CHALMERS
UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Minimizing search time for finding an effective treatment

Learning a near-optimal policy using constrained algorithms, approximations, and causal inference

Master's thesis in Computer science and engineering

Samuel Håkansson & Viktor Lindblom

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Minimizing search time for finding an effective treatment

Learning a near-optimal policy using constrained algorithms, approximations, and causal inference

Samuel Håkansson & Viktor Lindblom

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Minimizing search time for finding an effective treatment
Learning a near-optimal policy using constrained algorithms, approximations, and causal inference
Samuel Håkansson & Viktor Lindblom

Supervisor: Fredrik Johansson, Department of Computer Science and Engineering
Examiner: Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Treatments. How should the right one be found?

Typeset in LaTeX
Gothenburg, Sweden 2020

Minimizing search time for finding an effective treatment
Learning a near-optimal policy using constrained algorithms, approximations, and causal inference

Samuel Håkansson & Viktor Lindblom
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Patients sometimes have to try several treatments before the one that best alleviates their symptoms is found. Since each trial of an unsuccessful treatment can be both costly and prolong patient suffering, making this search as efficient as possible is of great importance. We have developed a solution in two parts. (i) A constraint that balances the need to find a better treatment versus the desire to minimize the number of treatments tried. (ii) A dynamic programming algorithm and a greedy algorithm that uses the constraint for finding a policy that finds a good treatment in as few trials as possible. We also develop different methods of estimating potential outcomes and computing the constraint. The algorithms are trained on observational data using causal inference to learn a policy based on true causal effects. The novel algorithms are then evaluated and compared to baseline algorithms on synthetic and real-world antibiotic resistance data.

Keywords: Machine learning, dynamic programming, causal inference, optimal decision-making.

Acknowledgements

We wish to express our deepest gratitude to our supervisor Fredrik Johansson for helping us throughout the project with his enthusiasm and encouragement, and never hesitating to tutor. We would also like to thank our examiner, Devdatt Dubhashi, for his support.

Samuel Håkansson & Viktor Lindblom, Gothenburg, June 2020

Contents

List of Figures

List of Tables

Notation

The notation presented here will be used throughout the rest of the thesis.

A subscript t means the variable at trial t, e.g. $P(A_t = a_i)$ means the probability that the treatment assigned at trial t is a_i.

X	Measured covariates of a subject
A	Treatments
Z	Unmeasured moderators
$Y(a)$	Potential outcome of treatment a
Y^a	Actual outcome of treatment a
h_t	The ordered history, e.g. pairs of treatments and outcomes, including the covariates $x, (a_{t-1}, y_{t-1})...(a_0, y_0)$, to trial t
T	Number of trials
δ	Threshold for better treatment
ϵ	Cost of changing treatment
$\overline{A}_s$	Sequence of treatments $(A_1 \dots A_s)$

Potential outcomes and unmeasured variables are explained in section 2.1.1 and section 3.1, respectively.

1

Introduction

When a patient presents with a medical condition, sometimes several different treatments have to be tried before the treatment that best alleviates the patient's symptoms is found [2]. These conditions could both be chronic, where different treatments are tried over a long time, or acute, where a sufficient treatment has to be found quickly. Trying different treatments can be seen as a search for the most effective treatment for this patient. This search should find a treatment that is as good as possible, while still being as short as possible, since each unsuccessful treatment tried can be both costly and prolong patient suffering.

A balance has to be struck between the desire to find the best possible treatment and the desire to try as few treatments as possible. To guarantee that the most effective treatment is found, all available treatments can be tried, and then the most effective one is selected. On the other hand, to have the shortest possible search time, only one treatment should be tried. Deciding the balance between minimizing the number of trials and the desire to find the most effective treatment is not trivial. Thus some rule for when to continue searching and when to settle for the currently best found treatment is needed.

1.1 Context

The search for an effective treatment can be seen as a sequence of actions, controlled by a policy that recommends either which treatment to try next or to terminate the search. The policy should try as few treatments as possible while still finding an effective treatment. To achieve this, it has to account for differences between patients, as well as how the patients have reacted to the already tried treatments. The policy also has to weigh the value of information gained by trying a sub-optimal treatment.

Policies that individualize treatments are often called Dynamic Treatment Regimes (DTR) in statistical and medical literature [3]. The purpose of a DTR is to tailor the treatment types and dosages to the patient's changing state during some time to ensure that some value is maximized or minimized once all treatments are done. There has been previous work done in finding optimal treatment policies us-

ing Reinforcement Learning (RL) [4, 5, 6, 7, 8, 9]. For example, Huang et al. uses a modified Q-learning algorithm to reduce accumulated bias to solve the problem of accumulated outcome of treatments [4] and Tao et al. uses tree-based reinforcement learning with a purity measure to estimate the optimal strategy [5]. While there are several challenges in using RL in medical settings [10], the technique also has a lot of promise. The methods mentioned above are focused on finding the optimal outcome or reducing the regret over a fixed number of interventions. Other work in this field focuses on minimizing asymptotic regret, which is irrelevant in our setting where we evaluate a small number of actions. In contrast, our purpose is to find a near-optimal final treatment while minimizing the number of interventions done to reduce cost and patient suffering.

Learning optimal treatment policies requires access to data, representative of the model that should be learned, to both train and evaluate the algorithms. While it is possible to gather such data through randomized experiments, it can be both unethical and expensive to do so in medical settings, especially for sequential treatments. Instead, one can use already existing data from a non-interventional setting, so-called observational data. However, using observational data means that the patterns found might be associative rather than causative, and care must be taken to find the causal effects of treatments. This can be done using causal inference [11].

1.2 Problem

The main goal of the thesis is to develop algorithms that, given an observational data set, finds a policy for quickly finding effective treatments for a population of patients. The policy has two goals; to find an as effective treatment as possible and to try as few treatments as possible. As stated earlier, these two goals are at odds. An exhaustive search will always find the best possible treatment and a one-step search will always have the minimum possible number of trials. Thus a trade-off has to be made between the search time and the efficacy when developing the algorithms.

1.3 Contributions

Instead of directly weighting the search time against the efficacy, a constraint for when the algorithm is allowed to stop searching for a better treatment is developed. This constraint causes the algorithm to keep searching until the probability of finding a better treatment is low. The constraint is realized using different methods to approximate the probability of finding a better treatment. Then, two policy optimization algorithms are implemented, both with and without the constraint. The algorithms include greedy variants as well as algorithms based on reinforcement learning. Since there is no natural best trade-off between time and efficacy, the variants are compared on their search time and mean efficacy under different conditions to evaluate their performance.

To perform the evaluation, both real-world data and generated synthetic data are used. The real-world data is of antibiotic resistance, taken from the MIMIC-III data set [12]. Several different data sets of synthetic data are generated, both to highlight differences in the capabilities of the algorithms and to provide more varied data to evaluate on.

The main contributions of the thesis are:

- A constraint balancing the need to find a better treatment versus the desire to stop searching as quickly as possible.
- An algorithm using the constraint to find a policy that in turn finds a good treatment in the minimum number of trials.

1.4 Limitations

In order to reach the goal of the thesis, some limitations were also set. The thesis limits itself to a small number of treatments, outcomes, and measured covariates. This ensures avoidance of a combinatorial explosion when trying to adjust for all possible treatment and outcome combinations. In the same vein, only discrete values are considered to avoid having to apply function approximation to all calculations. Function approximation is used in the thesis, but as an option rather than as a requirement. Another limitation is that only settings where the state of the patient does not change was studied, except for the outcome of the tried treatments.

2

Background

In this chapter, the background and context for the methods developed in this thesis are presented, as well as the theory behind the concepts that will be used in the Methods and Results chapters.

2.1 Causality

A concept that is often repeated is that correlation does not always equal causation. This statement means that while two variables have very similar trends, there might not be any cause-and-effect relation between the variables, as demonstrated in Figure 2.1 where margarine consumption is closely correlated to the divorce rate in Maine. Because of this, it is possible to perform a study and come to a conclusion that would not help in decision-making. For example, that a ban on sales of margarine would strengthen marriages.

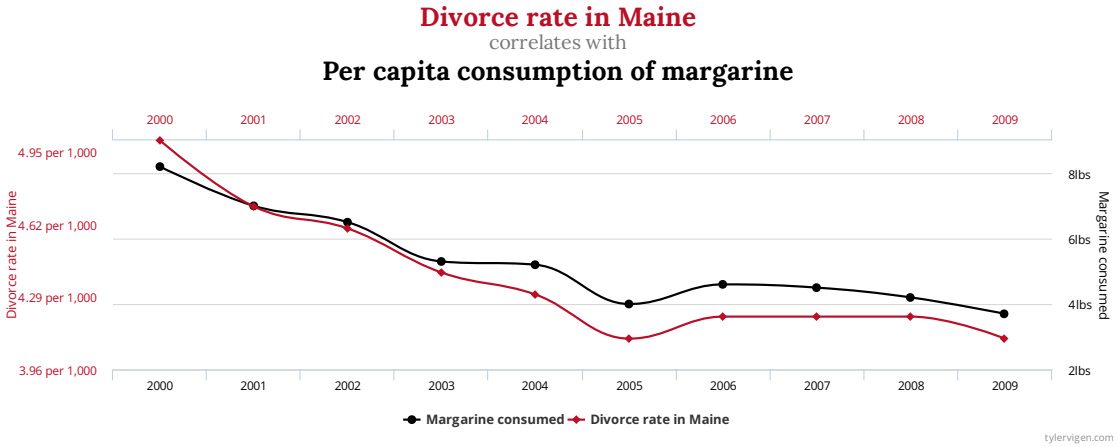


Figure 2.1: Correlation does not always equal causation. Image retrieved from https://www.tylervigen.com/spurious-correlations. Data sources: National Vital Statistics Reports and U.S. Department of Agriculture.

The concept of causality is very important when you want to discover the effect of some intervention done on a population of individuals [13, Chapter 3]. If the effect of some intervention was to be examined, the gold standard would be to conduct an

	A	B
Small	93% (81/87)	87% (234/270)
Large	73% (192/263)	68% (55/80)
Success rate	78% (273/350)	83% (289/350)

Table 2.1: Confounding by size of kidney stone. The success rate of treatment A is lower than B, even though A is better for both small and large stones.

experiment where the researcher can randomly assign individuals to either receive the intervention or be in the control group. After the experiment, the difference between the two groups can be measured, and the effect of the intervention can be calculated. Since the only difference between the groups is the intervention, any change in outcome has to be because of it [13, Chapter 2]. Thus, the causal effect of the intervention can be established and used to inform decisions. Unfortunately, in some instances, for example in medical settings, it is either very expensive or unethical to collect completely randomized data. As such, researchers often have to use pre-existing data in their studies, so-called observational data [14]. In observational data, there might be factors contributing to both which intervention the individual is receiving as well as the outcome of that intervention. Factors that both affect the assignment of an intervention as well as the outcome are called confounders and can confuse the conclusions of a study if they are not accounted for. This makes conclusions drawn from observational studies risky to use unless proper causal reasoning has been applied when calculating and interpreting the results of the study.

Example. Consider the setting where there are two treatments, A and B, for patients with kidney stones [15]. Treatment A is mostly assigned to people with large kidney stones while B is mostly assigned to patients with small ones. As can be seen in Table 2.1, A works better than B for both small and large kidney stones. However, if the size of the kidney stone is not taken into consideration, B has a higher success rate and appears to be better when comparing the whole population. This happens because the treatments are assigned unevenly to patients with small and large stones, causing the combined statistics to be dominated by A for large stones and B for small stones, where patients with large kidney stones have an overall lower success rate.

To help reason about a causal problem, a model graph of the covariates, treatments, and outcomes is useful [16]. Such models can help visualize the causal assumptions made and which variables have to be accounted for to avoid confounding [13, Chapter 7]. Arrows between nodes indicate a causal relationship between the variables in the nodes, with the direction of the arrow indicating the direction of the causality. If there exists a path between the intervention and the outcome even after all outgoing arrows from the intervention are removed, then there exists a so-called *backdoor path* from the intervention to the outcome. The presence of a backdoor path means that there exists confounding on the association between the intervention and the outcome of interest. It is possible to account for this confounding by adjusting for a variable on the backdoor path, thereby *blocking* it [17, Chapter 3].

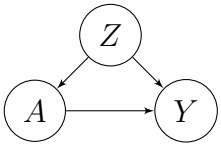


Figure 2.2: A simple causal graph representing the example in Table 2.1.

Using the previous example, the size of the kidney stones is the confounder Z, which affects both the treatment A and the outcome Y, as seen in Figure 2.2. The backdoor path through Z can confound the results of the study unless accounted for. When you block the backdoor path by splitting the data, thereby accounting for the size of the kidney stones, it can clearly be seen that treatment A is overall better.

There are several methods to account for confounding when all backdoor paths can be blocked by measured variables, some examples include: regression, inverse probability of treatment weighting, and standardization. An introduction to regression in a causal setting can be found in Section 2.1.3.

While this introduction to causality is mostly focused on the setting where a single treatment either is given or not, and we then want to calculate the causal effect of that treatment, causality can also be used to compare the effects of two treatments as in the example with the kidney stones or compare the effect of different sequences of treatments.

2.1.1 Potential outcomes

Data gathered from observational studies, so called observational data, is in general incomplete. It is not possible to observe both what would have happened if an individual received an intervention and what would have happened if it did not. In the setting with sequential treatments, not all treatments will be tried for all individuals, so outcomes of treatments that were not tried will not be known. Since not all outcomes are observed, it is impossible to compare the causal effects of different treatments at the individual level. Instead, the effect can be calculated on average on a population using the potential outcomes framework [18].

