Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Optimistic Monte Carlo Tree Search with Sampled Information Relaxation Dual Bounds

Daniel R. Jiang, Lina Al-Kanj, Warren B. Powell

http://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Articles in Advance, pp. 1–20
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Methods

Optimistic Monte Carlo Tree Search with Sampled Information Relaxation Dual Bounds

Daniel R. Jiang,[a] **Lina Al-Kanj,**[b] **Warren B. Powell**[b]

[a] Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania 15213; [b] Department of Operations Research and Financial Engineering, Princeton University, Princeton, New Jersey 08544

Contact: drjiang@pitt.edu, https://orcid.org/0000-0002-5388-8061 (DRJ); lalkanj@princeton.edu, https://orcid.org/0000-0003-4429-4525 (LAK); powell@princeton.edu, https://orcid.org/0000-0002-4364-7555 (WBP)

Received: April 17, 2017
Revised: August 15, 2018; April 22, 2019; July 20, 2019
Accepted: August 9, 2019
Published Online in Articles in Advance: September 11, 2020

Subject Classifications: decision analysis: sequential, dynamic programming
Area of Review: Stochastic Models

https://doi.org/10.1287/opre.2019.1939

Abstract. Monte Carlo tree search (MCTS), most famously used in game-play artificial intelligence (e.g., the game of Go), is a well-known strategy for constructing approximate solutions to sequential decision problems. Its primary innovation is the use of a heuristic, known as a *default policy*, to obtain Monte Carlo estimates of downstream values for states in a decision tree. This information is used to iteratively expand the tree toward regions of states and actions that an optimal policy might visit. However, to guarantee convergence to the optimal action, MCTS requires the entire tree to be expanded asymptotically. In this paper, we propose a new "optimistic" tree search technique called *primal-dual MCTS* that uses sampled *information relaxation* upper bounds on potential actions to make tree expansion decisions, creating the possibility of ignoring parts of the tree that stem from highly suboptimal choices. The core contribution of this paper is to prove that despite converging to a partial decision tree in the limit, the recommended action from primal-dual MCTS is optimal. The new approach shows promise when used to optimize the behavior of a single driver navigating a graph while operating on a ride-sharing platform. Numerical experiments on a real data set of taxi trips in New Jersey suggest that primal-dual MCTS improves on standard MCTS (upper confidence trees) and other policies while exhibiting a reduced sensitivity to the size of the action space.

Keywords: Monte Carlo tree search • dynamic programming • information relaxation

1. Introduction

Monte Carlo tree search (MCTS) is a technique popularized by the artificial intelligence (AI) community (Coulom 2007) for solving sequential decision problems with finite state and action spaces. To avoid searching through an intractably large decision tree, MCTS instead iteratively builds the tree and attempts to focus on regions composed of states and actions that an optimal policy might visit. A heuristic known as the *default policy* is used to provide estimates of downstream values, which serve as a guide for MCTS to explore promising regions of the search space. When the allotted computational resources have been expended, the hope is that the best first-stage decision recommended by the *partial decision tree* is a reasonably good estimate of the optimal decision that would have been implied by the full tree. The adaptive sampling algorithm by Chang et al. (2005), introduced within the operations research (OR) community, leverages a well-known bandit algorithm called *upper confidence bound* (UCB) for solving Markov decision processes (MDPs). The UCB approach is also used extensively for successful implementations of MCTS (Kocsis and Szepesvári 2006).

The applications of MCTS are broad and varied, but the strategy is traditionally most often applied to game-play AI (Chaslot et al. 2008). To name a few specific applications, these include Go (Chaslot et al. 2006, Gelly and Silver 2011, Gelly et al. 2012, Silver et al. 2016), Othello (Hingston and Masek 2007, Nijssen 2007, Osaki et al. 2008, Robles et al. 2011), Backgammon (Van Lishout et al. 2007), Poker (Maitrepierre et al. 2008, Van den Broeck et al. 2009, Ponsen et al. 2010), 16 × 16 Sudoku (Cazenave 2009), and even general game-playing AI (Méhat and Cazenave 2010). We remark that a characteristic of games is that the transitions from state to state are deterministic; because of this, the standard specification for MCTS deals with deterministic problems. The "Monte Carlo" descriptor in the name of MCTS therefore refers to stochasticity in the default policy. A particularly thorough review of both the MCTS methodology and its applications can be found in Browne et al. (2012).

The OR community has generally not taken advantage of the MCTS methodology in applications, with the exception of two recent papers. Bertsimas et al. (2017) compare MCTS with rolling horizon mathematical optimization techniques (a standard method in OR) on two problems: the large-scale dynamic resource allocation problem of tactical wildfire management and queuing network control. Al-Kanj et al. (2016) apply MCTS to an information-collecting vehicle routing problem, which is an extension of the classic vehicle routing model where the decisions depend on a *belief state*. Not surprisingly, both of these problems are intractable via standard MDP techniques, and results from these papers suggest that MCTS could be a viable alternative to other approximation methods (e.g., approximate dynamic programming). However, Bertsimas et al. (2017) find that MCTS is competitive with rolling horizon techniques only on smaller instances of the problems, and their evidence suggests that MCTS can be quite sensitive to large action spaces. In addition, they observe that large action spaces are more detrimental to MCTS than large state spaces. These observations form the basis of our *first research motivation*: can we control the action branching factor by making "intelligent guesses" at which actions may be suboptimal? If so, potentially suboptimal actions can be ignored.

Next, let us briefly review the currently available work on convergence theory. The paper by Chang et al. (2005) gives the first provably convergent algorithm using the UCB idea in an MDP setting, with the stages being solved sequentially in a "backward" manner. The work of Kocsis and Szepesvári (2006) uses the UCB algorithm to sample actions in MCTS, resulting in an algorithm called *upper confidence trees* (UCT). A key property of UCT is that every action is sampled infinitely often; Kocsis and Szepesvári (2006) exploit this to show that the probability of selecting a suboptimal action converges to zero at the root of the tree. Silver and Veness (2010) use the UCT result as a basis for showing convergence of a variant of MCTS for partially observed MDPs. Couëtoux et al. (2011) extend MCTS for deterministic finite-state problems to stochastic problems with continuous state spaces using a technique called *double progressive widening*. The paper by Auger et al. (2013) provides convergence results for MCTS with double progressive widening under an action sampling assumption. In these papers, the asymptotic convergence of MCTS relies on some form of *exploring every node infinitely often*. However, given that the spirit of the algorithm is to build partial trees that are biased toward nearly optimal actions, we believe that an alternative line of thinking deserves further study. Thus, our *second research motivation* builds off the first motivation by asking a theoretical question: can we design a version of MCTS that asymptotically *does not* build the full decision tree yet is still optimal?

Our use of optimistic estimates means that we no longer have to sample every action. From a practical perspective, this means that we are less vulnerable to large action spaces. But we should also emphasize that this is an important theoretical result that is fundamentally different from the UCB-based proofs. Ultimately, this is the core contribution of this paper.

A good default policy for MCTS can take significant effort to construct. For example, consider the case of AlphaGo, the first computer to defeat a human player in the game of Go, of which MCTS plays a major role. As Silver et al. (2016, p. 484) state, "The strongest current Go programs are based on MCTS, enhanced by policies that are trained to predict human expert moves." To be more precise, the default policy used by AlphaGo is carefully constructed through several steps: (1) a classifier to predict expert moves is trained using 29.4 million game positions from 160,000 games on top of a deep convolutional neural network (consisting of 13 layers); (2) the classifier is then played against itself, and a policy gradient technique is used to develop a policy that aims to win the game rather than simply mimic human players; (3) another deep convolutional neural network is used to approximate the *value function* of the heuristic policy; and (4) a combination of the two neural networks, dubbed the *policy* and *value* networks (also known as *actor* and *critic* networks), provides an MCTS algorithm with the default policy and the estimated downstream values. This illustrates our *third research motivation*: typically, researchers spend significant effort designing the default policy and use it solely to provide rough estimates of the value function, whereas one could also extract additional value from the default policy by using it to generate a dual upper bound.

