Timing Objectives in Dynamic Kidney Exchange

A thesis presented
by

Anson Kahng

To
Computer Science
in partial fulfillment of the honors requirements
for the degree of
Bachelor of Arts
Harvard College
Cambridge, Massachusetts

April 1, 2016

Acknowledgements

I would like to thank my advisor, Prof. David Parkes, for his support and guidance. I could not have asked for a better advisor, and I learned a lot about the research process through this endeavor.

I would also like to thank soon-to-be Dr. John Dickerson for his help with the FutureMatch framework, helping me understand dynamic kidney exchange in general, and responding to far too many of my frantic last-minute requests.

Last but not least, I would like to thank my family and friends for their support and friendship throughout this process.

Contents

Chapter 1

Introduction

Chronic kidney disease is a very serious health issue that eventually leads to loss of organ function and kidney failure. The treatment options at the point of kidney failure are either continual dialysis or a kidney transplant. The preferred treatment for kidney failure is a transplant, as dialysis is not only less effective, but also requires more time and money. In the United States alone, there are over 100,000 patients on a waiting list for kidney transplants, and demand is increasing at a rate that far outstrips supply. Patients either receive a transplant from a living or deceased donor. Approximately two-thirds of transplants come from cadavers, and one-third come from live donors. Because organ sales are illegal in the United States, it is impossible to use traditional market mechanisms in order to incentivize people to donate more kidneys. In addition, kidney exchange is a very highly sensitized process, and many *compatibility* issues such as blood type or tissue type make finding matches for transplantation quite difficult in practice.

Kidney exchange is a framework that allows patients to enter a pool with a willing but incompatible donor in order to potentially find other patient-donor pairs with whom to arrange a mutual exchange. For example, if pairs (P_1, D_1) and (P_2, D_2) enter the exchange and (P_1, D_2) and (P_2, D_1) are compatible, then the two pairs can arrange a *swap*, i.e., a two-cycle, that results in each patient receiving a life-saving organ.

Traditionally, this problem is represented as a directed graph in which the vertices are patient-donor pairs and edges represent compatibility matchings. In particular, if there is an edge from vertex i to vertex j, this means that the patient in pair j is compatible with the donor in pair i, and we can think of the direction of the edge as the direction in which a kidney will be transplanted. In theory, finding a cycle of any length in a kidney exchange graph will result in all patients in the cycle receiving a kidney, but there are logistical constraints that cap cycles at a certain length in practice. This is because all surgeries in a large swap must be performed simultaneously in order to ensure no donor can back out after his or her patient has received a kidney, thus depriving patients further along in the swap of transplants and leaving at least one remaining patient (the first transplant in the cycle) without a donor.

Altruistic donors in the realm of kidney exchange are donors who enter the exchange without a matching patient. This allows the formation of long *chains* starting with this altruistic donor because now at each step there is never a patient still needing a kidney who has lost her donor.

Much of the early work in kidney exchange treated the problem as fundamentally static, and focused on finding optimal matches assuming no entrances and exits from the pool. While this is an NP-complete problem in its own right, it has been solved in practice [4] and people have shifted their attention to dynamic kidney exchange.

Kidney exchange is actually a *dynamic* problem. There are two fundamental types of dynamics in kidney exchange: patient-donor pairs enter and leave the exchange over time, and blood-compatible matches prescribed by matching algorithms can fail at later stages, before actual transplantation, for other compatibility issues that require more extensive tests to diagnose. [1] Throughout this paper, mentions of *dynamic kidney exchange* refer to the entrance and exit of pairs into and out of the pool over time; we refer to the second type of dynamics as *failure-aware kidney exchange.* Although we focus primarily on dynamic kidney exchange, we do keep in mind possible extensions of our framework to account for the second; for example, much recent work has been done on trying to either maximize the

[1] Post-match failure can also occur because of death, a donor getting cold feet, or logistical issues. In general, very few algorithmically matched patients actually receive transplants. [5]

expected number of matches in a failure-aware setting [11] or trying to optimally allocate a constant number of additional tests per vertex in order to minimize the number of post-match failures [8].

In particular, the state-of-the-art UNOS kidney exchange deals with dynamics of the first kind by clearing the exchange in batches, which means that it runs a matching algorithm in fixed intervals.

Current kidney exchange mechanisms clear the exchange in batches in order to maximize an offline medically-prescribed objective. This objective can range from a simple maximum cardinality metric to more complex functions that involve various conceptions of fairness. I discuss examples of various offline objective functions more in Chapter 3. In general, this framework involves a step of learning how to maximize a certain objective based on historical data, and the final output is a batch-clearing algorithm that specifies how to output matchings in each interval in order to maximize the original objective.

In particular, given a specific offline objective function, work by Sandholm, Procaccia, and Dickerson [9] has suggested that each vertex type has a different 'future value' that weights its future usefulness. In other words, it is sometimes wise to avoid matching as many patients as possible during one timestep in order to preserve vertices in order to match with more highly sensitized patients in the future. To this end, they have developed a framework, FutureMatch, that uses two stages of machine learning to learn weights on edges and vertices in kidney exchange graphs that represent satisfying a medical objective and the future value of vertices (patient-donor pairs), respectively. I discuss the specifics of this process in Chapter 3.

However, this means that the exchange can take a long time to match certain people because it deems them potentially more 'useful' in the future. Medical practitioners are not always willing to wait as long as the matching algorithm may ideally want to because many patients' health deteriorates increasingly rapidly the longer they wait for a transplant, and it is more dangerous to operate on less healthy patients. This leads to friction between the generated matches and physicians' wishes. As of yet, there is no timing objective in state-of-the-art kidney exchange models that could potentially balance these two dynamics.

In some stylized models, notably those by Anderson [3] and Akbarpour [2], conceptions of timing and 'criticality' — essentially, closeness to death — are explored. In particular, Akbarpour's model suggests that, given knowledge of when patients will leave the exchange, presumably due to death, waiting for the market to thicken and only matching critical patients — those who will perish in the next time period — leads to superior performance. However, Anderson's model, which purely minimizes the total amount of people spend in the exchange, shows that a purely greedy batching algorithm performs optimally. Therefore, there is no clear incorporation of timing objectives in kidney exchange mechanisms designed to achieve specific objectives.

My thesis is about finding a new way to make a tradeoff between temporal (i.e., myopic) considerations and optimizing an offline medical objective.

1.1 Main Results

In this paper, I adapt a state-of-the-art dynamic kidney exchange framework to account for timing considerations. In order to do this, I introduce an additional timing variable in the state of the art kidney exchange framework in order to examine timing-aware extensions of offline medical objectives.

I develop algorithms that learn an objective corresponding to various weighted averages of the timing and offline medical objectives and examine their performance with respect to both the total number of matches and the average amount of time patients spend in the exchange. I then examine the tradeoff between the performance of these algorithms with respect to both timing and match quality in order to determine whether introducing timing considerations in these objective functions results in more effective matching strategies.

The results suggest that, at low levels of death in the kidney graph (i.e., patients do not die quickly), timing-aware algorithms generally achieve comparable results on the medical objective while reducing the average amount of time people spend in the exchange. However, at higher levels of death, the dynamics get much noisier and there are no clear trends.

1.2 Outline

In Chapter 2, I introduce two stylized kidney exchange models. These models provide theoretical intuition behind the difficulties of dynamic kidney exchange and, notably, motivate a disucssion of fairness with respect to time. In Chapter 3, I rigorously define the kidney exchange problem and introduce the FutureMatch framework [13]. In Chapter 4, I extend a version of FutureMatch to account for timing objectives. In Chapter 5, I present my empirical results. I then conclude in Chapter 6.

Chapter 2

Theoretical Kidney Exchange Models

In order to gain intuition about the potential tradeoffs between timing and medical objectives, I first approached this problem through the lens of two toy models and a corresponding simulation. Below, I discuss the relevant models and describe how I hybridized them in order to observe a tradeoff between timing and medical objectives for some range of death probabilities.

2.1 Kidney Exchange Models

Theoretical models of kidney exchange offer useful intuition about the dynamics and the tradeoffs between different objectives inherent in the matching process. There have been many theoretical models proposed over time, and there is no clear gold standard model, but I will focus on two toy models that highlight the tradeoff between timing and other objectives.

2.1.1 Erdős-Rényi Graphs

The simplest random graph model used to represent kidney exchange graphs is the Erdős-Rényi model [15]. This model is parameterized by the total number of nodes in the graph n and a probability p of an edge existing between any pair of vertices in the graph $G(n, p)$. Note that we can represent the kidney exchange case where exchanges are limited to swaps as an undirected Erdős-Rényi graph. If we introduce three-cycles and altruistic chains, both of which require directed edges, we can model it with a directed Erdős-Rényi graph.

For notational purposes, let $ER_u(n, p)$ be an undirected Erdős-Rényi graph with n nodes where every two nodes form an edge with probability p. Let $ER_d(n, p)$ be a directed Erdős-Rényi graph with n nodes where every two nodes form a directed edge with probability p. We can also define closely related notions of $ER_u(n, M)$ and $ER_d(n, M)$, where in this case M undirected or directed edges are chosen out of all possible edges in the graph. The two models are essentially equivalent and have been used interchangeably in the literature [3]. Both models discussed below make use of Erdős-Rényi graphs.

Although Erdős-Rényi graphs are simple to implement and easy to rigorously analyze, they are often too idealistic for accurate simulations. Most generative models of kidney graphs contain additional features that allow for richer generation of realistic kidney graphs, but Erdős-Rényi graphs are still very useful for building intuition about the dynamics underlying kidney exchange.

2.1.2 Akbarpour 2014

Akbarpour proposed a stochastic continuous-time model of kidney exchange in order to capture the dynamics of entrances and exits from the pool over time. Crucially, Akbarpour only allows swaps (i.e., two-cycles) in his model. This is for simplicity's sake, as dealing with three-cycles and altruistic chains makes the mathematical foundation of the model much harder to deal with. Agents (patient-vertex pairs) are drawn from an underlying distribution of blood types $\gamma \sim F_{ABO}$, where F_{ABO} is the blood-type distribution in the population, and arrive according to a Poisson process parameterized by a rate m, $T \sim \text{Poisson}(\alpha)$.

Per the Erdős-Rényi model, each arriving agent is compatible with any other agent with probability p. Each agent then becomes *critical* according to another, independent, Poisson process with rate λ. Critical agents leave the market immediately, and the last possible timestep at which they can be matched is the one where they are declared critical. This means that if an agent a enters the market at time t_0, she becomes critical at some time $t_0 + X$, $X \sim \exp(\lambda)$. If a critical agent is not immediately matched, she perishes.

Note that this means that any agent a that enters the pool at time t_0 must leave at some time t_1 such that $t_0 \leq t_1 \leq t_0 + X$. We define the sojourn of a as $s(a) = t_1 - t_0$. From this, we define the utility of a as follows:

$$u(a) := \begin{cases} e^{-rs(a)} & \text{if } a \text{ is matched} \\ 0 & \text{otherwise.} \end{cases}$$

There are thus three ways an agent a can leave the market: she could be matched with another agent b; she could become critical and get matched immediately; or she could become critical, remain unmatched, and perish. Akbarpour then explores the tradeoffs between greedily matching agents as soon as they enter the exchange and waiting until agents become critical in order to allow the market to thicken. Throughout his paper, he looks at the special case where the cost of waiting r is negligible compared to the cost of leaving the market unmatched, which means that the goal of the planner is simply to match the maximum number of agents because the utility to each agent of being matched is 1. The updated utility function for being matched then becomes

$$u(a) := \begin{cases} 1 & \text{if } a \text{ is matched} \\ 0 & \text{otherwise.} \end{cases}$$

From here, we define the social welfare of an online algorithm ALG as the normalized expected sum of the utility of all agents in the interval $[0, T]$:

$$W(ALG) := \mathbb{E}\left[\frac{1}{mT}\sum_{a \in A} u(a)\right].$$

Note that this corresponds exactly to maximizing the number of agents matched over a certain time interval. We can also define a complementary measure of loss as follows. Let

$L(ALG)$ define the loss function under a specific algorithm ALG at time T as the ratio of the expected number of perished agents to the expected size of the agent pool A, or

$$L(ALG) := \frac{\mathbb{E}[|A - ALG(T) - A_T|]}{\mathbb{E}[|A|]} = \frac{\mathbb{E}[|A - ALG(T) - A_T|]}{mT},$$

where m is the rate at which agents arrive according to a Poisson process, A_T is the set of all agents that enter the pool at time T, A is the set of all agents at time T, and $ALG(T)$ is the set of all matched agents by time T.

From these definitions, we then try to maximize the welfare (or, equivalently, minimize the loss) for each online algorithm. We now discuss the different types of online algorithms Akbarpour considered.

Let OPT^c represent the optimal algorithm given the knowledge of which agents will become critical at any time. Akbarpour observed that, when waiting time is negligible, OPT^c will only clear matches where at least one of the vertices is critical (otherwise, it can wait to clear them and be weakly better off), and it will match all possible critical nodes at all times. Therefore, OPT^c is inherently patient and waits until people become critical in order to match them.

He then considers OPT, the optimal algorithm but without any knowledge of the criticality of agents. Because this algorithm is not omniscient, its performance is much more constrained and, because we have no special information about the imminent danger of any agent, the best we can do is approximately the myopically greedy approach (i.e., greedily match agents as soon as they enter the exchange).

To formalize this, let ALG represent any online algorithm that does not know the set of critical agents at each time step and let OMN be the maximum omniscient matching. Akbarpour shows that

$$L(ALG) \geq L(OPT) \geq L(OPT^c) \geq L(OMN).$$

In order to further examine the tradeoffs between critical-aware (patient) and critical-unaware (greedy) matching, he proposes three algorithms:

- Greedy algorithm: When a new agent a enters the market, match her with an arbitrary neighbor. If she does not have any compatible neighbors, do nothing.
- Patient algorithm: If an agent a becomes critical at time t, match her uniformly at random with a neighbor. If she has no compatible neighbors, do nothing and let her perish.
- Patient(α) algorithm: Assign independent exponential clocks with rate $1/\alpha$ to each agent. If an agent becomes critical or her clock 'ticks' at time t, match her uniformly at random with a neighbor. If she has no compatible neighbors, do nothing and treat the agent as if she has perished and never match her again.

Given knowledge of agents' criticality, the Patient algorithm outperforms (i.e., results in more matches) than the Greedy algorithm. In worlds with small waiting costs, the Greedy algorithm results in a perpetually thin market, whereas the Patient algorithm results in thicker, Erdős-Rényi [[formatting]] graphs with average degree d (Proposition 4.1). This increased thickness allows the market to better react to critical nodes, resulting in more matches overall. In particular, exponentially (in d) fewer agents perish under the Patient algorithm than the Greedy algorithm. For $d \geq 2$ and as $T, m \to \infty$,

$$L(Greedy) \geq \frac{1}{2d+1}$$
$$L(Patient) \leq \frac{1}{2} \cdot e^{-d/2}$$

and therefore we have

$$L(Patient(\alpha)) \leq (d+1) \cdot e^{-d/2} \cdot L(Greedy).$$

Relaxing the assumption that the set of critical agents can be accurately elicited yields additional relative bounds on the performance of the Patient and Greedy algorithms. Per Theorem 3.10, for $d \geq 2$, and as $T, m \to \infty$,

$$\frac{1}{2d+1} \leq L(OPT) \leq L(Greedy) \leq \frac{\log(2)}{d}$$
$$\frac{e^{-d}}{d+1} \leq L(OMN) \leq L(Patient) \leq \frac{1}{2} \cdot e^{-d/2}.$$

The Patient(α) algorithm is interesting because it not only represents a natural interpolation of the Patient and Greedy algorithms, but also addresses timing concerns with the Patient algorithm. In practice, matching people only when their health deteriorates to a critical point is not feasible or practical. By introducing an exponential clock to speed up the matching process, Akbarpour demonstrates that as long as the clocks do not tick too quickly (i.e., α is not 'too small'), the Patient(α) algorithm still significantly outperforms the Greedy algorithm, meaning that waiting for even a moderate amount of time can result in very substantial gains.

2.1.3 Anderson 2015

Anderson considered a stylized discrete-time dynamic model of a barter marketplace where, at each time step, exactly one node enters the graph. The arriving node v desires the item of every other node in the system with constant probability p, and each node in the system desires v's item with the same probability p. This means the ensuing structure is a directed Erdős-Rényi graph, where a directed edge (a, b) from a to b means that b desires a's item; equivalently, arrows represent the potential flow of items in the graph. In particular, let $G(t) = (V(t), E(t))$ be the directed graph of compatibilities observed before time t, where $V(t)$ and $E(t)$ denote the vertices and (directed) edges in the graph.

Agent arrival and departure is significantly different than in the Akbarpour model. At each timestep, exactly one agent arrives with an item (i.e., a patient-donor pair arrives with an available donor kidney). Additionally, agents can only leave after being matched in a desirable exchange. Therefore, there is no perishing of agents and no criticality present in this model, but allowing agents to sit in the market for long (or infinite) periods of time would result in worse performance under the objective function.

Under this framework, Anderson examined how the matching policy and types of allowed exchanges affected agent outcomes. Under the assumption that agents in a barter marketplace want to quickly find and complete transactions, the performance metric he uses is the average waiting time of agents in steady state, meaning the optimal policy is one that minimizes the average waiting time. He then showed that, under a timing-centered objective function, allowing more complex exchanges (i.e., three-cycles and chains) and greedily matching at each timestep results in an essentially optimal outcome.

Defining cycles and chains:

As previously mentioned, Anderson defines a k-cycle as a cycle in the barter exchange graph of length k. When a cycle is matched, all nodes are removed from the graph and each agent in the cycle receives an acceptable item. However, his treatment of altruistic donors is much different from previous work. At the first time period, there is exactly one altruistic donor present in the system, and no additional altruists arrive later. An altruistic donor is willing to give up her item without receiving anything in return, leading to the possibility of long (theoretically unbounded) chains in the graph. When these chains are matched, the last agent in the chain becomes the new altruistic node because she has already received a desirable item and now is willing to give her own away. As usual, an *allocation* is a collection of disjoint cycles and chains in the graph, representing a set of mutually exclusive exchanges.

Greedy algorithms:

- *Cycle removal*: By the greedy assumption, the compatibility graph does not contain any cycle of length at most k at the beginning of each time period. If the arrival of a new node results in the formation of at least one cycle of length at most k, one is uniformly chosen to be matched.

- *Chain removal*: At each time step, the algorithm finds an allocation that includes the longest chain originating from the altruistic node (with ties being broken uniformly at random). These nodes are then matched and removed from the system, and the last node in the altruist-initiated chain becomes the new altruist in the system, as described above.

In order to examine the effect of different matching policies and types of allowed exchanges on the average time spent in the exchange, Anderson considered three types of exchanges and looked for a policy in each setting that would minimize the expected waiting time at steady state.

- Two-way cycles: For small p, the greedy policy achieves the optimal scaling and results in an average waiting time of $\Theta(1/p^2)$.
- Two-way cycles and three-way cycles: For small p, the greedy policy is scaling optimal among monotone policies [1] and achieves an average waiting time of $\Theta(1/p^{3/2})$.
- Two-way cycles, three-way cycles, and altruistic chains: For small p, the greedy policy is scaling optimal and achieves an average waiting time of $\Theta(1/p)$.

These results show that, in all three settings, a greedy policy is nearly optimal. Furthermore, for small p, including three-cycles and altruist-initiated chains can greatly reduce the average waiting time.

2.2 A Hybridized Model

In order to examine the potential tradeoffs between match quality and timing considerations, I hybridized both models. I used a discrete time model, as in Anderson, in which one agent arrives at each time period; this allowed me to define the same timing objective function. However, I introduced Akbarpour's concept of critical nodes in order to examine the timing performance of his Patient and Greedy algorithms. In particular, multiple nodes could become critical at each discrete time step. I also limited the exchanges to swaps instead of adding three-cycles and chains in order to abide by Akbarpour's framework.