If there exits two treatments, $A = 1$ and $A = 0$, in a study, then there is also two potential outcomes for each individual. The potential outcome $Y(A = 1)$ where the individual received treatment $A = 1$, and $Y(A = 0)$ where it received $A = 0$. However, since both these treatments can not be observed at the same time for the same individual, the effect of the treatment is calculated as the average effect over several individuals, some of which receive $A = 1$ and some who receive $A = 0$. As long as the individuals compared are similar enough, the average effect between the two populations is a way to calculate the efficacy of the treatments.

2.1.2 Assumptions

To distinguish between causal effects and spurious correlations in the observational setting, additional assumptions are needed [13, Chapter 3]. There are three commonly used assumptions when attempting to infer causal effects:

- Consistency: $Y^a = Y(a)$
- Ignorability: $\{Y(0), Y(1)\} \perp A|X$
- Positivity: $\forall x, a \quad P(A = a|X = x) > 0$

where Y is outcome, a is action, and x is covariates.
Consistency means that we assume that the potential outcome of a treatment is the same as the actual measured outcome. An example of violation: if the patient does not take the given treatment as prescribed.
Ignorability means that the outcome of a treatment a does not depend on if the subject was assigned to the treatment group or control group, conditional on the covariates x. This means that we assume that we can ignore how patients ended up in the treatment or control group when regarding Y. Ignorability is satisfied as long as all backdoor paths are blocked [13, Chapter 7].
Positivity means that we assume that the probability of receiving any treatment, conditional on x is greater than 0. Thus, all parts of the population can receive any treatment. If this is not the case, it means that some potential outcomes do not exist, and therefore no causal effect exists.

Without these assumptions, it would be harder to infer causal effects from observational data due to potential bias or confounding. For ignorability and consistency, there is also no statistical method to test whether the assumptions hold or not, and thus prior knowledge has to be used to reason about if they are plausible or not in the setting.

2.1.3 Regression estimates of causal effects

Regression is a method of estimating the relationship between one variable, called the outcome, and one or more other variables, called covariates. To infer causal effects from regression analysis, care has to be taken to fit the regression model using the correct covariates. If some covariates are left out it can lead to confounding where the incorrect conclusions are drawn because there are factors affecting both the treatment assignment and the outcome. On the other hand, if the wrong covariates are included in the model, there might be correlations found that are not causative. Thus, prior knowledge has to be used to choose which covariates are included in the model and which are not.

Example. Consider an intervention done for unemployed people for them to get a job. The intervention could for example be to let people attend a course. The outcome is binary, either the person gets a job or they do not. To calculate the

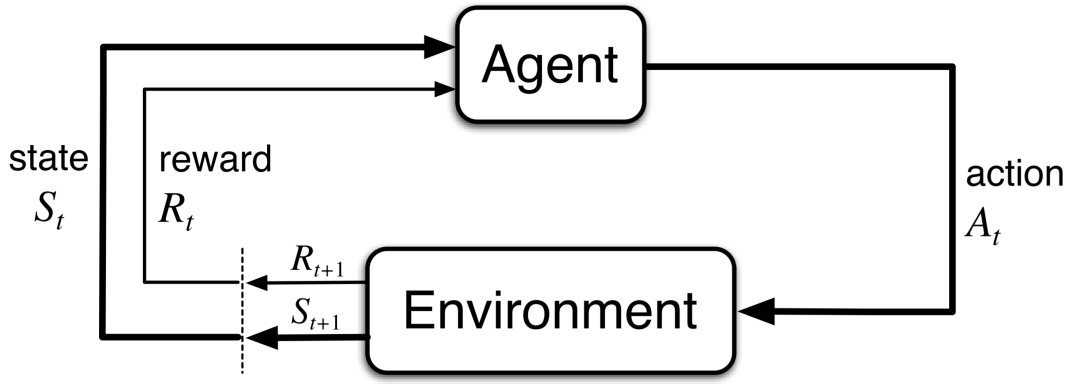


Figure 2.3: Overview of Reinforcement learning. The agent interacts with the environment by selecting an action depending on the state. The environment responds with a new state and a reward. Image retrieved from Sutton and Barto [1].

efficacy of the intervention using regression, both the relevant data of the person as well as if they attended the course or not has to be taken into account when fitting the regression model. Otherwise there might be bias where e.g. highly motivated people both attend the course in greater number and has a higher chance of getting an employment, thus skewing the results.

2.2 Reinforcement learning

Reinforcement learning (RL) is a branch of machine learning that specializes in finding the optimal action with respect to future events [19]. An RL algorithm consists of an agent interacting with an environment to learn the optimal actions to take for each state in the environment. Each state S_t and action A_t of the environment has an associated reward R_t, and by exploring the environment, the agent learns how to maximize the total reward received during an episode, see Figure 2.3. The reward is specified using a reward function that in turn determines which policy the RL algorithm will find. The reward given can be determined by the current state, previously taken actions, or some other criteria. For example, a constant negative reward will incentivize the RL algorithm to find the fewest number of actions required to reach a stopping state.

In the setting of the thesis, a state can be seen as a vector of the covariates and all previously tried treatments and the outcomes of those treatments, while the possible actions are the untried treatments. This allows RL to be used to find polices in the space of treatments and outcomes.

2.2.1 Dynamic Programming

Dynamic programming can be used to solve reinforcement learning problems where a model of the environment is known. This is achieved by breaking down the larger

problem into smaller sub-problems, that combine to solve the larger problem iteratively. For a stochastic problem, this is done by solving the Bellman equation (Equation 2.1) recursively from terminal states and backwards [19, Chapter 3]. The Bellman equation looks like follows:

$$V(s) = \max_a(R(s,a) + \gamma \cdot \sum_{s'} P(s,a,s')V(s')) \tag{2.1}$$

where V is the value function, s is a state, a is an action, R is the reward function, P is the probability of going from state s to s' by action a, and $\gamma \in [0,1]$ is a discount factor. While the Bellman equation can be used to find an optimal policy, it is symmetrical to the state-action Q-function [19, Chapter 6]. The Q-function looks as follows:

$$Q(s,a) = R(s,a) + \gamma \cdot \max_{a'} Q(s',a') \tag{2.2}$$

The equation states that at each state, the algorithm should pick the action which will return maximum accumulated future reward. Each step of the DP algorithm uses information about the next state that will be reached to calculate the value of the current state until it reaches a final state. The value of a final state is often defined to simply be the reward for that state, since no future state exists. Because of this, the value for all the final states is known and can be used to calculate the value for all possible previous states. This is then recursively done until the value for the initial state has been calculated. This means that every state will have an associated value since every reachable state will have been evaluated. The optimal policy is then extracted by starting at the initial state and choosing the action with the highest Q-value.

3

Methods

This chapter describes in detail the methodology used in the thesis. First, a model for the type of problems the thesis is attempting to solve is presented. Then, the building blocks for the algorithms are presented before the implementation is discussed.

3.1 Graphical causal model

Notation reminder. Recall that patients are represented by covariates X and unmeasured moderators Z which both may affect the outcomes Y. Outcomes are generated by treatments A.

Several assumptions have been made about how the observational data used in the thesis was generated. The assumptions have been used to create a causal graph as seen in Figure 3.1. The model assumes that all relevant confounders, X, have been measured meaning that ignorability holds. There are unmeasured moderators, Z, but they do not directly affect the treatments. Each treatment A affects its outcome but does not affect further outcomes or the patient's state. This can be reasonable for treatments that only affect the symptoms and not the underlying disease. The outcomes, Y, have an effect on which treatments are tried in the future. For example, if a treatment has a bad outcome, then similar treatments might have a lower chance to be tried.

From the model it can be seen that both the previous treatments and outcomes have to be accounted for when calculating the causal effects of a new treatment. If any treatment or outcome is left out, it opens up a backdoor path through Z and introduces potential confounding. For example, if we are trying to estimate Y_2, we have a potential backdoor path through Z to Y_1 and then to A_2.

3.1.1 Outcome Stationarity

In addition to the three common assumptions from section 2.1.2, an additional assumption is made for the model used in the thesis:

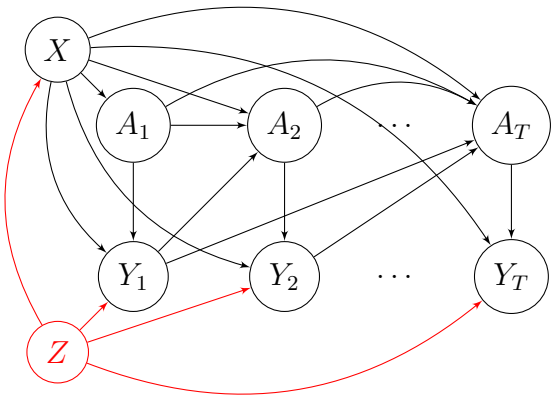


Figure 3.1: The assumed causal model for the problem presented in the thesis. X are the baseline covariates, A_t are the actions, Y_t are the outcomes, and Z are hidden moderators. Note how Z does not directly affect the treatments A_t and is therefore a moderator and not a confounder.

- Outcome stationarity: $Y_t(a) = Y_s(a), \forall a \in \mathcal{A}, t, s \in \{1, ..., T\}$

Outcome stationarity means that when estimating the outcome probability of a treatment, the order of previously tried treatments does not matter as long as the outcomes match. This means that a history with pairs of treatments and outcomes ([0, 1], [1, 2]) is equal to ([1, 2], [0, 1]) and can be used interchangeably when estimating causal effects. Outcome stationarity is a consequence of the treatments not affecting the covariates and future outcomes.

A result of the assumptions is Equation 3.1 which means that the observational distribution $p(Y|A = a)$ is indistinguishable from the interventional distribution $p(Y(a))$ given any order of the history of treatments [17, Chapter 3].

$$p(Y_s(a) \mid H_{s-1}) = p(Y_s \mid A_s = a, H_{s-1}) \ . \tag{3.1}$$

Outcome stationarity allows the algorithms to use more data when trying to predict the outcomes of a treatment for a patient with a specific history. It also lowers the number of unique histories that need to be considered since treatments tried in different orders are equivalent.

Theorem 1 (Outcome Stationarity)**.** *Let τ be a permutation of the sequence $(1, ..., s)$. Then, under the assumptions of Consistency (Section 2.1.2), and Stationarity, $Y_s(a) = Y_r(a) =: Y(a), \forall a \in A, s, r \in \mathbb{N}$,*

$$p(Y(a) \mid X, A_1, ..., A_s, Y_1, ..., Y_s) = p(Y(a) \mid X, A_{\tau(1)}, ..., A_{\tau(s)}, Y_{\tau(1)}, ..., Y_{(s)}) \tag{3.2}$$

Proof. Let τ be a permutation of $1, ..., T$ and $\tau(s)$ the index assigned to s. We use the short-hands $p(a) = p(A = a)$ and $p(A \mid b) = p(A \mid B = b)$. Equation 3.1 allows us to exchange $Y(a)$ for Y_t in the proof.

$$p(Y_t \mid H_t = h_t, A_t = a_t)$$

$$= \frac{p(Y_t, h_t, a_t)}{p(h_t, a_t)} = \frac{\sum_z p(Y_t, h_t, a_t, z)}{\sum_z p(h_t, a_t, z)} \qquad \text{prob. laws}$$

$$= \frac{\sum_z p(Y_t \mid h_t, a_t, z)p(a_t \mid h_t, z)p(h_t \mid z)p(z)}{\sum_z p(a_t \mid h_t, z)p(h_t \mid z)p(z)} \qquad \text{expand}$$

$$= \frac{\sum_z p(Y_t \mid h_t, a_t, z)p(a_t \mid h_t, z)\prod_s p(y_s \mid h_s, a_s, z)p(a_s \mid h_s, z)p(z)}{\sum_z \prod_s p(y_s \mid h_s, a_s, z)p(a_s \mid h_s, z)p(z)} \qquad \text{expand history}$$

$$= \frac{\sum_z p(Y_t \mid a_t, z)p(a_t \mid h_t)\prod_s p(y_s \mid a_s, z)p(a_s \mid h_s)p(z)}{\sum_z p(a_t \mid h_t)\prod_s p(y_s \mid a_s, z)p(a_s \mid h_s)p(z)} \qquad A_t \perp\!\!\!\perp Z \mid H_t$$

$$= \frac{\sum_z p(Y_t(a_t) \mid z)\prod_s p(y_s(a_s) \mid z)p(z)}{\sum_z \prod_s p(y_s(a_s) \mid z)p(z)} \qquad \text{cancel terms}$$

$$= \frac{\sum_z p(Y_{\tau(t)}(a_t) \mid z)\prod_s p(y_{\tau(s)}(a_s) \mid z)p(z)}{\sum_z \prod_s p(y_{\tau(s)}(a_s) \mid z)p(z)} \qquad \text{stationarity}$$

□

Notice how the final expression does not contain any references to the order of the treatments, only the results of them. Thus proving that the order of the treatments in the history does not matter as long as the outcomes match, and more information can be extracted from the observational data.

Example. We have one patient (V) in the data set with history of pairs of treatments and outcomes ([0, 1], [1, 1], [2, 2]). We also have a patient (W) that we want to treat that has the treatment history ([1, 1], [0, 1]). Both patients have the same covariates. If we would not assume historical equivalence, then we would not be able to infer anything from patient V when treating W, even though it seems that they are very similar considering that they reacted the same to two different treatments. Using this, we are more likely to believe that giving treatment 2 to patient W will generate an outcome of 2, the same outcome as for patient V.