In this paper, we address each of these questions by proposing a novel optimistic MCTS method, called *primal-dual MCTS*—the name is inspired by Andersen and Broadie (2004)—that takes advantage of the *information relaxation bound* idea (also known as *martingale duality*) first developed by Rogers (2002) and later generalized by Brown et al. (2010). The essence of information relaxation is to relax nonanticipativity constraints (i.e., allow the decision maker to use future information) in order to produce upper bounds (hence the "optimistic" descriptor) on the objective value (assuming a maximization problem). To account for the issue that a naive use of future information can produce weak bounds, Brown et al. (2010) describe a method to penalize the use of future information so that one may obtain a tighter (smaller) upper bound. This is called a *dual approach*, and it is shown that the value of the upper bound can be made equal to the optimal value if a particular penalty function is chosen that depends on the optimal value function of the original problem. Information relaxation has been used successfully to estimate the suboptimality of policies in a number of application domains, including option pricing (Broadie and Glasserman 1997, Andersen and Broadie 2004), portfolio optimization (Brown and Smith 2011), valuation of natural gas (Lai et al. 2011, Nadarajah et al. 2015), optimal stopping (Desai et al. 2012), and vehicle routing (Goodson et al. 2016). More specifically, the contributions of this paper are as follows:

- We propose a new MCTS method called *primal-dual MCTS* that uses the information relaxation methodology of Brown et al. (2010) to generate dual upper bounds. These bounds are used when MCTS needs to choose actions to explore (this is known as *expansion* in the literature). When the algorithm considers performing an expansion step, we obtain *sampled* upper bounds (i.e., in expectation, they are greater than the optimal value) for a set of potential actions and select an action with an upper bound that is *better* than the value of the current optimal action. Correspondingly, if all remaining unexplored actions have upper bounds lower than the value of the current optimal action, then we do not expand further. The proposed procedure is related to branch-and-bound (Land and Doig 1960) and alpha-beta pruning (Knuth and Moore 1975), except using noisy bounds for MDPs. This addresses our first research motivation of reducing the branching factor in a principled way.
- The core contribution of this paper is a proof that our method converges to the optimal action (and optimal value) at the root node. This holds, even though our proposed technique does not preclude the possibility of a partially expanded tree in the limit. By carefully using the upper bounds, we are able to "provably ignore" entire subtrees, thereby reducing the amount of computation needed. This addresses our second research motivation, which extends the current convergence theory of MCTS.
- Although there are many ways to construct the dual bound, one special instance of primal-dual MCTS uses the default policy (the heuristic for estimating downstream values) to induce a penalty function. This addresses our third research motivation: the default policy can provide actionable information in the form of upper bounds, in addition to its original purpose of estimating downstream values.
- Lastly, we present a model of the stochastic optimization problem faced by a single driver who provides transportation for fare-paying customers while navigating a graph. The problem is motivated by the need for ride-sharing platforms (e.g., Uber and Lyft) to be able to accurately simulate the operations of an entire ride-sharing system/fleet. Understanding human drivers' behaviors is crucial to a smooth integration of platform-controlled driver-less vehicles with the traditional contractor model (e.g., in Pittsburgh, Pennsylvania). Our computational results show that primal-dual MCTS dramatically reduces the breadth of the search tree when compared with standard MCTS.

This paper is organized as follows. In Section 2, we describe a general model of a stochastic sequential decision problem and review the standard MCTS framework along with the duality and information relaxation procedures of Brown et al. (2010). We present the primal-dual MCTS algorithm in Section 3 and provide the convergence analysis in Section 4. The ride-sharing model and the associated numerical results are discussed in Section 5, and we provide concluding remarks in Section 6.

2. Preliminaries

In this section, we first formulate the mathematical model of the underlying optimization problem as an MDP. Because we are in the setting of decision trees and information relaxations, we need to extend traditional MDP notation with some additional elements. We also introduce the existing concepts, methodologies, and relevant results that are used throughout this paper.

2.1. Mathematical Model

As is common in MCTS, we consider an underlying MDP formulation with a finite horizon $t = 0, 1, \ldots, T$, where the set of decision epochs is $\mathcal{T} = \{0, 1, \ldots, T-1\}$. Let $\mathcal{S}$ be a *state space* and $\mathcal{A}$ be an *action space*, and we assume a finite state and action setting: $|\mathcal{S}| < \infty$ and $|\mathcal{A}| < \infty$. The set of feasible actions for state $s \in \mathcal{S}$ is $\mathcal{A}_s$, a subset of $\mathcal{A}$. The set $\mathcal{U} = \{(s,a) \in \mathcal{S} \times \mathcal{A} : a \in \mathcal{A}_s\}$ contains all feasible state–action pairs.

The dynamics from one state to the next depend on the action taken at time t, written $a_t \in \mathcal{A}$, and an exogenous (i.e., independent of states and actions) random process $\{W_t\}_{t=1}^{T}$ on $(\Omega, \mathcal{F}, \mathbf{P})$ taking values in a finite space $\mathcal{W}$. For simplicity, we assume that W_t are independent across time t. The *transition function* for period t is given by $f_t : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{S}$. We denote the deterministic initial state by $s_0 \in \mathcal{S}$ and let $\{S_t\}_{t=0}^{T}$ be the random process describing the evolution of the system state, where $S_0 = s_0$ and $S_{t+1} = f_t(S_t, a_t, W_{t+1})$. To distinguish from the random variable $S_t \in \mathcal{S}$, we shall refer to a particular element of the state space by lowercase variables, for example, $s \in \mathcal{S}$. The contribution (or reward) function at stage $t < T$ is given by $c_t : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathbb{R}$. For a fixed state–action pair $(s,a) \in \mathcal{U}$, the contribution is the random quantity $c_t(s, a, W_{t+1})$, which we assume is bounded.

Because there are a number of other "policies" that the MCTS algorithm takes as input parameters (to be discussed in Section 2.2), we call the main MDP policy of interest the *operating policy*. A *decision* *function* $\pi_t : \mathcal{S} \to \mathcal{A}$ is a deterministic map from the state space to the action space such that $\pi_t(s) \in \mathcal{A}_s$ for any state $s \in \mathcal{S}$. An *admissible policy* $\pi = \{\pi_0, \pi_2, \ldots, \pi_{T-1}\}$ is a set of such decision functions, one for each time period in $\mathcal{T}$ (the qualifier "admissible" refers to the fact that these policies are nonanticipative and only use information contained in the state s). We also let Π be the set of all admissible policies for the MDP. Finally, we define the optimal value function at time t and state s, which is the maximum expected cumulative contribution achieved over all policies:

$$V_t^*(s) = \max_{\pi \in \Pi} \mathbf{E}\left[\sum_{\tau=t}^{T-1} c_\tau(S_\tau, \pi_\tau(S_\tau), W_{\tau+1}) \,|\, S_t = s\right]. \quad (1)$$

The objective of the MDP is to find an operating policy that achieves $V_0^*(s_0)$ for some initial state s_0. The optimal value function satisfies the standard Bellman optimality recursion

$$\begin{aligned} V_t^*(s) &= \max_{a \in \mathcal{A}_s} \mathbf{E}\left[c_t(s, a, W_{t+1}) + V_{t+1}^*(S_{t+1})\right], \\ &\qquad\qquad \text{for all } s \in \mathcal{S},\ t \in \mathcal{T}, \\ V_T^*(s) &= 0, \qquad\qquad \text{for all } s \in \mathcal{S}. \end{aligned}$$

The state–action formulation of the Bellman recursion is also necessary for the purposes of MCTS because the decision tree contains both state and state–action nodes. The state–action value function is defined as

$$\begin{aligned} Q_t^*(s, a) &= \mathbf{E}\left[c_t(s, a, W_{t+1}) + V_{t+1}^*(S_{t+1})\right], \\ &\qquad \text{for all } (s, a) \in \mathcal{U},\ t \in \mathcal{T}. \end{aligned}$$

For consistency, it is also useful to let $Q_T^*(s, a) = 0$ for all (s, a). It thus follows that $V_t^*(s) = \max_{a \in \mathcal{A}_s} Q_t^*(s, a)$. Likewise, the optimal policy $\pi^* = \{\pi_0^*, \ldots, \pi_{T-1}^*\}$ from the set Π is characterized by $\pi_t^*(s) = \arg\max_{a \in \mathcal{A}_s} Q_t^*(s, a)$.

