Real-time Elective Admissions Planning for Health Care Providers

by

George Zhu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2012

© George Zhu 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Efficient management of patient admissions plays a critical role in increasing a hospital's resource utilization and reducing health care costs. We consider the problem of finding the best available admission policy for *elective hospital admissions* under real time constraints. The problem is modeled as a Markov Decision Process (MDP) and we investigate current state-of-the art real time planning methods.

Due to the complexity of the model, traditional model-based planners are limited in scalability since they require an explicit enumeration of the model dynamics. To overcome this challenge, we apply sample-based planners along with efficient simulation techniques that given an initial start state, generate an action on-demand while avoiding portions of the model that are irrelevant to the start state.

Results show that given reasonable resources, our approach generates improved decisions over existing alternatives that fail to scale as model complexity increases. We also propose a parameter tuning method that can be easily and efficiently implemented.

Acknowledgements

I would like to thank my supervisor Professor Jesse Hoey for his guidance, patience, and support throughout my time here at the University of Waterloo.

Table of Contents

Chapter 1

Introduction

Efficient management of patient admissions plays a critical role in reducing the overall cost of hospitalization at health care facilities. In this thesis, we consider *elective patient admissions* for non-emergency care. A typical scenario is admissions for rehabilitation hospitals where patients stay for extended periods and are not expected to require emergency care.

Under this setting, hospital administrators are given increased flexibility to plan an efficient admission policy in order to maximize the amount of available resources at their disposal. Admission schedules can be planned at a tactical level in order to reach strategic targets of resource utilization as well as limit potential bottlenecks during a patient's course of stay.

Determining an optimal admission policy is a challenging problem. Several competing factors must be considered in order to balance the number of patients admitted at any given time with the current state of resource availability. These factors include:

1. Patient flow - a typical patient follows a treatment path during her course of stay

that may require difference resources (e.g. equipment, nurses, physicians) at various stages along the way.

2. Resource utilization - a resource that is at maximum capacity blocks the patient's treatment path (e.g. patients must wait until the resource becomes free) as well as limits other resources from being used to their full capacity.

3. Stochasticity - demand for resources may fluctuate at any given time. A patient's course of treatment and resource requirements are typically non-deterministic. For example, a change in the patient's condition may result in a different set of resource requirements than was originally planned.

As the number of resources and patients increases, the complexity of the problem increases exponentially. A systematic framework is needed to assist hospital administrators in making the most efficient admission policy decisions.

Numerous studies have been conducted on the topic of patient scheduling and resource allocation within the Operations Research and Artificial Intelligence communities. A popular approach is to model the problem as a Markov Decision Process (MDP) - a well established model for decision planning under uncertainty. For our work, we adopt an MDP model from [34] that leverages similar resource consumption as *treatment patterns* in order to reduce the overall domain space. The model seeks to maximize resource utilization and can be easily extended to handle emergency admissions as well as minimizing patient wait-times.

We investigate several approaches to solving for the optimal or best available admission policy (given limited resources). Traditional MDP solvers that are applied in the research literature typically assume a very small/toy domain size and attempt to find an optimal

solution. Although ideally an optimal solution is preferred, it is quite clear that alternative methods are required due to the well-known *curse of dimensionality* issue that arises as the problem size increases. Our goal is to find an efficient real-time solution that is able to scale up under real-world settings. Our work includes applying established offline and online solvers against a real-time solving approach that has gained increasing popularity from the game playing world called UCT (Upper Confidence Bound Applied to Trees) [27]. Our results show that as the problem size increases, the UCT type solving approach that does not require an enumeration of state transition probabilities appears to be the only viable option.

The following chapters are organized as follows. Chapter 2 gives a background description of the general MDP model and planning approaches. Chapter 3 gives a literature review of related works on patient admissions and real time planning. Chapter 4 gives a formal description of the Elective Admissions MDP model and chapter 5 offers a performance analysis over the various planning algorithms we investigated. We conclude in chapter 6 with a discussion on future work.

Chapter 2

MDP and Planning Background

In this chapter we provide an overview of Markov Decision Processes (MDP) and several planning approaches used to solve them.

2.1 Markov Decision Process

MDPs are a fundamental modeling approach in decision theory and planning. They are used extensively within the Artificial Intelligence and Operations Research communities to model problems that require sequential decision making in an uncertain environment. Comprehensive studies of MDPs can be found in [38, 31].

An MDP is generally defined by the following elements:

1. S is a finite set of fully-observable possible states.
2. A is a finite set of possible actions depending on the states.

3. $P : S \times A \times [0, 1]$ is the transition function where $P(s, a, s')$ is the probability of moving to state s' when action a is applied in state s.

4. R is a real-valued reward function where $R(s, a)$ represents the expected reward for taking action a in state s.

A policy $\pi : S \to A$ specifies an action $a = \pi(s)$ to be taken when in state s. The value function:

$$V_\pi(s) = R(s, a) + \gamma \cdot \sum_{s' \in S} P(s, a, s') \cdot V_\pi(s')$$

represents the long term expected reward of executing a policy where $0 \leq \gamma \leq 1$ is a discount rate on future rewards. An optimal policy that maximizes the long term expected rewards is defined via the optimal value function:

$$V^*(s) = \max_\pi V_\pi(s) = \max_{a \in A} R(s, a) + \gamma \cdot \sum_{s' \in S} P(s, a, s') \cdot V^*(s').$$

In planning problems where time is discretized, each time period when a decision is made is referred to as a decision epoch denoted by $t \in 0, 1, \ldots, H$. If H is finite the MDP is referred to as a finite horizon MDP and an infinite horizon MDP when H is infinite.

2.1.1 Finite Horizon MDP with Fixed Start State

An MDP can be defined over a fixed start state. This is a common extension used for many probabilistic planning problems for reducing the size of the overall state space. They are also referred to as *stochastic shortest path* problems in the literature when there is a goal state and future rewards are not discounted [6].

The extension is straightforward where the value function takes in an additional parameter t representing the current decision epoch:

$$V^*(s,t) = \max_{a \in A} R(s,a) + \gamma \cdot \sum_{s' \in S} P(s,a,s') \cdot V^*(s', t-1).$$

The above equation is also known as the Bellman equation. $V^*(s_0, H)$ is the optimal value function for a finite horizon MDP with a fixed start state s_0 and a maximum horizon H. The state space for a finite horizon MDP includes only states that are reachable from s_0 up to H time periods. Therefore, there is potential to drastically reduce the overall model complexity when we are only concerned with a limited number of start states and a finite horizon.

2.2 Planners

The goal of a planner is to solve for an optimal or the best available policy for a given MDP. There has been extensive research activity on various planning approaches. In the following sections, we give an brief overview of two standard planning algorithms (Value Iteration and Real Time Dynamic Programming) as well as a more recent approach (Monte Carlo Tree Search) that has gained increasing popularity when dealing with MDPs with very large state spaces.

2.2.1 Value Iteration (VI)

Value Iteration is a classic planning algorithm widely used in the planning community for solving MDPs optimally. Algorithm 1 is the finite horizon version where the optimal value

function is generated after performing Bellman updates over all states for each horizon step. It is an offline planning algorithm and also referred to as a *synchronous* dynamic programming solution.

Algorithm 1: ValueIterationFH

Global Parameter: H **(Horizon)**

for $s \in S$ **do**
  $V_{s,0} = 0$

for $h = 1$ **to** H **do**
  for $s \in S$ **do**
    for $a \in A$ **do**
      $Q_{s,a,h} = R(s,a) + \sum_{s' \in S} P(s'|s,a) V_{s,h-1}$
    $V_{s,h} = \max_{a \in A} Q_{s,a,h}$

VI requires a complete enumeration of the state and action spaces as well as the transition dynamics. Although it generates an optimal solution it is often impractical for handling large MDPs as the complexity runs in $O(mn^2)$ per horizon step, for m actions and n states.

2.2.2 Real time Dynamic Programming (RTDP)

In recent years, there has been growing interest in real time or online MDP planning approaches. In real-world settings, practitioners often prefer an anytime solution and may only be interested in a limited number of initial start states (typically a much smaller subset of all possible states) at any given time. Whereas offline planners solve for all

possible states, real time and online planners can be used to solve for a single initial state under real time constraints.

RTDP (algorithm 2) is a real time planner that solves MDPs based on heuristic search. It is an *asynchronous* solution where updates are performed through simulated *greedy* searches over the visited states. Provided the value function is initiated with an *admissible* upper bound defined as $V(s) \geq V^*(s)$ for all states s, then RTDP converges to the optimal value function asymptotically [6].