3.2 When to stop searching

As has been mentioned, there are two goals in this thesis, finding an effective treatment and minimizing the search time. In practice, this is done by searching for better treatments until it is certain, within some limit, that no better treatment

exists. This balances both the need to try treatments that give more information about which treatments will work and finding the best possible treatment. For a treatment a, this is expressed mathematically as

$$\Pr\left[\max_{a' \notin h} Y(a') > \max_{a \in h} Y(a) + \epsilon \middle| H = h\right] \leq \delta \tag{3.3}$$

where $a \in h$ means action a in history h, and conversely, $a' \notin h$ is an untried action. The parameter δ controls how likely a new better treatment has to be and ϵ controls how much better the new treatment has to be than the previous best to be worth pursuing. If $\delta = 0$ we want to be absolutely sure that we cannot find another better treatment, and if $\epsilon = 0$, any better treatment will do, and since that is the case in the rest of the thesis, it will be omitted where it is not relevant. The two parameters will be set depending on the application of the algorithm. In settings where there are a low number of discrete outcomes $\epsilon = 0$ is reasonable, and δ might be set to the minimum probability you want a future treatment to be better than your current best treatment.

3.3 Optimization problem

We formulate the problem of finding a good treatment in few trials through an optimization problem which yields a policy π. π then chooses the actions i.e. what treatment to give to a patient or if to stop searching.

$$\begin{aligned} \min_{\pi \in \Pi} \quad & \mathbb{E}_{X,\overline{Y},h,T \sim p_\pi}[T] \\ \text{s.t.} \quad & \forall t \in \mathbb{N}, \forall h \in H : \Pr\left[\max_{a \notin h_T} Y(a) > \max_{a' \in h_T} Y(a') + \epsilon \;\middle|\; H_t = h, T = t\right] \leq \delta \end{aligned} \tag{3.4}$$

The goal is to find a policy π that minimizes the expected number of trials while, in each step of the algorithm, fulfilling the constraint where a new treatment is only tried if the algorithm expects it to be better with a certain margin ϵ and probability δ.

3.4 Estimating potential outcomes

A model of the potential outcomes for an individual is used in the constraint (Equation 3.3) to estimate the probability that a given treatment is better than the current best treatment.

3.4.1 Frequentist and smoothing approach

The simplest way to estimate the potential outcome of a treatment is to directly estimate the probability function of the outcomes for that treatment using the collected data. If the outcomes are discrete or has been discretized, this can be done

by a simple frequency analysis conditioned on history, covariates, and treatment as seen in Equation 3.5.

$$P(Y = y \mid A = a, H = h) \approx \frac{n}{N} \tag{3.5}$$

n is the number of patients where $Y = y, A = a, H = h$ and N is number of patients where $A = a, H = h$.

This approach works fine as long as there is sufficient data and all variables are discrete. However, for some combinations of covariates, history, and treatments there might not exist any data at all or only very limited data, leading to a high uncertainty about the probability estimate. To still get an estimate even in cases with high uncertainty, a technique called *smoothing* is used. Since the distribution of the outcomes for a treatment can be seen as a categorical distribution, a Dirichlet prior is used and the mean of the posterior is used as an estimate of the outcome probabilities. If the Dirichlet prior probability estimate for outcome i is ω_i, the number of samples with outcome i is n_i, and the total number of samples is N, then the expectation of the probability for outcome i can be calculated according to Equation 3.6.

$$\mathbb{E}[P(y_i)] = \frac{n_i + \omega_i}{N + \sum_j \omega_j} \tag{3.6}$$

Two different priors are used in the thesis, presented below.

Historical smoothing

Historical smoothing assumes that estimates based on histories that are similar to the current patients are more informative. Thus, the prior is a weighted sum of the probabilities for the estimated treatment at all possible previous histories.

$$w_i = \frac{e^{-(|h|-|h_i|-1)^2}}{|h| \cdot 2^{(|h|-|h_i|-1)}} \tag{3.7}$$

The calculation for the weight can be seen in Equation 3.7, where the histories indexed by i are those that are potential precursors to the history at this point in time. The norm of a history is simply the number of recorded treatment and outcome pairs in the history. The weight is constructed to ensure that the closest histories contribute the most to the prior. Since there are multiple histories that can construct the current one, the denominator acts as a normalizer for the number of possible histories. We define two histories to be similar if they have the same

treatments and outcomes, but one or more treatment and outcome pairs that are tested in one history is untested in the other.

$$\omega_i = \sum_j w_i \cdot p_{i,j} \tag{3.8}$$

Historical smoothing assumes that if two similar histories are close in the number of treatments tried, then the distribution of outcomes for a treatment should be similar. This allows the prior to make use of more data when estimating the potential outcomes, especially for situations where data might be limited. While the assumption that close histories have similar distributions of outcomes might seem reasonable, it is not necessarily so for histories that are extremely predictive of future outcomes. Thus the historical prior can be arbitrarily wrong when no data is available, but since no inference can be made without data this is seen as acceptable. It can also skew results slightly when there exists only a few data points.

Uninformed smoothing

Uninformed smoothing works by simply assuming a uniform prior on the outcomes, and weighting the smoothing very lightly, thus only using the prior when no data is available. This is described in Equation 3.9.

$$\omega_i = \mathbb{I}(N = 0) \cdot \frac{1}{\sum_i 1} \tag{3.9}$$

The uninformed smoothing is an unbiased estimator as long as there is data, but suffers from higher variance when the amount of data is low which can easily happen when there are long sequences of treatments. When data is missing entirely, the uninformed smoothing does not try to predict the outcome using other data, instead it assumes that all outcomes are equally likely, which can lead to bad predictions.

3.4.2 Function approximation

Function approximation can also be used to estimate the probability distribution of outcomes of a treatment. Here, a function is fit such that the coviariates, history, and last action is the input and the probability distribution of outcomes of the last action is the output, see Equation 3.10. An advantage of this is that the estimate can find relationships between covariates and histories which the frequentist approach can not. As long as all relevant covariates are measured, regression can be used to estimate causal effects, which is taken advantage of by the function approximation. A Random Forest regression algorithm was used to estimate the outcomes for combinations of covariates, treatments and histories [20].

$$f(h, a) \approx P(Y(a)|h) \tag{3.10}$$

3.5 Computing the stopping constraint

To calculate the bound $\rho(h)$, any possible future combination of treatments and outcomes have to be evaluated. If we define $\mu = \max_{a \in h} Y(a)$, i.e. the best found outcome so far, this can be done as follows

$$
\begin{aligned}
\rho(h) =& \mathbb{C}(h) \\
\mathbb{C}(h) =& 0 \ \textit{if} \ a \notin h = \emptyset \ \textit{else} \\
& \max_{a \notin h} \sum_{y} P(Y(a) = y) \cdot \mathbf{I}(y > \mu) + \\
& \sum_{y} P(Y(a) = y) \cdot \mathbf{I}(y <= \mu) \cdot \mathbb{C}(h \cup (y, a))
\end{aligned}
$$

It is the sum of the probability to find a better treatment in the current step and the probability of finding a better treatment in any of the following steps. This estimate is uncertain in general since each possible future treatment and outcome adds to the uncertainty. Also, the further into the future, the more uncertain the estimate, especially for combinations of treatments and outcomes that are uncommon in the data.

Thus, it may be useful to instead approximate the constraint using a bound. A trivial lower bound is the maximum probability of finding a better outcome in the next tried treatment (Equation 3.11).

$$\rho(h) \geq \max_{a \notin h} \left[p \left(Y(a) > \mu \mid h \right) \right] \tag{3.11}$$

The fact that this is a lower bound is trivial since any further treatment has to have a 0 or greater probability of having an outcome greater than μ and thus the probability is at least equal to Equation 3.11.

An upper bound can be found using Boole's inequality. It states that the sum of probabilities of n events is always greater than or equal to the probability of the union of those events. Thus a simple upper bound for the probability of finding a better treatment is

$$\rho(h) \leq \sum_{a \notin h} \left[p \left(Y(a) > \mu \mid h \right) \right] \tag{3.12}$$

The upper bound guarantees that the probability of finding a better treatment is never underestimated. On the other hand, the upper bound might return a value

> 1 which does not have an immediate clear interpretation in the probability of finding a better treatment in relation to δ.

3.5.1 Using the constraint

To calculate the constraint with the frequentist approach, data is collected on the probability of outcomes conditioned on a given history, $P(Y = y|H = h)$. That information is then used to calculate where the constraint should allow further exploration and where it should allow the algorithms to stop.

Example. There are 6 patients with the same history except the last intervention, i.e. they have tried treatment a_2 and it had an outcome of 1 on a scale where 2 is the highest. All patients have the same covariates x.

	Treatment #1	Treatment #2
Patient 1	$Y(a_2) = 1$	$Y(a_0) = 1$
Patient 2	$Y(a_2) = 1$	$Y(a_0) = 1$
Patient 3	$Y(a_2) = 1$	$Y(a_1) = 0$
Patient 4	$Y(a_2) = 1$	$Y(a_1) = 1$
Patient 5	$Y(a_2) = 1$	$Y(a_1) = 1$
Patient 6	$Y(a_2) = 1$	$Y(a_1) = 2$

Assuming that this data set represents the population as a whole we can infer how other patients will react to a given treatment. There is a 0% chance of getting a strictly better outcome with treatment a_0 since we already scored 1 with the previous treatment, and there is a 100% chance of getting an outcome of 1. If we pick treatment a_1 there is 25% chance of getting 0, 50% chance of getting 1, and 25% chance of getting 2. This means that there is 25% chance of improving the outcome Y if we choose a_1.

If $\epsilon = 0$ and $\delta < 0.25$, we will choose to continue by trying treatment a_1. If $\epsilon = 0$ and $\delta \geq 0.25$ then we will not try a_1 but instead stop the search since the probability of finding a better treatment is below the δ limit.

3.6 Policy optimization

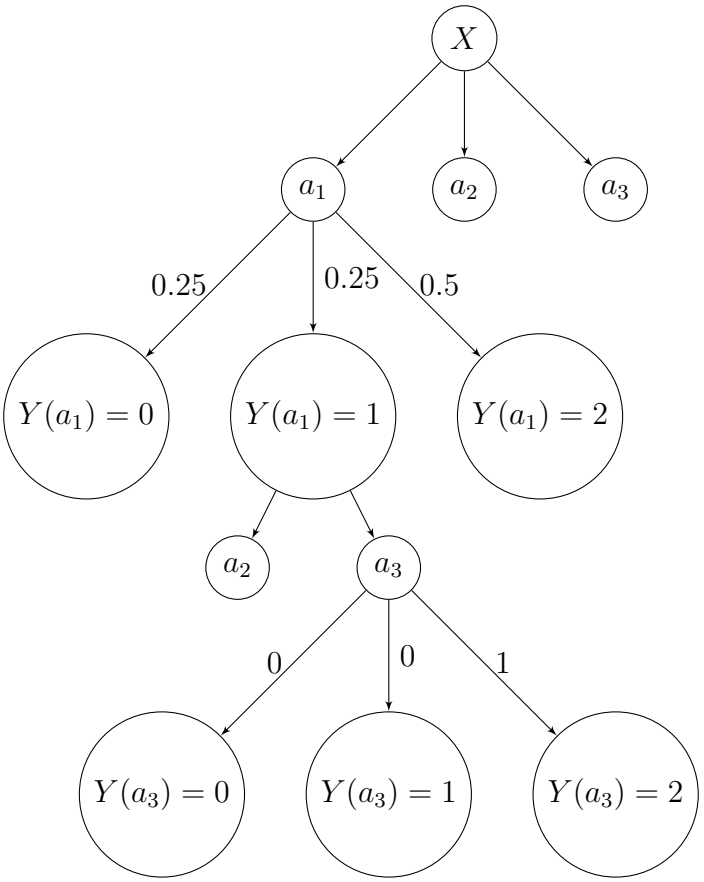
In this section, two different algorithms for optimizing the policy is presented. Both algorithms are based on the constraint in Equation 3.3.

3.6.1 Constrained Dynamic Programming

While a model of the environment is not know, the previous sections has shown how to approximate such a model. Thus dynamic programming becomes a possible method for solving our problem. Dynamic programming has the benefit of predicting which treatment to select based on all future treatment and outcome possibilities. Thus it is useful in our setting as it allows the algorithm to incorpo-

rate the value gained from information gathered from selecting a treatment and not just its outcome. While dynamic programming is very powerful, it is also susceptible to misspecification of the model which can lead to faulty results. It also requires a quickly growing state-space to calculate any policy.

Example. Below is a representation of a search tree where there are 3 treatments and 3 potential outcomes i.e. $\{0, 1, 2\}$. The numbers on the arrows are transition probabilities $p(Y(A) = y|H)$.



Here the algorithm first chose treatment a_1 and the outcome turned out to be 1. In this case, it turned out that $p(Y(A) = 2|A = a_3, Y(a_1) = 1) = 1$ i.e. that the probability of finding a treatment with outcome greater than 1 was certain if treatment a_1 had the outcome 1.