It is also useful for us to define the value of a particular operating policy π starting from a state $s \in \mathcal{S}$ at time t, given by the value function $V_t^\pi(s)$. If we let $S_{t+1}^\pi = f_t(s, \pi_t(s), W_{t+1})$, then the following recursion holds:

$$\begin{aligned} V_t^\pi(s) &= \mathbf{E}\left[c_t(s, \pi_t(s), W_{t+1}) + V_{t+1}^\pi(S_{t+1}^\pi)\right], \\ &\qquad\qquad \text{for all } s \in \mathcal{S},\ t \in \mathcal{T}, \\ V_T^\pi(s) &= 0, \qquad\qquad \text{for all } s \in \mathcal{S}. \end{aligned} \quad (2)$$

Note that $V_t^*(s) = V_t^{\pi^*}(s)$ for all $s \in \mathcal{S}$. Similarly, we have

$$\begin{aligned} Q_t^\pi(s, a) &= \mathbf{E}\left[c_t(s, a, W_{t+1}) + V_{t+1}^\pi(S_{t+1}^\pi)\right], \\ &\qquad \text{for all } (s, a) \in \mathcal{S} \times \mathcal{A},\ t \in \mathcal{T}, \\ Q_T^\pi(s, a) &= 0, \qquad \text{for all } (s, a) \in \mathcal{S} \times \mathcal{A}, \end{aligned} \quad (3)$$

the state–action value functions for a given operating policy π.

Suppose that we are at a fixed time t. Because of the notational needs of information relaxation, let $s_{\tau,t}(s, \mathbf{a}, \mathbf{w}) \in \mathcal{S}$ be the deterministic state reached at time $\tau \geq t$ given that we are in state s at time t, implement a fixed sequence of actions $\mathbf{a} = (a_t, a_{t+1}, \ldots, a_{T-1})$, and observe a fixed sequence of exogenous outcomes $\mathbf{w} = (w_{t+1}, w_{t+2}, \ldots, w_T)$. For succinctness, the time subscripts have been dropped from the vector representations. Similarly, let $s_{\tau,t}(s, \pi, \mathbf{w}) \in \mathcal{S}$ be the deterministic state reached at time $\tau \geq t$ if we follow a fixed policy $\pi \in \Pi$.

Finally, we need to refer to the future contributions starting from time t, state s, and a sequence of exogenous outcomes $\mathbf{w} = (w_{t+1}, w_{t+2}, \ldots, w_T)$. For convenience, we slightly abuse notation and use two versions of this quantity, one using a fixed sequence of actions $\mathbf{a} = (a_t, a_{t+1}, \ldots, a_{T-1})$ and the other using a fixed policy π:

$$\begin{aligned} h_t(s, \mathbf{a}, \mathbf{w}) &= \sum_{\tau=t}^{T-1} c_\tau(s_{\tau,t}(s, \mathbf{a}, \mathbf{w}), a_\tau, w_{\tau+1}), \quad h_t(s, \pi, \mathbf{w}) \\ &= \sum_{\tau=t}^{T-1} c_\tau(s_{\tau,t}(s, \pi, \mathbf{w}), a_\tau, w_{\tau+1}). \end{aligned}$$

Therefore, if we define the random process $\mathbf{W}_{t+1,T} = (W_{t+1}, W_{t+2}, \ldots, W_T)$, then the quantities $h_t(s, \mathbf{a}, \mathbf{W}_{t+1,T})$ and $h_t(s, \pi, \mathbf{W}_{t+1,T})$ represent the *random downstream cumulative reward* starting at state s and time t following a deterministic sequence of actions $\mathbf{a}$ or a policy π. For example, the objective function to the MDP given in (1) can be rewritten more concisely as $\max_{\pi \in \Pi} \mathbf{E}\left[h_t(s_0, \pi, \mathbf{W}_{1,T})\right]$.

2.2. Monte Carlo Tree Search

The canonical MCTS algorithm iteratively grows and updates a decision tree using the default policy as a guide toward promising subtrees. Because sequential systems evolve from a (predecision) state S_t, to an action a_t, to a postdecision state or a state–action pair (S_t, a_t), to new information W_{t+1}, and finally, to another state S_{t+1}, there are two types of nodes in a decision tree: *state nodes* (or *predecision states*) and *state–action nodes* (or *postdecision states*). The layers of the tree are chronological and alternate between these two types of nodes. A child of a state node is a state–action node connected by an edge that represents a particular action. Similarly, a child of a state–action node is a state node where the edge represents an outcome of the exogenous information process W_{t+1}.

Because we are working within the decision tree setting, it is necessary to introduce some additional notation that departs from the traditional MDP style. A state node is represented by an augmented state that contains the entire path down the tree from the root node s_0:

$$\mathbf{x}_t = (s_0, a_0, s_1, a_1, s_2 \ldots, a_{t-1}, s_t) \in \mathcal{X}_t,$$

where $a_0 \in \mathcal{A}_{s_0}, a_1 \in \mathcal{A}_{s_1}, \ldots, a_{t-1} \in \mathcal{A}_{s_{t-1}}$, $s_1, s_2, \ldots, s_t \in \mathcal{S}$, and $\mathcal{X}_t$ is the set of all possible $\mathbf{x}_t$ (representing all possible paths to states at time t). A state–action node is represented via the notation $\mathbf{y}_t = (\mathbf{x}_t, a_t)$, where $a_t \in \mathcal{A}_{s_t}$. Similarly, let $\mathcal{Y}_t$ be the set of all possible $\mathbf{y}_t$. We can take advantage of the Markovian property along with the fact that any node $\mathbf{x}_t$ or $\mathbf{y}_t$ contains information about t to write (again, a slight abuse of notation)

$$V^*(\mathbf{x}_t) = V_t^*(s_t) \quad \text{and} \quad Q^*(\mathbf{y}_t) = Q_t^*(\mathbf{x}_t, a_t) = Q_t^*(s_t, a_t).$$

At iteration n of the MCTS, each state node $\mathbf{x}_t$ is associated with a value function approximation $\bar{V}^n(\mathbf{x}_t)$, and each state–action node $(\mathbf{x}_t, a_t)$ is associated with the state–action value function approximation $\bar{Q}^n(\mathbf{x}_t, a_t)$. Moreover, we use the following shorthand notation:

$$\begin{aligned}\mathbf{P}(S_{t+1} = s_{t+1} \,|\, \mathbf{y}_t) &= \mathbf{P}(S_{t+1} = s_{t+1} \,|\, \mathbf{x}_t, a_t) \\ &= \mathbf{P}(f_t(s_t, a_t, W_{t+1}) = s_{t+1}).\end{aligned}$$

There are four main phases in the MCTS algorithm: selection, expansion, simulation, and backpropagation (Browne et al. 2012). Often the first two phases are called the *tree policy* because they traverse and expand the tree; it is in these two phases that we will introduce our new methodology. Let us now summarize the steps of MCTS while employing the so-called double progressive widening (DPW) technique (Couëtoux et al. 2011) to control the branching at each level of the tree. As its name suggests, DPW means that we slowly expand the branching factor of the tree, in both state nodes and state–action nodes. The following steps summarize the steps of MCTS at a particular iteration n. See Browne et al. (2012, figure 2) for an illustration of these steps.

Selection. We are given a *selection policy*, which determines a path down the tree at each iteration. When the algorithm encounters an iteration where progressive widening is not used, it traverses the tree until it reaches a leaf node (i.e., a state node that is not fully expanded) and proceeds to the simulation step. On a progressive widening iteration, the traversal is performed until an *expandable node* (i.e., one for which there exists a child that has not yet been added to the tree) is reached. This could be either a state node or a state–action node; the algorithm now proceeds to the expansion step in order to add a node to the tree.