[1] Monotone policies are defined as follows. Given a pair of nodes (i, j) and any compatibility graph G such that the edge (i, j) is present, if we remove the edge (i, j) and create a new graph G', the policy must act in an identical way up until $T_{ij} = \min(T_i, T_j)$, where T_i and T_j are the times at which i and j are removed from the network. Essentially, monotone policies ensure that the same cycles and chains are removed at the same times in each case, up until the time the first of the two nodes is removed from the graph. In order for a policy to be monotone, this property must hold for every pair of nodes (i, j) and all possible graphs G containing edge (i, j).

By combining the two frameworks in this manner, I was able to explore the tradeoffs between timing and match quality in Akbarpour's algorithms. The Patient algorithm only matches critical patients, which leads to better performance but worse average steady state waiting time than the Greedy algorithm. Conversely, the Greedy algorithm seems to minimize average steady state waiting time at the expense of the total number of matches.

In order to examine the tradeoffs between timing objectives and match quality, I implemented Akbarpour's Patient and Greedy algorithms, along with variations I termed Smart-Patient and Smart-Greedy. The details of the algorithms are as follows.

- Patient: In the discrete time model, given a set of critical nodes c at time t, this algorithm returns a matching with as many of them as possible. Crucially, this algorithm randomly breaks ties between matchings that contain the maximal number of critical nodes; in particular, it is indifferent between allocations with the same number of critical nodes but different numbers of non-critical nodes.

- Greedy: This algorithm is the only one that does not know the set of critical nodes at each discrete time step t. If the node a that just entered the exchange can be matched (i.e., $N(a) \neq \emptyset$), this returns a matching containing that node. Else, it returns nothing.

- Smart-Patient: Given a set of critical nodes c at time t, this algorithm returns a matching that contains the maximal number of critical nodes and the minimal number of non-critical nodes. In this way, it tries to preserve as many non-critical nodes as possible in order to maintain a thicker market. This is in contrast to the Patient algorithm's method of randomly breaking ties between all matchings that contain the maximal number of critical nodes. However, given multiple matchings with the same number of critical and non-critical nodes, Smart-Patient breaks ties at random.

- Smart-Greedy: This is a hybrid of the Smart-Patient and Greedy algorithms that returns a matching containing the maximal number of critical nodes and, secondarily, the maximal number of non-critical nodes. Note that this allows matches between two non-critical nodes, whereas both Patient variants require at least one node in each prescribed swap to be critical. Therefore, this both maximizes the number of matched critical nodes, but also returns a matching that will lead to the thinnest remaining market (in particular, the market at the end of each time step will have no possible swaps).

I ran simulations with death probabilities of $p_d = 0.001$, 0.003, 0.005, 0.01, 0.03, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, and 0.6, 100 time steps in which an agent arrived, and a constant probability of Erdős-Rényi attachment of $p = 0.1$ or 0.3. For each death probability, I ran 1000 trials for each of the four algorithms and recorded the total number of matches and the average amount of time spent in the exchange for each run. I then plotted the average performance of each algorithm for both objectives at each death probability p_d. The standard error for each sample of 1000 runs was negligible in the scale of each plot; see Figures 2.5 and 2.6 at the end of this section for reference. The results are summarized below.

As illustrated in Figures 2.1 and 2.3, we see that the Smart-Greedy algorithm encapsulates the best of the Greedy and (Smart-) Patient algorithms. At low death probabilities, it matches as well as the greedy algorithm, and at higher death probabilities, it matches as well as the Patient and Smart-Patient algorithms.

With respect to timing, as shown in Figures 2.2 and 2.4, the Greedy and Smart-Greedy algorithms perform about equally, as do the Patient and Smart-Patient algorithms. However, the Greedy family of algorithms far out-performs the Patient family at low death probabilities. At higher death probabilities, as the primary mode of exit from the pool shifts from matching to death, the four algorithms' timing performances converge.

Note that the runs with $p = 0.1$ and $p = 0.3$ are qualitatively very similar; the biggest difference between them is a smaller difference in time performance between the Greedy and Patient families at low death probabilities due to the sparser nature of the kidney graph in the $p = 0.1$ case. However, all qualitative observations hold for both connection probabilities, but the effect is less pronounced at $p = 0.1$.

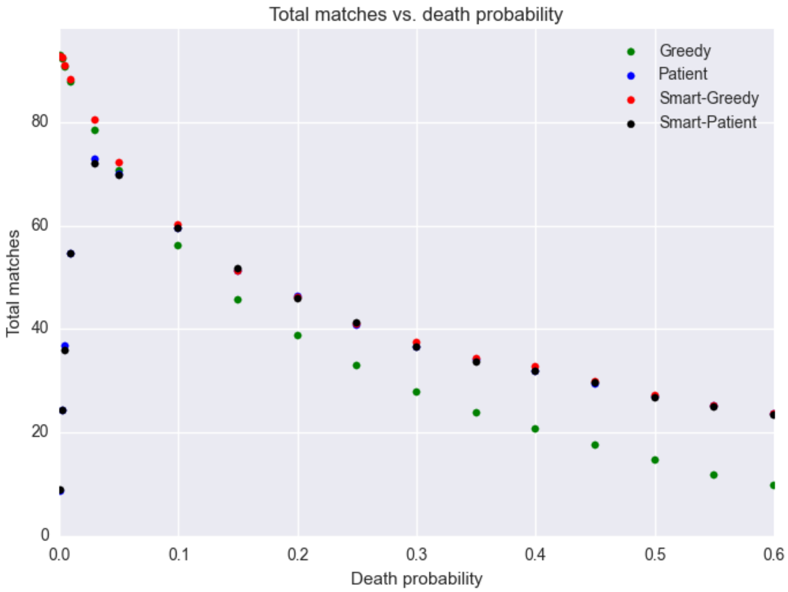

Figure 2.1: Total number of matches vs. death probability for each algorithm. $p = 0.3$.

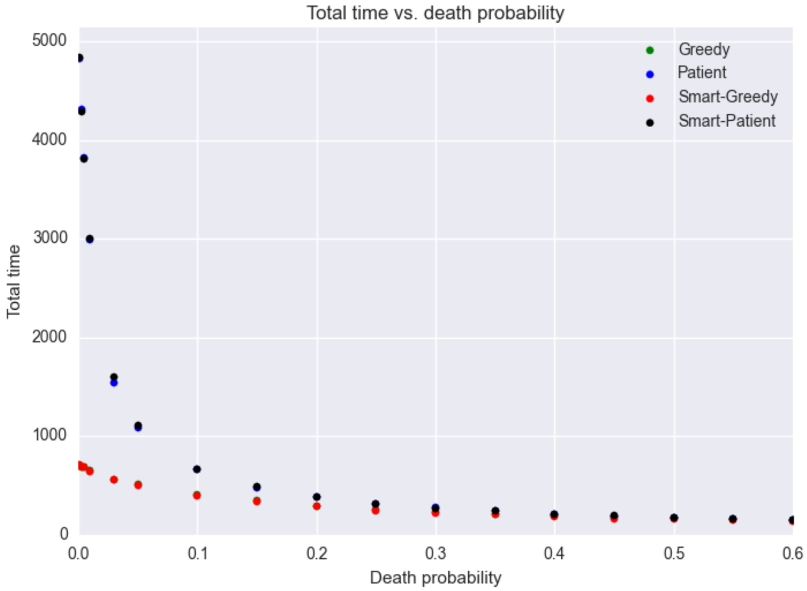

Figure 2.2: Total time spent in exchange vs. death probability for each algorithm. $p = 0.3$.

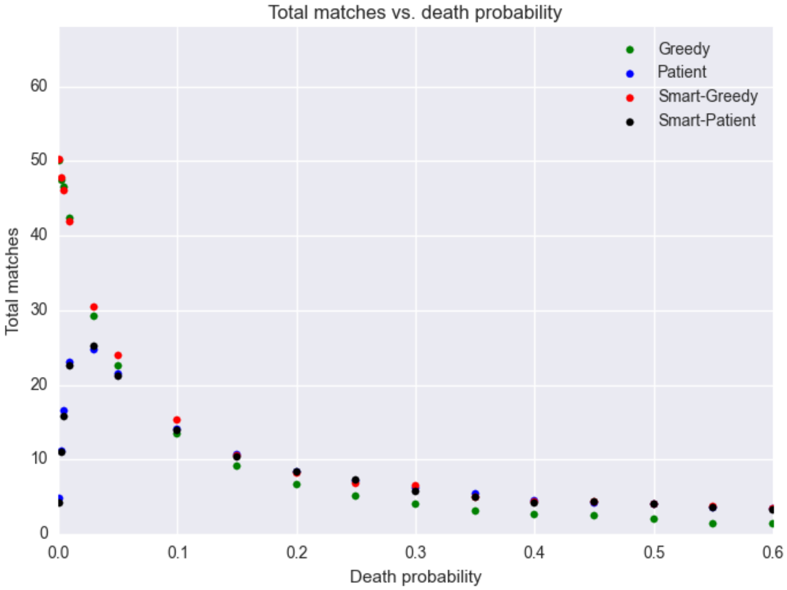

Figure 2.3: Total number of matches vs. death probability for each algorithm. $p = 0.1$.

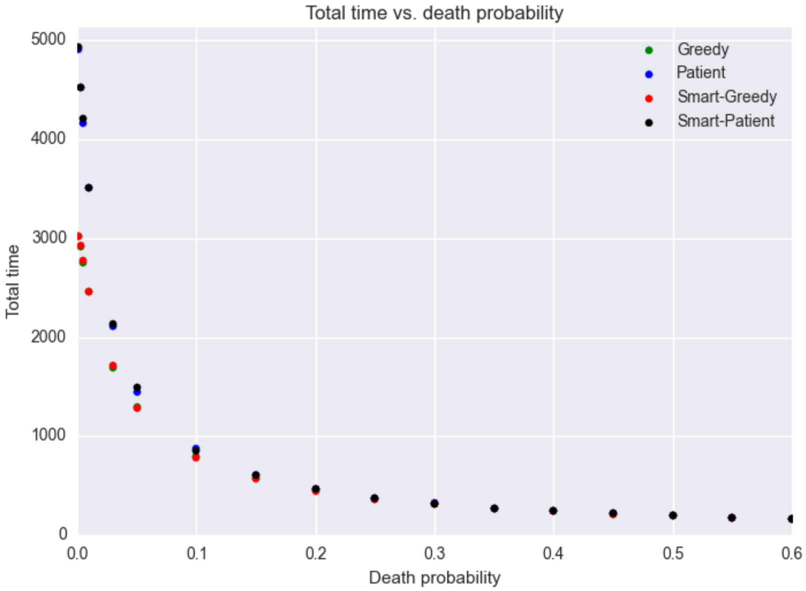

Figure 2.4: Total time spent in exchange vs. death probability for each algorithm. $p = 0.1$.

In general, Smart-Greedy performs as well as Greedy at low death probabilities and as well as Patient and Smart-Patient at high death probabilities. The other three algorithms each had cases in which they performed poorly (i.e., Smart-Patient and Patient perform very poorly with respect to both timing and match cardinality at low death probabilities, whereas Greedy performs poorly with respect to match cardinality at high death probabilities), but Smart-Greedy performed as well as the best-performing algorithm in each case.

Part of this was by design. Smart-Patient was designed to be optimal with respect to the cardinality of critical matches (crucially, not the total number of matches) and as unfair as possible with respect to waiting time, whereas Greedy was designed to take a myopic approach to minimizing the average wait time. Additionally, Patient, Smart-Patient, and Smart-Greedy were allowed to see the set of critical nodes, whereas Greedy was not. In this sense, Smart-Greedy illustrated that an algorithm designed to take advantage of the strengths of both the Greedy and Patient approaches could, in fact, perform as well as the better algorithm with respect to both timing and match cardinality at various levels of departure rates.

However, the most interesting part of these results is the region of ‘medium’ death probability (around $p_d \in [0.1, 0.2]$). In this region, we see that the Greedy algorithm performs worse than the other three algorithms, which had knowledge of critical nodes, but the Greedy class of algorithms still performed noticeably better with respect to timing. The special case of Smart-Greedy aside, this points to a tradeoff between match cardinality and average time spent in the exchange for moderate probabilities of death, which provided the initial motivation behind later work. The tradeoffs between the four algorithms at various death probabilities also motivated further exploration of dynamics at low, medium, and high levels of death.

In summary:

- As the death probability increases, the timing performance of all algorithms converges because the principle dynamic is now death.
- Smart-Greedy has the same behavior as Greedy at low death probabilities and the

same behavior as Smart-Patient at high death probabilities. In between, it shares the best behavior of both Smart-Patient and Greedy.

- There is a tradeoff between cardinality and average time for moderate probabilities of death.

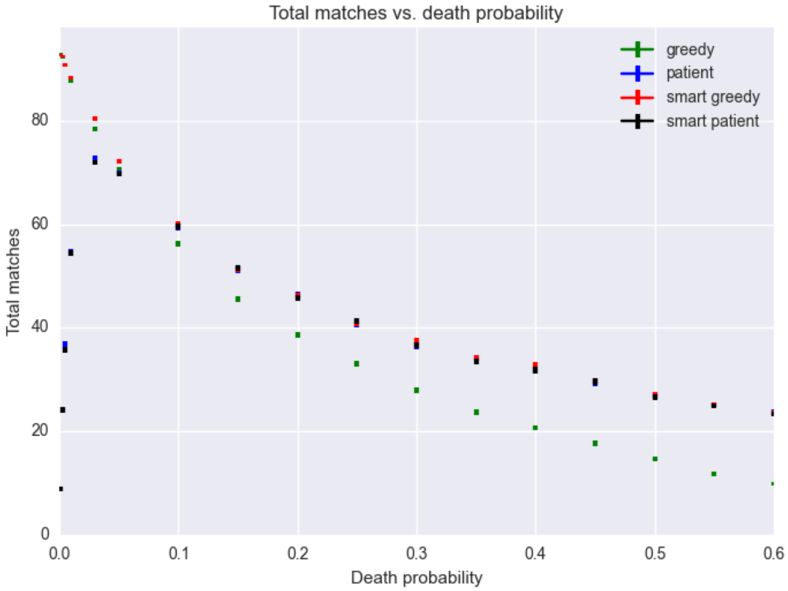

Figure 2.5: Total number of matches vs. death probability for each algorithm with error bars. $p = 0.3$.

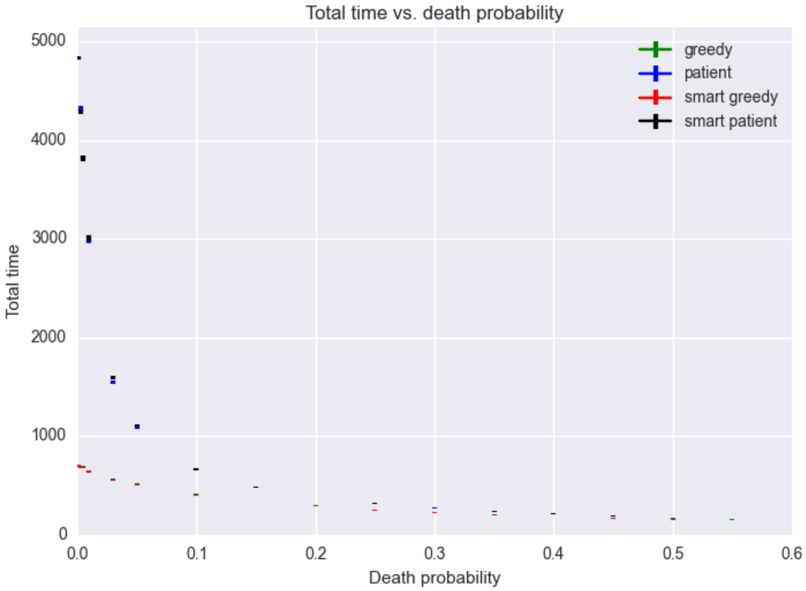

Figure 2.6: Total time spent in exchange vs. death probability for each algorithm with error bars. $p = 0.3$.

Chapter 3

Dynamic Kidney Exchange

In this chapter, I lay out the relevant framework for the dynamic kidney exchange problem. In order to do this, I first introduce the pertinent data structures in the static model of kidney exchange and then trace the development of various approaches to dynamic kidney exchange based on this structure.

After developing both the static and dynamic models of kidney exchange, I then provide a brief overview of the various approaches to dynamic kidney exchange, especially focusing on the development of FutureMatch [13], a general framework for learning to match under a specific prescribed objective in a general dynamic model.

3.1 Static Model

As touched upon in Chapter 1, much intial work was done on the original NP-complete problem of finding maximum matchings consisting of two-cycles, three-cycles, and altruist-initiated chains in kidney exchange graphs. Here, I introduce all relevant terminology and data structures needed for the static model of kidney exchange, all of which are relevant for the dynamic problem as well.

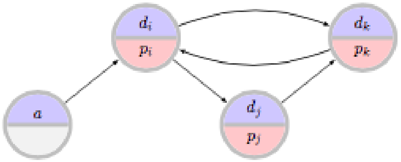

Figure 3.1: A small example of a kidney exchange graph [13].

3.1.1 Kidney Graph

Kidney exchange for n patients is traditionally modeled as a directed *compatibility graph G(n)*. Recall that patients and donors enter the exchange in patient-donor pairs, where each pair is internally incompatible (i.e., each patient cannot accept her donor's kidney). G is constructed by creating a vertex for each patient-donor pair and an edge e from v_i to v_j if the patient in v_j wants the donor kidney in pair v_i (the direction of edges in the kidney exchange model corresponds to the movement of kidneys in the eventual exchange). The weight w_e of edge e represents the utility to the patient at v_j of obtaining v_i's donor kidney[1].

In this model, a donor is only willing to part with her kidney only if her corresponding patient receives a kidney. Accordingly, any cycle c in the graph G represents a possible kidney swap, where the patient at each vertex in the cycle obtains a kidney from the donor at the previous vertex. If c containts k patient-donor pairs, then c is referred to as a *k-cycle*. In practice, due to logistical concerns[2], the length of allowable cycles in kidney exchange graphs is bounded by some upper length L. In most fielded kidney exchanges, including UNOS (the United Network for Organ Sharing), the United States' exchange, $L = 3$, which means that only two- and three-cycles are allowed.

The model also allows for *altruistic donors*, which are willing donors without an attached

[1] While in theory patients are equally happy receiving any compatible, functional kidney, issues of blood type compatibility and tissue type compatibility, among other measures, impact how well patients survive with new kidneys. I will discuss these issues of compatibility in greater detail later in this chapter.

[2] When performing kidney swaps, all transplants in a cycle must be performed simultaneously in order to ensure no patient is left without a willing transplant should any donor down the line renege. This also means that the surgeries must be performed at the same location, further constraining the logistical practicality of large transplant cycles.

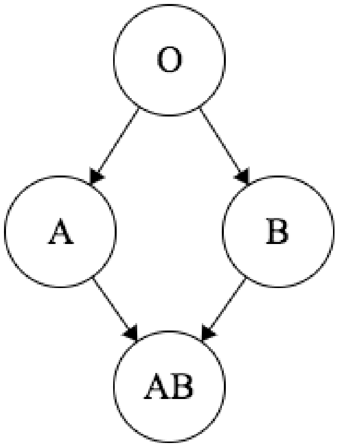

Figure 3.2: A partial ordering on ABO blood types.

patient, to enter the exchange. The presence of altruistic donors allows for the formation of chains of kidney transplants instead of cycles, leading to much more flexible exchanges. It has been shown that the addition of chains adds great utility to fielded kidney exchanges. In theory, the length of these chains is unbounded, but most eventually terminate with a patient who lacks a paired donor.

A *matching* M is a collection of disjoint cycles and chains in a kidney exchange graph G. Note that the cycles and chains must be disjoint because each donor can only give away one kidney (and each patient will only accept one kidney). Every vertex in a matching M will both receive and give a kidney, unless it is an altruistic donor, in which case it only gives a kidney.