The quality function (Equation 3.13) for the algorithm is the sum of the immediate reward for the history-action pair plus the sum over all possible outcomes for that action times the quality value of those outcomes.

$$\begin{aligned} &Q(h, a) = \\ &r(h, a) + \textstyle\sum_{y \in \mathcal{Y}} p(Y(a) = y|h) \max_{a' \in A_{-h}} Q\left(h \cup \{(a, y)\}, a'\right) \end{aligned} \tag{3.13}$$

where h is history, a is action, y is outcome, and $r(h, a)$ is the reward function.

$$r_{\epsilon,\delta}(h,a) = \begin{cases} -\infty, & \text{if } a = 0, \gamma_{\epsilon,\delta}(h) = 0 \\ 0, & \text{if } a = 0, \gamma_{\epsilon,\delta}(h) = 1 \\ -1, & \text{if } a > 0 \end{cases} \tag{3.14}$$

The idea of the reward function $r_{\epsilon,\delta}(h, a)$ is, for the first condition, to force the algorithm to not stop the search if there is a sufficient probability that it will find a better treatment. The second condition states that if there is not sufficient probability, then there will be no negative reward for stopping. The third condition states that we get a reward of -1 for each treatment that is tried.

$$\gamma_{\epsilon,\delta}(h) := \mathbf{1}\left[\Pr\left[\max_{a' \notin h} Y(a') > \max_{a \in h} Y(a) + \epsilon | h\right] \leq \delta\right] \tag{3.15}$$

$\gamma_{\epsilon,\delta}(h)$ is the realization of the constraint. It states that, given history h, the probability of finding a treatment that is better than the best treatment found so far plus ϵ among the ones not tested, should be smaller than δ in order to allow to stop the search.

$$V(h) = \max_a Q(h,a) \tag{3.16}$$

The value function $V(h)$ chooses the treatment a that maximizes that Q-function for each history h. Before the Q-function is calculated, all possible histories are sorted in reverse order of size so that the longest histories are considered first.

Algorithm 1 Constrained Dynamic Programming

Input: Slack ϵ, confidence δ

$H' \leftarrow sort(H)$
for $h \in H'$ **do**
  for $a \in A_{-h}$ **do**
    if $a = a_{stop}$ **then**
      $Q(h,a) \leftarrow r_{\epsilon,\delta}(h,a)$
    else
      $Q(h,a) \leftarrow r_{\epsilon,\delta}(h,a) + \sum_{y \in \mathcal{Y}} p(Y(a) = y|a) \max_{a' \in A_{-h}} Q\left(h \cup \{(a,y)\}, a'\right)$
    end if
  end for
end for
Output: $\pi : \pi(h) = \arg\max_{a \in A_{-h}} Q(h,a)$

3.6.2 Constrained Greedy Algorithm

While the dynamic programming algorithm presented above should in general find better policies, there are also some drawbacks. Dynamic programming is slow and

computationally expensive since it calculates all values for the entire state-space, which grows exponentially in the number of treatments and outcomes. In contrast, greedy algorithms can be quicker since each decision is made using only the information in the current state.

The greedy algorithm tries to maximize the expected value of each tried treatment. In each step of the algorithm, it selects the treatment with the highest estimated treatment effect, given the previously tried treatments and the covariates x. In contrast to the Dynamic Programming algorithm, it does not take into account that a treatment with lower expected value might lead to a better outcome in the future. I.e., following history h, it selects the treatment with

$$\max_a \sum_y \mathcal{I}(y > \max_{a' \in h} Y(a')) \cdot P(Y(a) = y|h) \cdot y \tag{3.17}$$

If one does not want compare different values of y in this way, e.g. if the difference between values in y is not linear. Equation 3.18 could be used instead.

$$\max_a \sum_y \mathcal{I}(y > \max_{a' \in h} Y(a')) \cdot P(Y(a) = y|h) \tag{3.18}$$

It prioritizes finding a better treatment while valuing slight improvements as much as bigger improvements in outcomes. This would ensure that fewer treatments are tried with bad results, but might require more treatments to be tried overall.

Example. The difference in decision-making between Equation 3.17 and 3.18 can be shown in a small example. We have $y \in \{0, 1, 2\}$, two treatments, and a patient. $\delta = 0$ and $\epsilon = 0$. The patient will be chosen a treatment and has a current maximum outcome of 0. Treatment 1 has probabilities $[0.4, 0.6, 0]$ and treatment 2 has probability $[0.5, 0, 0.5]$ for outcomes y. For equation 3.17, treatment 2 is selected while treatment 1 is selected by Equation 3.18.

Algorithm 2 Constrained Greedy

Input: Slack ϵ, confidence δ

while $a \neq a_{stop}$ **do**
  if $\gamma_{\epsilon,\delta}(h) = 0$ **then**
    $a \leftarrow a : \max_{a \in A_{-h}} \sum_y \mathcal{I}(y > \max_{a' \in h} Y(a') + \epsilon) \cdot P(Y(a) = y|h) \cdot y$
    $h \leftarrow h \cup \{(a, y)\}$
  else
    $a \leftarrow a_{stop}$
  end if
end while

3.6.3 The greedy policy is sub-optimal

While the greedy policy often works well, in certain circumstances it will be sub-optimal. This is because learning a non-repeating policy amounts to learning a minimal decision list or tree, which in general is NP-hard both optimally [21] and approximately [22]. The greedy policy being sub-optimal happens when there is a treatment that has a high probability of working but provides almost no information, while there is another treatment that is close in probability of working and provides much more information.

Example. We have 3 treatments (A), binary outcome (Y), and 4 groups of the hidden moderator (Z). C is a matrix representing which treatments A are effective and not (i.e. a 1 and 0, respectively) for which moderator Z.

$$C = \begin{array}{c} \\ z_1 \\ z_2 \\ z_3 \\ z_4 \end{array} \begin{array}{c} \begin{array}{ccc} a_1 & a_2 & a_3 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array}$$

The cumulative probability that each treatment a is effective is given by: $p(Y(\cdot) = 1) = \begin{bmatrix} 0.60 & 0.40 & 0.65 \end{bmatrix}$.

The greedy policy would start with selecting treatment a_3 since the probability for it to work is the highest, 0.65. It would then select a_2, then a_1. The optimal policy would instead start with selecting treatment a_1 and then treatment a_2 since it is then guaranteed to finish and to have found a working treatment.

To extend this example we can calculate the expected number of trials until we find a treatment that works. Given

$$p(Z) = \begin{bmatrix} 0.20 \\ 0.15 \\ 0.20 \\ 0.45 \end{bmatrix}$$

The expected number of trials for the greedy policy is:

$$\begin{aligned} \mathbb{E}[T] &= p(Z \in \{1, 4\}) + 2p(Z = 3) + 3p(Z = 2) \\ &= (0.20 + 0.45) + (2 * 0.20) + (3 * 0.15) = 1.5 \end{aligned}$$

And for the optimal policy:

$$\begin{aligned} \mathbb{E}[T] &= p(Z \in \{2, 4\}) + 2p(Z \in \{1, 3\}) \\ &= (0.15 + 0.45) + (2 * (0.20 + 0.20)) = 1.4 \end{aligned}$$

The expected number of trials for finding an effective treatment is shorter for the optimal policy than for the greedy policy, showing that the greedy policy is not optimal in general.

4

Results

In this chapter, the process of generating two synthetic data sets are presented, as well as a real-world antibiotics resistance data set. Then the results from several tests with the different data sets are presented.

4.1 Data sets

While using a real data set to train a model is straightforward, it is not as simple to evaluate the performance of the resulting policy. To allow for an easier time evaluating the performance of our trained policies in different settings, synthetic data sets were created for testing purposes. In the synthetic data sets, we have full access to the outcome distribution which helps when evaluating how well the policies work.

The data sets are generated by data models which were produced to provide different challenges for the algorithms to solve. Each data generating model consists of some number of hidden moderators Z, covariates X, treatments A, and a range of possible outcomes Y. The data is generated according to the assumed causal model that can be seen in Figure 3.1.

4.1.1 A discrete toy data set

A simple data model was created to highlight a situation where the greedy policy is not optimal. The model is a discrete model with deterministic outcomes where the values of the hidden moderators Z completely decide the outcome of the treatments. Only one covariate X was used with 3 treatments A and 3 possible outcomes Y. The distribution of Z and the outcomes can be seen in Table 4.1.

The outcomes can be ranked from 0 to 2, with 2 being the best outcome. When generating data, the treatments are selected according to a weighted policy, where the weight of each treatment is proportional to the percentage of the population having the best outcome for that treatment. Additionaly, the weight for any already attempted treatments are set to 0, thus each treatment is only tried once. New treatments are stopped with probability 0.9 when an outcome of 2 is reached, or

z_0	z_1	z_2	$P(Z)$	$Y(a_0)$	$Y(a_1)$	$Y(a_2)$
0	0	0	0.30	2	2	1
0	0	1	0.16	2	1	0
0	1	0	0.35	1	0	2
0	1	1	0.13	1	2	0
1	0	0	0.04	1	1	1
1	0	1	0.01	1	1	2
1	1	0	0.01	0	2	0
1	1	1	0.00	0	0	0

Table 4.1: The distribution of Z and treatment outcomes for the discrete toy data generator.

when all treatments have been tried.

The distribution is designed such that treatment a_0 will have the highest success rate for the majority of the population, but treatment a_0 does not give any further information about which treatment is optimal for the rest of the population. As such, the optimal treatment regime is to first try treatment a_1, and then, depending on the outcome, try either a_0 or a_2. Another feature of the data is that there has to be a decision whether to stop searching for a more effective treatment when both a_0 and a_1 yield a result of 1. With a probability of 0.2, trying a_2 as well will give a better outcome, and with probability 0.8 it will not.

4.1.2 A random discrete data set

While the data set presented in Section 4.1.1 can highlight an interesting property in the algorithms, it is perhaps not very realistic. Thus another data set was implemented to enable testing of more complex environments while still retaining full information about all potential outcomes and outcome distributions.

The generation process for a single patient in a model with I moderators, J covariates, K treatments and M different outcomes, is described below.

1. The vectors α, β, γ, and ω are generated once for each model environment. The moderators, Z, and the covariates, X, are binary variables.

2. Each moderator Z_i is generated randomly, with probability of being 1 equal to α_i. Where $\alpha \in (0, 1)^I$.

3. Then, each covariate X_j is generated randomly, with probability of being 1 weighted by $Z \cdot \beta_j$. Where $\beta_j \in \mathbb{R}^I$.

When Z and X has been generated, a sequence of treatments are drawn until either a treatment with the best possible outcome is generated, all treatments have been tried, or a stop action is taken (which happens with probability 0.1). The probability

of a treatment being drawn is weighted by $P(A = a_k|H, X) = \frac{(1,H,X)\cdot\gamma_k}{\sum_k (1,H,X)\cdot\gamma_k}$, or 0 if the treatment has already been tried.

The outcome of each treatment is approximately normally distributed with a mean calculated as $\mathbb{E}(Y(a_k) = (1, Z, X) \cdot \omega_k$ such that $\mathbb{E}(Y(a_k) \in (0, M)$. The variance is also randomly generated in the range $(0, 1)$.

All of this follows the model laid out on Figure 3.1. When generating training data, the above procedure is followed, while during testing, the assignment of treatments is determined by the policy, rather than the data generating model.

4.1.3 An antibiotic resistance data set

To evaluate the algorithms in a realistic setting, we use data from observational studies of antibiotic resistance in humans. The data is collected from the MIMIC-III data set [12] which contains data points from over forty thousand ICU patients at Beth Israel Deaconess Medical Center. Antibiotic resistance is measured by collecting a sample from a patient and classifying the bacterial cultures as resistant, intermediate, or susceptible to the antibiotic. The growing of cultures can take some time and in urgent cases one may not have time to wait for the results before prescribing medication. These antibiotic tests are how outcomes are measured in this setting and means that we have measured all potential outcomes. Having access to all potential outcomes is good when evaluating, since it allows us to simulate as if we had real patients. The setting of antibiotic resistance makes it a good match for the assumptions of the causal model since the state of the patient is not changed when the efficacy of the treatment was measured unless the treatment was a success, in which case there is no need to try further treatments.

The problem is simplified compared to the actual problem of prescribing antibiotics in several ways. In reality, a patient is taking an antibiotic one or several times, perhaps with different doses. A patient may also take multiple antibiotics simultaneously. In our data, an antibiotic for a patient is represented as only one event, and the problem of choosing a dose is not addressed. Another tricky part is that the MIMIC-III database does not state for which organism or organisms an antibiotic was prescribed to eliminate. This is difficult both for when a patient has several organisms and when multiple antibiotics are given at the same time, so-called antibiotic cocktails. Patients in our data are only allowed to have a single organism assigned, thus, one solution is to "copy" the patient such that for each organism the patient had, a new patient is created with that organism and the same history of treatments and the same covariates. In this experiment, we also assume that we can observe the outcome of the previous treatment before selecting a new treatment. This is probably not the case in reality since different antibiotics sometimes are given to a patient within a very small time frame. Using the antibiotic tests as ground truth for the outcome of the antibiotic might also be departing from reality.

A patient in the data is represented by a hospital stay from MIMIC-III, which means

that a person could show up more than twice in the data by visiting the hospital multiple times. For each hospital stay the first occurrence of a treatment is picked as the time the treatment is used. The scale of outcomes for the antibiotics is:

Resistant	0
Intermediate	1
Susceptible	2

There are 3 different covariates consisting of 4 organisms, 4 age bins, and 2 comorbidities of which a patient can have none, one of them, or both. The used organisms are Escherichia Coli (E. coli), Pseudomonas aeruginosa, Klebsiella pneumoniae, and Proteus mirabilis. The reason these were picked was because there were a lot of available data compared to other bacteria, and these had many antibiotics in common. The four age bins are $[0, 15]$, $(15, 31]$, $(31, 60]$, and $(60, \infty)$, and the comorbidities are Infectious And Parasitic Diseases, and Diseases Of The Skin And Subcutaneous Tissue, as classified by International Statistical Classification of Diseases and Related Health Problems (ICD) 9th edition [23]. These covariates were chosen because Ghosh et al. showed in [24] that there is a significant association between these age bins and comorbidities, and some of the organisms we use. The 6 selected antibiotics used to treat the organisms are Ceftazidime, Piperacillin/Tazo, Cefepime, Tobramycin, Gentamicin, and Meropenem. There are 1362 patients split 70/30 in training and test data such data one patient with multiple organisms ends up in the same group.