Algorithm 2: Finite Horizon RTDP

Global Parameter: H **(horizon)**

Global Variables: $\hat{V}_{s,h}, h \in 1, \ldots, H$

Generative Model: $s' \sim G(s,a)$ `// draw` $s' \in S$ `with` $P(s'|s,a)$

repeat

  $s \longleftarrow s_H$ *// initial state*

  for $h \leftarrow H$ **to** 1 **do**

    if *first time visiting* s *at* h **then**

      $\hat{V}_{s,h} \leftarrow$ **admissible upper bound**

    for $a \in A$ **do**

      $Q(s,a) =$ **getQvalue**(s,a,h)

    $a^* \leftarrow \arg\max_{a \in A} Q(s,a)$

    $\hat{V}_{s,h} \leftarrow \max Q(s,a)$

    $s \leftarrow G(s,a)$

until *timeout* **or convergence met**

return $\arg\max_{a \in A}$ **getQValue**(s_H, a, H)

Algorithm 3: getQvalue(s, a, h)

getQvalue(s, a, h)

if $h = 1$ **then**

    return $R(s, a)$

else

    return $R(s, a) + \sum_{s' \in S} P(s'|s, a)\hat{V}_{s',h-1}$ `// where` $\hat{V}_{s',h-1} =$ ***admissible upper bound*** `if` s' `has not been visited before at horizon` $h - 1$

RTDP explores only the states that are relevant to the start state, i.e. states that are reachable from the start state given the max horizon. It generates an anytime solution that generally improves in quality as the execution time increases. However, since it requires a complete enumeration of state transition probabilities there is still limited scalability for large MDPs with dense transition dynamics.

2.2.3 Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search is a family of planning methods based on random sampling. It has had a number of notable advances recently in game playing domains such as Go [17] and Solitaire [7] where previous planners failed to solve. More recently, MCTS based planners such as UCT have become increasingly popular for solving general probabilistic planning problems (modeled as finite horizon MDPs) where states have a large number of transitions [26].

Conceptually, MCTS can be described as a real time search algorithm using a best-first approach. It generates a search tree through repeated simulation trials where the nodes

are states and branches are the actions available at each state. Sample statistics such as state-action value estimates are calculated and propagated up from the child nodes to the ancestor nodes. Figure 2.1 shows an outline of one iteration of the MCTS process [14]. New nodes are added to the tree incrementally based on a *current-best* evaluation defined by an algorithm specific *tree policy*.

One of the main advantages of MCTS is that it is *model free*, i.e. it does not require an enumeration of the model dynamics or state/action spaces. Whereas *model based* planners such as RTDP require not only a simulator but also a complete enumeration of the state transitions for all visited states, MCTS requires only the simulator. This is a significant advantage since simulators can be efficiently implemented even for very large and complex models (e.g. via a sampling function) without needing to calculate the exact transition probabilities for every possible transition. Another significant advantage is that since the search tree branches only on the available actions of each state node, the branching factor can be much smaller versus planners that branch over both next states and actions (e.g. RTDP).

2.2.4 Upper Confidence Bounds for Trees (UCT) and Multi-arm Bandits

UCT is widely considered as the current state-of-the art MCTS based planner. Its key idea is the application of a tree policy based on the *upper confidence bound* (UCB), a multi-arm bandit (MAB) sampling technique based on regret minimization.

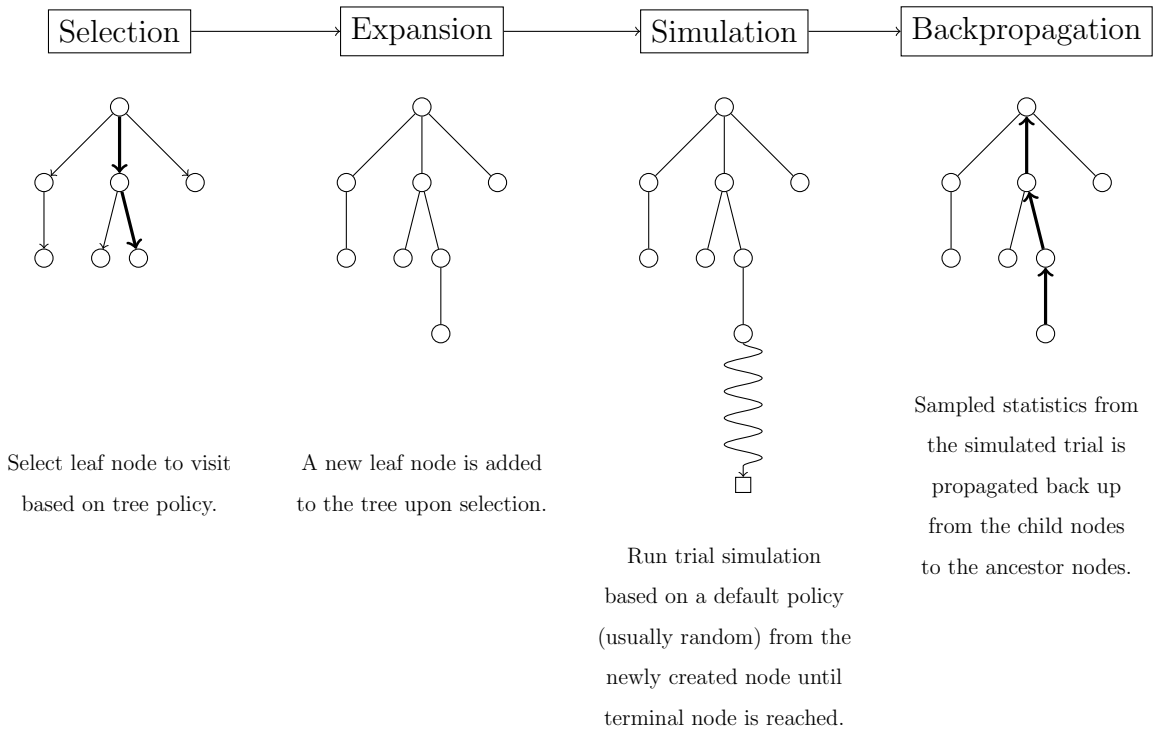

Figure 2.1: Outline of one iteration of the Monte Carlo Tree Search process

Multi-arm Bandits

Multi-arm bandits are a class of sequential decision problems where we must choose among K actions. Each action returns a random reward from an unknown distribution. The goal is to choose the action that will generate the highest cumulative reward from repeated action selections. Thus, the problem can be viewed as playing a slot machine with multiple arms. Since the reward distribution is unknown, this leads to the classic *exploration-exploitation dilemma* where one must balance between choosing what is believed to be the best action based on empirical results so far (exploitation) against trying other actions that may turn out to be more profitable in the long run (exploration).

A popular performance measure of an action selection strategy is *total expected regret*. For a bandit with K arms after T plays, it is defined as

$$R_T = T\mu^* - \sum_{t=1}^{T} \mu_j(t)$$

where μ^* represents the expected reward for the optimal action, and μ_j represents the expected reward for action $j = 1, 2, \ldots, K$.

MAB problems have been studied extensively. One of the earliest and simplest strategies is the ϵ-greedy family of algorithms. On each turn, it plays the arm with the best empirical mean reward with probability $1 - \epsilon$ and plays a random arm with probability ϵ. For constant values of ϵ, a linear bound on the expected regret in relation to the number of trials is achieved [28].

The idea of using upper confidence values was first proposed by Lai and Robbins [29] and later extended by Agrawal [1]. Based on these works, Auer et.al [3] proposed a family of more sophisticated upper confidence bound bandit algorithms including UCB1 - which

is notable for achieving logarithmic expected regret growth (uniformly and asymptotically) in relation to the number of plays.

Initially, UCB1 plays each arm once. Then at round t, it plays an arm a as follows:

$$a(t) = \arg\max_{1...k} \left(\bar{X}_i + \sqrt{\frac{2 \ln t}{n_i}} \right)$$

where $\bar{X}_i$ is the current sample mean reward for arm i and n_i is the number of times arm i has been played so far. The balance between exploration and exploitation is achieved by using both the sample mean reward which encourages exploitation (i.e. playing an arm that is the best so far) and $\sqrt{\frac{2 \ln t}{n_i}}$ which will increasingly favor arms that have been played less over time.

Lai and Robbins [29] showed that for the MAB problem, regret is bounded by $\Omega(\log n)$. Therefore, UCB1 is optimal (up to a multiplicative constant) and is the basis for the tree policy used in UCT.

UCT

UCT is a MCTS planning algorithm proposed by Kocsis and Szepesvári [27]. It is currently the most popular MCTS planning algorithm and one of the few available options for solving very large MDPs with dense transition dynamics (i.e. a high branching factor).

UCT views each node of the search tree as an independent multi-arm bandit problem corresponding to the random rewards generated through simulated trials. When selecting nodes for expansion, it applies the UCB1 bandit strategy for selecting an action a^* as follows:

$$a^* = \arg\max_{a \in A} \left(\hat{Q}_{s,a,h} + B \sqrt{\frac{2 \ln N_{s,h}}{N_{s,a,h}}} \right)$$

where $\hat{Q}_{s,a,h}$ is the current sampled mean reward for taking action a from state s at horizon h, $N_{s,h}$ is the total number of visits for state s at horizon h, $N_{s,a,h}$ is the total number of times action a has been selected for state s at horizon h, and B is a scaling constant.

A full listing of UCT is shown in algorithm 4. The scaling constant B is used as a bias modifier to adjust the amount for the exploration bonus and is domain dependent. Due to its model-free property, UCT is easy to implement and efficient. A typical implementation requires only two main components: a simulator for drawing next state transitions and a hash table for storing the sample statistics.

