning the performance of the regressor, we showed the benefit of adding randomly samples patients to extend the match run and

through the clinical interpretation of the returned rankings, we were able to conclude the ranking model can implicitly capture real-life variations such as clinicians' specifications of donor criteria. The good performance results and coherent clinicians interpretation of each component are promising and explain the high performance of the Organ Queue framework in the previous chapter.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

This dissertation identified a gap in clinical practice due to the discrepancy between theoretical and implemented policies and the lack of tools to analyse the latter. Our goal was to provide a policy agnostic framework to learn and model *implemented* allocation policies, which can be used to assess whether the implemented policy reaches its objectives, or to conduct further analyses to refine the current policy. Based on research and discussions with clinicians, to understand the key components and challenges of modelling an organ transplant system, we formulated the problem mathematically and proposed our first solution. Our solution is a framework which integrates a queuing system with (i) an unsupervised clustering algorithm to cluster organs into organ types and constructs priority queues associated with each organ type, (ii) a timing model to assign incoming patients to queues, and (iii) a learning to rank algorithm to place these patients within the queues.

Our case study — on the MELD-Na allocation policy for liver transplantation — showed the utility of each component of the framework to accurately model an implemented policy with which we can conduct meaningful analyses. Specifically, we were able to confirm that MELD-Na places emphasis on hyponatremia patients when allocating organs. Moreover, by reaching similar performance results with different allocation policies and providing clinical interpretation of our results in accordance with the theoretical policy, we showed our model is policy agnostic and able to learn any implemented policy from data. These initial experiments using UNOS data hint at the potential of our framework.

However, the framework does have some shortcomings. Indeed, the incomplete match runs and list of offers in the observed dataset cause certain aspects of the transplant system—such as the waitlist’s influence on the timing of offers to not be modelled by our framework. Moreover, the variations which create the discrepancy between the theoretical and implemented policies are only modelled implicitly through the data.

7.2 Future Work

To address the issue of large amounts of unobserved data notably in the list of offers received by a patient we could consider performing counterfactual analysis to predict which other queues the patient would have received an offer from had they not been removed from the waitlist. However, even with these estimates, it is not certain a neural network approach that considers both the patients and the waitlist would give satisfactory results due to the limited amount of patients we have for each policy.

Moving forward, we wish to extend and test our framework on other datasets. Our model was tested on various allocation policies and was shown to be policy agnostic. However, different datasets from different countries or different organs may record different covariates for the patient and the donors, these changes may affect the performance of our model. It is important to note that the UNOS data is one of the largest in the world, then all other datasets will have fewer patients and match runs to train on. This is not an issue for the ranking model as it is trained using pairs of patients within the same extended match runs. However, for the timing model, the use of simpler machine learning architectures such as random forests rather than complex models such as neural networks are essential to be able to expand to smaller datasets. Moreover, these models have the added benefit of remaining interpretable, a feature that is often desirable in a medical context.

In addition, we would like to more explicitly incorporate variations which modify the theoretical policy such as clinicians’ specification of donor criteria and their decision to accept/reject an offer. Currently, this is done implicitly through the data, but considering it has such a large influence on wait times and how the proposed policy is implemented, modelling this more explicitly could yield better results. For instance, an approach could be to learn a representation of clinicians’ decision-making behaviour when accepting/rejecting an organ offers with bayesian methods as in [Hüyük et al. \(2020\)](#). Modelling such decisions explicitly could allow to accurately predict which patient in the predicted match run will

accept an organ offer and thus permit the estimation of patient wait times. Furthermore, this information could be used across policies, which would reduce the requirement for data.

It is our hope that this dissertation will inspire more methods to focus on evaluating implemented policies to help inform the creation of new theoretical allocation policies with the goal of developing a more equitable, effective, and efficient allocation of organs.