Expansion. We now use a given *expansion policy* to decide which child to add to the tree. The simplest method, of course, is to add an action at random or add an exogenous state transition at random. Assuming that expansion of a state–action node always follows the expansion of a state node, we are now in a leaf state node.

Simulation. The aforementioned *default policy* is now used to generate a sample of the value function evaluated at the current state node. The estimate is constructed by running the default policy on a simulated future trajectory of the exogenous information process. This step of MCTS is also called a *rollout.*

Backpropagation. The last step is to recursively update the values up the tree until the root node is reached: for state–action nodes, a weighted average is performed on the values of its child nodes to update $\bar{Q}^n(\mathbf{x}_t, a_t)$, and, for state nodes, a combination of a weighted average and maximum of the values of its child nodes is taken to update $\bar{V}^n(\mathbf{x}_t)$. These operations correspond to a backup operator discussed in Coulom (2007) that achieves good empirical performance. This concludes one iteration of MCTS, and the next iteration begins at the selection step.

Once a prespecified number of iterations have been run, the best action out of the root node is chosen for implementation. After landing in a new state in the real system, MCTS can be run again with the new state as the root node. A practical strategy is to use the relevant subtree from the previous run of MCTS to initialize the new process (Bertsimas et al. 2017).

2.3. Information Relaxation Bounds

We next review the information relaxation duality ideas from Brown et al. (2010); see also Brown and Smith (2011) and Brown and Smith (2014). Here we adapt the results of Brown et al. (2010) to our setting, where we require the bounds to hold for arbitrary subproblems of the MDP. Specifically, we state the theorems from the point of view of a specific time t and initial state–action pair (s, a). Also, we focus on the *perfect information relaxation*, where one assumes full knowledge of the future in order to create upper bounds. In this case, we have

$$V_t^*(s) \le \mathbf{E}\left[\max_{\mathbf{a}} h_t(s, \mathbf{a}, \mathbf{W}_{t+1,T})\right],$$

which means that the value achieved by the optimal policy starting from time t is upper bounded by the value of the policy that selects actions using perfect information. As we described previously, the main idea of this approach is to relax nonanticipativity constraints to provide upper bounds. Because these bounds may be quite weak, they are subsequently strengthened by imposing penalties for usage of future information. To be more precise, we subtract away a penalty defined by a function z_t so that the right-hand side is decreased to $\mathbf{E}[\max_{\mathbf{a}}[h_t(s, \mathbf{a}, \mathbf{W}_{t+1,T}) - z_t(s, \mathbf{a}, \mathbf{W}_{t+1,T})]]$.

Consider the subproblem (or subtree) starting in stage t and state s. A *dual penalty* z_t is a function that maps an initial state, a sequence of actions

$\mathbf{a} = (a_t, a_{t+1}, \ldots, a_{T-1})$, and a sequence of exogenous outcomes $\mathbf{w} = (w_{t+1}, \ldots, w_T)$ to a penalty $z_t(s, \mathbf{a}, \mathbf{w}) \in \mathbb{R}$. As we did in the definition of h_t, the same quantity is written $z_t(s, \pi, \mathbf{w})$ when the sequence of actions is generated by a policy π. The set of *dual feasible penalties* for a given initial state s are those z_t that do not penalize admissible policies; it is given by the set

$$\mathcal{Z}_t(s) = \Big\{z_t : \mathbf{E}\big[z_t(s, \pi, \mathbf{W}_{t+1,T})\big] \le 0 \ \ \forall \pi \in \Pi\Big\}, \tag{4}$$

where $\mathbf{W}_{t+1,T} = (W_{t+1}, \ldots, W_T)$. Therefore, the only "primal" policies (i.e., policies for the original MDP) for which a dual feasible penalty z could assign a positive penalty in expectation are those that are not in Π.

We now state a theorem from Brown et al. (2010) that illuminates the dual bound method. The intuition is best described from a simulation point of view: we sample an entire future trajectory of the exogenous information $\mathbf{W}_{t+1,T}$, and using full knowledge of this information, the optimal actions are computed. It is clear that after taking the average of many such trajectories, the corresponding averaged objective value will be an upper bound on the value of the optimal (nonanticipative) policy. The dual penalty is simply a way to improve this upper bound by *penalizing the use of future information*; the only property required in the proof of Theorem 1 is the definition of dual feasibility. The proof is simple, and we repeat it here so that we can state a small extension later in the paper (in Proposition 1). The right-hand side of the inequality below is a penalized perfect information relaxation.

Theorem 1 (Weak Duality, Brown et al. 2010, p. 787)**.** *Fix a stage $t \in \mathcal{T}$ and initial state $s \in \mathcal{S}$. Let $\pi \in \Pi$ be a feasible policy, and let $z_t \in \mathcal{Z}_t(s)$ be a dual feasible penalty, as defined in* (4)*. It holds that*

$$V_t^\pi(s) \le \mathbf{E}\left[\max_{\mathbf{a}}\big[h_t(s, \mathbf{a}, \mathbf{W}_{t+1,T}) - z_t(s, \mathbf{a}, \mathbf{W}_{t+1,T})\big]\right], \tag{5}$$

where $\mathbf{a} = (a_t, \ldots, a_{T-1})$.

Proof. By definition, $V_t^\pi(s) = \mathbf{E}\,[h_t(s, \pi, \mathbf{W}_{t+1,T})]$. Thus, it follows by dual feasibility that

$$\begin{aligned} V_t^\pi(s_t) &\le \mathbf{E}\big[h_t(s, \pi, \mathbf{W}_{t+1,T}) - z_t(s, \pi, \mathbf{W}_{t+1,T})\big] \\ &\le \mathbf{E}\left[\max_{\mathbf{a}}\big[h_t(s, \mathbf{a}, \mathbf{W}_{t+1,T}) - z_t(s, \mathbf{a}, \mathbf{W}_{t+1,T})\big]\right]. \end{aligned}$$

The second inequality follows by the property that a policy using future information must achieve a higher value than an admissible policy. In other words, Π is contained within the set of policies that are not constrained by nonanticipativity.

Note that the left-hand side of (5) is known as the *primal problem* and the right-hand side is the *dual problem*, so it is easy to see that the theorem is analogous to classical duality results from linear programming. The next step, of course, is to identify some dual feasible penalties. For each t, let $\nu_t : \mathcal{S} \to \mathbb{R}$ be any function and define

$$\begin{aligned} \bar{\nu}_\tau(s, \mathbf{a}, \mathbf{w}) = {} & \nu_{\tau+1}(s_{\tau+1,t}(s, \mathbf{a}, \mathbf{w})) \\ & - \mathbf{E}\ \nu_{\tau+1}(f_t(s_{\tau,t}(s, \mathbf{a}, \mathbf{w}), a_\tau, W_{\tau+1})). \end{aligned} \tag{6}$$

Brown et al. (2010) suggest the following additive form for a dual penalty:

$$z_t^\nu(s, \mathbf{a}, \mathbf{w}) = \sum_{\tau=t}^{T-1} \bar{\nu}_\tau(s, \mathbf{a}, \mathbf{w}), \tag{7}$$

and they show that this form is indeed dual feasible. We refer to this as the *dual penalty generated by* $\nu = \{\nu_t\}$.

However, in situations where the standard dual upper bound is too weak, a good choice of ν can generate tighter bounds. It is shown that if the optimal value function V_τ^* is used in place of ν_τ in (6), then the best upper bound is obtained. In particular, a form of *strong duality* holds: when Theorem 1 is invoked using the optimal policy $\pi^* \in \Pi$ and $\nu_\tau = V_\tau^*$, the inequality (5) is achieved with equality. The interpretation of the case where $\nu_\tau = V_\tau^*$ for all τ is that $\bar{\nu}_\tau$ can be thought of informally as the "value gained from knowing the future." Thus, the intuition behind this result is as follows: if one knows precisely how much can be gained by using future information, then a perfect penalty can be constructed so as to recover the optimal value of the primal problem.