Algorithm 4: UCT

Global Parameters: H **(horizon),** B **(Bias Modifier)**

Global Variables: $N_{s,h}, N_{s,a,h}, \hat{Q}_{s,a,h}, h \in 1, 2, \ldots, H$

Generative Model: $s' \sim G(s,a)$ `// draw` $s' \in S$ `with` $P(s'|s,a)$

$s \longleftarrow s_H$ `// initial state`

repeat
  search(s, H)

until *timeout*

ϵ-UCT

Motivated by the initial trial results of UCT and the uniform random strategy on our test instances (see pages 41 and 42, chapter 5), we also propose a new performance tuning method that applies the concept of ϵ-greedy to UCT. We call this method ϵ-UCT where ran-

Algorithm 5: search(s, h)

if $h = 1$ **then**
  return $\arg\max_{a \in A} R(s, a)$
if $N_{s,h} = 0$ **then**
  $a \leftarrow$ random action;
  $q \leftarrow R(s, a)$+monteCarloRollout$(G(s, a), h - 1)$;
else
  $a \leftarrow \arg\max_{a \in A} \left(\hat{Q}_{s,a,h} + B\sqrt{\frac{2 \ln N_{s,h}}{N_{s,a,h}}} \right)$; `// all actions are run once initially so` $N_{s,a,h} \geq 1$
  $q \leftarrow R(s, a)+$ search$(G(s, a), h - 1)$;
$\hat{Q}_{s,a,h} \leftarrow \frac{N_{s,a,h} \cdot \hat{Q}_{s,a,h} + q}{N_{s,a,h}+1}$;
$N_{s,h} \leftarrow N_{s,h} + 1$;
$N_{s,a,h} \leftarrow N_{s,a,h} + 1$;
return q;

Algorithm 6: monteCarloRollout(s, h)

if $h = 1$ **then**
  return $\arg\max_{a \in A} R(s, a)$
$a \leftarrow$ *random action*
return $R(s, a)+$ monteCarloRollout$(G(s, a), h - 1)$

dom action selection is applied with ϵ probability and the UCT tree policy is applied with probability $1-\epsilon$ in order to further fine-tune the balance between exploration-exploitation.

Chapter 3

Literature Review

3.1 Patient Scheduling

In recent years, health care planning and control has received considerable attention as providers face increasing pressure to reduce costs and improve operational efficiency. A large body of work exists in the Operations Research and Management Science literature on patient scheduling and resource allocation. Studies typically offer tailored solutions for specific settings such as outpatient clinics [15, 18], emergency departments [30, 44] , and various types of services (e.g. diagnostic, rehabilitation). Comprehensive surveys of patient scheduling can be found in [20, 11, 18, 23].

Hans et al. [21] proposed a hierarchical framework that separates health care planning into the strategic, tactical, and operational levels. Strategic planning involves setting static long-term targets for resource capacity and staffing levels while tactical and operational planning implement policies for reaching strategic targets under a dynamic setting.

Our work in elective admissions planning falls within the tactical and operational stages

that operate with short to intermediate time horizons. Relatively few studies have been published in this area. The existing literature on elective admissions tend to focus on the modeling aspect of the problem while offering solutions that do not scale up to realistic settings.

The model we adopt for our work is a finite horizon version of the infinite horizon MDP proposed by Nunes et al [34]. It is a general framework for multiple specialties/departments and resource types. The objective is to maximize resource utilization in accordance to pre-defined targets on multiple resource types. In the original study, they were able to generate an optimal policy (using VI) for a test instance on a minimal setting but concluded that alternative planning approaches should be investigated for more realistic settings. One of the key features of the model is the use of *treatment patterns* to group multiple patients with similar demand dynamics. This allows health care providers to use their historical treatment data to potentially make drastic reductions in the model complexity.

The idea of treatment patterns was first proposed by Kapadia et al. [25, 24]. Their objective was to model the patient's treatment over the course of an extended period (e.g. days, weeks) and defined a treatment pattern as "a quantified configuration of services delivered a unit of time". Instead of using location changes or health states, treatment patterns represent the amount of resources consumed during each time step to mark a patient's treatment path over the course of his/her stay. Thus, a patient's course of treatment can be described as a sequence of treatment patterns. The study demonstrated that distinct treatment patterns as well as shifts between them can be identified using historical data from a real rehabilitation hospital and cluster analysis.

Other recent MDP-based approaches include [19, 36]. In Gocgun et al [19], a finite-horizon MDP model was applied for multi-category patient scheduling in a diagnostic hospital. They used data from a real hospital to model patient arrival patterns and used

net resource revenue in their objective function. As with Nunes et al [34], they found similar scalability issues with Value Iteration as their planner as the problem grew in size. Puterman and Queyranne [36] considered multi-priority patients and applied an approximate dynamic programming approach as their planner.

Mathematical programming is an alternative planning model that has been used extensively for general resource allocation problems. This approach was recently investigated in Hulshof et al. [22] in which they developed a Mixed Integer Programming (MIP) framework with the objective of providing equitable distribution of resources and patient access times.

The main focus of our work in the thesis is on applying a MCTS approach for generating an admission policy in real-time. Recent results from [26] show that this approach may be better suited for MDPs with very large state and action spaces. To our knowledge, there has not been a study that has investigated this approach for elective admissions planning.

3.2 Real Time Planning

Real time planning has received increasing focus as researchers look for ways to solve planning problems in an online fashion. Under many real-world settings, a complete solution of the model space is neither necessary nor achievable given limited resources. Practitioners are often interested in solving for states they are currently in and require results that can be generated within a few minutes. RTDP and more recently MCTS are two real-time planning approaches that have been studied extensively in the literature. In our case we do not hold "real time" to the same formal definition as in Computer Vision. We simply mean that the planning algorithm can be terminated at any time within a reasonable bound, e.g., in milliseconds or a few seconds.

3.2.1 RTDP

RTDP was proposed by Barto et al. [6], since then it has spawned several notable extensions. Bonet and Geffner proposed Labeled RTDP (LRTDP) [8] that used a label scheme to mark states for which the value function has stabilized. Whereas vanilla RTDP continues to visit states that have converged during its trials, LRTDP directs trial visits to only those states that have yet to converge, thereby increasing the overall convergence rate of the algorithm. Their experiments showed LRTDP converging orders of magnitude faster than RTDP on several test domains.

Later extensions of RTDP focused on maintaining lower and upper bounds on the value function to prioritize state visits where the values are the most uncertain. These include Bounded RTDP (BRTDP) [32], Focused RTDP (FRTDP) [40] and Baysian RTDP [39].

3.2.2 MCTS

MCTS based real time planning is an active research area within the Artificial Intelligence and Planning communities. Although their initial successes were primarily in the game playing domains [16, 17, 7, 5], there are increasing applications of MCTS and in particular UCT for general planning problems [26, 33, 12, 37]. A comprehensive survey of MCTS planners can be found in [9].

A major area of focus has been on improving the bandit based strategy that is central to UCT. In addition to UCB1, Auer et al. proposed UCB1-Tuned [3] which takes empirical variance into account when calculating the upper confidence bound. Their results showed UCB1-Tuned outperformed UCB1 on all test instances but they were not able to offer theoretical guarantees on its performance. Audibert et al. [2] proposed another variance based

bandit strategy called UCB-V along with analysis on the *concentration of regret*. Other notable bandit based strategies include *EXP3* [4], *Bayesian UCT* [42], and *Hierarchical Optimistic Optimization* (HOO) [10].

Pandey et al. [35] proposed a method to model dependencies between arms in the multi-arm Bernoulli bandit setting. It assumes the algorithm is given a clustering of arms where arms in the same cluster share similar rewards. It used a two-level policy to first find the best cluster, then play the best arm within that cluster. The experiments used UCB at both policy levels. Their results show a significant improvement over vanilla UCB for the online display ad problem where each arm is an ad and the reward is whether the user clicks on the ad or not. The intuition is that there are many ads that share similar click rates.

Nguyen et al. proposed a bootstrap MCTS [33] method that combined a heuristic policy along with UCT called UCT-Aux. They advocated coupling UCT with a heuristic policy that is *extreme*, i.e. either near optimal or as low as a random policy.

There is also ongoing research on the parallelisation of MCTS as multi-core processors become widely available. A discussion and analysis of MCTS parallelisation approaches can be found in [13].

Chapter 4

The Elective Admissions Model

Under the model proposed by Nunes et al. [34], we assume a setting where there is an infinite flow of patients waiting to be admitted and the decision is to output the number of patients to be admitted given the current state of the system and time horizon.

Hospital administrators can set individual resource utilization targets in order to match strategic goals. Planning is then carried out on a tactical level to admit the optimal number of patients in order to minimize resource costs which include cost penalties for when resources are over capacity, and below/over target utilization.

One of the key features of this model is the concept of *treatment patterns* that groups patients by their expected resource consumption during each time step. For example, at any given time step, there may be multiple patients who share the same consumption rate over the same set of resources. These patients are then grouped into a single treatment pattern rather than represented individually by the model. During a transition step, some patients in the pattern may transition into another pattern with different resource demands, and some patients may continue to stay in the same pattern.

Rather than enumerating an individual patient's health state or location as a marker during her "path" through the hospital, a patient's course of treatment can be represented as the amount of resources consumed during each time step. Therefore, a patient's course of treatment during her stay can be fully described by the sequence of treatment patterns they are assigned to during each time step [34]. The work from [25] shows that given a sufficiently long time step (e.g. a day or week) and sample size, the probability distribution over treatment pattern transitions can be learned from historical data.

4.1 Model Description

The overall objective is to generate an admission policy that will maximize the long term expected resource utilization under pre-set utilization targets.