4.2 Algorithm performance

In this section, a series of experiments have been performed to evaluate different parts of the algorithms. We evaluate different bounds, potential outcome estimation methods, and sizes of data set. We also compare the constrained algorithms to ones without the constraint as well as apply the algorithms to real data of antibiotic resistance. The standard setting of experiments on the synthetic data set is to use 3 different binary Z (moderators), 1 binary X (covariates), 5 different A (treatments), and 3 different Y (outcomes). When evaluating the algorithms for the test population, we have two different ways of estimating outcome performance. **Mean treatment effect** is the average best found treatment outcome in the population. **Efficacy** is the proportion of the test population that receives their maximum possible treatment outcome.

4.2.1 Baseline algorithms

We also compare the constrained algorithms to some baseline algorithms. This helps to contrast the constraint to other methods and evaluate the potential of our method. How the two baseline algorithms works are described below.

Naive greedy algorithm. In each step, the naive greedy algorithm picks the treatment with the highest probability of generating maximum outcome, given the

covariates and history. The treatment probabilities are calculated the same way as for the constrained algorithms, as described in Section 3.4. It also has a maximum number of treatments it is allowed to test and stops when it finds a treatment with the best possible outcome or the maximum number of steps have been reached.

Naive Dynamic Programming. The Naive Dynamic Programming algorithm is implemented similarly to Constrained Dynamic Programming, described in Section 3.6.1. The difference is that the naive version does not use the constraint. Instead, it has a fixed negative reward for each treatment it tries, and a positive reward for stopping equal to the numeric value of the best treatment found. The negative reward can be tweaked to get different results from the algorithm.

4.2.2 Comparison of Greedy and Dynamic Programming

Using the simple discrete data generating model from Section 4.1.1, a training set of 30 000 samples and a test set of 3 000 samples were generated. We compared the performance of the Constrained Dynamic Programming algorithm as described in Section 3.6.1 and a Constrained Greedy algorithm as described in Section 3.6.2. Both algorithms used $\delta = 0$ and $\epsilon = 0$. The results are presented in Figure 4.1.

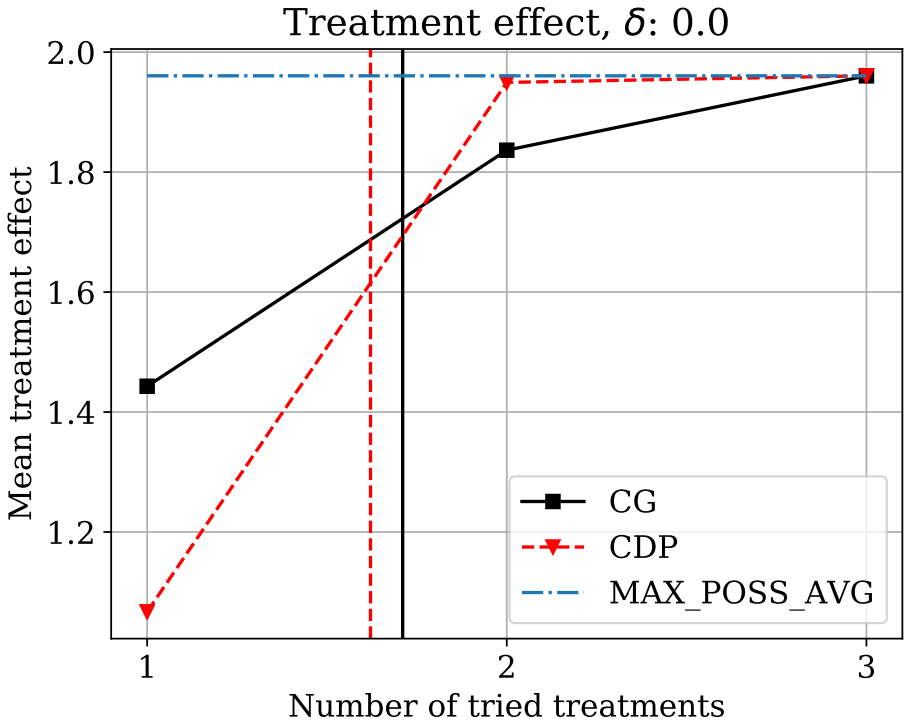


Figure 4.1: Results for the toy data set. The vertical lines in the graph represents the mean search times for the two algorithms.

Notable is that the greedy approach has a better mean treatment effect after the first attempted treatment as would be expected. After the second treatment, the Constrained Dynamic Programming algorithm has a better mean treatment effect

since it has gained an information advantage in the first step.

4.2.3 Effect of δ on search time and efficiency

To compare the two constrained algorithms to the two naive ones, a test was set up using the data set discussed in Section 4.1.2. Using 15000 training samples and 3000 test samples the results are averaged over 10 runs.

In the test, δ was varied between 0 and 1 for the constrained algorithms. The naive greedy algorithm had its maximum number of steps varied between 1 and 5 which equals the number of available treatments, and the naive dynamic programming algorithm had the negative reward set from 0 to -1. The results for this test are presented in Figure 4.2, note that the results on the y-axis are the proportion of the test population which receives their maximum possible treatment outcome, which does not have to match the best possible treatment outcome in the population.

As can be seen, there is a trade-off between the treatment efficacy and the mean number of trials with generally higher search times leading to better outcomes. To evaluate which algorithm performs the best, the same results are presented in Figure 4.3 where the search time and efficacy are plotted against each other. What can be seen is that all algorithms have roughly the same trade-off with the constrained greedy algorithm perhaps being somewhat better overall.

4.2.4 Comparing bounds

Figure 4.4 shows the mean number of trials in relation to the efficacy of the three bounds from Section 3.5. As can be seen, all bounds performs approximately the same. The same plots but done over δ can be seen in Figure B.3 and B.4. While the algorithms yield different results for the same δ, for a specific search time and efficacy of one bound, the same values can be achieved by adjusting δ for the other bounds.

4.2.5 Comparing potential outcome estimation

As discussed in section 3.4.1, to compensate for a low amount of data, smoothing is used to help estimate the model of outcomes. However, this runs the risk of introducing errors when trying to calculate the probabilities. In Figure 4.5, four different smoothing techniques are plotted with search time versus efficacy for both the DP and Greedy algorithms. The last letter tells which type of estimation is done, T is using the true probabilities, U is using Uninformed smoothing, H is using Historical smoothing and F is using function approximation.

Note how the true probabilities most often give the best result as would be expected. Of the others, the historical prior works best, and the uninformed prior seems to work the worst. The greedy algorithm also has a smaller spread between the estimators.

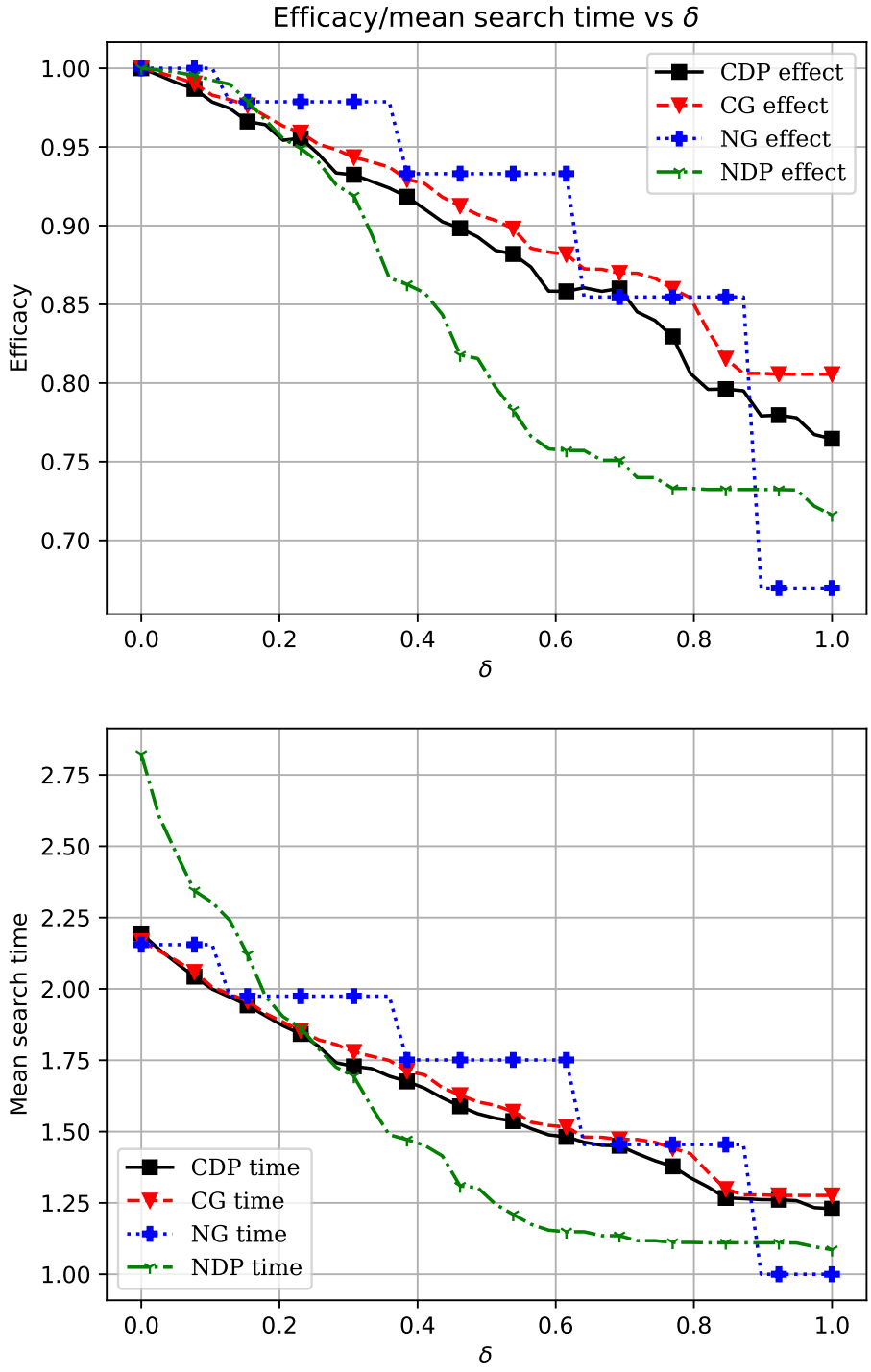


Figure 4.2: The two main algorithms and the two naive algorithms plotted over a range of δ, reward, and max-steps. As expected, both the search time and efficacy decreases with higher δ.

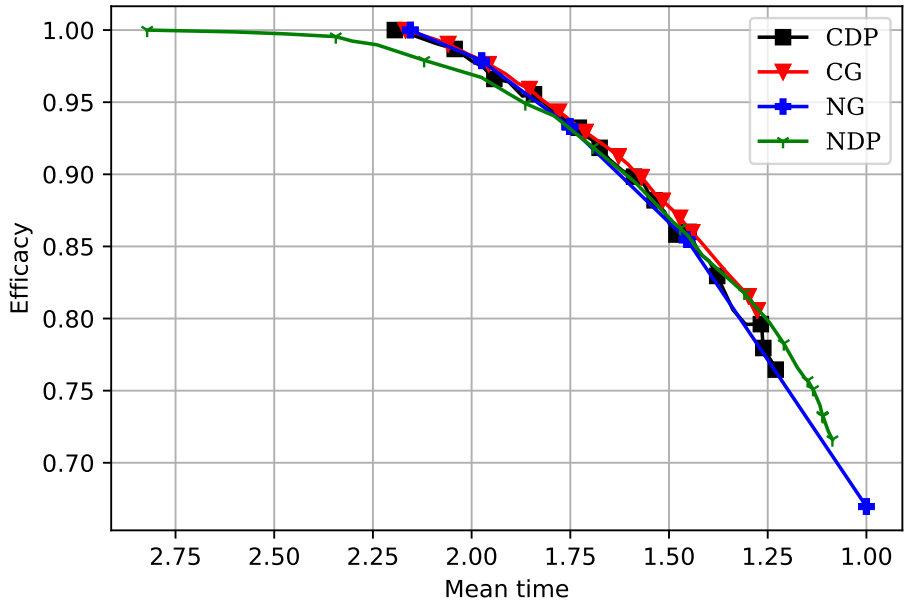
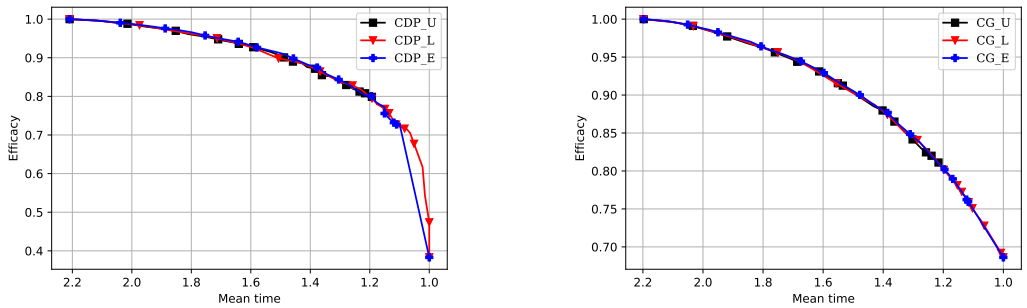
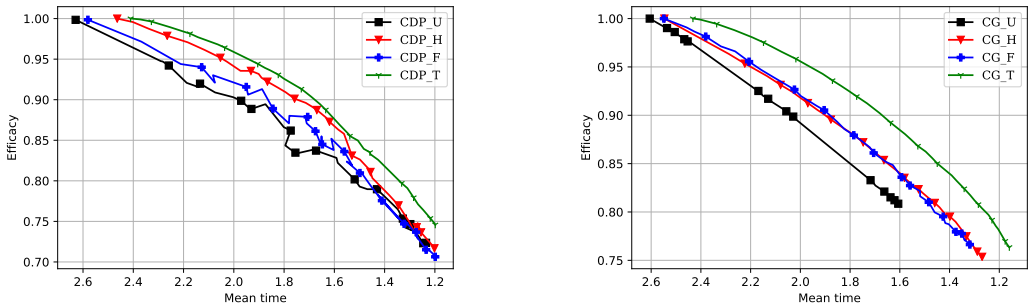


Figure 4.3: The same graph as in Figure 4.2, but plotted as efficacy vs time.