3.1.2 Blood Types

In the kidney exchange model, patients and donors are represented by their blood type (A, B, AB, or O). This is because the primary criterion for a possible kidney match is blood type compatibility. An O-type patient can only accept an O-type donor; an A-type patient can accept an O-type or A-type donor; a B-type patient can accept an O-type or B-type donor; and an AB-type patient can accept an O-type, A-type, B-type, or AB-type donor. In this sense, O-type patients are universal donors and AB-type patients are universal acceptors. As shown in Figure 3.2, each blood type can donate to all reachable nodes along the directed edges.

Based on this partial ordering over the blood types, we can, as in [19] and [18], define under-demanded (UD), over-demanded (OD), reciprocal (R), and self-demanded (S) pairs. Intuitively, pairs such as (B, O) are relatively easy to match, hence their classification as over-demanded. Conversely, (O, B) pairs are harder to match, leading them to be labeled under-demanded. Pairs consisting of patients and donors with the same bloodtype are self-demanded, and (A, B) and (B, A) pairs are reciprocal. Table 3.1 depicts the classification of patient-donor pairs under this framework.

Patient	Donor			
	O	**A**	**B**	**AB**
O	S	UD	UD	UD
A	OD	S	R	UD
B	OD	R	S	UD
AB	OD	OD	OD	S

Table 3.1: Patient-donor pairs grouped by pair type. [18]

3.1.3 Tissue Types

Patients and donors must also be tissue type compatible for a successful match. Determining tissue type compatibility requires additional, more extensive tests than checking blood type compatibility, and therefore most matches are initially made based on blood type compatibility and then only later checked for tissue type compatibility. Because tissue type compatibility is not as straightforward as blood type compatibility (there is no clean antibody-based system like the one based on blood types), people generally look at the *Panel Reactive Antibody (PRA)* sensitivity of patients, which broadly measures the percentage of the population with whom the patient will be tissue-type incompatible [18], where a higher PRA value means fewer compatible matches. Generally, people consider two models for PRA sensitivity: a uniform PRA model, in which all patients have the same PRA, and a non-uniform model, in which patients are sorted into three sensitivity groups (low, medium, and high), each associated with a different PRA value. Low sensitivity patients have lower PRA values and more available matches than high sensitivity patients. In this paper, I focus on the uniform PRA model for simplicity, as the non-uniform model requires additional population considerations.

3.2 Dynamic Model

Kidney exchange is fundamentally a dynamic problem. As mentioned in Chapter 1, there are two types of dynamism: entrances and exits from the patient-donor pool, and exchange failures after the initial match.

Entrances and exits from the pool over time are very interesting because they change the way in which an ideal clearing engine would match people during each timestep. Most kidney exchanges run in batches and clear periodically (anywhere from multiple times a week to a few times per year). However, this means that myopically greedily matching during each cycle may not yield the best results over time. In addition, the choice of an objective function matters more in dynamic kidney exchange. This is because different objectives lead to quantifiably different algorithms for generating matchings. In general, dynamic, online matching is much more difficult to align with more complex long-term goals than the static, myopic case. In this vein, I discuss the implications of various conceptions of fairness on the dynamic matching problem below.

Over the years, people have explored two main approaches to dynamic matching: online stochastic optimization via the sampling of future trajectories, and a learning-based model that matches via weighted myopia. I discuss both approaches in the next section.

3.2.1 Objective Functions and Fairness

There are many ways of formulating the problem of 'fairness' in kidney exchange. Each definition comes with its own set of medical objectives and its own objective function, and many of them are in some ways quite orthogonal to each other. In Dickerson et al.'s recent FutureMatch paper [13], they consider three objective functions: MaxCard, MaxCard-Fair, and MaxLife.

- MaxCard maximizes the total number of patients that are either algorithmically matched (in the deterministic, or non-failure-aware model) or receive transplants in expectation (in the failure-aware setting).
- MaxCard-Fair adds to this the concept of 'marginalized' patients, or patients who are by some measure harder to match, and up-weights them by some factor β.
- MaxLife, the most complicated objective function they use, attempts to maximize the total amount of time transplanted organs last in patients. Some of these objective functions are much harder to formally characterize than others, leading the team to devise a method for translating a medical professional-defined objective into a set of weights on edges in the kidney exchange graph.

3.3 Stochastic Optimization

Awasthi and Sandholm [7] introduced the distinction between static (myopic) and dynamic (non-myopic) kidney exchange. Although the authors had previously published a result in 2007 that optimally solves the kidney exchange problem given a certain pool of people [1], they noted that this often repeatedly left behind hard-to-match patients, which harmed performance in the long run. In order to address this problem, they introduce a trajectory-sampling approach that uses information about blood and tissue type distributions in the United States in order to sample possible future trajectories. Under each trajectory, they then consider the utility of possible actions in the current timestep. By aggregating these results across the sampled future trajectories, they can determine the current action with the best expected utility. They introduce three different algorithms that deal with sampling future trajectories.

However, there are some problems with this approach. There is an inherent tradeoff between sample size (the number of sample trajectories) and lookahead depth (the distance you look ahead along each trajectory), and increasing lookahead depth while keeping sample size constant can decrease the solution quality because a smaller fraction of future trajectories is sampled at each time step. Additionally, the number of future trajectories scales very quickly with the size of kidney exchange network, limiting the algorithm's scope in practice.

3.4 Weighted Myopia and FutureMatch

Stochastic optimization brought to light many important considerations when considering dynamic kidney exchange, but the trajectory-sampling approach did not scale to reasonably-sized kidney exchange graphs. In order to address this issue, Sandholm et al. moved toward another learning-based approach they termed weighted myopia. They propose to learn ways to re-weight kidney graphs such that taking the myopic matching on the altered graph at each time step will eventually satisfy a specified offline objective. An additional benefit of this approach is that it allows them to introduce measures of 'fairness' in dynamic kidney exchange (e.g., as included in the aforementioned MaxCard, MaxCard-Fair, and MaxLife objective functions), which addresses the real-world concerns of medical professionals and patients alike. This is described in greater detail below.

3.4.1 Dynamic Matching via Weighted Myopia

In a follow-up paper, Dickerson et al. [10] address many of the computational concerns. Because the previous trajectory-sampling approach does not scale beyond small exchanges, the authors propose a new solution framework for the dynamic kidney exchange problem: namely, the notion of *potentials*. Given a structure in the kidney exchange graph (vertex, edge, cycle, etc.), its potential can be thought of as the expected future utility that can be derived from that structure. For example, consider potentials on vertices[3]. An altruistic O donor should have high potential, as they have the ability to set off a long chain with high value, while a pair needing an O donor should have low potential, as they are generally hard to match and unlikely to enable many matchings. Any algorithm with this potential information should act accordingly, perhaps holding structures with high potentials until the system is in a state where that potential is reached (e.g. when the altruistic O donor is able to set off a long chain), while trying to immediately match things with lower potential. It is assumed here that all structures of the same variety (e.g. all vertices with the same blood type) have the same potential.

The paper introduces an algorithm that uses potentials on vertices to re-weight edges. Given potentials p_a and p_b on vertices a and b, respectively, the edge e from a to b would have weight $w(e) = 1 - \frac{1}{2}(p_a + p_b)$. In every period, the algorithm reweights all the edges according to the procedure above, and then uses the myopic clearing algorithm to determine the matching with the maximum total weight. Once the potentials are learned, this means that each stage is computationally similar to the myopic approach, as the computation of edge weights is generally a simple operation. Therefore, this addresses many of the computational complexity issues of the previous paper and results in an approach that will scale to larger instances of kidney exchange.

[3] Recall that each vertex contains a patient and a donor, each represented by their blood type: O, A, B, or AB, based on the presence of A and B antibodies. Exchanges are only possible if the donor's blood contains a subset of the patient's antibodies. O donors can donate to anyone, A donors can donate to A or AB patients, B donors can donate to B or AB patients, and AB patients can donate only to AB patients. O patients can only accept from O donors, A patients can accept from O or A donors, B patients can accept from A or B donors, and AB patients can accept from anyone.

3.4.2 Balancing Efficiency and Fairness in Dynamic Kidney Exchange

In 2014, Dickerson and Sandhom built upon their previous work in dynamic by articulating a solution framework that allowed for different objective functions. This enabled the incorporation of multiple definitions of fairness, which is a desirable trait in a real-world exchange. In particular, Dickerson and Sandholm propose various utility functions that incorporate various measures of fairness: MaxCard, MaxCard-Fair, and MaxLife.

In the conventional approach to kidney exchange (i.e., trying to maximize the total number of matches in a pool over time), there may be groups that are perpetually marginalized by the algorithm, such as difficult-to-match pairs. In addition, there may be sets of patients that one desires to preference due to social or moral concerns. Determining which categories of people qualify as 'marginalized' is a difficult task, which the authors recognize. Based on best practices within the medical community, they focus on two groups: highly sensitized and pediatric patients. Highly sensitized patients are those that are unlikely to be compatible with a random kidney (for reasons other than blood type). These people are therefore very difficult to match, but unsurprisingly, quite prevalent in the pools of kidney exchanges as difficult cases tend to persist. Pediatric patients are quite self-explanatory: they are preferenced because of longer future lifespan and also the potential for kidney disease to stunt growth.

The approach here increases the weight of any edge that is adjacent to a marginalized patient by a factor $(1 + \beta)$ for some value β. This is their notion of 'fairness': that by preferencing these margnalized groups, they have made the system more fair.

In order to incorporate measures of fairness into their model, Dickerson and Sandholm obtain an objective function from a panel of medical experts. They then translate that function into a set of weights on the edges in the compatibility graph. Then, given this weighting function, they learn potentials on graph structures, as in the previous paper. The potentials are then combined with the edge weighting function to give each edge a final weight, and the resulting re-weighted graph is then fed through the myopic matching algorithm at each time period. Given an edge e with a weight $w(e)$ and potentials p_a and

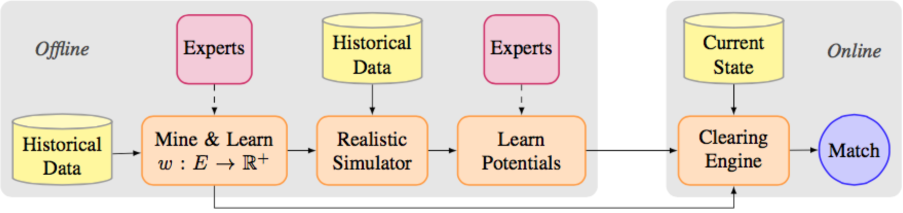

Figure 3.3: The FutureMatch pipeline. [13]

p_b at each of its vertices, its final weight would be $f_w(e) = w(e) \cdot (1 - p_a - p_b)$. Note that this approach closely mirrors that in the previous paper, but it adds in a medical objective that can incorporate measures of fairness.

In addition, this paper introduced the possiblity of *post-match failure* - when transplants do not occur due to unforeseen incompatibility, logistical issues, or death - to the paradigm of online matching. Therefore, under this assumption, the authors maximize the expected number of transplants instead of merely the number of algorithmically-prescribed matches. They further explore this form of dynamism in subsequent papers, as discussed below.

3.4.3 FutureMatch

The FutureMatch framework [13] learns from historical data how to match people in each time step in order to maximize some overarching objective function over time. In order to do this, it incorporates two steps of machine learning to learn *edge weights* and *vertex potentials*, respectively, resulting in a parameterized online matching algorithm that learns to match in the present in order to match some desired long-term behavior. One key consideration in the development of this framework is scability; the fielded clearing algorithm must be able to run relatively quickly in order to be implementable in practice. Also, note that this framework allows for two- and three-cycles as well as altruist-initiated chains.

At a very high level, the FutureMatch framework consists of three main steps: translating a medically-defined objective into a set of edge weights on the kidney exchange graph, learning vertex potentials, and incorporating both sets of weights into a final myopic weighted matching algorithm.

The first step in FutureMatch is deciding what objective function to maximize. The objective functions are defined by domain experts, generally a committee of medical and legal professionals, and they are generally quite hard to quantify. For example, if the goal is to maximize the total time patients survive after kidney transplantation, then much additional data about the relative quality of each match (i.e., the weight on each edge in the compatibility graph) is needed. Therefore, it is possible to learn a set of edge weights w in the compatibility graph corresponding to a medically-defined objective function by using historical data.

The resulting learned weight function w is then fed into a simulator based on real historical data and which therefore mimics the true underlying distribution of kidney exchange patients and donors. This simulator generates training and test data, which are then fed into a system for learning the potentials on vertices in the original kidney graph. The potential of a vertex in the graph is the expected utility to the overall exchange of that vertex in the future. Therefore, potentials intuitively quantify the value for waiting to use a certain vertex or set of vertices at a later time in the exchange. Although potentials can be defined for any graph element class (not just vertices), the authors focused on learning potentials on patient-donor blood type pairs, or what we defined as vertices in the graph.

The edge weights and vertex potentials are then combined in a final parameterized online matching (or clearing) algorithm. This is a very simple process that merely results in a re-weighting of the original input graph. Furthermore, note that now finding myopic, greedy matchings in the re-weighted kidney graph will result in the desired long-term behavior because of the incorporation of the learned edge weights and vertex potentials. Essentially, the weights encode the future, resulting in a ‘potential-aware’ kidney exchange graph in which greedily matching at each time step addresses the original medically-defined objective. In particular, the final step of online matching makes use of state-of-the-art myopic matching based on a branch-and-price algorithm developed by Dickerson, Procaccia, and Sandholm [11].

Match Failures

FutureMatch also allows for failure-aware considerations. As tangentially touched on before, FutureMatch matches in either a deterministic or a failure-aware setting. In the deterministic setting, FutureMatch learns to maximize the objective function subject to the assumption that all matches will go through to transplantation. In the failure-aware setting, FutureMatch learns to maximize the expected value of the objective function under some assumption of post-match failure. However, note that post-match failures are allowed in both cases, and after an algorithmically-prescribed match fails, the pair is re-entered into the patient-donor pool. The main difference between the deterministic and failure-aware settings is the fact that the deterministic model maximizes the given objective purely based on algorithmic matches, whereas the failure-aware model maximizes the given objective in expectation. Again, I focus on the deterministic setting in this thesis, but the framework is flexible and able to consider the failure-aware setting.

Learning Edge Weights

Once a medical objective is set by a committee of medical professionals, the FutureMatch framework learns a corresponding set of edge weights that encodes the desired function. As mentioned above, three objectives are considered (MaxCard, MaxCard-Fair, and MaxLife), each with a different resulting learned weight function based on historical data.

One issue that the FutureMatch team noticed was that experts often conflate the ends and means of their policy suggestions. However, the framework of FutureMatch ensures the separation of the ends and means. The ends (or goal) of the exchange, the expert-defined objective, is defined as weights on edges, whereas the means, potentials on vertices, are automatically optimized in a manner orthogonal to the initial expert-defined ends.

However, although the medical objective is initially translated into a set of weights on edges, when evaluating the quality of the final clearing algorithm, the output of the algorithm is judged based on the originally defined medical objective, and not based on the learned edge weights. This removes the effect of any inaccuracies or biases when learning the edge weights and allows for more correct evaluation overall.

Learning Vertex Potentials

The learned edge weights represent the value of an edge in the present. However, because the medical objective is defined in a dynamic setting (i.e., over many match cycles), it is also necessary to learn discount factors on parts of the graph that could potentially be more valuable in the future. The initial idea of potentials was proposed by Dickerson, Procaccia, and Sandholm [9], but the FutureMatch conception of potentials uses an algorithm that converges and more realistic training and test data in order to more effectively characterize the future expected utility from each graph element.

Potentials are defined on a set of features Θ that represent many element types in the pool. For each element type θ, there exists a potential $P_\theta \in \mathbb{R}$ that represents the expected future utility of that element type to the overall pool. In this case, potentials are defined over the blood types of patients and donors. Recall that humans have blood types O, A, B, or AB, depending on the presence or absence of two proteins (A and B), and blood type compatibility is based primarily on the presence and absence of the same proteins. A donor can donate blood to any patient with a superset of her blood proteins (O can donate to anyone, A can donate to A or AB, B can donate to B or AB, and AB can only donate to AB). Conversely, a recipient can accept blood from any donor with a subset of her blood proteins (O can only accept O, A can accept O or A, B can accept O or B, and AB can accept any blood type). This intuitively means that it is easier to find a match for an O donor or an AB patient, meaning that they are perhaps more 'valuable' overall.

Formally, potentials are defined on all types of vertices. In this case, there are 16 patient-donor blood type pairs and 4 altruistic blood types, so this means that there are 20 types of vertices: $\Theta_{ABO} = \{O-O, O-A, \ldots, AB-B, AB-AB\} \cup \{O, \ldots, AB\}$. There is a real-valued potential P_θ for each $\theta \in \Theta_{ABO}$, which we then learn.

After learning potentials on vertices, we again re-weight the edges in the graph. Specifically, the revised weight f_w of an edge, in terms of the learned edge weight $w(e)$ from the medical objective and the potentials on the donor (d) and patient (p) vertices, P_{θ_d} and P_{θ_p}, is $f_w(e) = w(e) \cdot (1 - P_{\theta_d} - P_{\theta_p})$. These new 're-weights' balance the myopic value of an edge in the framework of the objective function with the future value of each edge as represented by the two vertices it connects. As such, the new weights balance the medical objective and the potential expected utility through learned weights on both edges (myopic and goal-oriented) and vertices (long-term utility).

SMAC

The FutureMatch team uses SMAC [17], a sequential model-based algorithm configuration tool that searches through a parameter space in order to optimize a provided objective. In this case, the parameter vector is the vector of vertex potentials, and the evaluation metric is based on many trials through the kidney exchange simulator. In particular, SMAC loops through a cycle of hypothesizing and evaluating a vector of vertex potentials, and after convergence, it returns a list of potentials.

For the purposes of my thesis, I treated SMAC as a black box; that is to say, I did not worry about the specifics of its implementation, but only used it to search through the vertex potential parameter space and return a list of final potentials.

Online Clearing Algorithm

The online clearing algorithm is exactly a myopically greedy matching on the re-weighted kidney exchange graph. It incorporates the edge weighs learned to reflect the medical objective function in the first stage of FutureMatch, as well as the vertex potentials learned to discount the utility of various blood types in the second stage.

In each period, the online clearing algorithm maximizes the total weight of the matching it clears, where for each chain or cycle c, $u(c) = \sum_{e \in c} w_e$. A current myopic state-of-the-art algorithm based on work by Dickerson, Procaccia, and Sandholm [11] is used to efficiently find matchings on the edge- and potential-weighted graphs. The algorithm is then tested on current data from UNOS and evaluated based on the initially defined medical objective (crucially, not the learned edge-weight approximation of the objective).

3.5 Related Work

Although failure-aware kidney exchange is not the focus of this thesis, it is an orthogonal form of dynamism in the kidney exchange problem. Additionally, much work has been done on incentivizing truthful reporting from participating hospitals — i.e., making sure that they don't hide any cycles or chains they can match internally — and interesting results have come out of both of these angles on the kidney exchange problem. Ideally, the kidney exchange problem would be best addressed by a combination of a mechanism that incentivizes truthful reporting by hospitals and an algorithm that accounts for entrances and exits over time, as well as post-match failures. However, this has yet to be put together in a cohesive manner.

3.5.1 Failure-Aware Kidney Exchange

A second form of dynamism in kidney exchange involves the failure of algorithmic matches before actual transplantation. In fact, most matches do not ultimately result in transplants, leading to a very interesting problem of maximizing the expected number of transplants instead of the total number of deterministic matches, as addressed by Dickerson et al. [11]. This also introduces the idea of match quality into the framework of kidney exchange, a direction that has been explored by Blum et al. [8].