We begin by describing the three basic elements that the model uses to represent the overall state of the system:

1. Specialty - the facility is divided into a set of specialties. Each specialty admits patients independently. Patients are assumed to stay within the same specialty through out their stay.

2. Treatment pattern - as stated earlier, each pattern represents a group of patients that share the same resource consumption demands over the same set of resources for a given time period. It represents the overall resource utilization across all specialties with each specialty assigning the number of patients currently under each treatment pattern, i.e. resources are shared across all specialties.

3. Resource - any hospital resource with an assigned cost and capacity. A resource can

have different cost levels based on whether it is under or over pre-set target utilization levels and capacity.

To illustrate the model further, we introduce a real-world scenario where there are three patients currently waiting to be admitted. Patients a and b are diagnosed with the same condition and are expected to have the same resource demands (e.g.medical equipment, nurses, doctors) during the current time step. Patient c has different resource demands. At time step 0, patients a and b are both assigned treatment pattern 1 since the model considers both of them as belonging to the same class in terms of resource consumption, and patient c is assigned treatment pattern 2. To calculate the cost function, we multiply the number of patients in treatment pattern 1 over the resource consumption specified for the treatment pattern, and do the same for the single patient under treatment pattern 2. If during the transition to the next time step, patient a's condition changes and his/her resource demands change, then he/she will be assigned to a different treatment pattern that matches the new resource consumption demands.

For a more detailed example, we will use the test instance specifications from Table 5.1 to illustrate an exact calculation when we formally define the cost function (subsection 4.1.3).

4.1.1 State and Action Space

States in the MDP can be expressed as

$$s = \{(s_1^1, s_2^1, \ldots, s_n^1), (s_1^2, s_2^2, \ldots, s_n^2), \ldots, (s_1^m, s_2^m, \ldots, s_n^m)\}$$

for a facility consisting of m specialties and n treatment patterns, where s_i^j denotes the number of patients under treatment pattern i at specialty j in the current time period. By

default, we use the last indexed treatment pattern for each specialty as a *discharge* pattern representing all patients being discharged during the current time period. The state space S consists of all possible states s.

Actions represent the number of admissions into the system and can be expressed as

$$a = (a^1, a^2, \ldots, a^m),$$

where a^i represents the number of patients being admitted into specialty i. The action space A represents all possible actions subject to the constraints that will be described later.

Throughout the paper, we use superscripts to represent the specialty index and subscripts to represent the treatment pattern index whenever these distinctions are needed. A full notations table is listed under the appendix section.

4.1.2 Stochastic Dynamics

The model assumes the stochastic dynamics are independent for each specialty. Figure 4.1 shows a dynamic decision network for the transition to the next state given the current state and action. Patients can only transition from the current pattern to another pattern within the same specialty. The probability of reaching s' given s, a can be expressed as

$$P(s'|s,a) = \prod_{i=1}^{m} P(s^{i\prime}|s^i, a^i).$$

where $s^{i\prime}$ is the number of patients in specialty i at time step $t+1$, s^i is the number of patients in specialty i at time step t, and a^i is the number of patients being admitted to specialty i in the current step step.

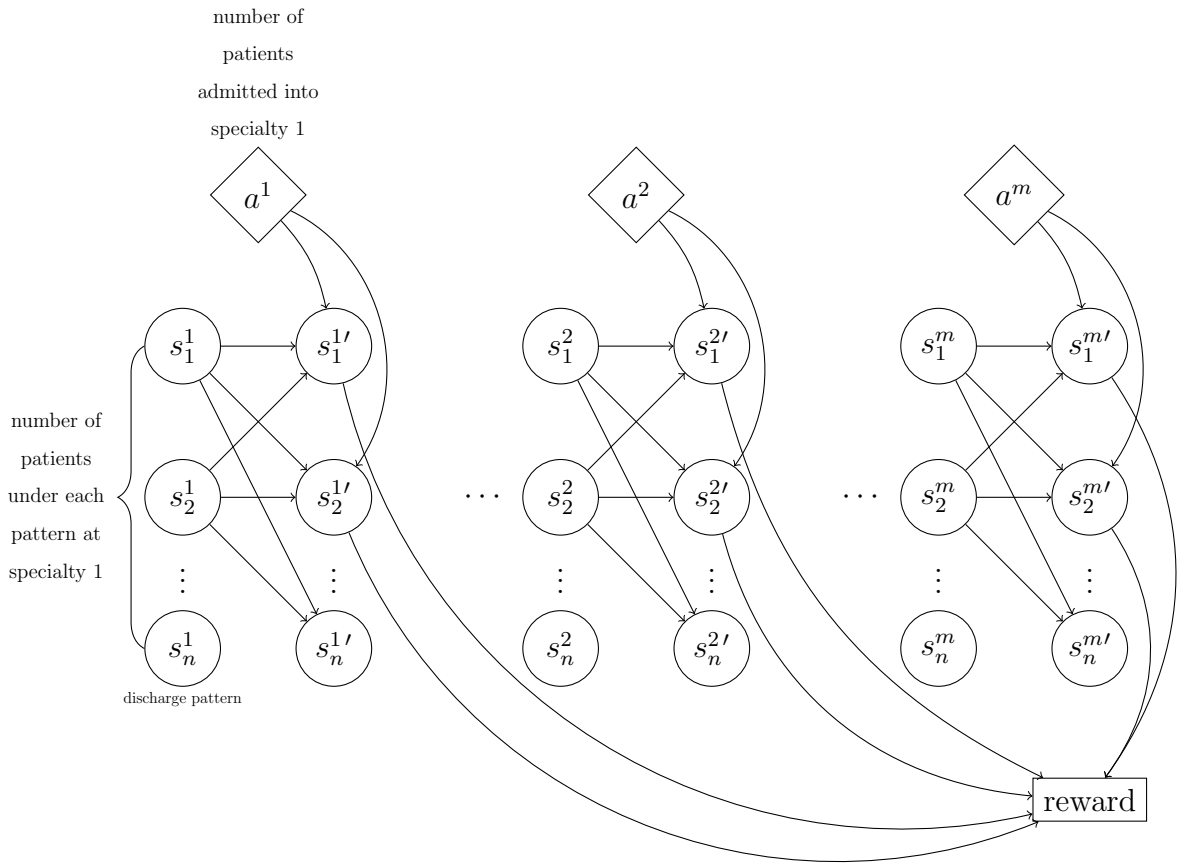

Figure 4.1: Dynamic decision diagram for the Hospital MDP with m specialties and n patterns.

In order to find $P(s^{i\prime}|s^i, a^i)$, we assume prior knowledge on the following data:

1. Transition probabilities between all treatment patterns within the specialty. This applies to all patients under the same pattern and can be expressed as

$$\begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ \vdots & \vdots & \vdots & \\ p_{n-1,1} & p_{n-1,2} & \dots & p_{n-1,n} \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

   where p_{ij} denotes the probability of any single patient currently under treatment pattern i at time t transitioning to treatment pattern j at time $t+1$.

2. The probability of an admitted patient entering each treatment pattern denoted by the entrance probability vector $(p_1, p_2, \dots, p_{n-1}, 0)$.

For clarity, we do not show the specialty superscript.

The above data allows us to calculate $P(s^{i\prime}|s^i, a^i)$. Figure 4.1 shows that the number of patients in each treatment pattern during a transition is dependent on all other treatment patterns that have a non-null probability of transition into it, as well as the probablity of an admitted patient entering the pattern. To find $P(s^{i\prime}|s^i, a^i)$, we must consider every pattern to pattern transition and entrance possibility that would cause s^i to transition into $s^{i\prime}$, given a^i. Let

$$\mathbf{x} = ((x_{11}, x_{12}, \dots, x_{1n}), (x_{21}, x_{22}, \dots, x_{2n}), \dots, (x_{n1}, x_{n2}, \dots, 0))$$

be a vector of n random vectors where $\mathbf{x}_{ij}, i < n$ represents the number of patients transitioning from treatment pattern i to j and $\mathbf{x}_{nj}$ represents the number of admitted patients

entering treatment pattern j. Let $\mathbf{R}_{\mathbf{x}s^{i\prime}|a^i,s^i}$ be the support of $\mathbf{x}$ with respect to the transition $(s^{i\prime}|a^i, s^i)$ such that $\forall \mathbf{x} \in \mathbf{R}_{\mathbf{x}s^{i\prime}|a^i,s^i}$:

$$\mathbf{x} = \left\{ x \in \{1, 2, \ldots\}^{n^2} : \forall k \in \{1, 2, \ldots, n\}, \sum_{j=1}^{n} \mathbf{x}_{jk} = s_k^{i\,\prime} \wedge \forall j \in \{1, 2, \ldots, n\}, \sum_{k=1}^{n} \mathbf{x}_{jk} = s_j^i \right\}.$$