References

- Abellán, J., Armero, C., Conesa, D., Pérez-Panadés, J., Martínez-Beneito, M., Zurriaga, O., García-Blasco, M., and Vanaclocha, H. (2006). Analysis of the renal transplant waiting list in the país valencià (spain). *Statistics in medicine*, 25(2):345–358.
- Barone, M., Avolio, A. W., Di Leo, A., Burra, P., and Francavilla, A. (2008). Abo blood group-related waiting list disparities in liver transplant candidates: effect of the meld adoption. *Transplantation*, 85(6):844–849.
- Berrevoets, J., Alaa, A., Qian, Z., Jordon, J., Gimson, A. E., and Van Der Schaar, M. (2021). Learning queueing policies for organ transplantation allocation using interpretable counterfactual survival analysis. In *International Conference on Machine Learning*, pages 792–802. PMLR.
- Berrevoets, J., Jordon, J., Bica, I., Gimson, A., and van der Schaar, M. (2020). Organite: Optimal transplant donor organ offering using an individual treatment effect. <https://proceedings.neurips.cc/paper/2020>, 33.
- Biggins, S. W., Kim, W. R., Terrault, N. A., Saab, S., Balan, V., Schiano, T., Benson, J., Therneau, T., Kremers, W., Wiesner, R., et al. (2006). Evidence-based incorporation of serum sodium concentration into meld. *Gastroenterology*, 130(6):1652–1660.
- Biggins, S. W., Rodriguez, H. J., Bacchetti, P., Bass, N. M., Roberts, J. P., and Terrault, N. A. (2005). Serum sodium predicts mortality in patients listed for liver transplantation. *Hepatology*, 41(1):32–39.
- Boulware, L., Troll, M., Wang, N.-Y., and Powe, N. (2007). Perceived transparency and fairness of the organ allocation system and willingness to donate organs: a national study. *American journal of transplantation*, 7(7):1778–1787.
- Boxma, O. J., David, I., Perry, D., and Stadge, W. (2011). A new look at organ transplantation models and double matching queues. *Probability in the Engineering and Informational Sciences*, 25(2):135–155.
- Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., and Shah, R. (1993). Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.
- Burges, C., Ragno, R., and Le, Q. (2006). Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19:193–200.

- Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96.
- Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.
- Cecka, J. M. (2000). The unos scientific renal transplant registry–2000. *Clinical transplants*, pages 1–18.
- Colvin, M., Smith, J., Ahn, Y., Skeans, M., Messick, E., Goff, R., Bradbrook, K., Foutz, J., Israni, A., Snyder, J., et al. (2021). Optn/srtr 2019 annual data report: heart. *American Journal of Transplantation*, 21:356–440.
- Cossock, D. and Zhang, T. (2006). Subset ranking using regression. In *International Conference on Computational Learning Theory*, pages 605–619. Springer.
- Cragg, J. G. (1971). Some statistical models for limited dependent variables with application to the demand for durable goods. *Econometrica: Journal of the Econometric Society*, pages 829–844.
- Cramer, K., Singer, Y., et al. (2001). Pranking with ranking. In *Nips*, volume 1, pages 641–647.
- Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.
- Drekic, S., Stanford, D. A., Woolford, D. G., and McAlister, V. C. (2015). A model for deceased-donor transplant queue waiting times. *Queueing Systems*, 79(1):87–115.
- Eick, C. F., Zeidat, N., and Zhao, Z. (2004). Supervised clustering-algorithms and benefits. In *16Th IEEE international conference on tools with artificial intelligence*, pages 774–776. IEEE.
- Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.
- Finley, T. and Joachims, T. (2005). Supervised clustering with support vector machines. In *Proceedings of the 22nd international conference on Machine learning*, pages 217–224.
- Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969.
- Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.
- Goyal, N. and Weiner, D. E. (2018). The affordable care act, kidney transplant access, and kidney disease care in the united states.

- Green, L. (2006). Queueing analysis in healthcare. In *Patient flow: reducing delay in healthcare delivery*, pages 281–307. Springer.
- Haldar, P., Pavord, I. D., Shaw, D. E., Berry, M. A., Thomas, M., Brightling, C. E., Wardlaw, A. J., and Green, R. H. (2008). Cluster analysis and clinical asthma phenotypes. *American journal of respiratory and critical care medicine*, 178(3):218–224.
- Hall, R. W. (1991). *Queueing methods: for services and manufacturing*. Pearson College Division.
- Hart, A., Lentine, K., Smith, J., Miller, J., Skeans, M., Prentice, M., Robinson, A., Foutz, J., Booker, S., Israni, A., et al. (2021). Optn/srtr 2019 annual data report: kidney. *American Journal of Transplantation*, 21:21–137.
- Hartigan, J. A. (1975). *Clustering algorithms*. John Wiley & Sons, Inc.
- Herbrich, R., Graepel, T., and Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. *Advances in large margin classifiers*, 88(2):115–132.
- Heuman, D. M., Abou-Assi, S. G., Habib, A., Williams, L. M., Stravitz, R. T., Sanyal, A. J., Fisher, R. A., and Mihas, A. A. (2004). Persistent ascites and low serum sodium identify patients with cirrhosis and low meld scores who are at high risk for early death. *Hepatology*, 40(4):802–810.
- Hu, M.-C., Pavlicova, M., and Nunes, E. V. (2011). Zero-inflated and hurdle models of count data with extra zeros: examples from an hiv-risk reduction intervention trial. *The American journal of drug and alcohol abuse*, 37(5):367–375.
- Hüyük, A., Jarrett, D., Tekin, C., and van der Schaar, M. (2020). Explaining by imitating: Understanding decisions by interpretable policy learning. In *International Conference on Learning Representations*.
- Järvelin, K. and Kekäläinen, J. (2017). Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, volume 51, pages 243–250. ACM New York, NY, USA.
- Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.
- Kalra, A. and Biggins, S. W. (2018). New paradigms for organ allocation and distribution in liver transplantation. *Current opinion in gastroenterology*, 34(3):123–131.
- Kandaswamy, R., Stock, P., Miller, J., Skeans, M., White, J., Wainright, J., Kyaw, N., Niederhaus, S., Israni, A., and Snyder, J. (2021). Optn/srtr 2019 annual data report: Pancreas. *American Journal of Transplantation*, 21:138–207.
- Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons.
- Kim, W. R., Biggins, S. W., Kremers, W. K., Wiesner, R. H., Kamath, P. S., Benson, J. T., Edwards, E., and Therneau, T. M. (2008). Hyponatremia and mortality among patients on the liver-transplant waiting list. *New England Journal of Medicine*, 359(10):1018–1026.