However, strong duality is hard to exploit in practical settings, given that both sides of the equation require knowledge of the optimal policy. Instead, a viable strategy is to use approximate value functions $\bar{V}_\tau$ on the right-hand side of (6) in order to obtain "good" upper bounds on the optimal value function V_t^* on the left-hand side of (5). This is where we can potentially take advantage of the default policy of MCTS to improve upon the standard dual upper bound; the value function associated with this policy can be used to generate a dual feasible penalty. We now state a specialization of Theorem 1 that is useful for our MCTS setting. □

Proposition 1 (State–Action Duality)**.** *Fix a stage $t \in \mathcal{T}$ and an initial state–action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$. Assume that the dual penalty function takes the form given in* (6) *and* (7) *for some $\nu = \{\nu_t\}$. Then it holds that*

$$\begin{aligned} Q_t^*(s, a) \le \mathbf{E}\Big[\, & c_t(s, a, W_{t+1}) + \max_{\mathbf{a}}\big[h_{t+1}(S_{t+1}, \mathbf{a}, \mathbf{W}_{t+2,T}) \\ & - z_{t+1}^\nu(S_{t+1}, \mathbf{a}, \mathbf{W}_{t+2,T})\big]\Big], \end{aligned} \tag{8}$$

where $S_{t+1} = f_t(s, a, W_{t+1})$ and the optimization is over the vector $\mathbf{a} = (a_{t+1}, \ldots, a_{T-1})$.

Proof. Choose a policy $\tilde{\pi}$ (restricted to stage t onward) such that the first decision function maps to a and the remaining decision functions match those of the optimal policy π^*:

$$\tilde{\pi} = \left(a, \pi^*_{t+1}, \pi^*_{t+2}, \ldots, \pi^*_{T-1}\right).$$

Using this policy and the separability of z_t^ν given in (7), an argument analogous to the proof of Theorem 1 can be used to obtain the result.

For convenience, let us denote the dual upper bound generated using the functions ν by

$$u_t^\nu(s,a) = \mathbf{E}\Bigg[c_t(s,a,W_{t+1}) + \max_{\mathbf{a}}\big[h_{t+1}(S_{t+1},\mathbf{a},\mathbf{W}_{t+1,T}) - z_{t+1}^\nu(S_{t+1},\mathbf{a},\mathbf{W}_{t+1,T})\big]\Bigg].$$

Therefore, the dual bound can be simply stated as $Q_t^*(s,a) \le u_t^\nu(s,a)$. For a state–action node $\mathbf{y}_t = (s_0, a_0, \ldots, s_t, a_t)$ in the decision tree, we use the notation $u^\nu(\mathbf{y}_t) = u_t^\nu(s_t, a_t)$. The proposed algorithm will keep estimates of the upper bound on the right-hand side of (8) in order to make tree expansion decisions. As the algorithm progresses, the estimates of the upper bound are refined using a stochastic gradient method. □

3. Primal-Dual MCTS Algorithm

In this section, we formally describe the proposed Primal-Dual MCTS algorithm. We assume that c_t and f_t are known for each t and the ability to generate samples of $\{W_t\}$ from its true distribution. This could be accomplished in practice either through the availability of a *black-box simulator* or a large data set.[1] The core of the algorithm is MCTS with double progressive widening (Couëtoux et al. 2011), except in our case the dual bounds generated by the functions ν_t play a specific role in the expansion step. Let $\mathcal{X} = \cup_t \mathcal{X}_t$ be the set of all possible state nodes, and let $\mathcal{Y} = \cup_t \mathcal{Y}_t$ be the set of all possible state–action nodes. After each iteration $n \ge 0$, our tree $\mathcal{T}^n = (n, \mathcal{X}^n, \mathcal{Y}^n, \bar{V}^n, \bar{Q}^n, \bar{u}^n, v^n, l^n)$ is described by the set $\mathcal{X}^n \subseteq \mathcal{X}$ of expanded state nodes, the set $\mathcal{Y}^n \subseteq \mathcal{Y}$ of expanded state–action nodes, the value function approximations $\bar{V}^n : \mathcal{X} \to \mathbb{R}$ and $\bar{Q}^n : \mathcal{Y} \to \mathbb{R}$, the estimated upper bounds $\bar{u}^n : \mathcal{Y} \to \mathbb{R}$, the number of visits $v^n : \mathcal{X} \cup \mathcal{Y} \to \mathbb{N} \cup \{0\}$ to expanded nodes, and the number of information relaxation upper bounds, or "lookaheads," $l^n : \mathcal{Y} \to \mathbb{N} \cup \{0\}$ performed on unexpanded nodes. The term "lookahead" is used to mean a stochastic evaluation of the dual upper bound given in Proposition 1. In other words, we "lookahead" into the future and then exploit this information (thereby relaxing nonanticipativity) to produce an upper bound.

The root node of $\mathcal{T}^n$, for all n, is $\mathbf{x}_0 = s_0$. Recall that any node contains full information regarding the path from the initial state $\mathbf{x}_0 = s_0$. Therefore, in this paper, the edges of the tree are implied and we do not need to explicitly refer to them; however, we will use the following notation. For a state node $\mathbf{x} \in \mathcal{X}^n$, let $\mathcal{Y}^n(\mathbf{x})$ be the child state–action nodes (i.e., already expanded nodes) of $\mathbf{x}$ at iteration n (dependence on $\mathcal{T}^n$ is suppressed) and let $\tilde{\mathcal{Y}}^n(\mathbf{x})$ be the unexpanded state–action nodes of $\mathbf{x}$:

$$\mathcal{Y}^n(\mathbf{x}) = \{(\mathbf{x}, a') : a' \in \mathcal{A}_{\mathbf{x}},\ (\mathbf{x}, a') \in \mathcal{Y}^n\},$$
$$\tilde{\mathcal{Y}}^n(\mathbf{x}) = \{(\mathbf{x}, a') : a' \in \mathcal{A}_{\mathbf{x}},\ (\mathbf{x}, a') \notin \mathcal{Y}^n(\mathbf{x})\}.$$

Furthermore, we write $\tilde{\mathcal{Y}}^n = \cup_{\mathbf{x}\in\mathcal{X}^n} \tilde{\mathcal{Y}}^n(\mathbf{x})$.

Similarly, for $\mathbf{y} = (s_0, a_0, \ldots, s_t, a_t) \in \mathcal{Y}^n$, let $\mathcal{X}^n(\mathbf{y})$ be the child state nodes of $\mathbf{y}$ and let $\tilde{\mathcal{X}}^n(\mathbf{y})$ be the unexpanded state nodes of $\mathbf{y}$:

$$\mathcal{X}^n(\mathbf{y}) = \{(\mathbf{y}, s) : s \in \mathcal{S},\ (\mathbf{y}, s) \in \mathcal{X}^n\},$$
$$\tilde{\mathcal{X}}^n(\mathbf{y}) = \{(\mathbf{y}, s) : s \in \mathcal{S},\ (\mathbf{y}, s) \notin \mathcal{X}^n(\mathbf{y})\}.$$

For mathematical convenience, we have $\bar{V}^0$, $\bar{Q}^0$, $\bar{u}^0$, v^0, and l^0 taking the value zero for all elements of their respective domains. For each $\mathbf{x} \in \mathcal{X}^n$ and $\mathbf{y} \in \mathcal{Y}^n$, let $\bar{V}^n(\mathbf{x})$ and $\bar{Q}^n(\mathbf{y})$ represent the estimates of $V^*(\mathbf{x})$ and $Q^*(\mathbf{y})$, respectively. Note that although $\bar{V}^n(\mathbf{x})$ is defined (and equals zero) prior to the expansion of $\mathbf{x}$, it does not gain meaning until $\mathbf{x} \in \mathcal{X}^n$. The same holds for the other quantities.