That is, $\sum_{j=1}^{n} \mathbf{x}_{jk} = s_k^{i\,\prime}$ specifies that the total flow of patients transitioning into pattern k from all other patterns j (including $j = k$) must equal to the number of patients in pattern k in the next time step. Similarly, $\sum_{k=1}^{n} \mathbf{x}_{jk} = s_j^i$ specifies the total number of patients transitioning out of pattern j to all other patterns (including $k = j$) must equal to the number of patients in pattern j in the current time step. Therefore, each $\mathbf{x} \in \mathbf{R}_{\mathbf{x}s^{i\prime}|a^i,s^i}$ represents the pattern to pattern transitions and admission entrance possibilities that can result in the specialty transition $(s^{i\prime}|s^i, a^i)$. The random vectors in $\mathbf{x}$ follow a multinomial distribution with parameters that can be taken directly from the state and action variables and the patient transition and entrance probabilities tables. Therefore, the transition probability for a single specialty can be expressed as:

$$P(s^{i\prime}|s^i, a^i) = \sum_{\mathbf{x} \in \mathbf{R}_{s^i|a^i,s^{i\prime}}} f\left(\mathbf{x}_{n\cdot}, a^i, (p_1, p_2, \ldots, p_{|\mathbf{x}_n|})\right) \prod_{j=1}^{n-1} f\left(\mathbf{x}_j, s_j^i, (p_{j1}, p_{j2}, \ldots, p_{jn})\right),$$

where $\mathbf{x_n}$ is the vector containing the number of patients being admitted into the specialty under each pattern, $p_1, \ldots, p_{|\mathbf{x}_n|}$ is the corresponding entrance probability vector, $\mathbf{x}_j$ is the vector containing the number of patients transitioning from pattern j to all other patterns (including j itself), and $p_{j1}, \ldots, p_{jn}$ are the corresponding transition probabilities, and $f\left(\mathbf{u}, s, (p_1, p_2, \ldots, p_{|\mathbf{u}|})\right)$ is the probability mass function for a multinomial distribution over a set of random variables U where $U_1 = \mathbf{u}_1, \ldots, U_{|\mathbf{u}|} = \mathbf{u}_{|\mathbf{u}|}$ and $p_1 = P(U_1 = \mathbf{u}_1), \ldots, p_{|\mathbf{u}|} = P(U_{|\mathbf{u}|} = \mathbf{u}_{|\mathbf{u}|})$:

$$f\left(\mathbf{u}, s, (p_1, p_2, \ldots, p_{|\mathbf{u}|})\right) = \begin{cases} \frac{s!}{\mathbf{u}_1!\mathbf{u}_2!\ldots\mathbf{u}_{|\mathbf{u}|}!} p_1 \ldots p_{|\mathbf{u}|} & \text{when } \sum_{i=1}^{|\mathbf{u}|} \mathbf{u}_i = s \\ 0 & \text{otherwise} \end{cases}$$

4.1.3 Reward Function

The reward function models the costs incurred when the utilization of a resource is over its maximum capacity, as well as whether it is over or under its targeted utilization level set by the hospital administrator.

For a hospital with l resources, let

$$C = ((c_{11}, c_{12}, \ldots, c_{1l}), \ldots, (c_{n1}, \ldots, c_{nl}))$$

where C_{ij} represents the per patient consumption from treatment pattern i on resource j. Let $O = (o_1, o_2, \ldots, o_l)$, $B = (b_1, b_2, \ldots, b_l)$, $D = (d_1, d_2, \ldots, d_l)$, $E = (e_1, e_2, \ldots, e_l)$ and $F = (f_1, f_2, \ldots, f_l)$, where O_i, B_i, D_i, E_i, F_i represents the over cost, excess cost, idle cost, target and maximum capacity for resource i, respectively. The over cost represents the cost for when a resource is over its maximum capacity, the excess cost represents the cost for when a resource is over its target utilization level, and the idle cost represents the cost for when a resource is under its target utilization level.

Given the above definitions, the immediate reward for being in a state can be expressed as

$$R(s) = -\sum_{k=1}^{l} \left\{ \begin{array}{l} O_k \times \max\left(\sum_{j=1}^{m}\sum_{i=1}^{n} C_{ik} s_i^j - F_k; 0\right) + \\ B_k \times \max\left(\sum_{j=1}^{m}\sum_{i=1}^{n} C_{ik} s_i^j - E_k; 0\right) + \\ D_k \times \max\left(E_k - \sum_{j=1}^{m}\sum_{i=1}^{n} C_{ik} s_i^j; 0\right) \end{array} \right\} \tag{4.1}$$

for a hospital with l resources.

For example, under the instance specifications from table 5.1 and a test state $s = [(0,0,0)(0,0,0)]$,

$$R(s) = -(0+0+(1.0*(4-0)) + (0+0+(1.6*(4-0)) = 10.4$$

as a state with no patients in any specialty would incur only the idle cost for each resource.

The best possible immediate reward of a state is 0, i.e. the state is in the ideal utilization level that does not incur any over, excess or idle costs.

4.1.4 Model Constraints

The main challenge in planning for this domain is how quickly the model grows in complexity with only a slight increase to the instance settings (e.g. resource capacity, max number of admissions per time period, number of specialties and treatment patterns). In order to make the model scalable to a realistic setting, we first apply the following constraints:

1. any state that is expected to consume a resource over its capacity in the next time period stops admitting new patients.

2. at time 0, only states that are not expected to be over the capacity limit for any resource is allowed, i.e. states that will incur an over capacity cost in the next time step are not allowed to be an initial start state.

3. we define the MDP over a finite horizon, the maximum number of transitions for any state is bounded by the horizon parameter.

We implement these constraints in the model by specifying a default empty state, i.e. $[(0, \ldots, 0), \ldots, (0, \ldots, 0)]$ as the initial seed state and a no-op action where we admit 0 patients across all specialties if the current state is expected to be over capacity in the next time step. This allows us to reduce the number of possible states up to a manageable level while still maintaining a realistic setting for a typical treatment facility.

Chapter 5

Performance Analysis

In this chapter we present the trial results and analysis on the planning algorithms described in chapter 2.

We tested the algorithms on two instances of the model - one with a small number of domain elements where we were able to compare results against an optimal policy and a larger instance with more realistic settings where the optimal planner fails to solve.

5.1 Experimental Setup

We measure the performance of a planner based on the cumulative rewards received from executing the planner's policy over a number of time steps. Give an initial start state $s = s_H$, and horizon H and a planner that takes parameters s and $h \in 1, 2, \ldots, H$ and returns an action a, each trial consists of the following steps:

1. Query the planner with s and h and receive an action $a \in A$ as the result.

2. Simulate the next state by drawing s' given s and a based on the model's transition probabilities and record the immediate reward $R(s')$. Set $s = s'$ and $h = h - 1$.

3. Continue to step 1 until $h = 0$.

The trial reward is the sum of the immediate rewards (undiscounted) recorded during step 2. All trials were executed on machines with an AMD Opteron @ 2.2 GHz processor and a max memory setting of 8.0 GB. For each of the real-time planners, we ran trials on time-out settings of 100, 1,000, 10,000, 100,000, 1,000,0000 milliseconds. A total of 100 trials were executed for each time-out setting and we used the average trial reward along with the standard error as the score.

5.2 Instance Specification

In this section we present the specifications for each test model instance.

5.2.1 Small Test Instance

For the small test instance, we use the same specifications from [34] which consists of 2 specialties, 3 treatment patterns (including the discharge pattern), and 2 resources. Table 5.1 shows a complete listing of the specifications used to generate the instance. The total number of possible states in this instance is 5,765 states with over 17 million state transitions.

Average resource utilization per period

	resource 1	resource 2
pattern 1	2.2	2.6
pattern 2	2.6	2.2
pattern 3 (discharge)	-	-

Resource costs, capacity, and target level

	resource 1	resource 2
over cost	1.0	1.0
excess cost	1.5	1.0
idle cost	1.0	1.6
capacity	5	5
target level	4	4

Treatment pattern transition probabilities

	pattern 1	pattern 2	pattern 3
specialty 1			
pattern 1	0.4	0.1	0.5
pattern 2	0.1	0.3	0.6
pattern 3	0.0	0.0	1.0
specialty 2			
pattern 1	0.2	0.1	0.7
pattern 2	0.1	0.2	0.7
pattern 3	0.0	0.0	1.0

Entrance probabilities

	specialty 1	specialty 2
pattern 1	0.5	0.4
pattern 2	0.5	0.6
pattern 3	0.0	0.0

Maximum number of admissions per specialty = 2 patients

Table 5.1: Small instance specification

5.2.2 Large Test Instance

Table 5.2 lists the specifications for the large test instance. We increased the settings to 4 specialties, 4 treatment patterns (including the discharge pattern), and 4 resources while keeping the same number of maximum admissions per specialty and resource capacities.

Each specialty can admit from 0 to 2 patients which makes $3^4 = 81$ possible actions. The resulting instance proved too large for Value Iteration and RTDP to solve.. Therefore, the results include comparisons between planners that rely solely on a simulator for execution.

5.3 Results

In this section we present the trial results for both test instances.

5.3.1 Small Test Instance

For the small test instance we were able to generate an optimal policy using the Finite Horizon Value Iteration algorithm described in Algorithm 1. Figure 5.1 and table 5.3 shows the results for running the planners for $H = 10$ horizon steps. The average trial reward for the optimal policy is measured at -42.64 (std. error = 0.59) by returning the optimal action to the simulator for each horizon step. Although the settings for this instance are extremely small, the goal is to allow a comparison against an optimal policy that would be impossible to generate for larger and more realistic settings.