- Köppel, M., Segner, A., Wagener, M., Pensel, L., Karwath, A., and Kramer, S. (2019). Pairwise learning to rank by neural networks revisited: Reconstruction, theoretical analysis and practical performance. *arXiv preprint arXiv:1909.02768*.
- Kumar, R. and Vassilvitskii, S. (2010). Generalized distances between rankings. In *Proceedings of the 19th international conference on World wide web*, pages 571–580.
- Kwong, A., Kim, W., Lake, J., Smith, J., Schladt, D., Skeans, M., Noreen, S., Foutz, J., Booker, S., Cafarella, M., et al. (2021). Optn/srtr 2019 annual data report: Liver. *American Journal of Transplantation*, 21:208–315.
- Lan, Y., Zhu, Y., Guo, J., Niu, S., and Cheng, X. (2014). Position-aware listmle: A sequential learning process for ranking. In *UAI*, pages 449–458.
- Leise, M. D., Kim, W. R., Kremers, W. K., Larson, J. J., Benson, J. T., and Therneau, T. M. (2011). A revised model for end-stage liver disease optimizes prediction of mortality among patients awaiting liver transplantation. *Gastroenterology*, 140(7):1952–1960.
- Levenshtein, V. I. et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.
- Li, H. (2011). A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862.
- Li, P., Wu, Q., and Burges, C. (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. *Advances in neural information processing systems*, 20:897–904.
- Liao, M., Li, Y., Kianifard, F., Obi, E., and Arcona, S. (2016). Cluster analysis and its application to healthcare claims data: a study of end-stage renal disease patients who initiated hemodialysis. *BMC nephrology*, 17(1):1–14.
- Liu, T.-Y. (2011). Learning to rank for information retrieval.
- Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.
- Malinchoc, M., Kamath, P. S., Gordon, F. D., Peine, C. J., Rank, J., and Ter Borg, P. C. (2000). A model to predict poor survival in patients undergoing transjugular intrahepatic portosystemic shunts. *Hepatology*, 31(4):864–871.
- Neuberger, J., Gimson, A., Davies, M., Akyol, M., O’Grady, J., Burroughs, A., Hudson, M., Blood, U., et al. (2008). Selection of patients for liver transplantation and allocation of donated livers in the uk. *Gut*, 57(2):252–257.
- OPTN (2021). Organ procurement and transplantation network: Organ donation and transplantation can save lives.
- Procurment, T. O. and Network, T. (2011). Guidance for abo subtyping organ donors for blood groups a and ab.
- Qin, T., Liu, T.-Y., Xu, J., and Li, H. (2010). Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374.