Each unexpanded state–action node $\mathbf{y}^{\mathrm{u}} \in \tilde{\mathcal{Y}}^n$ is associated with an estimated dual upper bound $\bar{u}^n(\mathbf{y}^{\mathrm{u}})$. A state node $\mathbf{x}$ is called *expandable* on iteration n if $\tilde{\mathcal{Y}}^n(\mathbf{x})$ is nonempty. Similarly, a state–action node $\mathbf{y}$ is *expandable* on iteration n if $\tilde{\mathcal{X}}^n(\mathbf{y})$ is nonempty. In addition, let $v^n(\mathbf{x})$ and $v^n(\mathbf{y})$ count the number of times that $\mathbf{x}$ and $\mathbf{y}$ are visited by the selection policy (so v^n becomes positive after expansion). The tally $l^n(\mathbf{y})$ counts the number of dual look-aheads performed at each unexpanded state–action node. We also need step sizes $\alpha^n(\mathbf{x})$ and $\alpha^n(\mathbf{y})$ to track the estimates $\bar{V}^n(\mathbf{x})$ generated by the default policy π^{d} for leaf nodes $\mathbf{x}$ and $\bar{u}^n(\mathbf{y})$ for leaf nodes $\mathbf{y} \in \tilde{\mathcal{Y}}^n$.

Lastly, we define two sets of *progressive widening iterations*, $\mathcal{N}_x \subseteq \{1, 2, \ldots\}$ and $\mathcal{N}_y \subseteq \{1, 2, \ldots\}$. When $v^n(\mathbf{x}) \in \mathcal{N}_x$, we consider expanding the state node $\mathbf{x}$ (i.e., adding a new state–action node stemming from $\mathbf{x}$), and when $v^n(\mathbf{y}) \in \mathcal{N}_y$, we consider expanding the state–action node $\mathbf{y}$ (i.e., adding a downstream state node stemming from $\mathbf{y}$). Note that the progressive widening formula given in Couëtoux et al. (2011) is a particular choice of $\mathcal{N}_x$ and $\mathcal{N}_y$; they suggest frequent widening in early iterations and reducing this frequency as the algorithm progresses.

3.1. Selection

Let π^{s} be a *selection policy* that steers the algorithm down the *current version* of the decision tree $\mathcal{T}^{n-1}$.

Each call to π^{s} is independent from the rest of the system and depends only on the current node and its child nodes in the decision tree. We use the same notation for both types of nodes: for $\mathbf{x} \in \mathcal{X}^{n-1}$ and $\mathbf{y} \in \mathcal{Y}^{n-1}$, we have

$$\pi^{\mathrm{s}}(\mathbf{x}, \mathcal{T}^{n-1}) \in \mathcal{Y}^{n-1}(\mathbf{x}) \quad \text{and} \quad \pi^{\mathrm{s}}(\mathbf{y}, \mathcal{T}^{n-1}) \in \mathcal{X}^{n-1}(\mathbf{y}).$$

Let us emphasize that π^{s} contains no logic for expanding the tree and simply provides a path down the partial tree $\mathcal{T}^{n-1}$. The most popular MCTS implementations (Chang et al. 2005, Kocsis and Szepesvári 2006) use the UCB1 policy (Auer et al. 2002) for π^{s} when acting on state nodes. The UCB1 policy balances exploration and exploitation by selecting the state–ction node $\mathbf{y}$ by solving

$$\pi^{\mathrm{s}}(\mathbf{x}, \mathcal{T}^{n-1}) \in \operatorname*{arg\,max}_{\mathbf{y} \in \mathcal{Y}^{n-1}(\mathbf{x})} \bar{Q}^{n-1}(\mathbf{y}) + \sqrt{\frac{2 \ln \sum_{\mathbf{y}' \in \mathcal{Y}^{n-1}(\mathbf{x})} v^{n-1}(\mathbf{y}')}{v^{n-1}(\mathbf{y})}}. \tag{9}$$

The second term is an *exploration bonus* that decreases as nodes are visited. Other multiarmed bandit policies may also be used; for example, we may instead prefer to implement an ϵ-greedy policy where we exploit with probability $1 - \epsilon$ and explore with probability (w.p.) ϵ:

$$\pi^{\mathrm{s}}(\mathbf{x}, \mathcal{T}^{n-1}) = \begin{cases} \operatorname*{arg\,max}_{\mathbf{y} \in \mathcal{Y}^{n-1}(\mathbf{x})} \bar{Q}(\mathbf{y}), & \text{w.p.} \quad 1-\epsilon, \\ \text{a random element from } \mathcal{Y}^{n-1}(\mathbf{x}), & \text{w.p.} \quad \epsilon. \end{cases}$$

When acting on state–action nodes, π^{s} selects a downstream state node; for example, given $\mathbf{y}_t = (s_0, a_0, \ldots, s_t, a_t)$, the selection policy $\pi^{\mathrm{s}}(\mathbf{y}_t, \mathcal{T}^{n-1})$ may select $\mathbf{x}_{t+1} = (s_0, a_1, \ldots, s_{t+1}) \in \mathcal{X}^{n-1}(\mathbf{y}_t)$ with probability $\mathbf{P}(S_{t+1} = s_{t+1} \,|\, \mathbf{y}_t)$, normalized by the total probability of reaching expanded nodes $\mathcal{X}^n(\mathbf{y}_t)$. We require the condition that once all downstream states are expanded, the sampling probabilities match the transition probabilities of the original MDP. We now summarize the *selection phase* of the primal-dual MCTS.

- Start at the root node and descend the tree using the selection policy π^{s} until one of the following is reached: condition (S1), an expandable state node $\mathbf{x}$ with $v^n(\mathbf{x}) \in \mathcal{N}_x$, condition (S2), an expandable state–action node $\mathbf{y}$ with $v^n(\mathbf{y}) \in \mathcal{N}_y$, or condition (S3), a leaf state node $\mathbf{x}$ is reached.
- If the selection policy ends with conditions (S1) or (S2), then we move on to the *expansion* step. Otherwise, we move on to the *simulation and backpropagation* steps.

3.2. Expansion

Case 1. First, suppose that on iteration n, the selection phase of the algorithm returns $\mathbf{x}^n_{\tau_{\mathrm{e}}} = (s_0, a_0, \ldots, s_{\tau_{\mathrm{e}}})$ to be expanded, for some $\tau_{\mathrm{e}} \in \mathcal{T}$. Given the possibly large set of unexpanded actions, we first sample a subset of candidate actions according sampling policy π^{a} that returns a random subset (e.g., each element of $\mathcal{A}$ that has not been expanded is independently included with some probability):

$$\pi^{\mathrm{a}}\left(\mathbf{x}^n_{\tau_{\mathrm{e}}}, \mathcal{T}^{n-1}\right) \subseteq \left\{a \in \mathcal{A}_{\mathbf{x}^n_{\tau_{\mathrm{e}}}} : \left(\mathbf{x}^n_{\tau_{\mathrm{e}}}, a\right) \in \tilde{\mathcal{Y}}^n\left(\mathbf{x}^n_{\tau_{\mathrm{e}}}\right)\right\}.$$

Then, for each candidate, we perform a look-ahead to obtain an estimate of the perfect information relaxation dual upper bound. The look-ahead is evaluated by solving a deterministic optimization problem on one sample path of the random process $\{W_t\}$. In the most general case, this is a deterministic dynamic program. However, other formulations may be more natural and/or easier to solve for some applications. If the contribution functions c_t are linear, then the deterministic problem could be as simple as a linear program (e.g., the asset acquisition problem class described in Nascimento and Powell (2009); see also Al-Kanj et al. (2016) for an example where the information relaxation is a mixed-integer linear program).

The resulting upper-bound estimate is then smoothed with the previous estimate via a stepsize $\alpha^n(\mathbf{x}^n_{\tau_{\mathrm{e}}}, a)$, which should be chosen to be a function of the number of visits $l^n(\mathbf{x}^n_{\tau_{\mathrm{e}}}, a)$. This smoothing process ensures convergence to a true upper bound, even if any single estimate is noisy. We select the action with the highest upper bound to expand, but only if the upper bound is larger than the current best value function $\bar{Q}^n$. Otherwise, we skip the expansion step because our estimates tell us that none of the candidate actions are optimal. In a typical MCTS implementation, a node is simply added at random. The following steps comprise the *expansion phase* of the primal-dual MCTS for $\mathbf{x}^n_{\tau_{\mathrm{e}}}$.