From the results we can see that of the MCTS planners, 0.50-UCT clearly has the best performance on all time-out settings. Comparing the ϵ-greedy planners, there is consistent

Average resource utilization per period

	resource 1	resource 2	resource 3	resource 4
pattern 1	1.0	0.5	0.5	1.0
pattern 2	0.5	1.0	2.0	0.25
pattern 3	2	0.5	1.5	0.5

Resource costs, capacity, and target level

	resource 1	resource 2	resource 3	resource 4
over cost	1.5	2.0	2.5	1.5
excess cost	1.0	1.5	1.5	0.5
idle cost	2.0	3.0	1.5	1.0
capacity	5	5	5	5
target level	4	4	4	4

Treatment pattern transition probabilities

	pattern 1	pattern 2	pattern 3	pattern 4
specialty 1				
pattern 1	0.40	0.10	0.20	0.30
pattern 2	0.10	0.30	0.40	0.20
pattern 3	0.80	0.10	0.00	0.10
pattern 4	0.00	0.00	0.00	1.00
specialty 2				
pattern 1	0.10	0.30	0.40	0.20
pattern 2	0.80	0.10	0.00	0.10
pattern 3	0.25	0.25	0.25	0.25
pattern 4	0.00	0.00	0.00	1.00
specialty 3				
pattern 1	0.80	0.10	0.00	0.10
pattern 2	0.40	0.10	0.00	0.50
pattern 3	0.30	0.30	0.20	0.20
pattern 4	0.00	0.00	0.00	1.00
specialty 4				
pattern 1	0.50	0.15	0.00	0.35
pattern 2	0.40	0.10	0.00	0.50
pattern 3	0.80	0.10	0.00	0.10
pattern 4	0.00	0.00	0.00	1.00

Entrance probabilities

	specialty 1	specialty 2	specialty 3	specialty 4
pattern 1	0.20	0.40	0.15	0.00
pattern 2	0.40	0.25	0.70	0.00
pattern 3	0.40	0.35	0.15	0.00
pattern 4	0.00	0.00	0.00	0.00

Maximum number of admissions per specialty = 2 patients

Table 5.2: Large instance specification

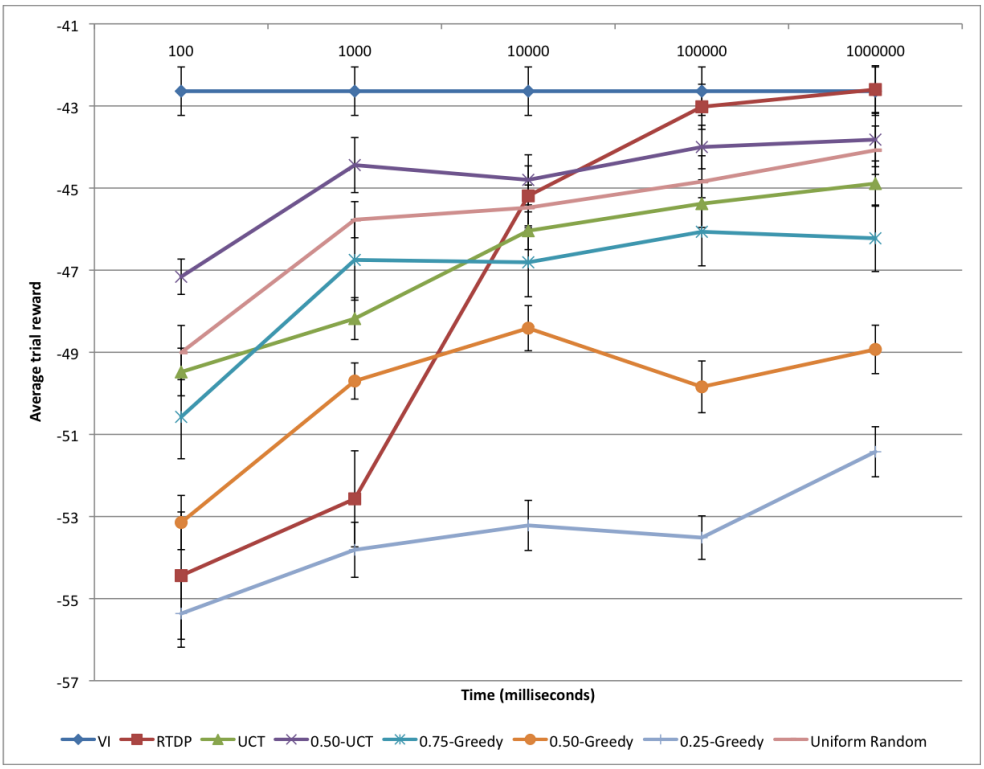

Figure 5.1: Small instance with horizon = 10

improvement as the ratio of random sampling increases. Surprisingly, the uniform random sampling method achieves the second best result of all MCTS methods including UCT.

For RTDP, its results are significantly lower compared to all but one other planner until the 10,000 millisecond time-out mark where it starts to overtake all other planners and eventually converges to the optimal policy at the 1,000,000 milliseconds time-out mark. Even on this small instance, there is significant ramp up time for RTDP to generate good results as compared to the MCTS planners.

Figure 5.2 and table 5.4 show the results for increasing the horizon steps to $H = 20$. As can be seen, the increased horizon has a signficant impact on RTDP's performance.

planner/timeout	100	1,000	10,000	100,000	1,000,000
VI	-42.64 (±0.59)	-42.64 (±0.59)	-42.64 (±0.59)	-42.64 (±0.59)	-42.64 (±0.59)
RTDP	-54.44 (±1.55)	-52.57 (±1.17)	-45.19 (±0.73)	-43.02 (±0.55)	-42.6 (±0.58)
UCT	-49.48 (±0.58)	-48.18 (±0.51)	-46.04 (±0.46)	-45.38 (±0.58)	-44.89 (±0.55)
0.50-UCT	-47.16 (±0.43)	-44.44 (±0.67)	-44.8 (±0.61)	-44.00 (±0.53)	-43.82 (±0.66)
0.75-Greedy	-50.57 (±1.02)	-46.75 (±0.98)	-46.81 (±0.84)	-46.06 (±0.83)	-46.22 (±0.81)
0.50-Greedy	-53.15 (±0.81)	-49.70 (±0.54)	-48.41 (±0.58)	-49.84 (±0.47)	-48.93 (±0.55)
0.25-Greedy	-55.36 (±0.82)	-53.81 (±0.67)	-53.22 (±0.61)	-53.51 (±0.53)	-51.42 (±0.61)
Uniform Random	-49.00 (±0.66)	-45.77 (±0.44)	-45.48 (±0.55)	-44.84 (±0.63)	-44.08 (±0.59)

Table 5.3: Small instance with horizon = 10

planner/timeout	100	1,000	10,000	100,000	1,000,000
VI	-80.47 (±0.88)	-80.47 (±0.88)	-80.47 (±0.88)	-80.47 (±0.88)	-80.47 (±0.88)
RTDP	-125.92 (±3.38)	-115.05 (±2.19)	-98.21 (±1.77)	-84.22 (±1.02)	-80.08 (±0.84)
UCT	-105.68 (±0.87)	-89.48 (±1.02)	-83.07 (±0.87)	-82.89 (±0.83)	-81.89 (±0.88)
0.50-UCT	-91.89 (±0.81)	-85.83 (±0.77)	-81.37 (±0.86)	-81.02 (±0.88)	-80.32 (±0.77)
0.75-Greedy	-98.86 (±1.31)	-89.52 (±0.81)	-88.61 (±0.72)	-87.22 (±0.83)	-87.40 (±0.9)
0.50-Greedy	-101.31 (±1.42)	-97.14 (±1.13)	-96.27 (±1.09)	-96.12 (±1.20)	-96.67 (±1.20)
0.25-Greedy	-104.55 (±1.65)	-103.08 (±1.41)	-101.16 (±1.39)	-102.20 (±1.46)	-101.86 (±1.43)
Uniform-random	-97.62 (±1.35)	-87.27 (±0.88)	-84.47 (±0.86)	-82.96 (±0.9)	-81.79 (±0.87)

Table 5.4: Small instance with horizon = 20

RTDP's results are lower against all but one other planner until the 100,000 millisecond time-out mark while the performance of the MCTS planners are not impacted significantly.

Overall, 0.50-UCT again shows the best performance where it converged to the optimal policy at the 10,000 miliseconds time-out mark. Between UCT and uniform-random, their results are much closer to each other at the increase horizon setting. Although UCT starts lower, its results are statistically tied with uniform-random after the 10,000 milliseconds time mark.