(a) Constrained Dynamic Programming algorithm. **(b)** Constrained Greedy algorithm.

Figure 4.4: Upper, exact, and lower bounds for the two algorithms. Note that all bounds are very similar.



(a) Four different approximators plotted for the constrained dynamic programming algorithm.

(b) Four different approximators plotted for the constrained greedy algorithm.

Figure 4.5: Plots of efficacy vs time for different approximators. Note that an algorithm closer to the upper right has a better performance than an algorithm closer to the bottom left.

4.2.6 Effect of data set size

The size of the data set used to train the algorithms can influence how accurate they represent the whole population. A low amount of data can lead to variance playing a larger part in the estimates of the model. Figure 4.6 illustrates how well the different algorithms perform with different sized data sets.

While all algorithms are unreliable at extremely small data set sizes, all of them seem to stabilize and get better performance at a data set size of around 2500 samples, with the constrained greedy and constrained dynamic programming perhaps a bit earlier. A better performing algorithm will have a larger gap between the outcome and the search time. Also noteworthy is that Uninformed smoothing is very cautious when the data set size is low and gets both a high efficacy and a high mean time.

4.2.7 Comparison of the constrained and naive algorithms

To get another comparison between the naive and constrained versions of the algorithms, 100 test runs were done for each of the algorithms, and the mean effect and mean search time was calculated. Each run was made on a data set of 15000 samples with δ set to 0.3. The results are presented in Table 4.2. All algorithms performed similarly, with a higher mean number of trials leading to lower regret.

4.3 Experiments on antibiotic resistance data set

Evaluation with antibiotic resistance data is performed in two different settings. The first is captured in Figure 4.7 and is the mean treatment effect for each tried antibiotic where we compare CDP and CG to NDP and an emulated doctor. The second setting is the mean efficacy vs mean time for different values of δ captured

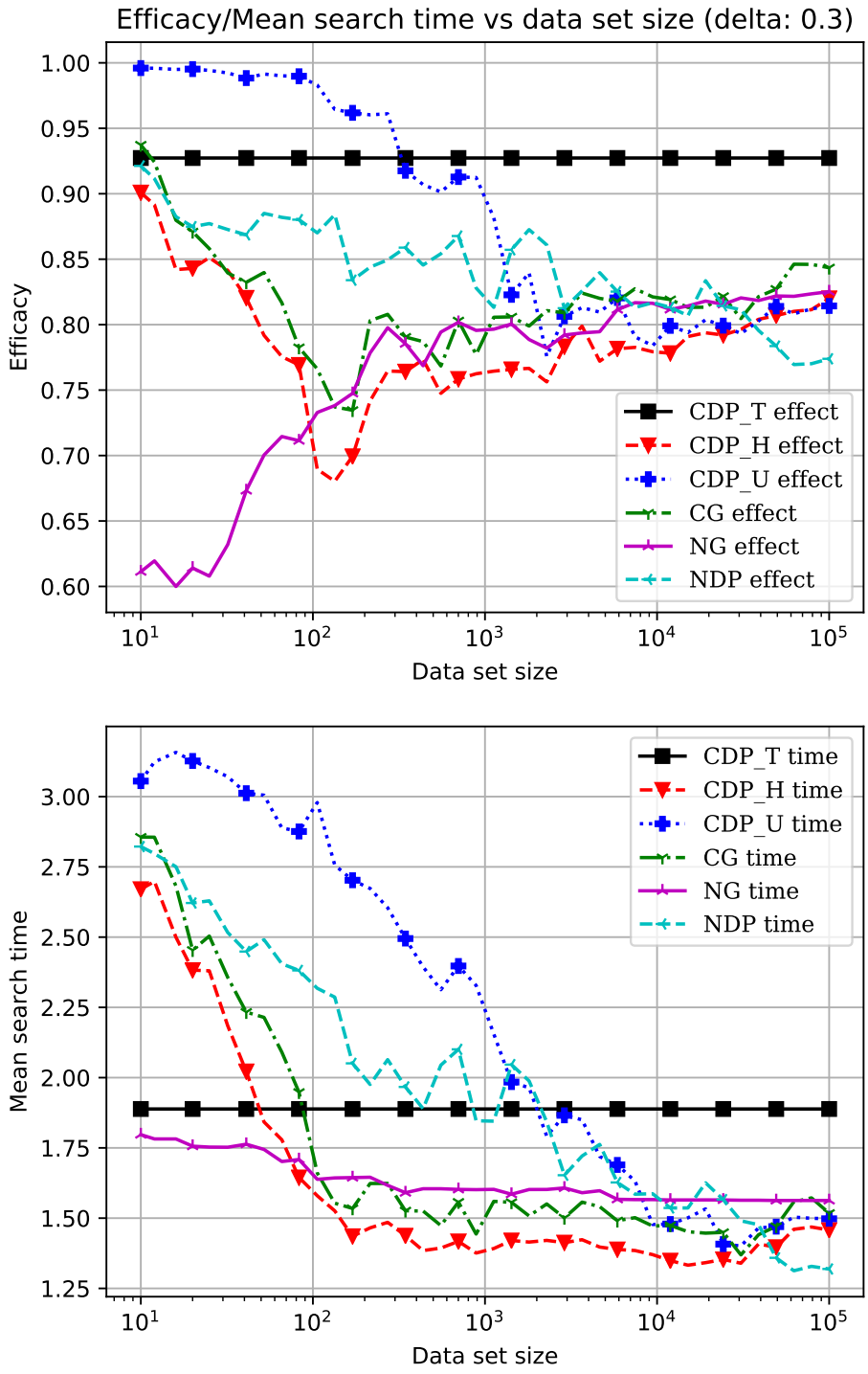


Figure 4.6: How the algorithms respond to different data set sizes. Note that the CDP_T does not depend of the amount of data since it has direct access to the model of the outcomes. Also note that the x-axis is logarithmic.

Algorithm	Regret (1-efficacy)	Regret var	Mean trials	Trials var
Naive Greedy (4 steps)	0.0291	0.0002	2.1729	0.1581
Constrained Greedy	0.0656	0.0012	1.8679	0.0755
Naive Dynamic P. (r=-0.25)	0.0880	0.0037	1.8023	0.1203
Constrained Dynamic P.	0.0829	0.0017	1.8009	0.0574
Constrained Dynamic P. T	0.0572	0.0009	1.8660	0.0683

Table 4.2: Performance comparison of different algorithms. The data was averaged over 100 runs.

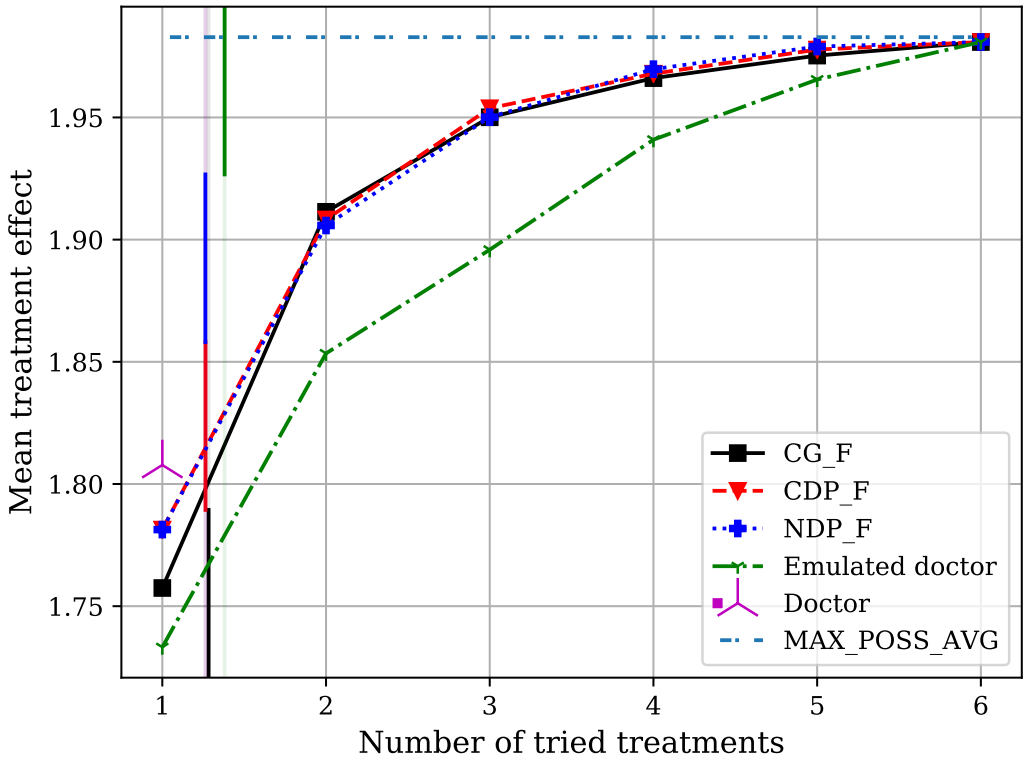


Figure 4.7: Antibiotic resistance data evaluated with algorithms that used function approximation. $\delta = 0$. CG_F performs slightly better than CDP_F, NDP_F, and the Emulated Doctor. Doctor is only evaluated at the first trials to avoid incorrect inference. The vertical lines represent the mean number of trials for the different algorithms.

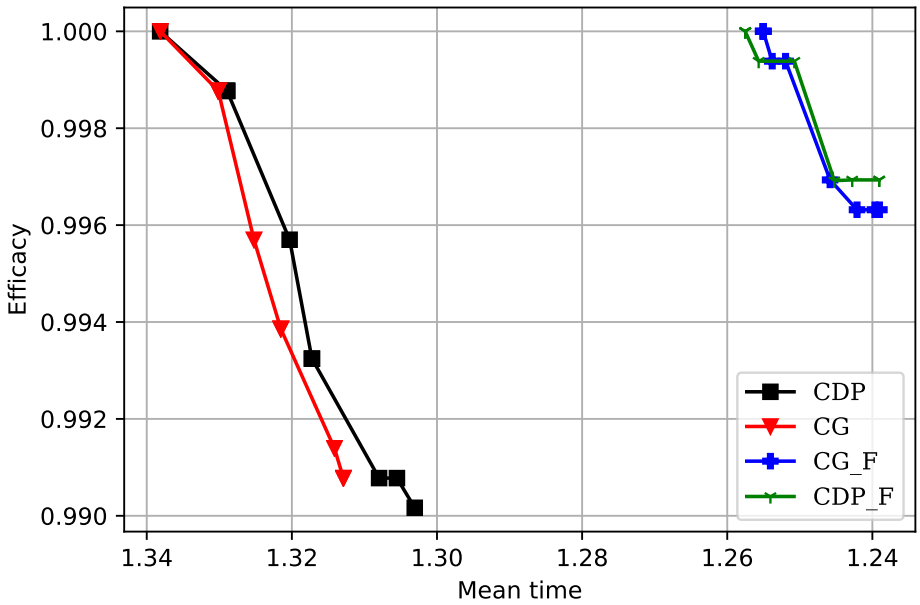


Figure 4.8: The mean time and efficacy evaluated over 10 different $\delta \in [0, 1]$. Note that the upper right area is better and that function approximation works well compared to frequentist approach with historical smoothing.

in Figure 4.8. CDP, CG, and NDP used function approximation as described in Section 3.4.2 for the potential outcome estimation, both in the algorithm and in the constraint.

Doctor in Figure 4.7 is what the actual doctor prescribed to the patient. Doctor is only shown at the first trial because we can't know why the doctor stopped treating the patient. it could e.g. be that the patient passed away, which would lead us to believe that the doctor stopped before finding an antibiotic that the organism was susceptible to. The Doctor achieved a higher treatment effect in the first step which suggests that the doctor considered more covariates than the algorithms when antibiotics were prescribed to a patient. **Emulated doctor** is a greedy algorithm such that it chooses the treatment that the doctor was most likely to give to a patient given the covariates and the history of the patient.

Figure 4.8 shows how function approximation in both CDP and CG works well compared to the frequentist approach with historical smoothing. The more a point is to the upper right, the better the algorithm performs. Function approximation performs well because it finds similarities between different covariates, which the frequentist approach can not. This is particularly important in cases with small amount of data. The plot also suggests that a larger δ is more advantageous for CDP than CG, although differences are very small and might depend on variance. The values in the plot were averaged over 5 different shuffles of the training and test set.

5

Related Work

5.1 Reinforcement learning in dynamic treatment regimes

Reinforcement learning has previously been used to search for optimal dynamic treatment regimes, however, previous work often focus on finding the optimal sequence of treatments for a predetermined number of steps. Tao et al. [5] used tree-based reinforcement learning to find the optimal treatment for each step in a simulated study, but lacked the notion of finding a "good enough" treatment [5]. Instead they were interested in finding a policy that when deciding between a combination of three treatments at two sequential decision points find the best treatment outcome. Their method worked well, both when the estimated causal model was correctly and incorrectly specified.

Huang et. al. [4] used a modified version of Q-learning to decide between two treatments at two stages during a simulated treatment of cancer. Accounting for causal factors in deciding the models for the Q-learning improved the accuracy and improved the robustness when unmeasured confounding was present in the data compared to the unmodified reinforcement learning. Although promising, their method was also focused on finding the optimal treatment on two separate sequential stages and did not have the notion of searching for an effective treatment.