- Let $\tilde{\mathcal{A}}^n(\mathbf{x}^n_{\tau_{\mathrm{e}}})$ be an independent sample generated by $\pi^{\mathrm{a}}(\mathbf{x}^n_{\tau_{\mathrm{e}}}, \mathcal{T}^{n-1})$. This is the subset of candidate unexpanded actions that we will consider.
- Obtain a single sample path $\mathbf{w}^n_{\tau_{\mathrm{e}}+1,T} = (w^n_{\tau_{\mathrm{e}}+1}, \ldots, w^n_T)$ of the exogenous information process. For each candidate action $a \in \tilde{\mathcal{A}}^n(\mathbf{x}^n_{\tau_{\mathrm{e}}})$, compute the optimal value of the deterministic optimization "inner" problem of (8):

$$\hat{u}^n\left(\mathbf{x}^n_{\tau_{\mathrm{e}}}, a\right) = c_{\tau_{\mathrm{e}}}\left(s, a, w^n_{\tau_{\mathrm{e}}+1}\right) + \max_{\mathbf{a}}\left[h_{\tau_{\mathrm{e}}+1}\left(S_{\tau_{\mathrm{e}}+1}, \mathbf{a}, \mathbf{w}^n_{\tau_{\mathrm{e}}+1,T}\right) - z^v_{\tau_{\mathrm{e}}+1}\left(S_{\tau_{\mathrm{e}}+1}, \mathbf{a}, \mathbf{w}^n_{\tau_{\mathrm{e}}+1,T}\right)\right].$$

- For each candidate action $a \in \tilde{\mathcal{A}}^n(\mathbf{x}^n_{\tau_{\mathrm{e}}})$, smooth the newest observation of the upper bound with the previous estimate via a stochastic gradient step:

$$\bar{u}^n\left(\mathbf{x}^n_{\tau_{\mathrm{e}}}, a\right) = \left(1 - \alpha^n\left(\mathbf{x}^n_{\tau_{\mathrm{e}}}, a\right)\right) \bar{u}^{n-1}\left(\mathbf{x}^n_{\tau_{\mathrm{e}}}, a\right) + \alpha^n\left(\mathbf{x}^n_{\tau_{\mathrm{e}}}, a\right) \hat{u}^n\left(\mathbf{x}^n_{\tau_{\mathrm{e}}}, a\right). \tag{10}$$

State–action nodes $\mathbf{y}$ elsewhere in the tree that are not considered for expansion retain the same upper bound estimates; that is, $\bar{u}^n(\mathbf{y}) = \bar{u}^{n-1}(\mathbf{y})$.

- Let $a^n = \arg\max_{a \in \mathcal{A}^n(\mathbf{x}^n_{\tau_e})} \bar{u}^n(\mathbf{x}^n_{\tau_e}, a)$ be the candidate action with the best dual upper bound. If no candidate is better than the current best value, that is, $\bar{u}^n(\mathbf{x}^n_{\tau_e}, a^n) \le \bar{V}^{n-1}(\mathbf{x}^n_{\tau_e})$, then we skip this potential expansion and return to the *selection phase* to continue down the tree.
- Otherwise, if the candidate is better than the current best, that is, $\bar{u}^n(\mathbf{x}^n_{\tau_e}, a^n) > \bar{V}^{n-1}(\mathbf{x}^n_{\tau_e})$, then we *expand* action a^n by adding the node $\mathbf{y}^n_{\tau_e} = (\mathbf{x}^n_{\tau_e}, a^n)$ as a child of $\mathbf{x}^n_{\tau_e}$. We then immediately sample a downstream state $\mathbf{x}^n_{\tau_e+1}$ using π^s from the set $\tilde{\mathcal{X}}^n(\mathbf{y}^n_{\tau_e})$ and add it as a child of $\mathbf{y}^n_{\tau_e}$ (every state–action expansion triggers a state expansion). After doing so, we are ready to move on to the *simulation and backpropagation phase* from the leaf node $\mathbf{x}^n_{\tau_e+1}$.

Case 2. Now suppose that we entered the *expansion phase* via a state–action node $\mathbf{y}^n_{\tau_e}$. In this case, we simply sample a single state $\mathbf{x}^n_{\tau_e+1} = (\mathbf{y}^n_{\tau_e}, s_{\tau_e+1})$ from $\tilde{\mathcal{X}}^n(\mathbf{y}^n_{\tau_e})$ such that

$$\mathbf{P}\left(S_{\tau_e+1} = s_{\tau_e+1} \,|\, \mathbf{y}^n_{\tau_e}\right) > 0$$

and add it as a child of $\mathbf{y}^n_{\tau_e}$. Next, we continue to the *simulation and backpropagation phase* from the leaf node $\mathbf{x}^n_{\tau_e+1}$.

3.3. Simulation and Backpropagation

We are now at a leaf node $\mathbf{x}^n_{\tau_s} = (s_0, a_0, \ldots, s_{\tau_s})$, for some $\tau_s \le T$. At this point, we cannot descend further into the tree, so we proceed to the *simulation and backpropagation phase*. The last two steps of the algorithm are relatively simple: first, we run the default policy to produce an estimate of the leaf node's value and then update the values "up" the tree via equations resembling (2) and (3). The steps are as follows:

- Obtain a single sample path $\mathbf{w}^n_{\tau_s+1,T} = (w^n_{\tau_s+1}, \ldots, w^n_T)$ of the exogenous information process and, using the default policy π^d, compute the value estimate

$$\hat{V}^n\left(\mathbf{x}^n_{\tau_s}\right) = h_{\tau_s}\left(s_{\tau_s}, \pi^d, \mathbf{w}^n_{\tau_s+1,T}\right) \mathbf{1}_{\{\tau_s < T\}}. \tag{11}$$

If $\tau_s = T$, then the value estimate is simply the terminal value of zero. The value of the leaf node is updated by averaging the new observation with previous observations according to the equation

$$\bar{V}^n\left(\mathbf{x}^n_{\tau_s}\right) = \bar{V}^{n-1}\left(\mathbf{x}^n_{\tau_s}\right) + \frac{1}{v^n\left(\mathbf{x}^n_{\tau_s}\right)}\left[\hat{V}^n\left(\mathbf{x}^n_{\tau_s}\right) - \bar{V}^{n-1}\left(\mathbf{x}^n_{\tau_s}\right)\right].$$

- After simulation, we backpropagate the information up the tree. Working backward from the leaf node, we can extract a *path*, or a sequence of state and state–action nodes

$$\mathbf{x}^n_{\tau_s}, \mathbf{y}^n_{\tau_s-1}, \ldots, \mathbf{x}^n_1, \mathbf{y}^n_0, \mathbf{x}^n_0.$$

Each of these elements is a *subsequence* of the vector $\mathbf{x}^n_{\tau_s} = (s_0, a^n_0, \ldots, s^n_{\tau_s})$, starting with s_0. For $t = \tau_s - 1, \tau_s - 2, \ldots, 0$, the backpropagation equations are

$$\bar{Q}^n\left(\mathbf{y}^n_t\right) = \bar{Q}^{n-1}\left(\mathbf{y}^n_t\right) + \frac{1}{v^n\left(\mathbf{y}^n_t\right)}\left[c_t\left(s^n_t, a^n_t, w^n_{t+1}\right) + \bar{V}^n\left(\mathbf{x}^n_{t+1}\right) - \bar{Q}^{n-1}\left(\mathbf{y}^n_t\right)\right], \tag{12}$$

$$\tilde{V}^n\left(\mathbf{x}^n_t\right) = \tilde{V}^{n-1}\left(\mathbf{x}^n_t\right) + \frac{1}{v^n\left(\mathbf{x}^n_t\right)}\left[\bar{Q}^n\left(\mathbf{y}^n_t\right) - \tilde{V}^{n-1}\left(\mathbf{x}^n_t\right)\right], \tag{13}$$

$$\bar{V}^n\left(\mathbf{x}^n_t\right) = (1 - \lambda^n)\, \tilde{V}^n\left(\mathbf{x}^n_t\right) + \lambda^n \max_{\mathbf{y}_t} \bar{Q}^n\left(\mathbf{y}_t\right), \tag{14}$$

where $\mathbf{y}_t \in \mathcal{Y}^n(\mathbf{x}^n_t)$, and $\lambda^n \in [0,1]$ is a mixture parameter. Nodes $\mathbf{x}$ and $\mathbf{y}$ that are not part of the path down the tree retain their values; that is,

$$\bar{V}^n(\mathbf{x}) = \bar{V}^{n-1}(\mathbf{x}) \quad \text{and} \quad \bar{Q}^n(\mathbf{y}) = \bar{Q}^{n-1}(\mathbf{y}). \tag{15}$$