- Rose, C. E., Martin, S. W., Wannemuehler, K. A., and Plikaytis, B. D. (2006). On the use of zero-inflated and hurdle models for modeling vaccine adverse event count data. *Journal of biopharmaceutical statistics*, 16(4):463–481.
- Sanfilippo, F. P., Vaughn, W. K., Peters, T. G., Shield, C. F., Adams, P. L., Lorber, M. I., and Williams, G. M. (1992). Factors affecting the waiting time of cadaveric kidney transplant candidates in the united states. *JAMA*, 267(2):247–252.
- Stanford, D. A., Lee, J. M., Chandok, N., and McAlister, V. (2014). A queuing model to address waiting time inconsistency in solid-organ transplantation. *Operations Research for Health Care*, 3(1):40–45.
- Tan, M., Xia, T., Guo, L., and Wang, S. (2013). Direct optimization of ranking measures for learning to rank models. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 856–864.
- Taylor, M., Guiver, J., Robertson, S., and Minka, T. (2008). Sofrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 77–86.
- Thompson, D., Waisanen, L., Wolfe, R., Merion, R. M., McCullough, K., and Rodgers, A. (2004). Simulating the allocation of organs for transplantation. *Health care management science*, 7(4):331–338.
- Tosto, G., Monsell, S. E., Hawes, S. E., Bruno, G., and Mayeux, R. (2016). Progression of extrapyramidal signs in alzheimer’s disease: clinical and neuropathological correlates. *Journal of Alzheimer’s Disease*, 49(4):1085–1093.
- Trivedi, J. R., Ising, M., Fox, M. P., Cannon, R. M., Van Berkel, V. H., and Slaughter, M. S. (2018). Solid-organ transplantation and the affordable care act: accessibility and outcomes. *The American Surgeon*, 84(12):1894–1899.
- Wu, Q., Burges, C. J., Svore, K. M., and Gao, J. (2010). Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270.
- Xu, C., Alaa, A., Bica, I., Ershoff, B., Cannesson, M., and Schaar, M. (2021). Learning matching representations for individualized organ transplantation allocation. In *International Conference on Artificial Intelligence and Statistics*, pages 2134–2142. PMLR.
- Yoon, J., Alaa, A., Cadeiras, M., and Van Der Schaar, M. (2017). Personalized donor-recipient matching for organ transplantation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.
- Yue, Y., Finley, T., Radlinski, F., and Joachims, T. (2007). A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278.
- Zenios, S. A. (1999). Modeling the transplant waiting list: A queuing model with renegeing. *Queueing systems*, 31(3):239–251.

Appendix A

Supplementary Material

A.1 Original Timing Model

In this section, we present the timing model we had originally designed. For the same reasons detailed in chapter 4, this model must predict both the probability of response (if a patient should be added to a given queue) and the value of the response (if a patient is to be added, when that should be). As stated in challenge (d), when a patient receives an offer is dependent of which other patients were in the waitlist. Then, in addition to the patient covariates, the wait list should also be an input to the model to predict T_n^k .

The network developed (Figure A.1) is inspired by siamese network architecture (Bromley et al., 1993) in that it takes two separate inputs. The patient's covariates \mathbf{X} and the covariate of the patients in the waitlist at the time of the patient's arrival, $\mathcal{X}(t)$. Both these inputs are passed through separate ϕ networks used to reduce their dimensionality. Then the outputs of these networks, $\phi(\mathbf{X}), \phi(\mathcal{X}(t))$ are concatenated and passed through a neural network denoted L . The aim of this neural network is to capture the complex interactions between the patient and the wait list and create a latent representation of them denoted $L(\phi)$. Simply concatenating the two vectors would not allow to learn such complex interactions. The resulting vector $L(\phi)$ is fed as input to a classification network and to a regression network denoted respectively \mathbf{C} and \mathbf{R} . The classification and regression are learnt within the same network for multiple reasons, firstly as it is more efficient computation wise, second as we want the same learnt interactions to affect the classification and the regression and lastly because this allows one task to be a regulariser for the other as learning them jointly and having a loss on the two allows to create a compromise between the performance of the two networks.

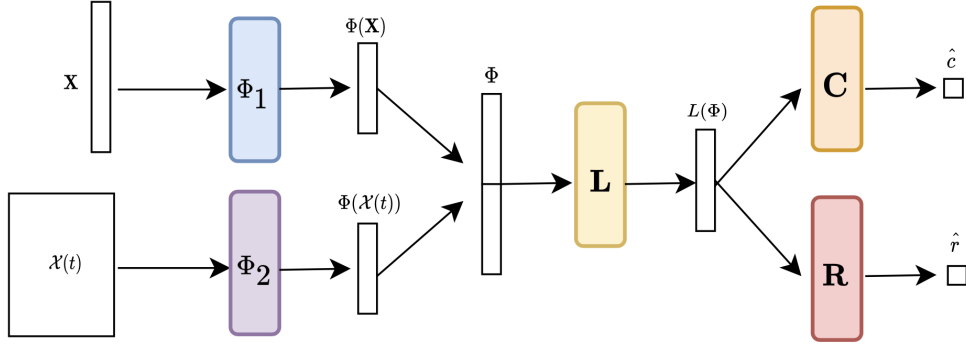


Fig. A.1 Initial Timing Model Architecture

The loss used to train the network is the sum of the loss from the regressor and the loss from the classifier. Denote y the target values of T_n^k , and g the binarized version of the target ($g = \mathbb{1}_{y>0}$). Denote \hat{r} the prediction of the regressor and \hat{c} the prediction of the classifier. We use the mean squared error (MSE) and the binary cross entropy (BCE) loss to compute the networks loss as follows:

$$L = MSE(\hat{r}, y) + BCE(\hat{c}, g) \quad (\text{A.1})$$

A.2 Additional Results

Table A.1 **Results of organ allocation for various number of clusters.** Jaccard index (higher is better) and the Levenstein distance ratio (lower is better) between the real match run \mathcal{R} and the predicted match run \mathcal{Q} . These results complete Table 5.1

Our model						
	8 clusters	12 clusters	20 clusters	32 clusters	36 clusters	48 clusters
Jac \uparrow	0.037 (0.061)	0.043 (0.058)	0.048 (0.073)	0.071 (0.097)	0.068 (0.122)	0.084 (0.166)
Lev \downarrow	0.974 (0.030)	0.969 (0.034)	0.963 (0.061)	0.941 (0.088)	0.941 (0.122)	0.927 (0.115)
Timing Ablation						
	8 clusters	12 clusters	20 clusters	32 clusters	36 clusters	48 clusters
Jac \uparrow	0.038 (0.077)	0.038 (0.079)	0.035 (0.067)	0.033 (0.069)	0.030 (0.065)	0.042 (0.082)
Lev \downarrow	0.979 (0.022)	0.977 (0.033)	0.987 (0.023)	0.981 (0.032)	0.983 (0.026)	0.975 (0.032)
Ranking Ablation						
	8 cluster	12 clusters	20 clusters	32 clusters	36 clusters	48 clusters
Jac \uparrow	0.043 (0.067)	0.046 (0.065)	0.040 (0.093)	0.062 (0.087)	0.061 (0.092)	0.082 (0.181)
Lev \downarrow	0.970 (0.032)	0.965 (0.037)	0.972 (0.044)	0.946 (0.081)	0.947 (0.088)	0.918 (0.181)
Our model - mixed						
	8 clusters	12 clusters	20 clusters	32 clusters	36 clusters	48 clusters
Jac \uparrow	0.041 (0.04)	0.058 (0.082)	0.061 (0.115)	0.086 (0.143)	0.080 (0.097)	0.099 (0.180)
Lev \downarrow	0.972 (0.040)	0.959 (0.037)	0.952 (0.102)	0.932 (0.137)	0.935 (0.112)	0.913 (0.173)

Table A.2 **Results of the timing model under different architectures.** Leaky RMSE (lower is better). Siamese network is the neural net described in Section A.1 which uses \mathbf{X}_n and $\mathcal{X}(t)$ whereas the hurdle models are based solely on \mathbf{X}_n . RFC/RFR = Random Forest Classifier/Regressor, GBC/GBR = Gradient Boosting Classifier/Regressor, LoR/LiR = Logistic/Linear Regression, SVM = Support Vector Machine Classifier. All these have undergone hyperparameter optimisation.

	Hurdle Model						Siamese Net
	RFC+RFR	GBC + GBR	LoR + LiR	SVM + RFR	RFC + GBR	LoR + RFR	
32 clusters	32.98 (14.75)	33.1 (15.08)	36.32 (15.56)	34.69 (13.04)	34.61 (14.88)	34.59 (15.86)	37.04 (17.89)
8 clusters	30.58 (14.03)	30.67 (13.92)	35.18 (16.22)	34.13 (13.85)	32.52 (15.97)	33.67 (15.24)	37.63 (16.41)

A.3 Evaluation Metrics

Benefit of NDCG vs Kendall's τ or Spearman's ρ Table A.3 shows Spearman's ρ and Kendall's τ are unable to take into account element relevance and positional information. Indeed Kendall and Spearman coefficient are the same for both rankings of Table A.3, while the NDCG is higher for the first ranking which places the relevant patient **X** higher.

Table A.3 **Illustrative example to compare NDCG, Spearman's ρ and Kendall's τ** . Consider 3 patients $\{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$ patient **X** is relevant for the query while patient **Y** and **Z** are equally irrelevant.

Ranking	NDCG	Kendall's τ	Spearman's ρ
$\mathbf{Y} \triangleright \mathbf{X} \triangleright \mathbf{Z}$	0.630	-0.499	-0.5
$\mathbf{Y} \triangleright \mathbf{Z} \triangleright \mathbf{X}$	0.5	-0.499	-0.5