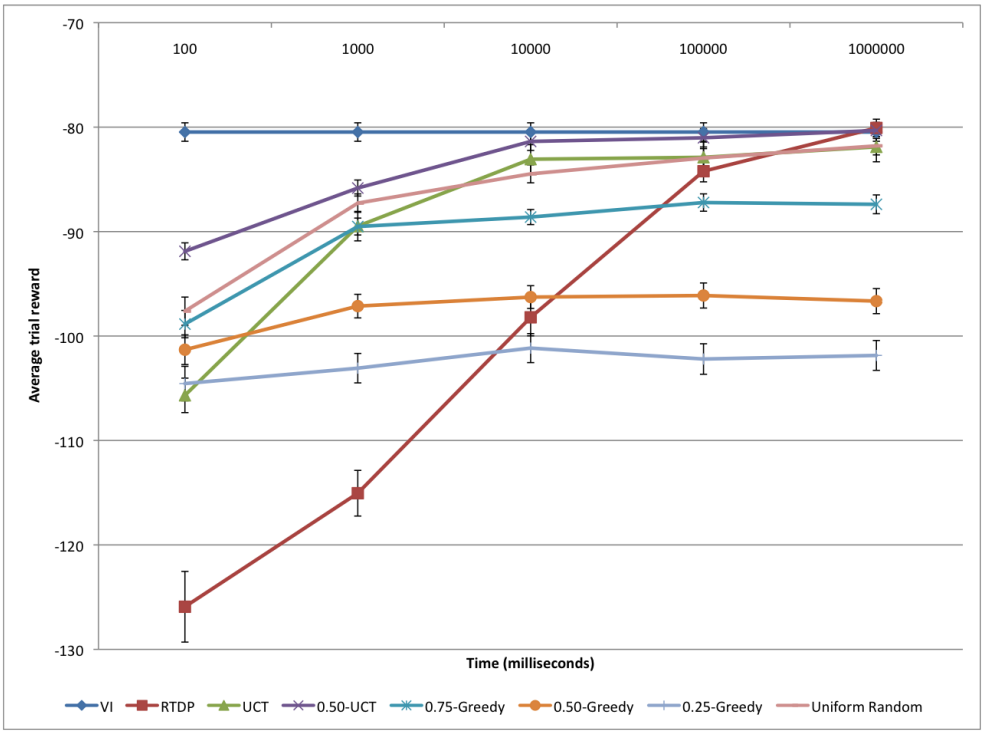

Figure 5.2: Small test instance with horizon = 20

5.3.2 Large Test Instance

Figure 5.3 and 5.5 shows the results of running trials on the large test instance with horizon $H = 10$. As previously mentioned, we were not able to generate the optimal policy on this increased setting using VI nor were we able to run RTDP in any reasonable amount of time (i.e. we were not able to complete a single RTDP trial in under an hour). Therefore, results are included for only the MCTS planners.

One noticeable difference in these results is the performance of UCT where it has the lowest initial score but then achieves almost best overall results at the time-out mark of 1,000 milliseconds. However, it experiences a significant dip at the next time-out mark

before recovering.

On the last two time-out marks, both 0.50-UCT and UCT are statistically tied for best performance with 0.50-UCT consistently improving after each time-out mark. Uniform-random has lower results for the first two time-out marks but overtakes the ϵ-greedy planners for the last three time-out marks.

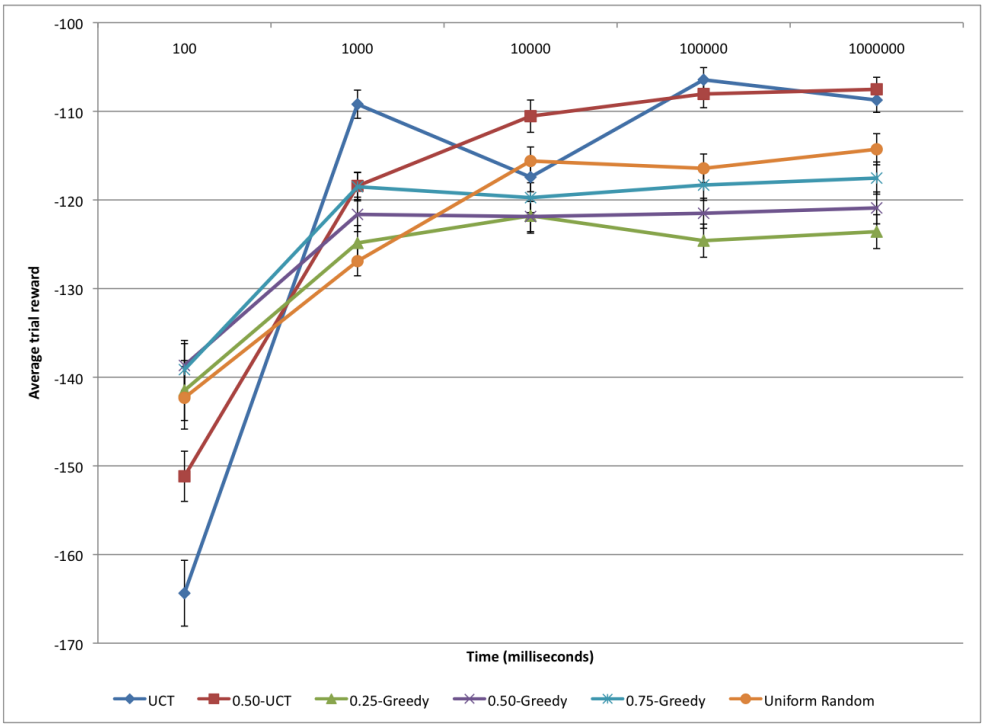

Figure 5.3: Large test instance with horizon = 10

5.4 Discussion

We were most surprised that the naive uniform random action selection policy performed well past expectations on all of the test instances, especially in relation to UCT. We

planner/timeout	100	1,000	10,000	100,000	1,000,000
UCT	-164.37 (±3.71)	-109.2 (±1.59)	-117.42 (±1.65)	-106.44 (±1.37)	-108.73 (±1.39)
0.50-UCT	-151.17 (±2.83)	-118.41 (±1.54)	-110.55 (±1.82)	-108.05 (±1.55)	-107.52 (±1.36)
0.75-Greedy	-139.14 (±2.94)	-118.51 (±1.60)	-119.74 (±1.68)	-118.32 (±1.73)	-117.53 (±1.81)
0.50-Greedy	-138.67 (±2.83)	-121.63 (±1.94)	-121.87 (±1.70)	-121.50 (±1.69)	-120.90 (±1.80)
0.25-Greedy	-141.49 (±3.38)	-124.85 (±1.96)	-121.78 (±1.96)	-124.59 (±1.87)	-123.58 (±1.91)
Uniform-random	-142.32 (±3.53)	-126.90 (±1.66)	-115.60 (±1.62)	-116.44 (±1.62)	-114.28 (±1.76)

Table 5.5: Large instance with horizon = 10

expected UCT to be the top ranked MCTS planner given its reputation and popularity. However, while UCT does indeed outperform uniform random on the large test instance on most time-out settings, it was tied or beaten on both horizon settings of the small test instance (on most time-out settings).

Reviewing UCT's sampling percentages over all actions, it was evident that we needed to tune the bias modifier to increase the exploration bonus, i.e. it was under-exploring potentially better actions on the small test instance. This raises an important and often overlooked issue - it is difficult to determine what is an appropriate bias modifier setting until we compare trial results from various settings, which brings us back to the exploration-exploitation dilemma. Although UCT and many of the bandit strategies offer theoretical guarantees asymptotically, in practice what is desired is a convergence condition that can be evaluated under a finite setting.

It is also notable that on the large test instance, UCT suffered a dip in performance as the time-out increased from 1,000 to 10,000 milliseconds. In theory UCT should return better results given more execution time. However, our results show that as instance size becomes large short term performance gains do not always hold as UCT can often over-exploit suboptimal actions due to sample variance. It will eventually recover due to the

exploration bonus increase over time but the process can be slow with a larger number of actions to sample from as well as a bias modifier that is not properly tuned.

Given our initial results, we were motivated to find a more efficient method of tuning the balance between exploration-exploitation. Applying the ϵ-greedy concept, we implemented ϵ-UCT and set ϵ to 0.50, i.e. apply a 50/50 split between random action selection and the UCT policy. As results show, 0.50-UCT was the overall best MCTS planner on all of the test instances. Furthermore, 0.50-UCT consistently improved as time-out settings increased and was able to converge to the optimal policy on both the small instance settings.

For RTDP, the main cause behind the slower ramp up time is due to it having to calculate action values for all possible next states for each state it visits at each horizon. Even on the small test instance setting, the search tree it generates is substantial as the horizon increases. On average, we measured an average of 19 next states for each visited state. Combined with 9 possible actions this results in a much longer execution time for each search trial. In contrast, the search tree for the MCTS planners branches only on the possible actions. They store a running average of the action values over the visited states and therefore do not require the branching on next states during the search rollout.

Overall, we are encouraged by the performance of 0.50-UCT. Combining random sampling with UCT offers a mechanism that can be tuned efficiently while providing behavior that can be easily understood by the practitioner. As model complexity increases, an efficient tuning method is desirable since trials become much more expensive to run.

5.4.1 Model Complexity and Scalability

Under this model, the number of specialties and treatment patterns alone does not necessarily imply increased complexity. Instance size can be dramatically affected by only a

slight adjustment in resource capacity, transition density between treatment patterns, and the maximum number of admissions per period for each specialty.

Although MCTS planners do not branch over next states, they are nevertheless affected by the number of next state transitions. For example, if we double the setting of our large test instance to 8 specialties and 8 treatment patterns while keeping similar resource capacities and transition density, the number of next state transitions can easily reach tens of thousands for a given state with many low probability transitions. Under this setting, UCT or any standard sample-based planner will end up defaulting to the random policy since they will rarely visit the same state again.

Chapter 6

Conclusion

In this thesis, we applied current state-of-the-art real time planning techniques for Elective Admissions Planning. To our knowledge, no existing studies have focused on applying the MCTS family of planning algorithms on this challenging domain. Through extensive trials, we showed MCTS to be a viable option for solving larger instances of the model where traditional planners (e.g. VI and RTDP) were unable to solve.