5.2 Incorporating causal factors in reinforcement learning

In a paper by Yu et al. [9], they incorporate causal factors into policy gradient reinforcement learning. This is done by having a causal factor $C_{(A|B)}$ that controls the direction of the update function. This paper suggests using sampling methods such as those by Merck and Kleinberg to find C, which does not rely on a causal graph [25].

In a paper by Nie et. al. [26] they present an *advantage doubly robust estimator* for deciding if a doctor should treat a patient or postpone the treatment.

5.3 Optimal stopping

Our work is also related to the theory of optimal stopping in mathematics, which is concerned with the problem of, during some sequence of events, taking an action that then censors all future events while still finding the stopping point with highest value [27]. This has also been used in economics and mathematical finance to determine the pricing of American options [28]. However, our problem differs in that the next event is controlled by our decision-making agent, rather than something that just happens.

5.4 Active learning

Active learning [29] has been used to learn policies that performs a minimum number of tests before an underlying hypothesis is found [30]. The difference between our case and the work by Golovin et al. [30] is that the distribution of hypotheses is unknown and the outcomes of actions are only partially observed.

6

Conclusion

This chapter discusses the results and possible further work on the topic.

6.1 Discussion

In this thesis we have presented a novel way of, given an observational data set, minimizing the number of treatments tried when trying to find a near-optimal treatment. This is done by imposing a constraint on the search process, where the search continues unless the probability of finding a better treatment is lower than some set limit δ. Using this constraint, two algorithms were implemented, one based on dynamic programming and one based on a greedy rule. Several variants of the algorithms were tested, with different strategies to handle missing data and calculating the constraint. The policies produced by these algorithms were then compared and evaluated to find which settings worked best.

Our problem does not allow us to have a single value assessing the algorithms, which means that to claim that an algorithm is better than another, the search time has to be shorter and the mean treatment effect has to be better, or one of them being equal compared to another algorithm.

In general, the difference between algorithms seem to be small on the synthetic data set, both for the constrained and the naive variants, which indicate that there are several policies that are close but not identical and that those policies can easily be found, regardless of method. While we proved that the greedy policy in general is sub-optimal in Section 3.6.3, this requires a particular set of circumstances. It requires that there is a treatment which has the highest success rate but contains almost no information about which other treatments might work. At the same time there has to be another treatment that works almost as well, but has a lot of information about which other treatments can work. The conditions required for this to be the case means that it is unlikely that the greedy policy is much worse than the optimal policy in a random setting.

Using an upper or lower bound for the constraint yields different performance for the same δ, but none of them are clearly better than the other since the same balance

between time and efficacy can be achieved using any bound. Thus the choice of bound falls to the interpretation you want to have of δ rather than any notion of performance. Using the lower bound means that the next treatment tried will always have at least a δ chance of working, while using the exact bound means that the overall chance of finding a better treatment is at least δ. The upper bound can be useful when there is not enough data to accurately calculate the exact bound since it still guarantees that there will be at least a δ chance of some treatment having a better outcome.

Choosing how to estimate the potential outcomes is quite important for the performance. With access to the true probabilities, both CDP and CG performed the best, but since the true probabilities are not generally available, the method based on the historical prior seemed to work almost as well on the synthetic data. The estimator based on function approximation also seemed to work well on synthetic data, while yielding better performance on a greater covariate space, as shown with the antibiotic resistance data set.

Most algorithms seem to find similarly performing policies at about a data set size of 4000 samples. It is reasonable to assume that estimates of potential outcomes become more reliable at that point. Presumably, more data yields better performance, even though the trend in Figure 4.6 does not seem to converge towards the true distribution even at 100 000 data samples. This is probably since most gains in time and mean treatment value is gained during the first two steps of the policy. The last few steps, e.g. when to try the fifth treatment vs. when to stop at the fourth is harder to predict since there is less data for those decisions.

CDP and CG seem to work well on real data, although the modified problem was very simplified compared to the real problem of prescribing antibiotics. For the first tried treatment of the patient, the algorithms were quite close to the performance of the real doctor. This suggests that the algorithms perform quite well, but could probably also perform better if more relevant covariates were used. However, for CDP, this would mean a great increase in runtime. While NDP also yields good performance, being able to have certain guarantees about the estimated probabilities of the outcomes of a selected treatment is favorable in medical applications.

6.2 Conclusion

The method of using a constraint to control when to search for a better treatment and when not to works well. However, there are no major improvements in performance in using that method rather than pure dynamic programming. Instead, the main reason for using the constrained versions would be that that the usage of δ lets the user have a more interpretable parameter to tune, rather than using a fixed limit on a reward that has no direct interpretation.

Constrained Greedy worked very well for most tests we did, even though it performs worse than Constrained Dynamic Programming in the worst case. However, this is

probably due to the way that our test data is structured, rather than a general idea that the greedy variant is always as good.

6.3 Future work

One of the major improvements that can be done to the method is to do all estimations using function approximation, perhaps using a neural network. This might help both the problem when our frequentist approach breaks down when we have too few data points for rare combinations of treatments and outcomes as well as allowing us to handle more treatments, outcomes, and covariates. Using function approximation instead of a table of values would also allow for use of continuous outcomes and covariates.

A possible improvement would be to try a data generating model that does not match the assumed model to try and evaluate how robust the method is when the causal assumptions do not match reality. This is very important, especially in medical settings where mistakes are extremely costly.

Bibliography

[1] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction Second edition, in progress. Technical report.

[2] Jasvinder A. Singh, Kenneth G. Saag, S. Louis Bridges Jr., Elie A. Akl, Raveendhara R. Bannuru, Matthew C. Sullivan, Elizaveta Vaysbrot, Christine McNaughton, Mikala Osani, Robert H. Shmerling, Jeffrey R. Curtis, Daniel E. Furst, Deborah Parks, Arthur Kavanaugh, James O'Dell, Charles King, Amye Leong, Eric L. Matteson, John T. Schousboe, Barbara Drevlow, Seth Ginsberg, James Grober, E. William St.Clair, Elizabeth Tindall, Amy S. Miller, and Timothy McAlindon. 2015 american college of rheumatology guideline for the treatment of rheumatoid arthritis. *Arthritis & Rheumatology*, 68(1):1–26, 2016. doi: 10.1002/art.39480. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/art.39480`.

[3] Susan A Murphy. Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(2):331–355, 2003.

[4] Xuelin Huang, Sangbum Choi, Lu Wang, and Peter F. Thall. Optimization of multi-stage dynamic treatment regimes utilizing accumulated data. *Statistics in Medicine*, 34(26):3424–3443, nov 2015. ISSN 02776715. doi: 10.1002/sim.6558. URL `http://doi.wiley.com/10.1002/sim.6558`.

[5] Yebin Tao, Lu Wang, and Daniel Almirall. Tree-based reinforcement learning for estimating optimal dynamic treatment regimes. *The annals of applied statistics*, 12(3):1914, 2018.

[6] Peng Liao, Kristjan Greenewald, Predrag Klasnja, and Susan Murphy. Personalized heartsteps: A reinforcement learning algorithm for optimizing physical activity. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–22, 2020.

[7] Ying Liu, Brent Logan, Ning Liu, Zhiyuan Xu, Jian Tang, and Yangzhi Wang. Deep Reinforcement Learning for Dynamic Treatment Regimes on Medical Registry Data. In *Proceedings - 2017 IEEE International Conference on Healthcare Informatics, ICHI 2017*, pages 380–385. Institute of Electrical and Electronics Engineers Inc., sep 2017. ISBN 9781509048816. doi: 10.1109/ICHI.2017.45.

[8] Junzhe Zhang and Elias Bareinboim. Near-Optimal Reinforcement Learning in Dynamic Treatment Regimes. Technical report.

[9] Chao Yu, Yinzhao Dong, Jiming Liu, and Guoqi Ren. Incorporating causal factors into reinforcement learning for dynamic treatment regimes in HIV. *BMC Medical Informatics and Decision Making*, 19, apr 2019. ISSN 14726947. doi: 10.1186/s12911-019-0755-6.

[10] Omer Gottesman, Fredrik Johansson, Joshua Meier, Jack Dent, Donghun Lee, Srivatsan Srinivasan, Linying Zhang, Yi Ding, David Wihl, Xuefeng Peng, et al. Evaluating reinforcement learning algorithms in observational health settings. *arXiv preprint arXiv:1805.12298*, 2018.

[11] Judea Pearl et al. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146, 2009.

[12] Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li Wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):1–9, may 2016. ISSN 20524463. doi: 10.1038/sdata.2016.35.

[13] MA Hernán and JM Robins. *Causal inference: What if*. 2020.

[14] Paul R Rosenbaum et al. *Design of observational studies*, volume 10. Springer, 2010.

[15] Steven A Julious and Mark A Mullee. Confounding and simpson's paradox. *Bmj*, 309(6967):1480–1481, 1994.

[16] Sander Greenland, Judea Pearl, and James M Robins. Causal diagrams for epidemiologic research. *Epidemiology*, pages 37–48, 1999.

[17] Judea. Pearl. *Causality : Models, Reasoning and Inference.* Cambridge University Press, 2009. ISBN 9781139638883.

[18] Jasjeet S Sekhon. The neyman-rubin model of causal inference and estimation via matching methods. *The Oxford handbook of political methodology*, 2:1–32, 2008.

[19] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, chapter 6.5 Q-Learning: Off-Policy TD Control. MIT press, 2018.

[20] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995.

[21] Ronald L Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

[22] Venkatesan T Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, and Mukesh Mohania. Decision trees for entity identification: Approximation algorithms and hardness results. In *Proceedings of the twenty-*

sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 53–62. ACM, 2007.

[23] World Health Organization. *International classification of diseases : [9th] ninth revision, basic tabulation list with alphabetic index*, 1978.

[24] Debabrata Ghosh, Shivam Sharma, Eeshan Hasan, Shabina Ashraf, Vaibhav Singh, Dinesh Tewari, Seema Singh, Mudit Kapoor, and Debarka Sengupta. Machine learning based prediction of antibiotic sensitivity in patients with critical illness. doi: 10.1101/19007153. URL `https://doi.org/10.1101/19007153`.

[25] Christopher A Merck and Samantha Kleinberg. Causal Explanation Under Indeterminism: A Sampling Approach. *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, 2016:1037–1043, feb 2016. ISSN 2159-5399. URL `http://www.ncbi.nlm.nih.gov/pubmed/31001456http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6465960`.

[26] Xinkun Nie, Emma Brunskill, and Stefan Wager. Learning When-to-Treat Policies. Technical report.

[27] James MacQueen and RG Miller Jr. Optimal persistence policies. *Operations Research*, 8(3):362–380, 1960.

[28] SD 1 Jacka. Optimal stopping and the american put. *Mathematical Finance*, 1(2):1–14, 1991.

[29] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[30] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems*, pages 766–774, 2010.

A

Appendix 1

A.1 Algorithms

A.1.1 Constrained Deep Q-learning

The q-table in the Constrained dynamic programming algorithm grows exponentially when the size of A, X, or Y is increased, which means that it is inconvenient to apply to bigger problems. A solution is to instead approximate the q-table with a function.

An attempt was made to use double q-learning as an estimator for the q-table. A feed-forward neural network is used as estimator of the q-function. The reward is calculated as before i.e. by the constraint and by equation 3.14 and 3.15. The algorithm itself is thus not constrained, but rather implied in the reward system. This method did not yield a stationary loss function, which caused the algorithm not to converge to a solution.

B

Appendix 2: Graphs

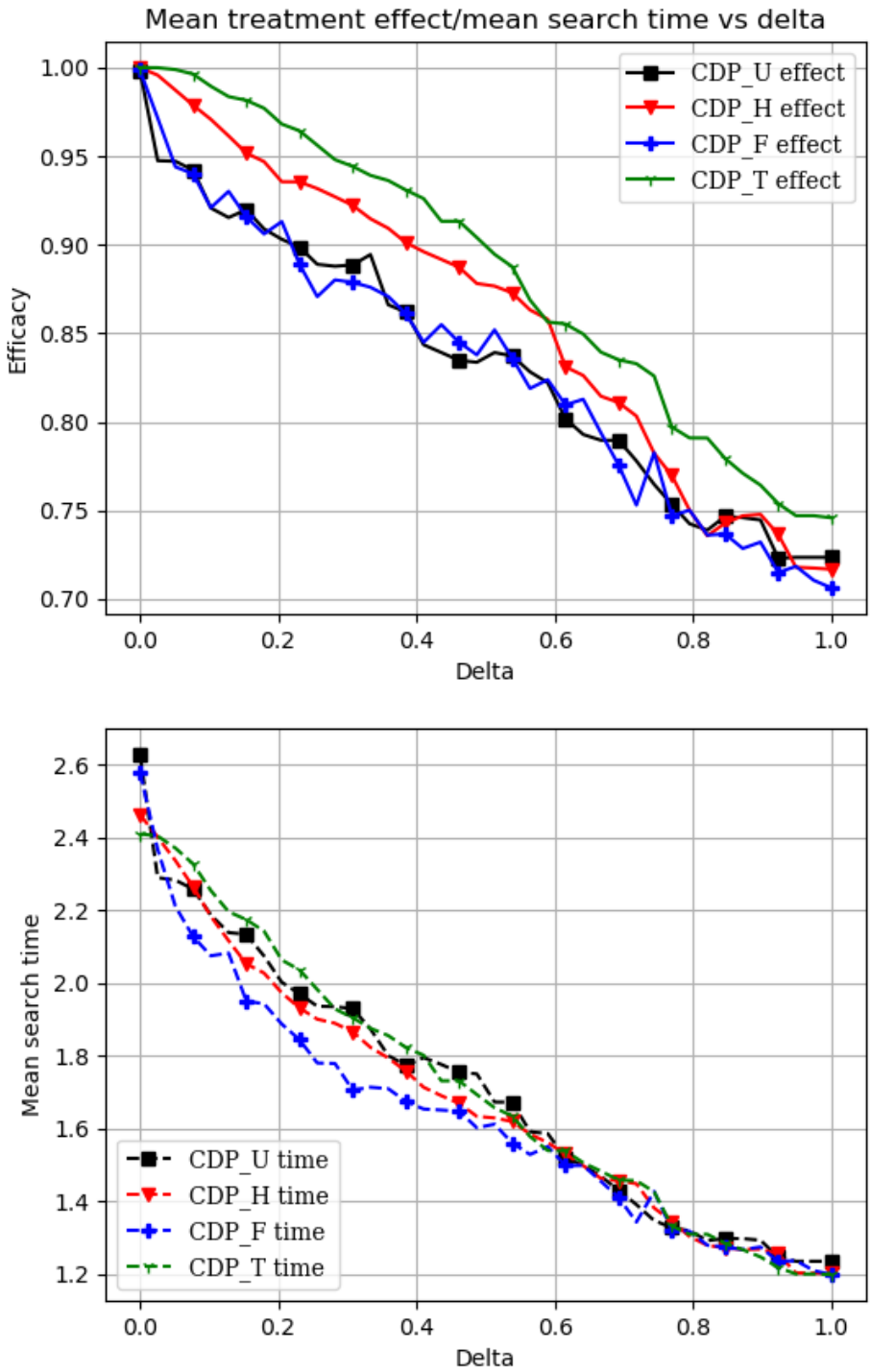


Figure B.1: Four different statistical approximators plotted over a range of delta values for the constrained dynamic programming algorithm.