In particular, Dickerson et al. addresses the problem of maximizing the expected number of lives saved in theory on random graph models, on real data from past kidney exchange runs, and on synthetic data generated by a realistic kidney exchange simulator. By formulating the problem as a probabilistic exchange, they design a scalable branch-and-price-based clearing algorithm that addresses the above objectives. However, they treat the process of determining whether a match will go to completion more or less as a black box and merely assign success probabilities q_e to each edge e, which they then use to determine the discounted utility of each cycle or chain. In particular, the utility $u(c)$ of a chain or cycle c is equal to the sum of all weights w_e on edges e in c multiplied by the product of the success probabilities on all edges e in c. In other words,

$$u(c) = \left[\sum_{e \in c} w_e\right] \cdot \left[\prod_{e \in c} q_e\right].$$

They then use this discounted utility per cycle or chain throughout the paper in their maximization of the total expected utility (equivalently, the total number of expected transplants) over time.

However, Blum et al. [8] approach this problem from a different perspective based on the medical tests needed in order to determine compatibility past the initial blood-type matching stage, which is generally the basis of algorithmically-prescribed matchings. In the standard operating procedure for kidney exchange, additional tests are administered to each matched pair in order to determine their actual compatibility. However, these tests are expensive and time-consuming, and, as mentioned before, many algorithmically matched swaps are ultimately found to be incompatible. In this way, tests between vertices yield additional information as to whether the edge between the two vertices truly exists (i.e., if the prescribed exchange will be carried out in reality).

Blum et al. consider the problem of querying a small number of edges per vertex in order to gain as much information as possible. They design an adaptive algorithm that queries a constant number of edges per vertex and achieves an aribtrarily close approximation of the optimal omniscient solution.

3.5.2 Truthful Reporting in Kidney Exchange

One other major problem in the realm of kidney exchange is incentivizing hospitals to truthfully report their pool of patients and donors. Many hospitals may be tempted to withhold pairs of patients and donors they can match themselves [18], but this harms overall exchanges by increasing the number of ‘hard to match’ pairs. Therefore, there has been much work in designing proper mechanisms in order to make sure that hospitals are incentivized to report all of their pairs.

When considering truthful reporting, it is necessary to consider both *individual rationality* and *strategyproofness*. A mechanism is individually rational if players are better off participating in the exchange than abstaining (i.e., matching their pairs purely internally). On the other hand, a mechanism is strategyproof if agents are best off truthfully entering all their pairs in the exchange. In particular, this means that no hospital will do better off by keeping some internal matches 'hidden' from the exchange. Additionally, note that strategyproofness implies individual rationality, but not the other way around; it is a strictly stronger claim. Establishing individual rationality for kidney exchange mechamisms is not particularly hard, but ensuring strategyproofness is much more difficult.

Toulis and Parkes [18] observe that the expected benefit of pooling scales with the square root of the number of pairs in each hospital. They then design the xCM algorithm, which incentives hospitals of at least moderate size to fully report their pairs.

In a similar vein, Ashlagi et al. [6] examine strategyproof mechanisms for incentivizing hospitals to truthfully report their pairs. They establish welfare loss bounds for randomized and deterministic mechanisms and propose a randomized mechanism that guarantees at least half of the maxmimum social welfare in the worst case, but which performs much closer to optimal in simulations.

Chapter 4

Dynamic Kidney Exchange with Timing Considerations

For the bulk of my research, I did not use the full version of FutureMatch, but rather wrote my own stripped-down version. I focused on the second step — learning potentials on vertices — and used the previously existing learned edge weights corresponding to FutureMatch's objective functions [13]. This allowed me to be more flexible in my experiments and better isolate the effect of introducing timing considerations into the general FutureMatch framework.

In general, my framework consisted of the following pieces, which I explain in more detail below. It is also visually represented in Figure 4.1.

- Graph generator: I wrote a simulator that generated kidney graphs $G = (V, E)$. This was used for training and testing various iterations of the online algorithm.

- Simulator: This took in a graph, a set of potentials, and edge weights and returned a matching resulting from carrying out a weighted myopic clearing algorithm on the re-weighted graph.

- Evaluator: This took in a matching and evaluated the timing and medical objective. It then combined the timing and medical objectives and reported a global score that

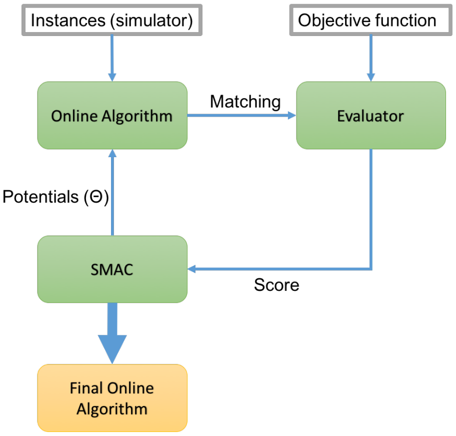

Figure 4.1: A schematic of the timing-aware learning framework.

SMAC could interpret.

- SMAC: As described in Chapter 3, this was used to learn a set of potentials (i.e., weights on vertices representing future utility) corresponding to a certain combination of a medical objective and timing considerations.

- Offline clearing algorithm: This is the output of the entire pipeline, and this takes in a set of learned potentials and edge weights and returns a weighted myopic algorithm designed to address a scoring rule corresponding to a certain combination of timing and a medical objective.

4.1 Graph Generation

I generated kidney exchange graphs $G = (V, E)$ that were then processed by a simulator in order to obtain matchings. In essence, each graph represents the change (i.e., entrances and exits) in a kidney exchange pool over time.

4.1.1 Vertices

I generate vertices consisting of patient-donor pairs by drawing two independent blood types from a distribution F_{ABO} [18]. In addition, each vertex pair is associated with an entrance and exit time from the exchange, as discussed below. Therefore, each vertex consists of four parts: a patient blood type, a donor blood type, an entrance time, and an exit time, or $V = (P_{ABO}, D_{ABO}, t_0, t_1)$.

Also, note that there are no altruistic vertices in my model. This is also for simplicity's sake, as adding unbounded-length chains significantly complicates the matching process. Dealing with the generation of altruists also adds another layer to entrance dynamics that unnecessarily complicates things in a proof-of-concept work.

Entrances and Exits

In my discrete-time model, I define an exchange as having a start time T_0 and an end time T_f. Vertices are uniformly generated at times $E \sim \mathrm{Unif}(T_0, T_f)$. However, not all generated vertices enter the exchange; if the generated vertex is self-compatible, then it does not enter the exchange. I explain this in greater detail in 4.1.3. Agents exit the exchange either when they are matched or if they perish. To this end, I again take the deterministic (i.e., not failure-aware) view in the FutureMatch framework.

Death Probability

As in [2], each agent a's lifespan is drawn from an exponential distribution parameterized by a variable p. In particular, if a enters the exchange at time t_0, she dies at some time $t_0 + X$, $X \sim \exp(p)$. This death probability is held constant across all agents in the exchange.

4.1.2 Edges

For each blood type compatible pair of a patient and donor at different vertices, I determined whether or not they were tissue compatible by flipping a biased coin, as described in 4.1.3. For every patient-donor combination deemed compatible, I introduced a directed edge from the vertex containing the donor to the vertex containing the intended recipient. I then stored all these potential edges.

Note that this generates all potential edges in a graph without taking timing into account. Specifically, some of these edges may never occur because the endpoint vertices may not be in the exchange at the same time. It is left to the simulator to determine which edges are actually viable when running the exchange.

Edge Weights

Due to the focus on the MaxCard objective, all weights on edges in the graph are trivially set to 1 because we only care about the total number of matches, implying that all vertices are of equal importance to us.

When considering other objectives (i.e., MaxCard-Fair and MaxLife), learning edge weights becomes much more complicated. In the MaxCard-Fair case, which can be viewed as a generalized version of MaxCard, we can multiplicatively re-weight edges that involve a marginalized patient. For example, we can define an edge re-weighting function $\Delta^\beta : E \to \mathbb{R}$, where β is a parameter that determines how much to prefer a certain marginalized subpopulation M, as follows:

$$\Delta^\beta(e) := \begin{cases} (1+\beta)w_e & \text{if } e \text{ ends in } V_p \in M \\ 1 & \text{otherwise.} \end{cases}$$

The MaxLife case is yet more complicated. First, it is necessary to run a Cox proportional hazards test in order to determine the effect of multiple features on survivability. The most significant features are then fed into a regression learning step in order to output an edge weighting function that takes features such as recipient age, the age difference between the recipient and donor, and components of each person's HLA profile in order to output a weight on the specified edge. However, this requires a much more detailed population profile.

For the purposes of this thesis, I used a uniform edge weighting scheme (i.e., without loss of generality, all edge weights were set to 1) because of my focus on the MaxCard objective.

4.1.3 Compatibility

As in [18], I take a two-stage approach to determining compatibility between two vertices: they must be both blood type and tissue type compatible. When generating a new patient-donor pair, I draw the blood type of each person from a pre-defined distribution as in [18] and [20]. If the pair is blood type incompatible, then it enters the exchange. However, if the pair is blood type compatible, I then flip a biased coin representing the PRA values of each person (as discussed in 3.1.3, this is assumed to be uniform across the entire population) and then only enter the pair into the exchange if the components are determined to be tissue type incompatible. Specifically, if the donor has a PRA value of p_d and the patient has a PRA value of p_p, the probability that they are tissue type compatible is $(1 - p_d) \cdot (1 - p_p)$.

More generally, this framework holds for determining general compatibility: the patient and donor must be both blood and tissue type compatible.

I obtained both the blood type distribution and the uniform PRA value I used from existing literature [18] [20].

4.2 Simulation

Given a graph G, a set of edge weights W, and a set of potentials Θ, the simulator generates a matching via weighted myopia, which means it re-weights the edges in the graph based on W and Θ and then carries out a greedy batching algorithm in order to obtain a final matching.

Formally, as discussed in Chapter 3, given an edge weight $w(e)$ and potentials P_{θ_d} and P_{θ_p} on the donor and patient vertices, respectively, the re-weighted edge has value $f_w(e) = w(e) \cdot (1 - P_{\theta_d} - P_{\theta_p})$. The new edge weight is then used to myopically generate matchings that incorporate both the medical objective (as represented by edge weights) and the future value of various vertices (as captured by the learned potentials).

4.2.1 Swaps

I only allowed swaps (i.e., two-cycles) in my framework. This significantly simplified the matching process because I could use a weighted version of Edmonds' Blossom algorithm [14] implementated by Galil [16] in order to find the maximum weight matchings in each time period.

As previous studies have shown, adding three-cycles and altruist-initiated chains greatly improves the overall performance of kidney exchange algorithms [3]. However, as a first step in examining the effects of timing on other aspects of kidney exchange, simplifying the problem allowed me to run a deterministic algorithm in order to find matchings, as opposed to the more complicated branch-and-price algorithms used to solve the NP-complete bounded-length cycle case.

4.3 Evaluation

This step takes a matching and returns a summary score that combines the timing and MaxCard objectives via a weighted average.

4.3.1 Timing Objective

I introduced a global timing parameter t_g that measured the average amount of time people spent in the exchange before leaving due to either death or transplant. The timing objective aimed to minimize this quantity.

4.3.2 Medical Objective

Throughout this thesis, I considered the MaxCard medical objective [9] [13]. Although this objective does not explicitly contain a particularly strong notion of fairness, as discussed earlier, this is the canonical kidney exchange objective and there is the richest literature on how to interpret and process this particular objective function.

4.3.3 Weighting the Timing Objective

I then combined the timing objective with the MaxCard objective score in order to create a new 'timing-weighted' score to feed into SMAC. I examined five different combinations of the timing and MaxCard objectives.

I used a 'slider' that allows for different weighted averages of timing and the medical objective to be used as an objective function for SMAC. I had five ranges of weighting that ranged from purely the timing objective to only the medical objective, with three ranges in between. If we let a and $1-a$ represent the weights given to the timing and medical objective, respectively, the five combinations I tested correspond to $a \in \{0, 0.25, 0.50, 0.75, 1\}$.

However, because the timing and MaxCard objectives measure drastically different things, merely taking a weighted average would not preserve the relative weights I desired in each of my test scenarios. To that end, in order to combine the timing and MaxCard objectives, I first (approximately) normalized each and then took a weighted average. In order to approximately normalize each objective, I first generated a large amount of data on a training set with the same death probability p but all constant edge weights and potentials. From this, I then extracted the mean μ and standard deviation σ for both the timing and MaxCard objectives, which I then used to transform the data into an approximately standard score: $X_{new} = \frac{X-\mu}{\sigma}$.

Additionally, because it is desirable to minimize timing and maximize cardinality, I then negated the normalized cardinality score before taking a weighted average of the two. In particular, given a weighted average parameterized by $a \in [0, 1]$, a normalized timing score s_t, and a normalized cardinality score s_c, the final aggregated score to feed into SMAC is $s = a \cdot s_t + (1 - a) \cdot (-s_c)$.

4.4 SMAC

As mentioned in Chapter 3, I treated the specifics of SMAC as a black box and merely used it to learn potentials corresponding to various timing-aware objective functions.

4.4.1 Convergence

However, something I did notice was that SMAC often took prohibitively long to converge, even when I ran it over long periods of time. In personal correspondence, John Dickerson also mentioned that in multiple cases in the FutureMatch paper, SMAC did not converge for all potentials. Therefore, I cut off all SMAC runs at a very high cutoff and took the last iteration of potentials as the input to my final offline clearing algorithm.

4.5 Offline Clearing Algorithm

The offline clearing algorithm takes in the final learned potentials Θ and the edge weights w_e in order to create a weighted myopic algorithm that first re-weights the edges in each kidney exchange graph according to the standard re-weighting function $f_w(e) = w(e) \cdot (1 - P_{\theta_d} - P_{\theta_p})$ and then chooses a greedy matching on this re-weighted graph during each batch. Because I only allowed swaps in my model, the greedy matching solution is well-defined up to equivalence classes with the same overall score (i.e., it is always possible to achieve an optimal matching under some objective function). This algorithm was then tested on a separate set of kidney exchange graphs in order to get a final score.

Chapter 5

Experimental Results

5.1 Goal

The overarching idea behind my implementation was to eventually observe a tradeoff between timing and the medical objective. In order to do this, I produced two kinds of graphs: summary graphs that show the timing and MaxCard performance of each algorithm, and scatterplots of the number of matches against the inverse average time for each death probability and weighted objective function. The summary plots provide a broad picture of the results, whereas the scatterplots allow us to examine specific tradeoffs in greater depth.

5.2 Methodology

I tested five different combinations of the timing and MaxCard objectives: 0timing, 25timing, 50timing, 75timing, and 100timing. As described in Chapter 4, these represent sliding weighted averages between the two (normalized) objectives.

Additionally, I explored six different death probabilities: $p = 0.001$, 0.003, 0.005, 0.01, 0.03, and 0.05. In general, $p = 0.001$ and 0.003 are considered low probabilities of death, $p = 0.005$ and 0.01 are considered medium probabilities of death, and $p = 0.03$ and 0.05 are high probabilities of death.

I ran exchanges consisting of 500 patient-donor pairs and 1000 timesteps over which they were uniformly distributed to enter the pool. As per [18] and [20], I assumed a constant tissue-type compatibility factor (PRA) of 0.2.

During the SMAC learning step, I repeatedly trained potentials on a set of 1000 training instances. I then evaluated each final offline algorithm on a set of 5000 previously unseen test instances.

5.3 Graphs

In Figures 5.1, 5.2, 5.3, and 5.4, we can see a bit of separation into two main types of algorithms — generally, the more timing-aware algorithms do better on the timing objective than the MaxCard objective, and the less timing-aware algorithms exhibit the opposite tradeoff. However, there is some noise in these plots, especially in 5.2 and 5.4, probably due to the fact that SMAC did not completely converge when training the potentials.

Additionally, the spread in the timing and MaxCard objectives is quite small due to the sparse nature of the kidney exchange graph. Figures 5.1 and 5.3 in particular suggest that there is a tradeoff between algorithms trained with different degrees of timing awareness.

At higher death probabilities, as seen in Figures 5.5 and 5.6, the relationship between timing-aware clearing algorithms and the timing-objective tradeoff begins to break down. This is due to the larger amount of noise due to the relatively high death probabilities.

The error bars on each summary graph represent the standard error in each direction ($S = \frac{s}{\sqrt{n}}$, where s is the standard deviation and n is the number of samples). However, they may be misleading in this case because they cannot fully encapsulate the case in which one algorithm out-performs another on most instances in the test set because they are aggregating averages over a very large test sample. Therefore, in order to examine this behavior more closely, we also look at scatterplots of the difference in performance between different algorithms with respect to the total number of matches and overall timing.

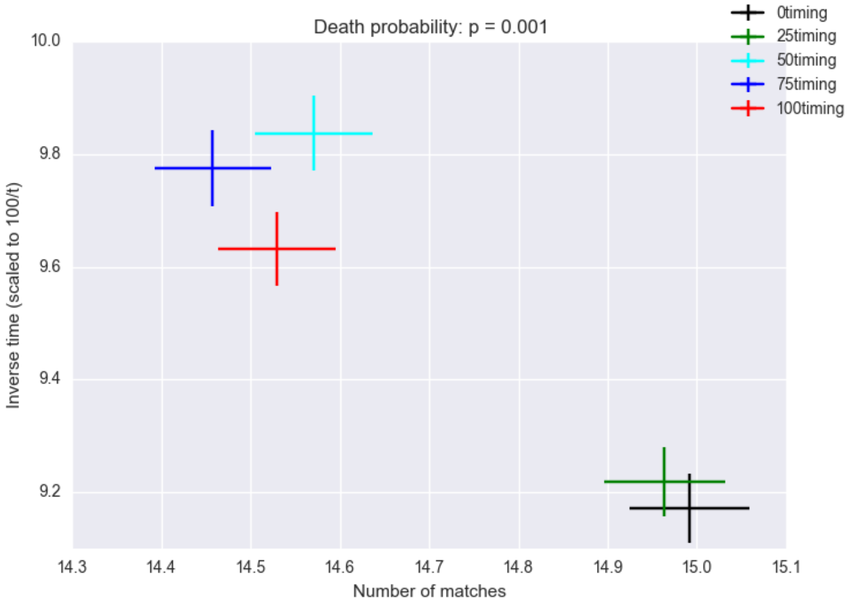

Figure 5.1: The timing-objective tradeoff with a death probability of 0.001.

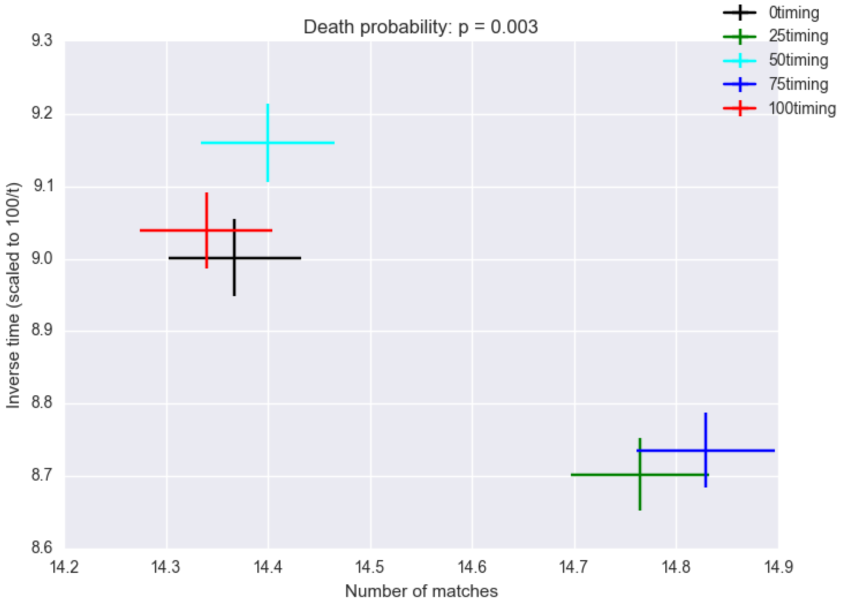

Figure 5.2: The timing-objective tradeoff with a death probability of 0.003.