6.1 Future Work

In our work, we did not consider patient wait-times nor different priority levels in our model. For a real hospital setting, we would need to extend the model to cover these additional requirements. These extensions should be investigated in order to find efficient ways of building a more complete model.

Given the lack of real hospital admission data, we were unable to apply any domain knowledge into the planning algorithms. Domain-specific adaptations have been shown to

dramatically improve MCTS planners (e.g. Gelly et al. [17]). There are also opportunities to apply machine learning techniques to better guide the search strategy if given access to real patient treatment patterns.

There is also potential in combining planning approaches other than the random action selection we have proposed. One interesting approach would be to combine the RTDP and UCT approaches together. Ideally, we would like to calculate the value function using the exact transition probabilities rather than the sampling approach. However, this is an expensive operation when dealing with large MDPs with dense transition dynamics. However, in a realistic setting there may be states with significantly fewer transitions than others. Then it would make sense to apply asynchronous backups over states with sparse transitions and UCT for states with dense transitions. The trick is identifying the transition densities cheaply without doing the actual enumeration (e.g. from domain knowledge). The result would likely generate more accurate estimates without sacrificing scalability.

Appendix

Notation Table	
n	number of treatment patterns
m	number of specialties
l	number of resources
s_i^j	state variable denoting number of patients currently under treatment pattern i and specialty j.
a_i	number of patients to be admitted for specialty i in the current time step.
$\mathbf{x}$	vector of vectors denoting the number of patient transitioning between treatment patterns for each specialty.
$\mathbf{x}_{ij}$	number of patients transitioning from treatment pattern i to treatment pattern j from the current time step to the next time step.
$\mathbf{R}_{\mathbf{x}s^i\|a^i,s^{i\prime}}$	the support of $\mathbf{x}$ with respect to the state transition $(s^{i\prime}\|s^i, a^i)$.

References

[1] R. Agrawal. Sample mean based index policies with O(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, pages 1054–1078, 1995.

[2] J.Y. Audibert, R. Munos, and C. Szepesvári. Tuning bandit algorithms in stochastic environments. In *Algorithmic Learning Theory*, pages 150–165. Springer, 2007.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 322–331. IEEE, 1995.

[5] R.K. Balla and A. Fern. UCT for tactical assault planning in real-time strategy games. In *21st International Joint Conference on Artificial Intelligence*, pages 40–45, 2009.

[6] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming, 1993.

[7] R. Bjarnason, A. Fern, and P. Tadepalli. Lower bounding Klondike solitaire with Monte-Carlo planning. In *Proc. ICAPS*, pages 26–33, 2009.

[8] B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *ICAPS*, volume 3, pages 12–21, 2003.

[9] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):1–43, 2012.

[10] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *J. Mach. Learn. Res.*, 999999:1655–1695, July 2011.

[11] Tugba Cayirli and Emre Veral. Outpatient scheduling in health care: A review of literature. *Production and Operations Management*, 12(4):519–549, 2003.

[12] T. Cazenave, F. Balbo, and S. Pinson. Monte-Carlo bus regulation. In *12th International IEEE Conference on Intelligent Transportation Systems*, pages 340–345, 2009.

[13] G. Chaslot, M. Winands, and H. van Den Herik. Parallel Monte-Carlo tree search. *Computers and Games*, pages 60–71, 2008.

[14] GMJ Chaslot, M.H.M. Winands, H. Herik, J. Uiterwijk, and B. Bouzy. Progressive strategies for Monte-Carlo tree search. *New Mathematics and Natural Computation*, 4(3):343, 2008.

[15] SG Elkhuizen, SF Das, PJM Bakker, and JAM Hontelez. Using computer simulation to reduce access time for outpatient departments. *Quality and Safety in Health Care*, 16(5):382–386, 2007.

[16] Sylvain Gelly and David Silver. Combining online and offline knowledge in UCT. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 273–280, New York, NY, USA, 2007. ACM.

[17] Sylvain Gelly and David Silver. Monte-Carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856 – 1875, 2011.

[18] Paul Gemmel and Roland Van Dierdonck. Admission scheduling in acute care hospitals: does the practice fit with the theory ? the case of Flemish hospitals. in 1997 Proceedings Decision Sciences Institute, volume 3, pp. 1461-1469, 1997.

[19] Yasin Gocgun, Brian W. Bresnahan, Archis Ghate, and Martin L. Gunn. A Markov Decision Process approach to multi-category patient scheduling in a diagnostic facility. *Artificial Intelligence in Medicine*, 53(2):73 – 81, 2011.

[20] D. Gupta and B. Denton. Appointment scheduling in health care: Challenges and opportunities. *IIE transactions*, 40(9):800–819, 2008.

[21] Erwin W. Hans, Mark Houdenhoven, and Peter J. H. Hulshof. A framework for health care planning and control. In Randolph Hall and Frederick S. Hillier, editors, *Handbook of health care System Scheduling*, volume 168 of *International Series in Operations Research & Management Science*, pages 303–320. Springer US, 2012.

[22] P.J.H. Hulshof, R.J. Boucherie, E.W. Hans, and J.L. Hurink. Tactical resource allocation and elective patient admission planning in care pathways. Technical report, Department of Applied Mathematics, University of Twente, 2011.

[23] J. B. Jun, S. H. Jacobson, J. R. Swisher, and Correspondence. Application of discrete-event simulation in health care clinics: A survey. *Journal of the Operational Research Society*, pages 109–123, February 1999.

[24] Asha Seth Kapadia, Wenyaw Chan, Ramesh Sachdeva, Lemuel A. Moye, and Larry S. Jefferson. Predicting duration of stay in a pediatric intensive care unit: A markovian approach. *European Journal of Operational Research*, 124(2):353 – 359, 2000.

[25] Asha Seth Kapadia, Shalom E. Vineberg, and C.Donald Rossi. Predicting course of treatment in a rehabilitation hospital: A Markovian model. *Computers & Operations Research*, 12(5):459 – 469, 1985.

[26] T. Keller and P. Eyerich. Prost: Probabilistic planning based on UCT. *ICAPS12*, 2012.

[27] Levente Kocsis and Csaba Szepesvri. Bandit based Monte-Carlo planning. In *In: ECML-06. Number 4212 in LNCS*, pages 282–293. Springer, 2006.

[28] V. Kuleshov and D. Precup. Algorithms for the multi-armed bandit problem. *Journal of Machine Learning*, 2010.

[29] T.L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[30] Marek Laskowski, Bryan C. P. Demianyk, Julia Witt, Shamir N. Mukhi, Marcia R. Friesen, and Robert D. McLeod. Agent-based modeling of the spread of influenza-like illness in an emergency department: A simulation study. *Trans. Info. Tech. Biomed.*, 15(6):877–889, November 2011.

[31] Mausam and Andrey Kolobov. *Planning with Markov Decision Processes: An AI Perspective*. Morgan & Claypool Publishers, 2012.

[32] H.B. McMahan, M. Likhachev, and G.J. Gordon. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In

Proceedings of the 22nd international conference on Machine learning, pages 569–576. ACM, 2005.

[33] T.H. Nguyen, W.S. Lee, and T.Y. Leong. Bootstrapping Monte Carlo tree search with an imperfect heuristic. *Machine Learning and Knowledge Discovery in Databases*, pages 164–179, 2012.

[34] Luiz Guilherme Nadal Nunes, Solon Veníncio de Carvalho, and Rita de Cássia Meneses Rodrigues. Markov Decision Process applied to the control of hospital elective admissions. *Artif. Intell. Med.*, 47(2):159–171, October 2009.

[35] Sandeep Pandey, Deepayan Chakrabarti, and Deepak Agarwal. Multi-armed bandit problems with dependent arms. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 721–728, New York, NY, USA, 2007. ACM.

[36] J. Patrick, M.L. Puterman, and M. Queyranne. Dynamic multipriority patient scheduling for a diagnostic resource. *Operations Research*, 56(6):1507–1525, 2008.

[37] D. Pellier, B. Bouzy, and M. Métivier. An UCT approach for anytime agent-based planning. *Advances in Practical Applications of Agents and Multiagent Systems*, pages 211–220, 2010.

[38] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[39] S. Sanner, R. Goetschalckx, K. Driessens, and G. Shani. Bayesian real-time dynamic programming. In *Proc. of IJCAI*, volume 9, 2009.

[40] T. Smith and R. Simmons. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *Proceedings of the National Conference on Arti-*

ficial Intelligence, volume 21, page 1227. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[41] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction (Adaptive computation and machine learning)*. The MIT Press, 1998.

[42] Gerald Tesauro, V Rajan, and Richard Segal. Bayesian inference in Monte-Carlo tree search. In *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 580–588, Corvallis, Oregon, 2010. AUAI Press.

[43] David Tom and Martin Mller. A study of UCT and its enhancements in an artificial game. In H. van den Herik and Pieter Spronck, editors, *Advances in Computer Games*, volume 6048 of *Lecture Notes in Computer Science*, pages 55–64. Springer Berlin Heidelberg, 2010.

[44] Junchao Xiao, Leon J. Osterweil, and Qing Wang. Dynamic scheduling of emergency department resources. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 590–599, New York, NY, USA, 2010. ACM.