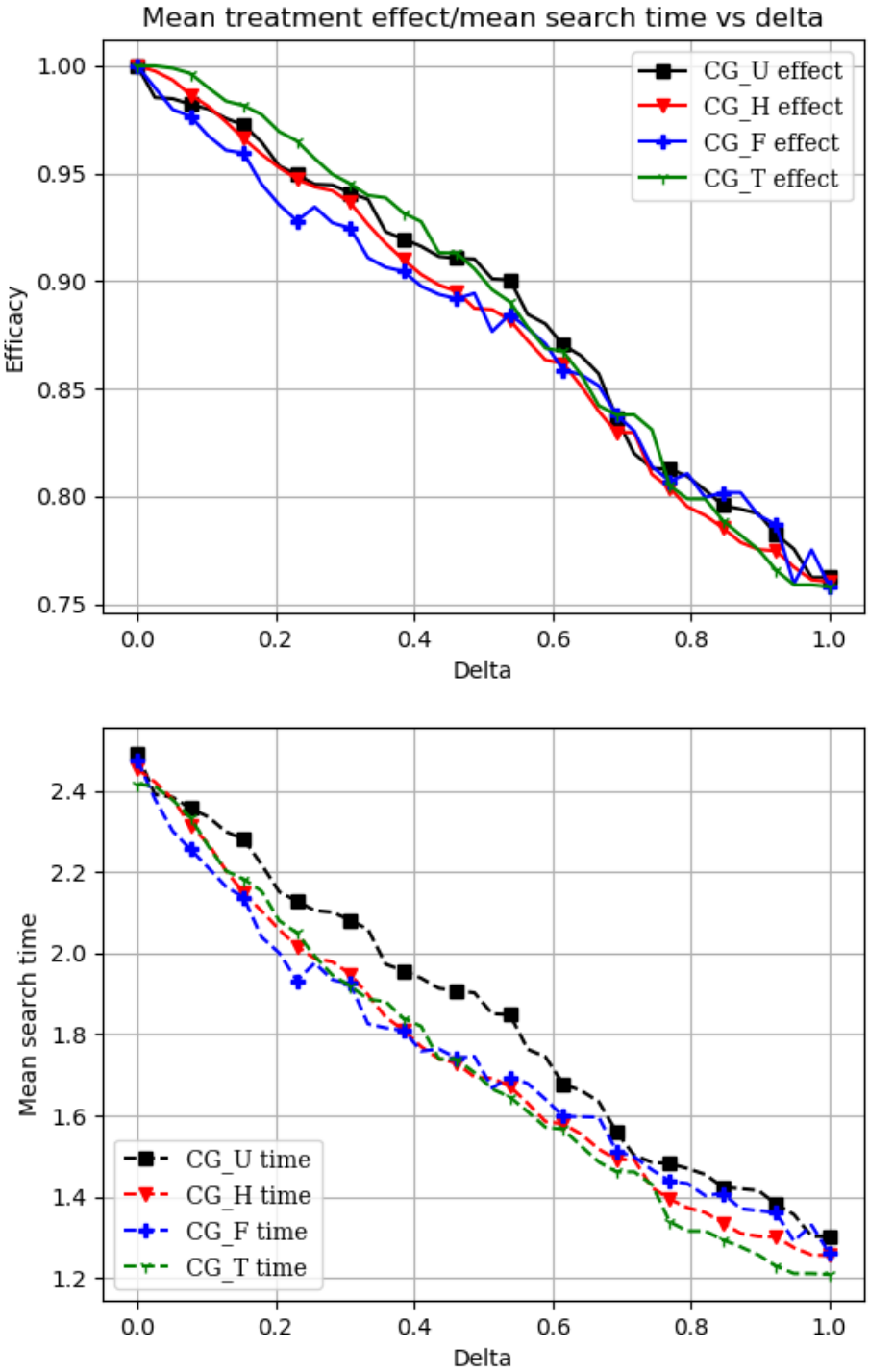


Figure B.2: Four different statistical approximators plotted over a range of delta values for the constrained greedy algorithm.

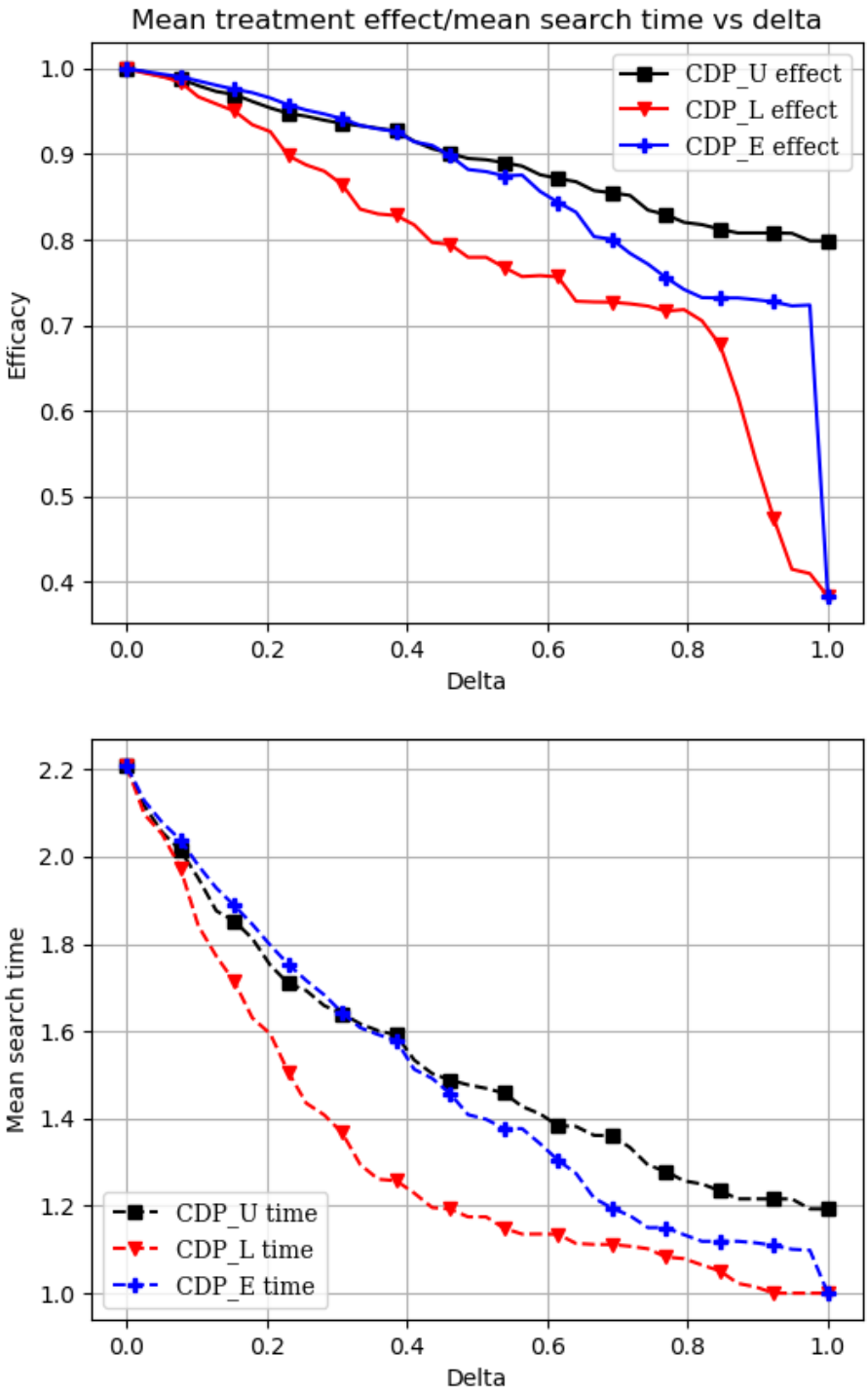


Figure B.3: Three different bounds, lower, upper and exact plotted for the CDP algorithm.

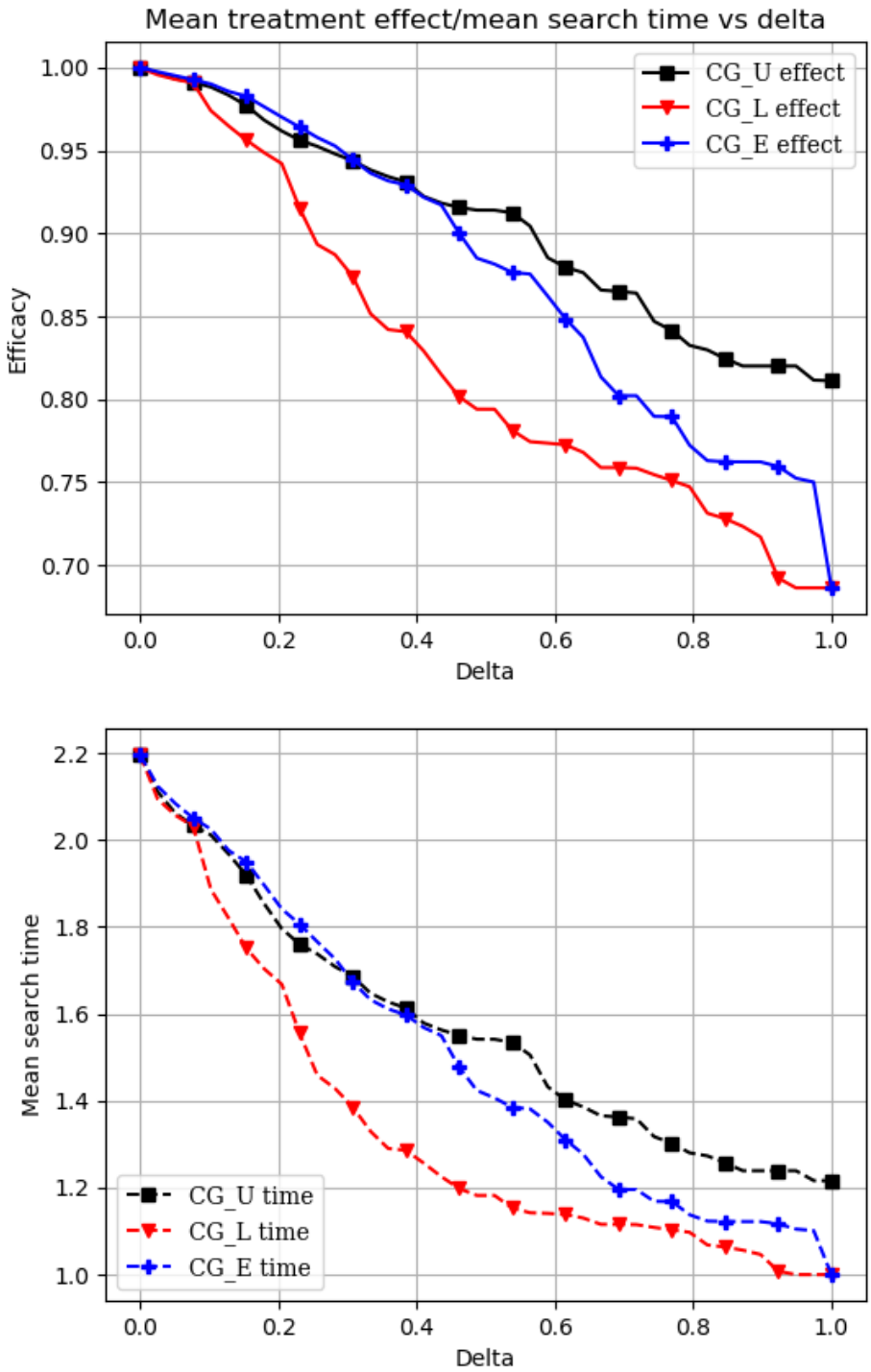


Figure B.4: Three different bounds, lower, upper and exact plotted for the CG algorithm.