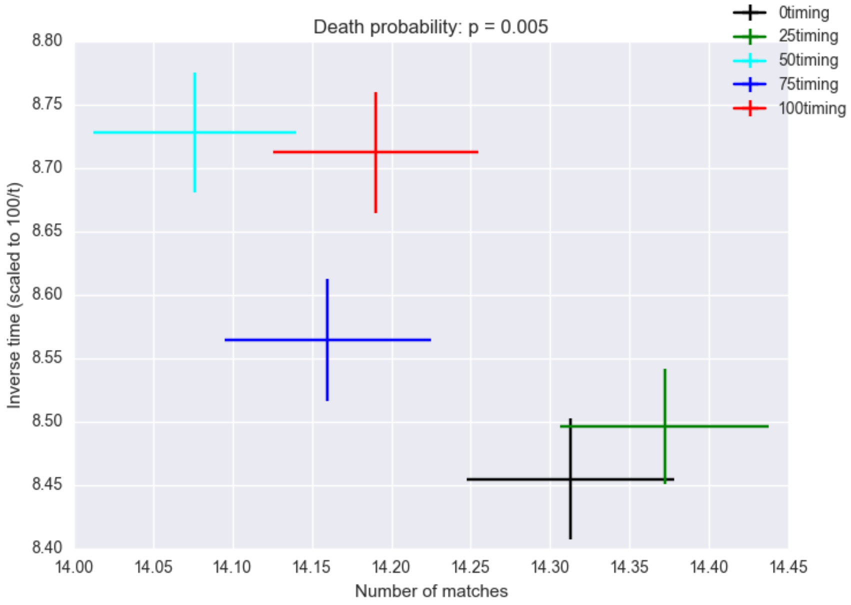

Figure 5.3: The timing-objective tradeoff with a death probability of 0.005.

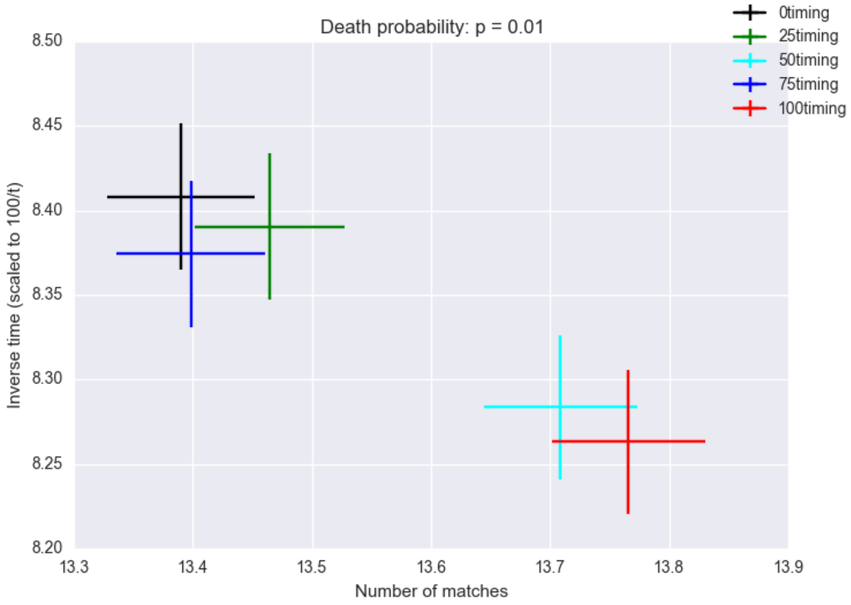

Figure 5.4: The timing-objective tradeoff with a death probability of 0.01.

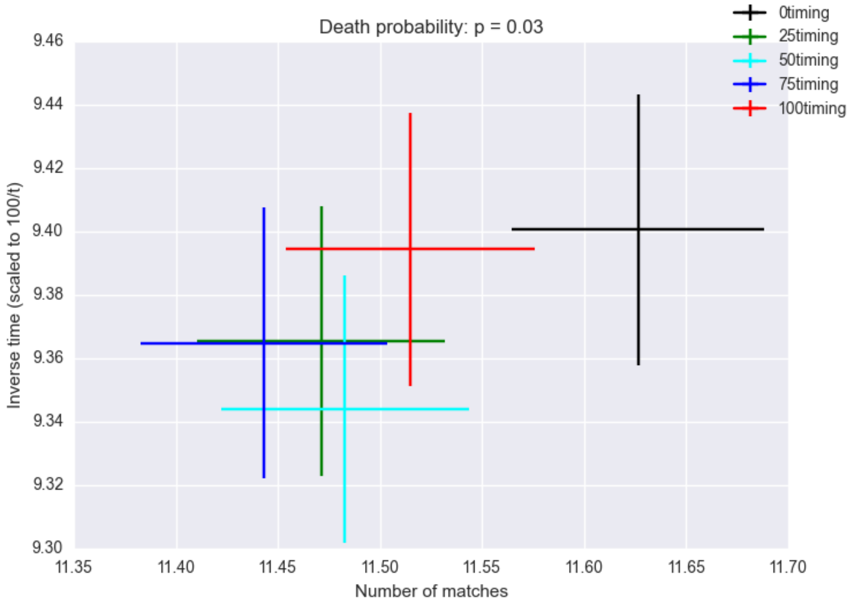

Figure 5.5: The timing-objective tradeoff with a death probability of 0.03.

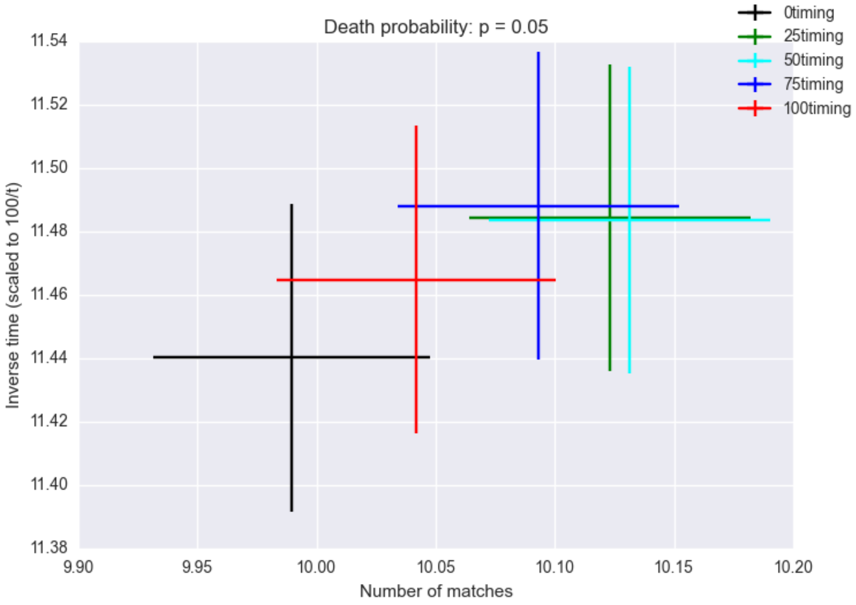

Figure 5.6: The timing-objective tradeoff with a death probability of 0.05.

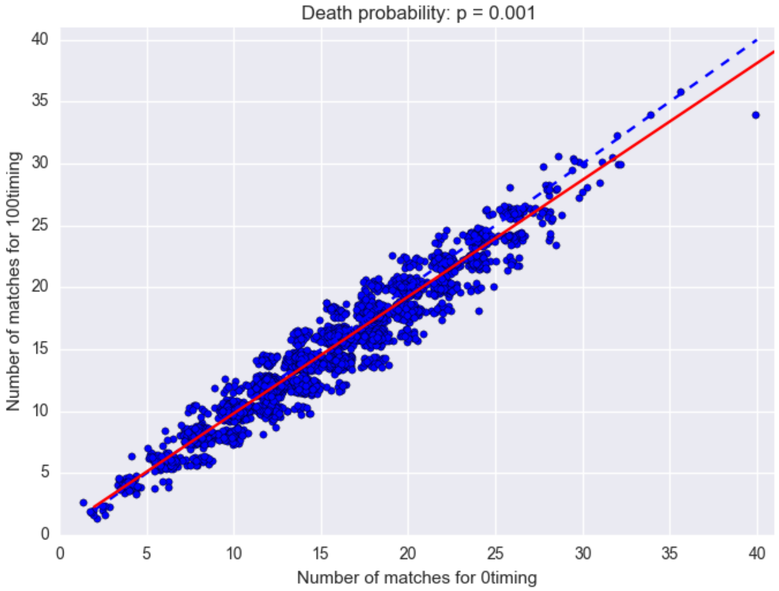

Figure 5.7: Scatterplot comparing the score performance of the 0timing and 100timing algorithms with a death probability of 0.001.

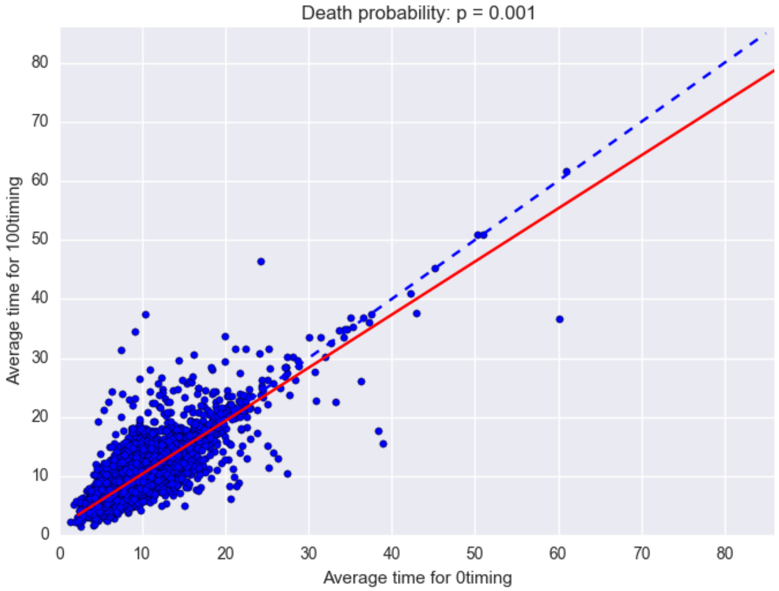

Figure 5.8: Scatterplot comparing the time performance of the 0timing and 100timing algorithms with a death probability of 0.001.

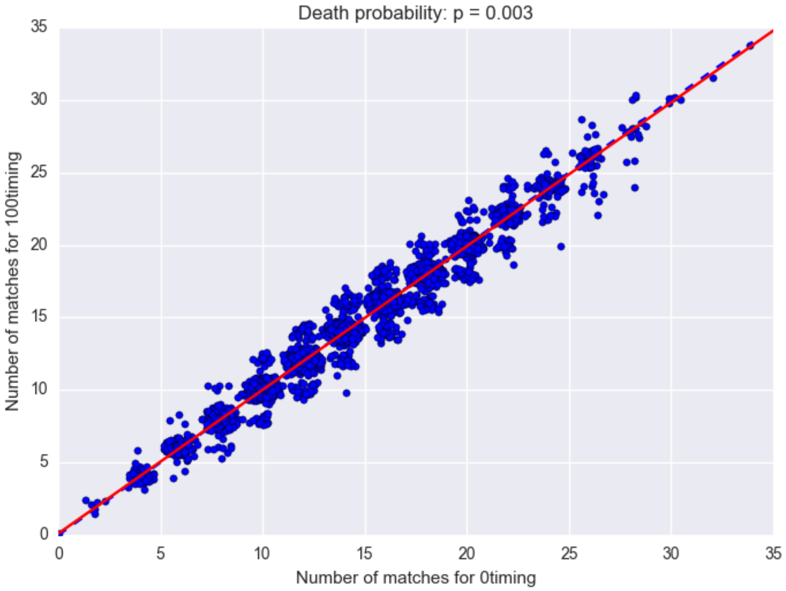

Figure 5.9: Scatterplot comparing the score performance of the 0timing and 100timing algorithms with a death probability of 0.003.

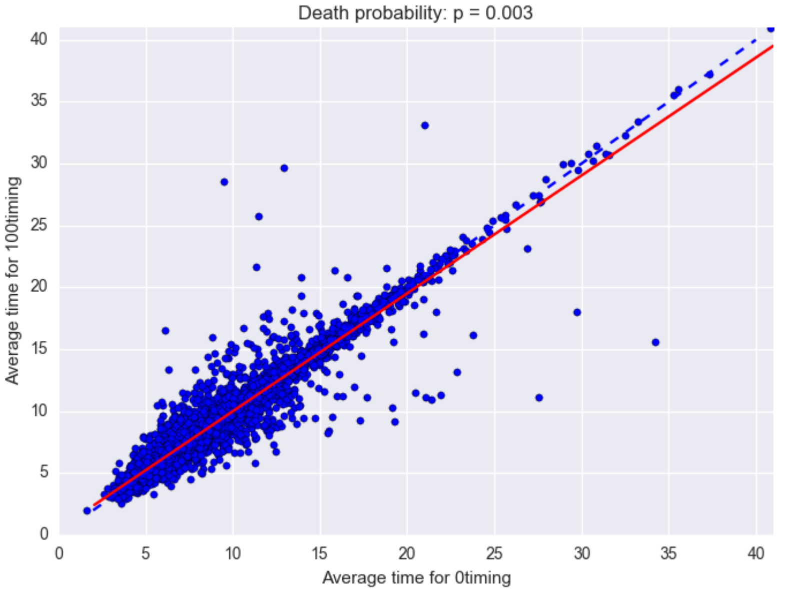

Figure 5.10: Scatterplot comparing the time performance of the 0timing and 100timing algorithms with a death probability of 0.003.

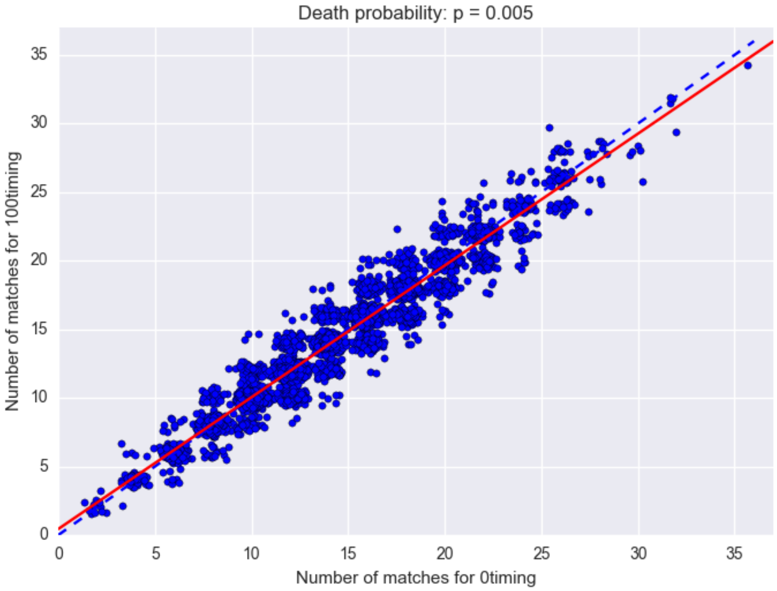

Figure 5.11: Scatterplot comparing the score performance of the 0timing and 100timing algorithms with a death probability of 0.005.

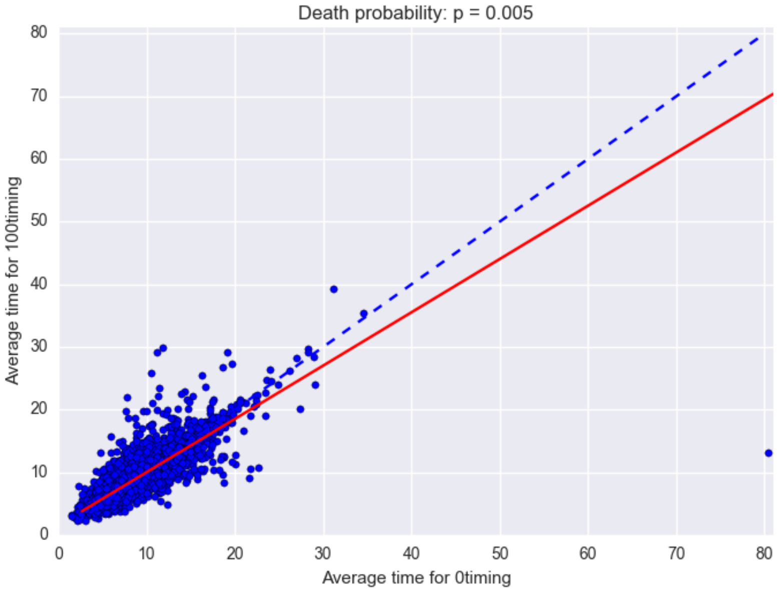

Figure 5.12: Scatterplot comparing the time performance of the 0timing and 100timing algorithms with a death probability of 0.005.

These scatterplots suggest more strongly that at $p = 0.001$, 0.003, and 0.005, timing-aware algorithms outperform timing-unaware algorithms with respect to timing while still achieving comparable MaxCard results. Additionally, even when comparing algorithms in between 0timing and 100timing (e.g,. comparing 25timing with 75timing), we can see that the more heavily timing-aware algorithms often out-perform the others in terms of timing, while performing similarly on the MaxCard objective. For more graphs, please see the Appendix.

Although none of these findings were especially strong, they serve as a hint that further tradeoffs between timing and match quality exist in the FutureMatch framework. These results certainly suggest that the various timing-aware algorithms perform about the same on the offline medical objective but better with respect to the timing metric. Therefore, we can say that perhaps it is possible to design timing-aware algorithms that keep people in exchanges for shorter periods of time while achieving roughly the same performance, which would be a very interesting and potentially useful result.

However, as we raise the death probability, this tradeoff becomes less noticeable. This is likely due to, again, the greater amount of noise in the system at higher death rates, and as the principal dynamic behind exits from the pool switches from matches to deaths, the performance of a timing-aware algorithm matters less as more vertices are forced to die as opposed to being matched. This is because the kidney graphs are relatively sparse to begin with, and therefore at certain levels of death, it is impossible to have a thick enough market to guarantee that any significant portion of them get successfully matched.

Of course, all these observations come with a very basic kidney exchange model. Perhaps introducing three-cycles and altruistic donors, using a richer population model, and implementing a non-uniform PRA model would lead to a different conclusion, but in this limited framework, it seems that there is evidence for a rather successful tradeoff between timing and the MaxCard objective where, at low death probabilities, it is possible to train timing-aware algorithms that result in people spending less time on average in exchanges while resulting in roughly the same cardinality of matches overall.

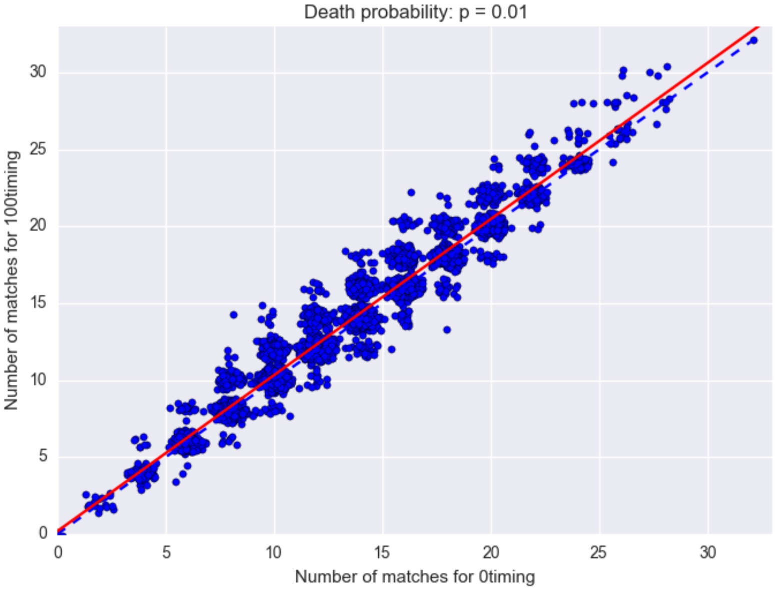

Figure 5.13: Scatterplot comparing the score performance of the 0timing and 100timing algorithms with a death probability of 0.01.

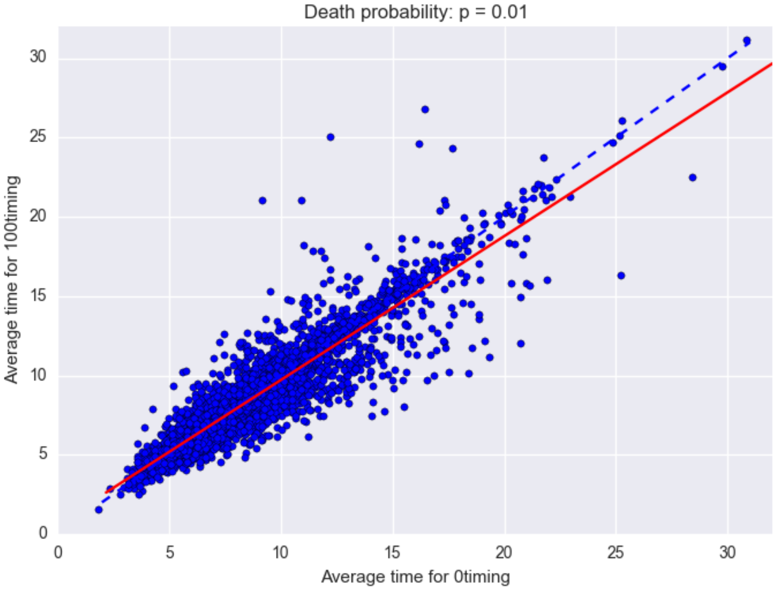

Figure 5.14: Scatterplot comparing the time performance of the 0timing and 100timing algorithms with a death probability of 0.01.

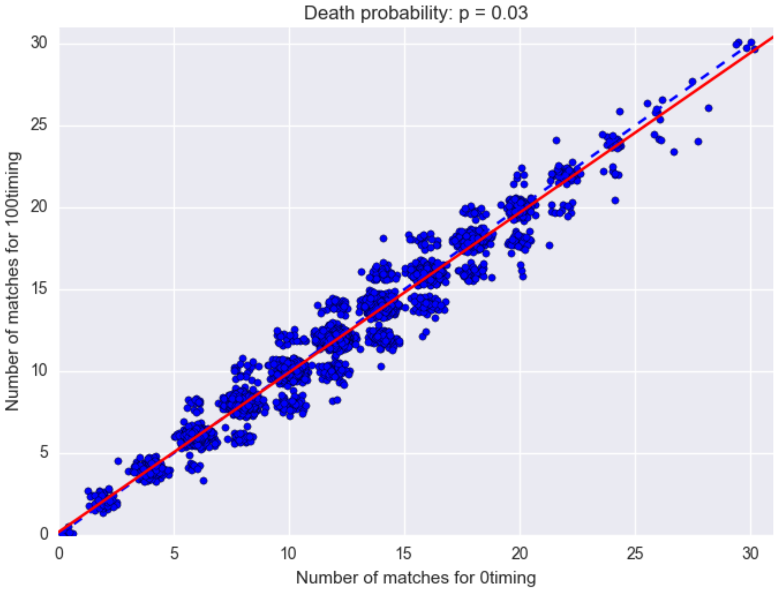

Figure 5.15: Scatterplot comparing the score performance of the 0timing and 100timing algorithms with a death probability of 0.03.

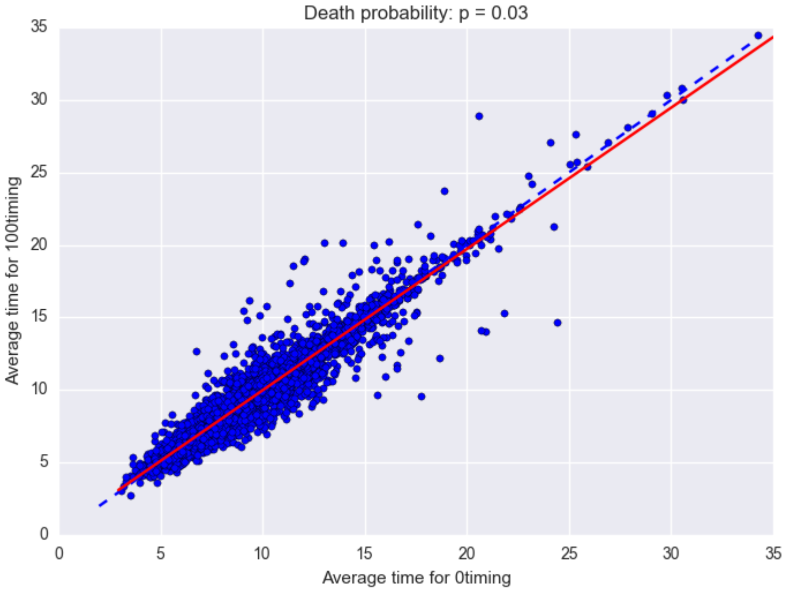

Figure 5.16: Scatterplot comparing the time performance of the 0timing and 100timing algorithms with a death probability of 0.03.

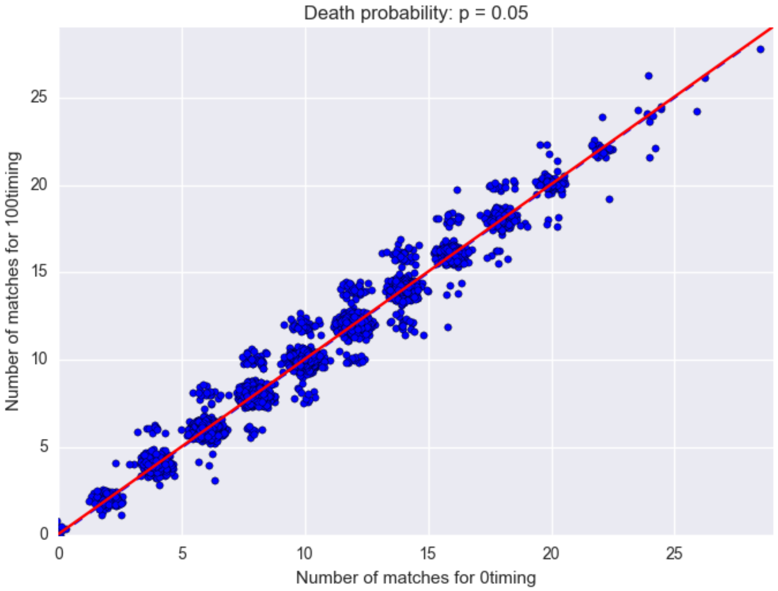

Figure 5.17: Scatterplot comparing the score performance of the 0timing and 100timing algorithms with a death probability of 0.05.

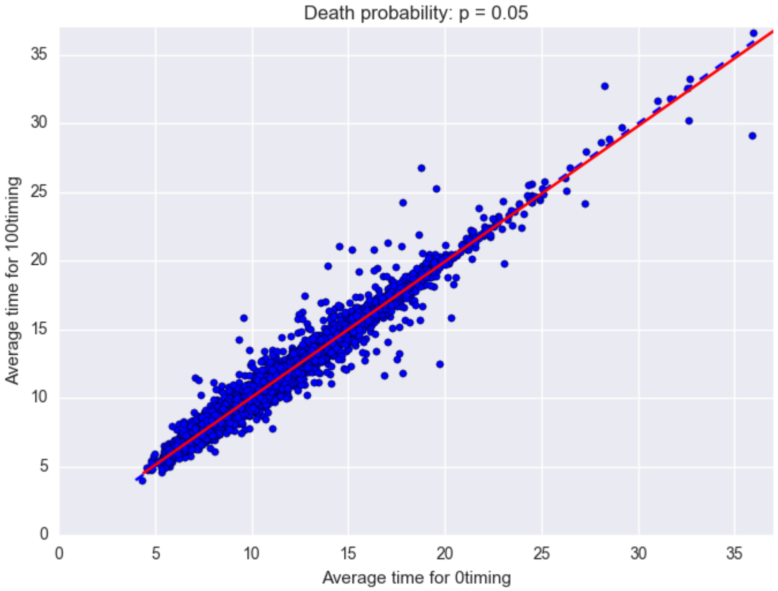

Figure 5.18: Scatterplot comparing the time performance of the 0timing and 100timing algorithms with a death probability of 0.05.

Chapter 6

Discussion and Conclusion

6.1 Relevance

Considering aspects of timing in dynamic kidney exchange is an important problem: not only do timing-aware algorithms reduce the risk of patients dying while waiting for a kidney, they also lessen the burden on physicians — performing a transplant on a healthier patient is both easier and less stressful than on a more fragile patient — and ease patient concerns about not being able to be matched in time. Perceived fairness does a lot to sway public opinion and lead to the adoption of new matching approaches, and incorporating explicit measures of timeliness into current kidney exchange algorithms will do much to ease peoples' minds. Additionally, even small improvements in the average amount of time spent in kidney exchanges are important, as improving matching speed by even a day saves real lives.

Although the model used in this thesis was admittedly simplistic and unrealistic, the implication that there is a possible tradeoff between timing considerations and medically-prescribed objectives could add an interesting dimension to this optimization problem: from a practical standpoint, is there an optimal blend of timing and fairness that leads to some provable level of performance? Or, taking this a bit further into the hands of patients in the exchange, is there perhaps a way for people to collectively specify a particular combination of timing and a medical objective based on an information aggregation-based measure of their preferences?

Additionally, this represents a first step toward combining timing objectives as highlighed in Anderson's theoretical model [3] with the real-world implications of Dickerson et al.'s FutureMatch framework [13], which could potentially lead to extensions down the line that incorporate more involved measures of timing or explicitly model time spent in the exchange in the medically-prescribed objective. It seems that a lot of work has been done on the 'end result' of these exchanges without much thought being given to the actual mechanics within the realm of the exchange itself, and introducing timing considerations in the model will hopefully lead people to not treat the exchange as a black box in terms of peoples' overall well-being. Of course, there are models that can specify the general health of patients in the exchange and which preferentially match them [12], but explictly trying to capture the amount of time people in general spend in the exchange in conjunction with the FutureMatch framework represents a step in a new direction.

6.2 Future Work

Firstly, given more time, I would extend my framework to contain the complete FutureMatch framework. In particular, this means extending to all three fairness objectives and their corresponding edge weights, as opposed to just considering MaxCard. I would also add three-cycles and altruist-initiated chains into my model, which would hopefully allow for the entirety of Anderson's results (i.e., a large decrease in average time spent in the exchange) [3] to propagate to the revised FutureMatch framework.

Additionally, I would simulate a more complex and realistic population, both in terms of blood-type distribution, which varies by race, among other factors, and in terms of tissue-type compatibility, because people aren't uniformly tissue-sensitive. As discussed in [18] and [20], there are three widely-accepted levels of PRA sensitivity, and generating patients and donors with a non-uniform level of PRA sensitivity would result in much more realistic data. It would also allow us to consider measures of fairness between people with different levels of PRA sensitivity, which adds a new facet to the problem: suppose that adding timing objectives forces algorithms to match too quickly, resulting in harder-to-match patients being neglected? Exploring the dynamics in a heterogeneous population would help address additional fairness questions that our previous homogeneous population never encountered.

In my model, the entrances and exits of vertex pairs over time were also highly stylized. One possible next step would be to introduce vertices with different exit functions, which could either be exponential functions with a different parameter or another family of function altogether. Allowing subpopulations to have different lifespan distributions within the exchange could also lead to interesting fairness results; perhaps the presence of a shorter-lived population S would force the algorithm to learn to match only longer-lived subpopulations L, as letting a member of S die would presumably increase the overall average amount of time spent in the exchange by less than if the algorithm matched people in S and let members of L die. A richer set of time features could also yield much more nuanced and interesting results. For example, we could care about the worst-case waiting time instead of the average, or we could try to minimize a combination of the average wait time and the population standard deviation. Additionally, we could introduce a new type of timing-based potential for a set of graph elements (e.g., vertices or edges), which could possibly enrich the expressibility of the timing aspect of the objective. Additionally, the timing and medical objectives could be combined in more complex ways than a simple weighted average in order to reflect the preferences of people in the exchange.

Lastly, it would be interesting to see how this expanded FutureMatch-esque framework dovetails with other aspects of kidney exchange: notably, failure-aware kidney exchange and truthful mechanisms. Although combining everything into one cohesive model could be very tricky due to the different basic assumptions in each case, this could lead to interesting results about the tradeoff between the different approaches themselves. For example, Blum et al.'s adaptive querying algorithm in the failure-aware FutureMatch framework could have more of a relative effect than trying to maximize the number of matches in expectation and not learning from each failed match. Additionally, combining the matching aspect of dynamic kidney exchange with a properly-incentivizing mechanism could lead to better real-world effects, a very important step when deploying these algorithms in reality.

6.3 Conclusion

Theoretical models by Anderson et al. [3] and Akbarpour et al. [2] suggested a tradeoff between the average time patients spent in the exchange and the total cardinality of all matches. I explored this in more detail through the lens of Dickerson et al.'s FutureMatch framework [13], and through the introduction of a coarse timing measure, showed that timing-aware algorithms may be able to reduce the average amount of time spent in the exchange at steady state without significantly reducing the total number of matches.

Chapter 7

Appendix

As mentioned in Chapter 5, the following are other scatterplots of the performance of various timing-aware algorithms on the timing and MaxCard objectives for various death probabilities. These support the observation that more timing-aware algorithms result in more efficient performance on timing objectives without sacrificing much match quality.

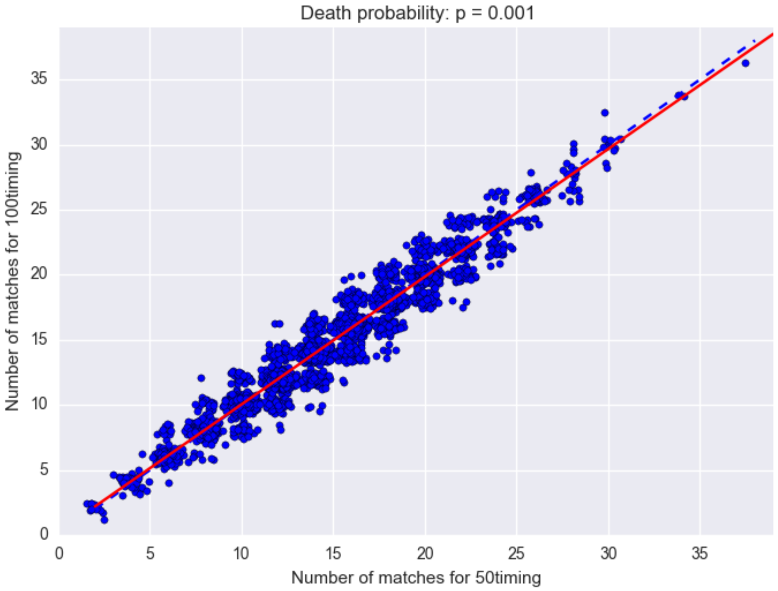

Figure 7.1: Scatterplot comparing the score performance of the 50timing and 100timing algorithms with a death probability of 0.001.

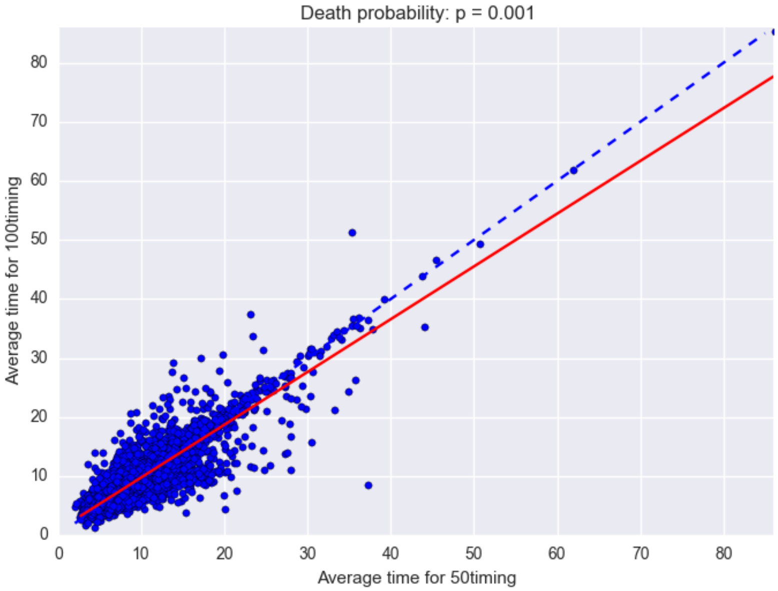

Figure 7.2: Scatterplot comparing the time performance of the 50timing and 100timing algorithms with a death probability of 0.001.

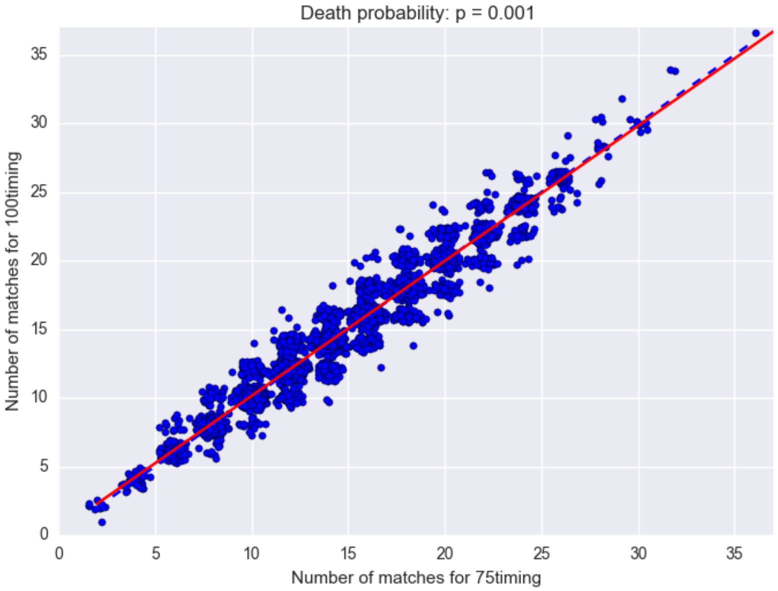

Figure 7.3: Scatterplot comparing the score performance of the 75timing and 100timing algorithms with a death probability of 0.001.

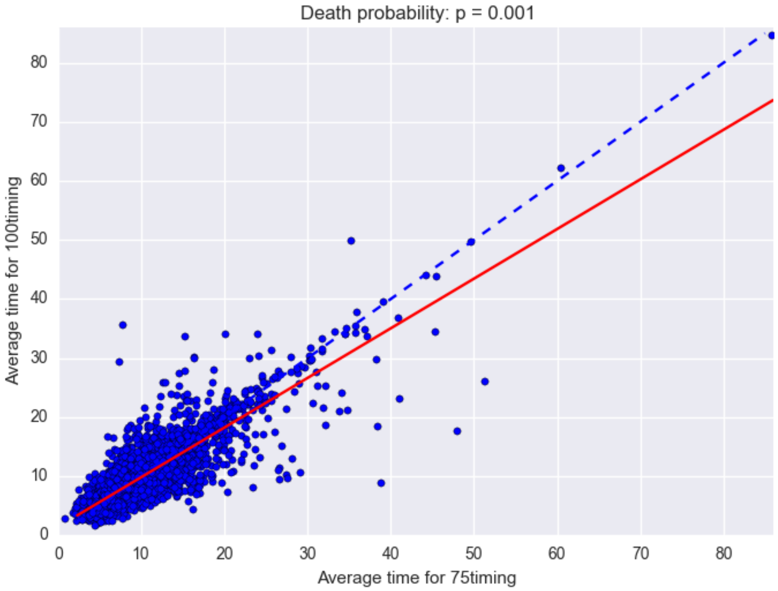

Figure 7.4: Scatterplot comparing the time performance of the 75timing and 100timing algorithms with a death probability of 0.001.

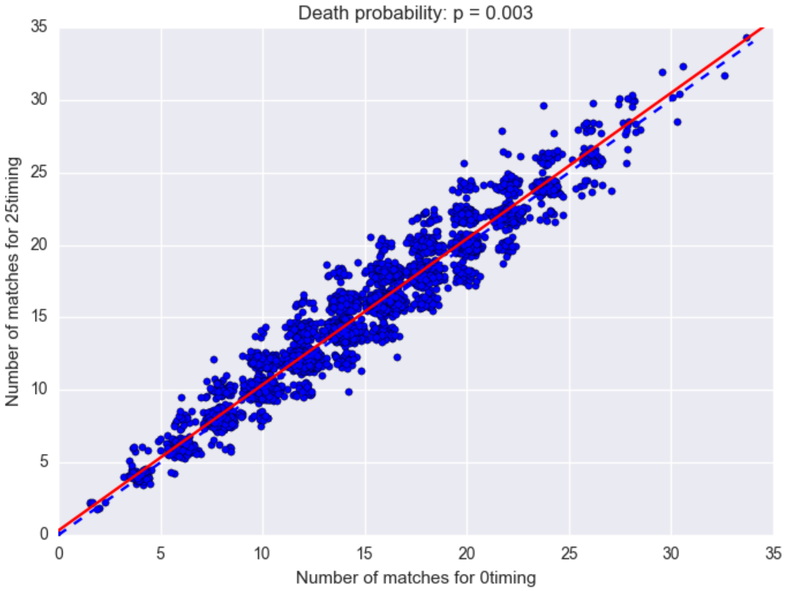

Figure 7.5: Scatterplot comparing the score performance of the 0timing and 25timing algorithms with a death probability of 0.003.

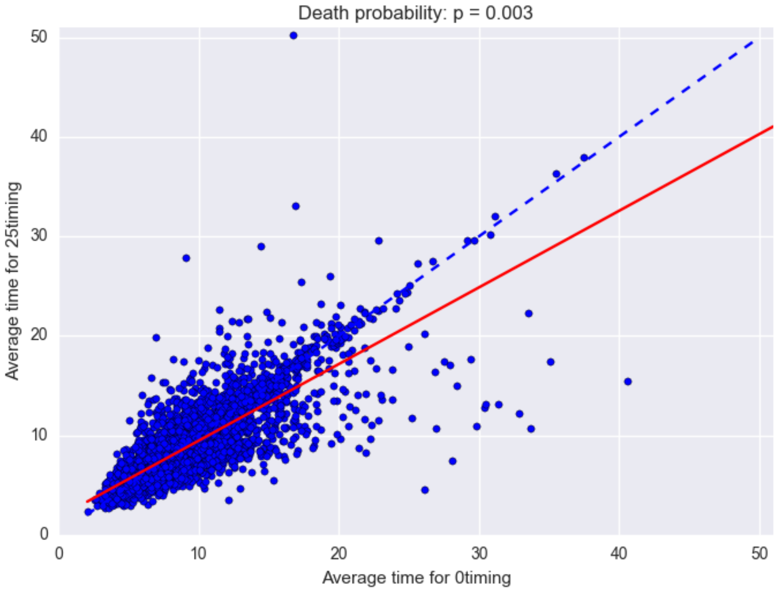

Figure 7.6: Scatterplot comparing the time performance of the 0timing and 25timing algorithms with a death probability of 0.003.

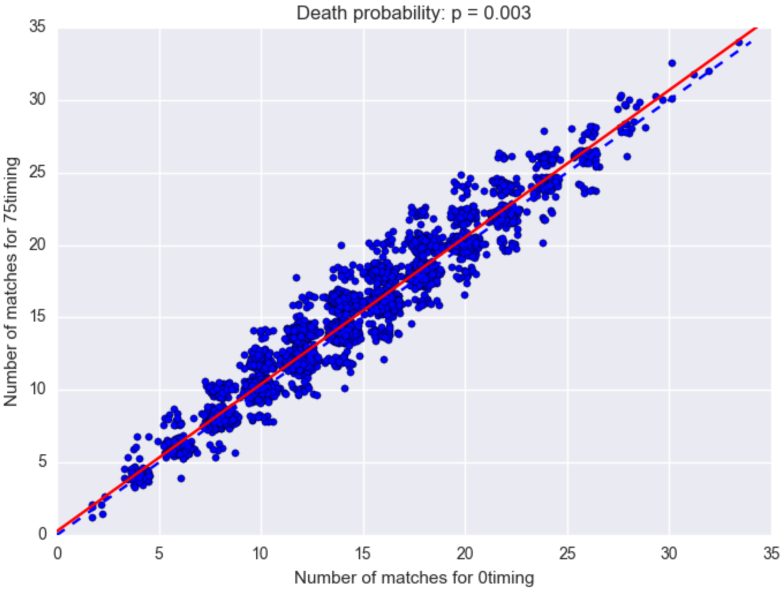

Figure 7.7: Scatterplot comparing the score performance of the 0timing and 75timing algorithms with a death probability of 0.003.

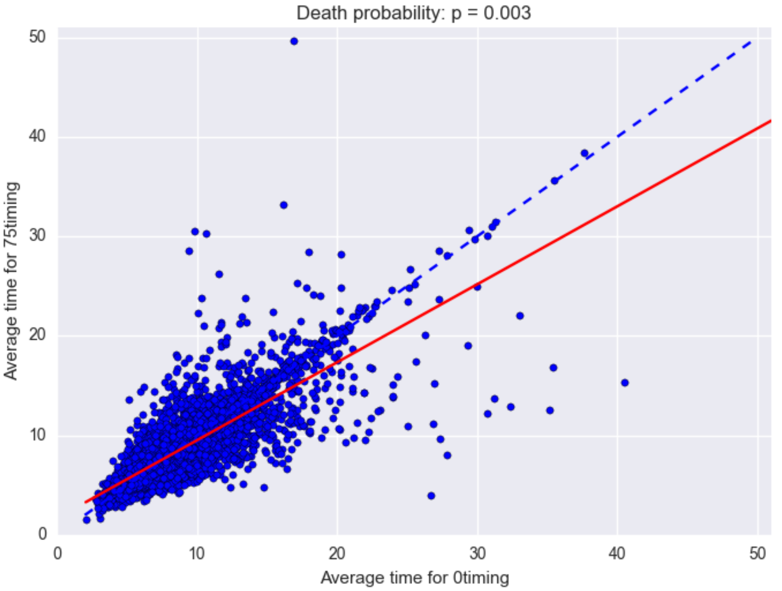

Figure 7.8: Scatterplot comparing the time performance of the 0timing and 75timing algorithms with a death probability of 0.003.

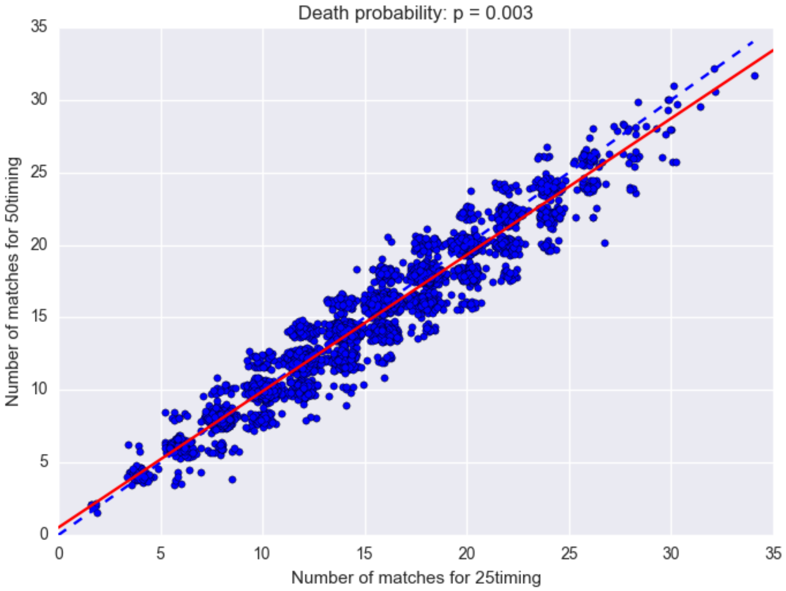

Figure 7.9: Scatterplot comparing the score performance of the 25timing and 50timing algorithms with a death probability of 0.003.

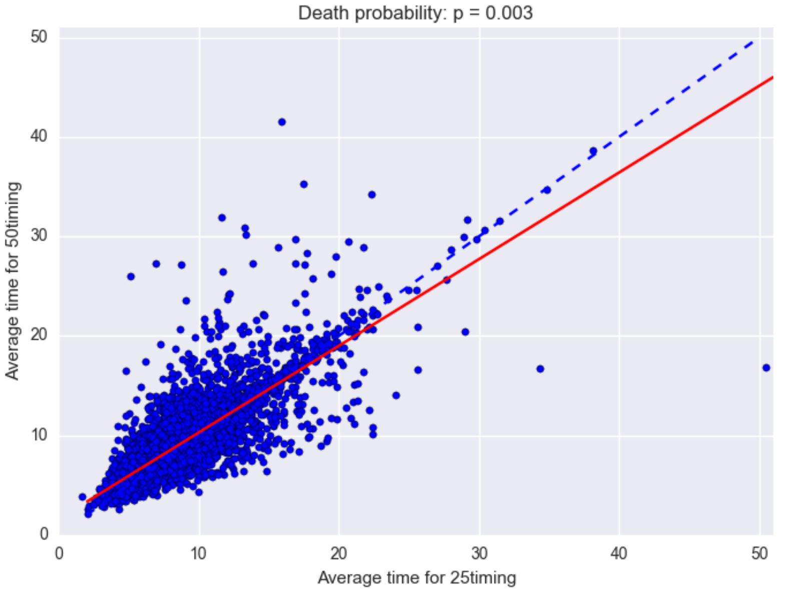

Figure 7.10: Scatterplot comparing the time performance of the 25timing and 50timing algorithms with a death probability of 0.003.

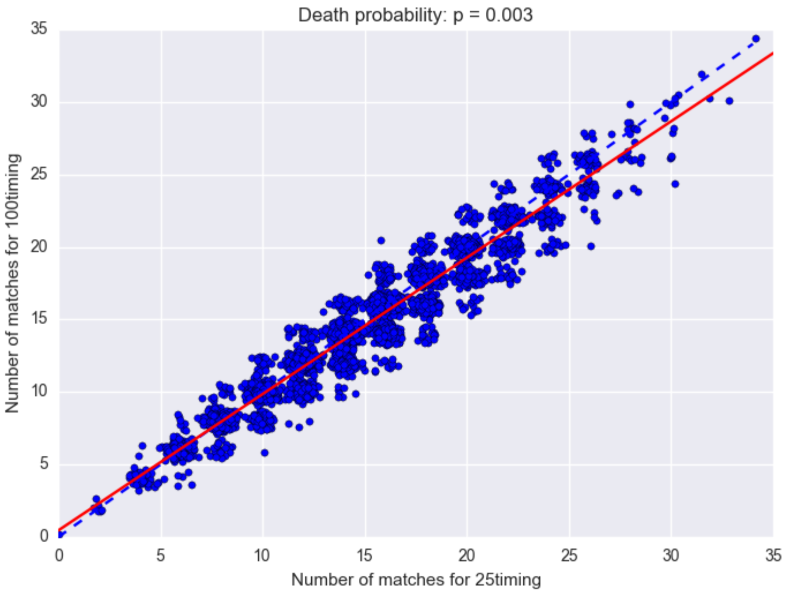

Figure 7.11: Scatterplot comparing the score performance of the 25timing and 100timing algorithms with a death probability of 0.003.

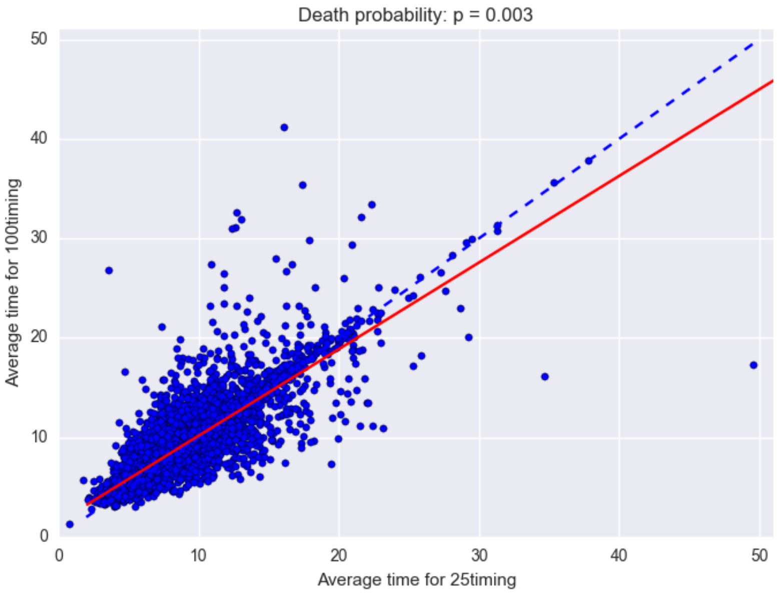

Figure 7.12: Scatterplot comparing the time performance of the 25timing and 100timing algorithms with a death probability of 0.003.

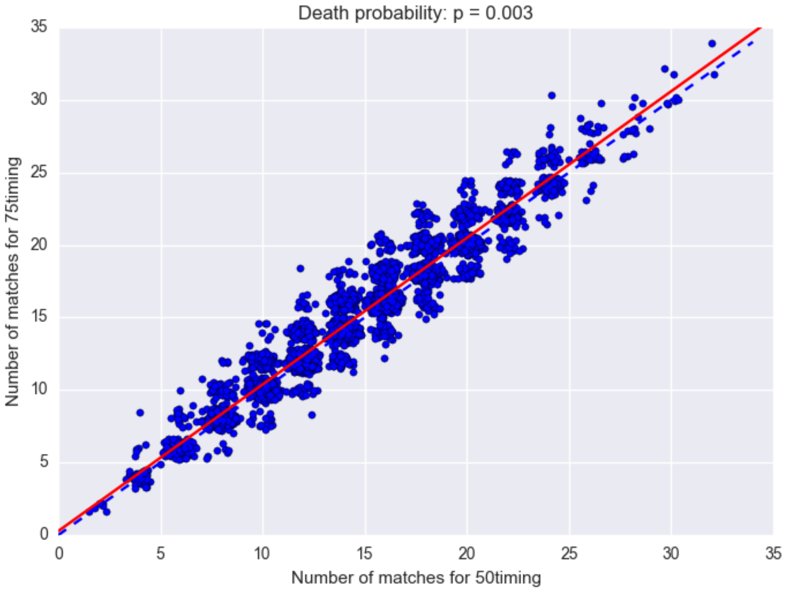

Figure 7.13: Scatterplot comparing the score performance of the 50timing and 75timing algorithms with a death probability of 0.003.

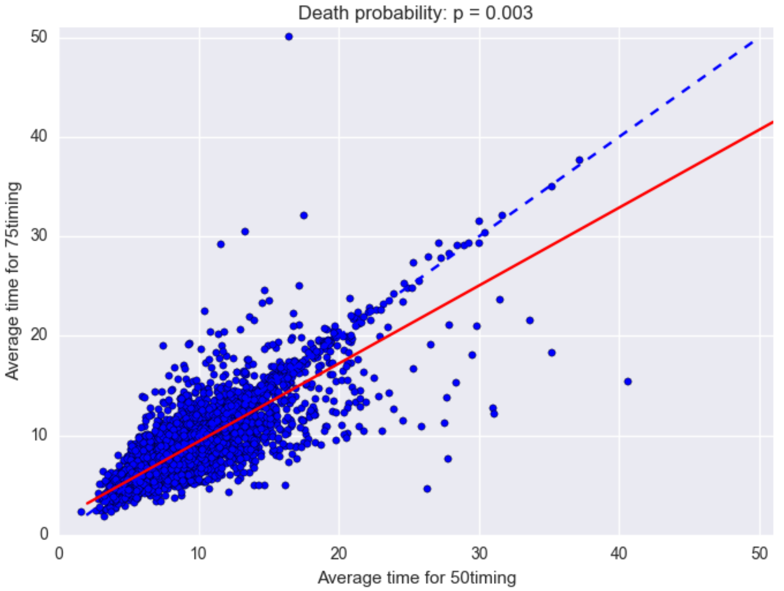

Figure 7.14: Scatterplot comparing the time performance of the 50timing and 75timing algorithms with a death probability of 0.003.

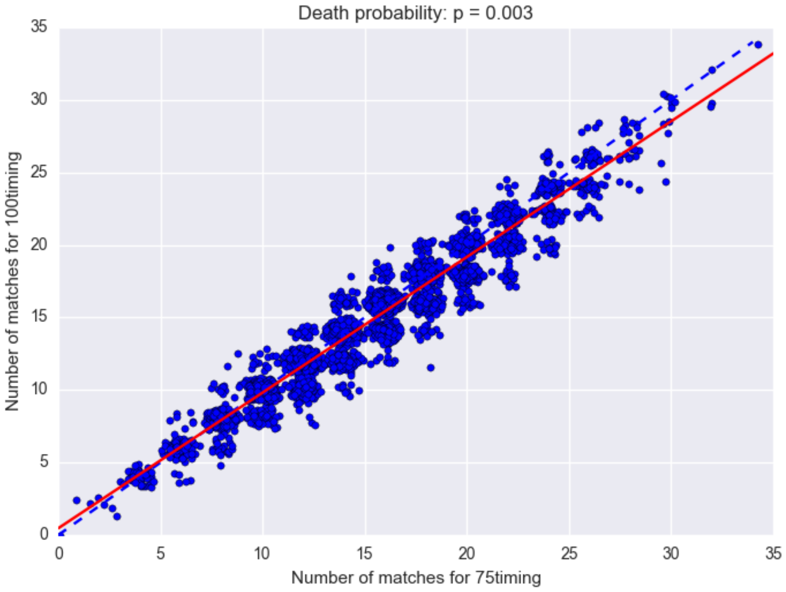

Figure 7.15: Scatterplot comparing the score performance of the 75timing and 100timing algorithms with a death probability of 0.003.

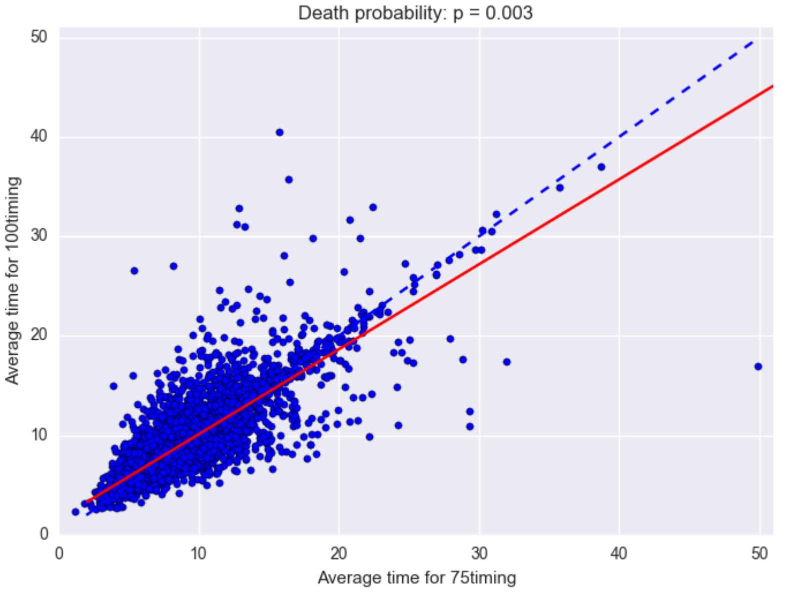

Figure 7.16: Scatterplot comparing the time performance of the 75timing and 100timing algorithms with a death probability of 0.003.

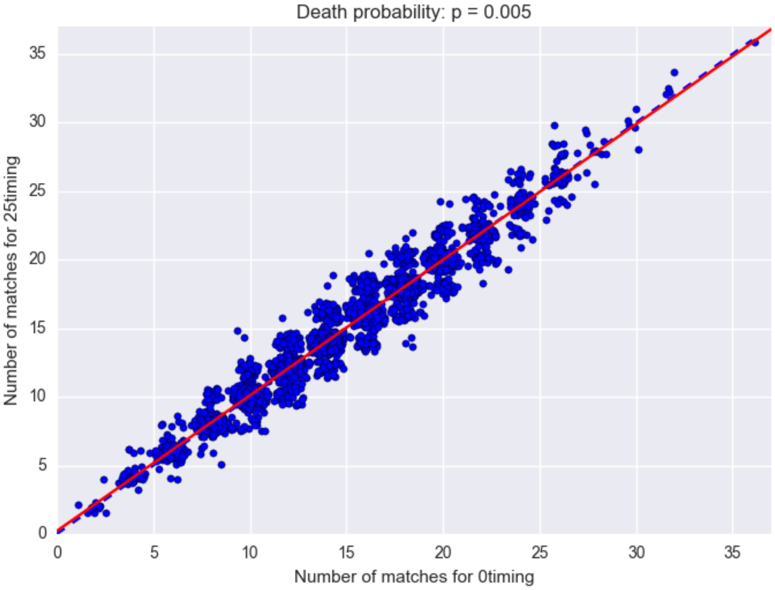

Figure 7.17: Scatterplot comparing the score performance of the 0timing and 25timing algorithms with a death probability of 0.005.

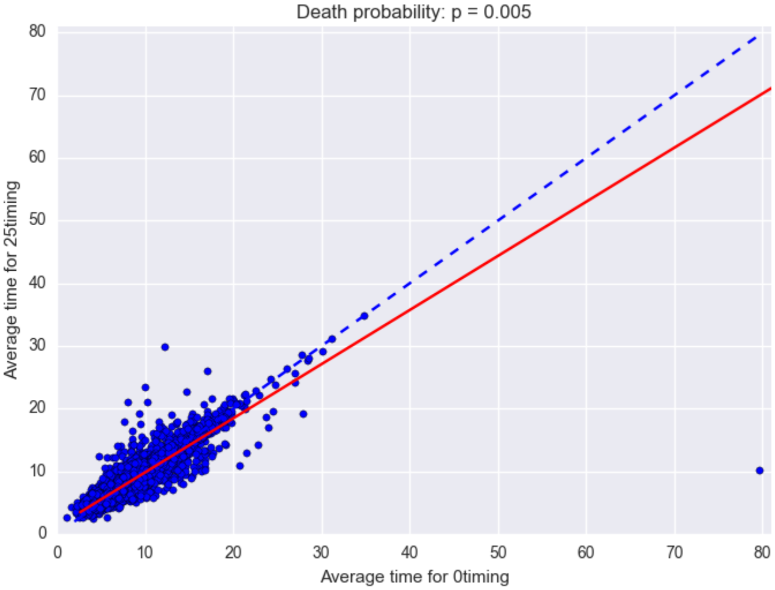

Figure 7.18: Scatterplot comparing the time performance of the 0timing and 25timing algorithms with a death probability of 0.005.

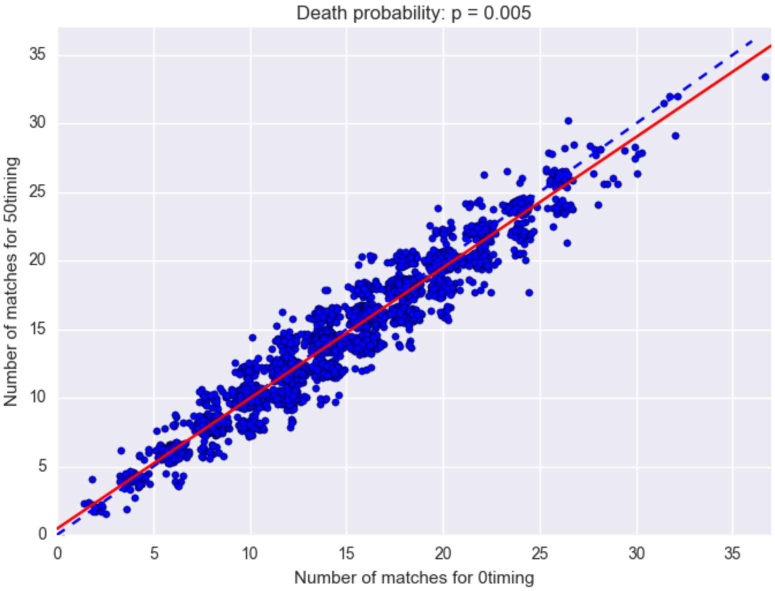

Figure 7.19: Scatterplot comparing the score performance of the 0timing and 50timing algorithms with a death probability of 0.005.

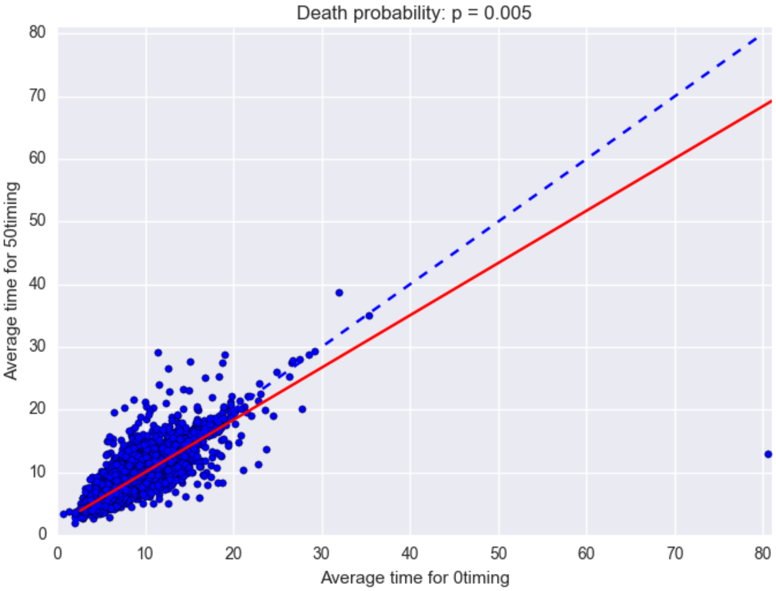

Figure 7.20: Scatterplot comparing the time performance of the 0timing and 50timing algorithms with a death probability of 0.005.

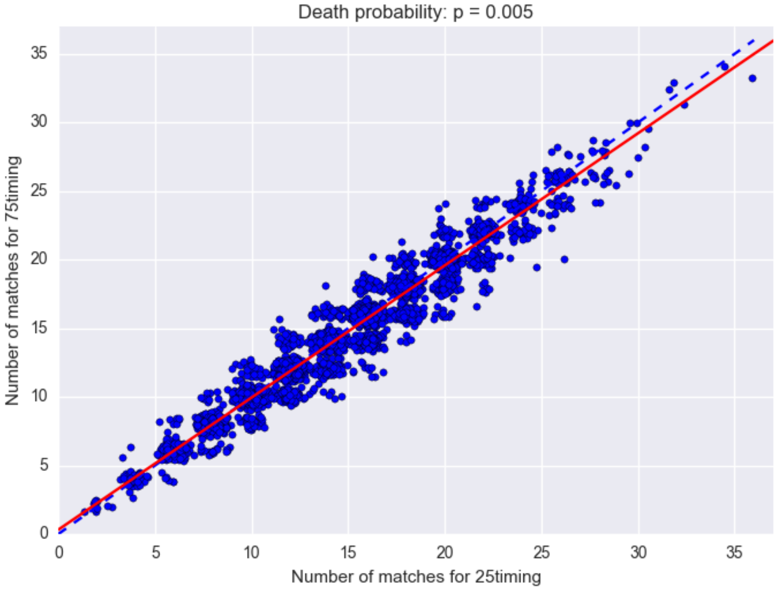

Figure 7.21: Scatterplot comparing the score performance of the 25timing and 75timing algorithms with a death probability of 0.005.

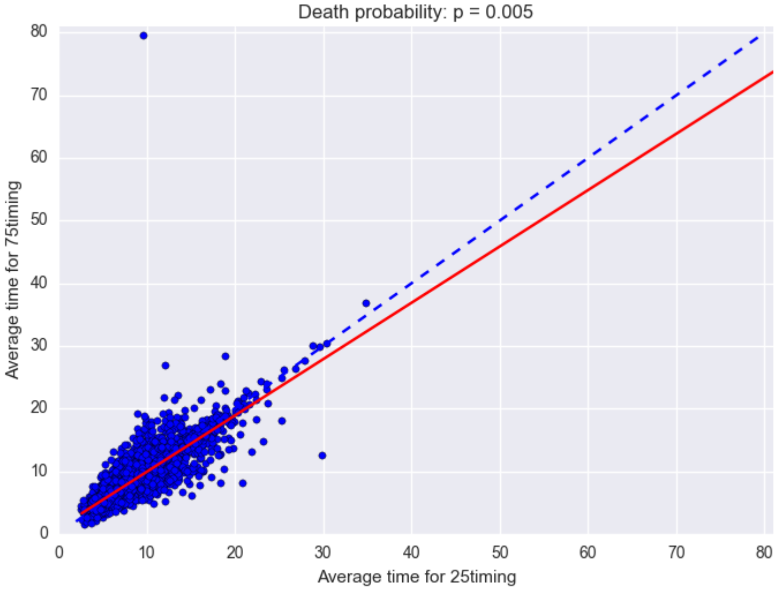

Figure 7.22: Scatterplot comparing the time performance of the 25timing and 75timing algorithms with a death probability of 0.005.

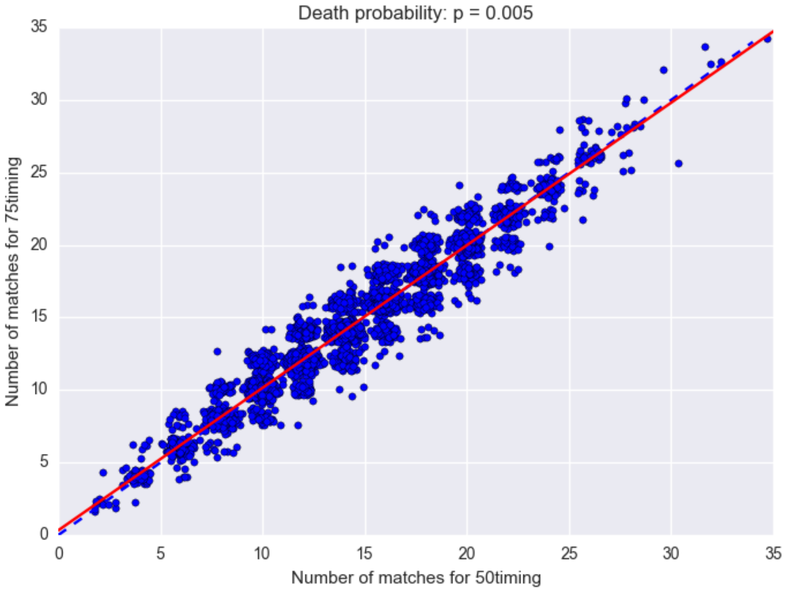

Figure 7.23: Scatterplot comparing the score performance of the 50timing and 75timing algorithms with a death probability of 0.005.

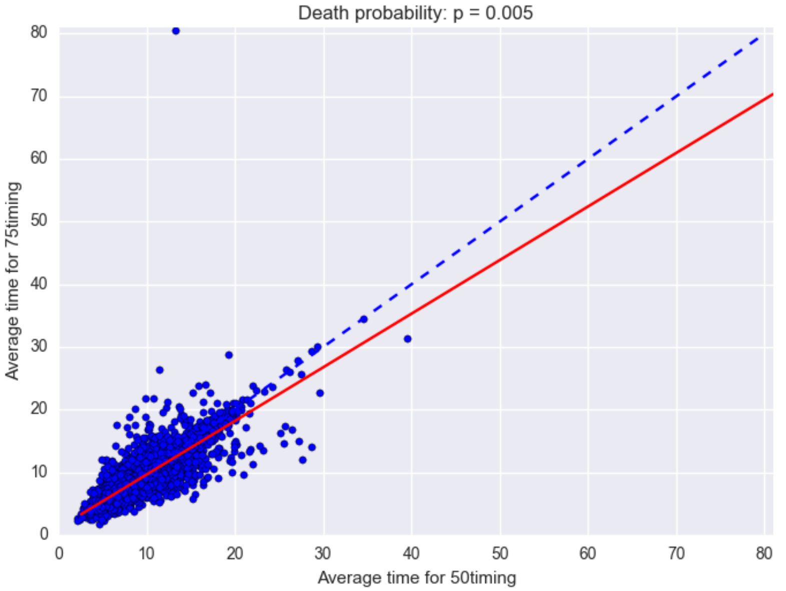

Figure 7.24: Scatterplot comparing the time performance of the 50timing and 75timing algorithms with a death probability of 0.005.

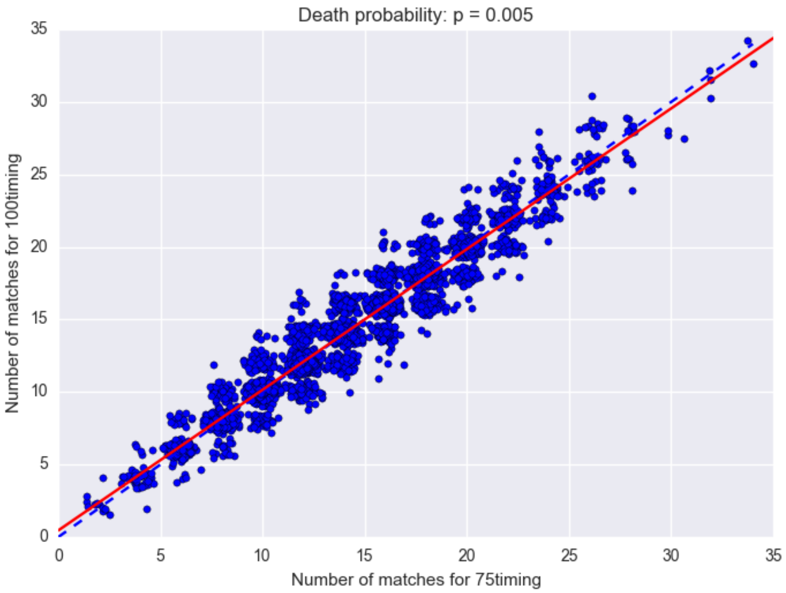

Figure 7.25: Scatterplot comparing the score performance of the 75timing and 100timing algorithms with a death probability of 0.005.

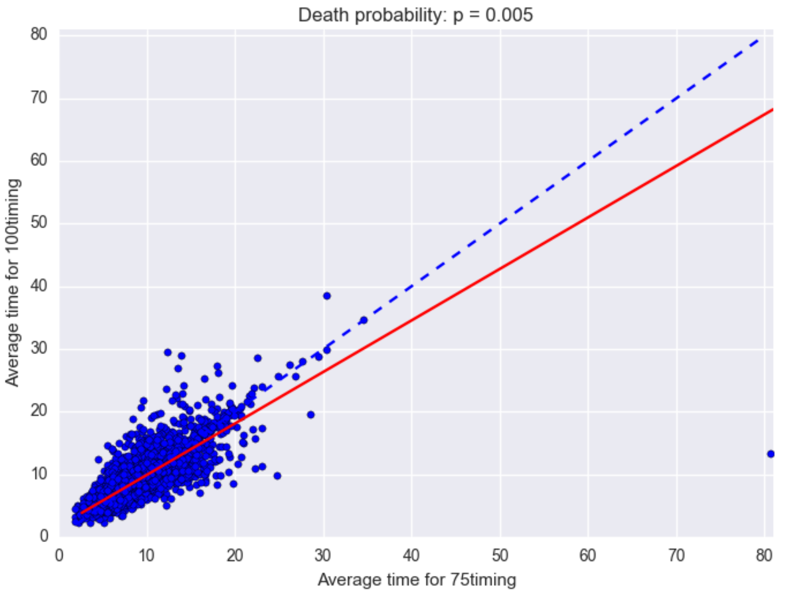

Figure 7.26: Scatterplot comparing the time performance of the 75timing and 100timing algorithms with a death probability of 0.005.

Bibliography

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

[2] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. Dynamic matching market design. In *ACM Conference on Economics and Computation, EC '14, Stanford , CA, USA, June 8-12, 2014*, page 355, 2014.

[3] Ross Anderson, Itai Ashlagi, David Gamarnik, and Yash Kanoria. A dynamic model of barter exchange. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1925–1933, 2015.

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

[5] I. Ashlagi, D. S. Gilchrist, A. E. Roth, and M. A. Rees. Nonsimultaneous chains and dominos in kidney- paired donationrevisited. *American Journal of Transplantation*, 11(5):984–994, 2011.

[6] Itai Ashlagi, Felix A. Fischer, Ian A. Kash, and Ariel D. Procaccia. Mix and match. *CoRR*, abs/1006.1881, 2010.

[7] Pranjal Awasthi and Tuomas Sandholm. Online stochastic optimization in the large: Application to kidney exchange. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 405–411, 2009.

[8] Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15-19, 2015*, pages 325–342, 2015.

[9] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Dynamic matching via weighted myopia with application to kidney exchange. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[10] John P Dickerson, Ariel D Procaccia, and Tuomas Sandholm. Dynamic matching via weighted myopia with application to kidney exchange. In *AAAI*, 2012.

[11] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Failure-aware kidney exchange. In *ACM Conference on Electronic Commerce, EC '13, Philadelphia, PA, USA, June 16-20, 2013*, pages 323–340, 2013.

[12] John P Dickerson and Tuomas Sandholm. Balancing efficiency and fairness in dynamic kidney exchange. In *AAAI Workshop on Modern Artificial Intelligence for Health Analytics (MAIHA)*, 2014.

[13] John P. Dickerson and Tuomas Sandholm. Futurematch: Combining human value judgments and machine learning to match in dynamic environments. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 622–628, 2015.

[14] Jack Edmonds. *Classic Papers in Combinatorics*, chapter Paths, Trees, and Flowers, pages 361–379. Birkhäuser Boston, Boston, MA, 1987.

[15] P Erdos and A Renyi. On random matrices ii. *Studia Sci. Math. Hungar*, pages 459–464, 1968.

[16] Zvi Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Comput. Surv.*, 18(1):23–38, March 1986.

[17] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration (extended version). Technical Report TR-2010-10, University of British Columbia, Department of Computer Science, 2010. Available online: http://www.cs.ubc.ca/˜hutter/papers/10-TR-SMAC.pdf.

[18] Panos Toulis and David C. Parkes. Design and analysis of multi-hospital kidney exchange mechanisms using random graphs. *Games and Economic Behavior*, 91:360–382, 2015.

[19] M Utku Ünver. Dynamic kidney exchange. *The Review of Economic Studies*, 77(1):372–414, 2010.

[20] S. A. Zenios, E. S. Woodle, and L. F. Ross. Primum non nocere: avoiding harm to vulnerable wait list candidates in an indirect kidney exchange. *Transplantation*, 72(4):648–654, Aug 2001.