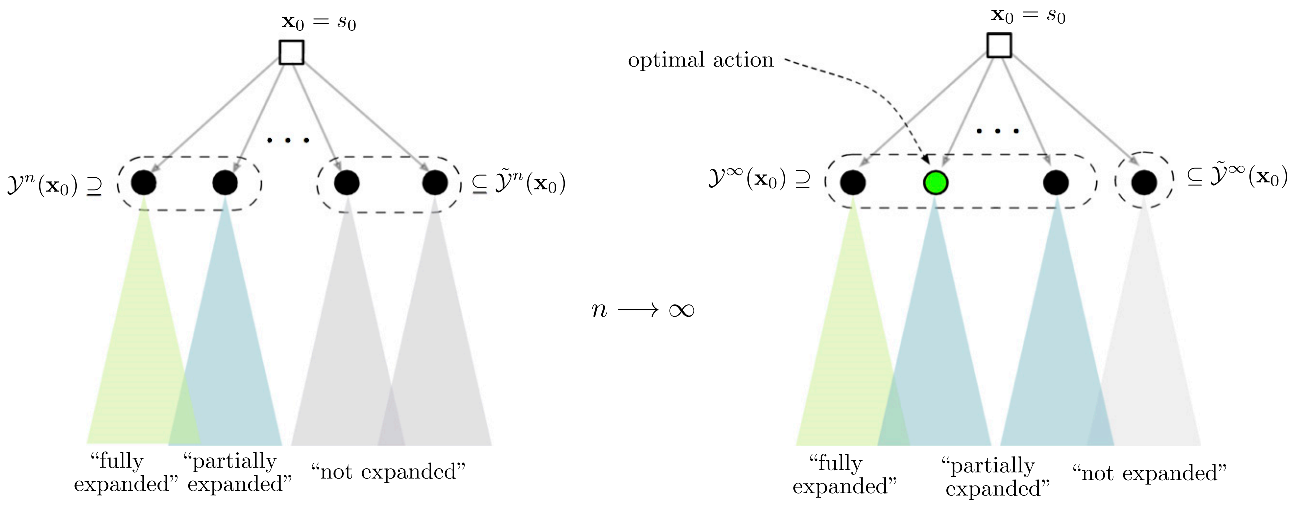
The first update (12) maintains the estimates of the state–action value function as weighted averages of child node values. The second update (13) similarly performs a recursive averaging scheme for the state nodes, and finally, the third update (14) sets the value of a state node to be a mixture between the weighted average of its child state–action node values and the maximum value of its child state–action nodes.

The naive update for $\bar{V}^n$ is to simply take the maximum over the state–action nodes (i.e., following the Bellman equation), removing the need to track $\tilde{V}^n$. Empirical evidence from Coulom (2007, p. 77), however, shows that this type of update can create instability; furthermore, the authors state that "the mean operator is more accurate when the number of simulations is low, and the max operator is more accurate when the number of simulations is high." Taking this recommendation, we impose the property that $\lambda^n \nearrow 1$ so that asymptotically we achieve the Bellman update yet allow for the averaging scheme to create stability in the earlier iterations. The update (14) is similar to the mix backup suggested by Coulom (2007) that achieves superior empirical performance.

The end of the simulation and backpropagation phase marks the conclusion of one iteration of the primal-dual MCTS algorithm. We now return to the root node and begin a new selection phase. Algorithm 1 gives a concise summary of primal-dual MCTS. Moreover, Figure 1 emphasizes several key properties of the algorithm:

- The left tree shows that on some given iteration n, state–action nodes of a state node ($\mathbf{x}_0$) may be (1) added to the tree and have a fully expanded subtree, (2) added to the tree and have a partially expanded subtree, or (3) not added to the tree.
- The utilization of dual bounds allows entire subtrees to be ignored, even in the limit (rightmost node in the right tree), thereby providing potentially significant computational savings.

Figure 1. (Color online) Properties of the Primal-Dual MCTS Algorithm


$\mathbf{x}_0 = s_0$
$\mathcal{Y}^n(\mathbf{x}_0) \supseteq$
$\subseteq \tilde{\mathcal{Y}}^n(\mathbf{x}_0)$
"fully expanded"
"partially expanded"
"not expanded"
$n \longrightarrow \infty$
$\mathbf{x}_0 = s_0$
optimal action
$\mathcal{Y}^\infty(\mathbf{x}_0) \supseteq$
$\subseteq \tilde{\mathcal{Y}}^\infty(\mathbf{x}_0)$
"fully expanded"
"partially expanded"
"not expanded"

• The optimal action at the root node (shown as the second circle in the right tree) can be found without its subtree necessarily being fully expanded.

We will analyze these properties in the next section, but we first present an example that illustrates in detail the steps taken during the expansion phase.

Algorithm 1 (Primal-Dual MCTS)

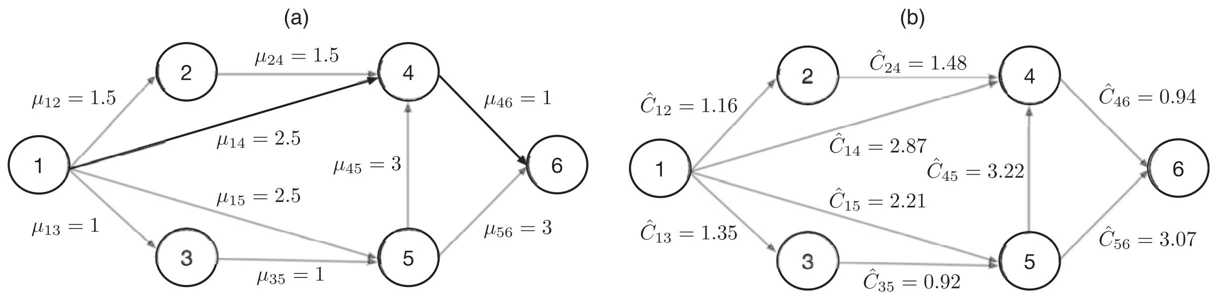
```
Input: An initial state node $\mathbf{x}_0$, a default policy $\pi^d$, a
       selection policy $\pi^s$, a candidate sampling policy
       $\pi^a$, a step-size rule $\{\alpha^n\}$, a backpropagation
       mixture scheme $\{\lambda^n\}$.
Output: Partial desicion trees $\{\mathscr{T}^n\}$.
  for $n = 1, 2, \ldots$ do
1     run Selection phase with policy $\pi^s$ from $\mathbf{x}_0$ and
        return either condition (S1) with $\mathbf{x}^n_{\tau_e}$, (S2) with
        $\mathbf{y}^n_{\tau_e}$, or (S3) with $\mathbf{x}^n_{\tau_s}$.
      if condition (S1) then
2         run Case 1 of Expansion phase with policy $\pi^a$
            at state node $\mathbf{x}^n_{\tau_e}$ and return leaf node $\mathbf{x}^n_{\tau_s} = \mathbf{x}^n_{\tau_e+1}$.
      else if condition (S2) then
3         run Case 2 of Expansion phase at state–action
            node $\mathbf{y}^n_{\tau_e}$ and return leaf node $\mathbf{x}^n_{\tau_s} = \mathbf{x}^n_{\tau_e+1}$.
      end
4     run Simulation and Backpropagation phase from
        leaf node $\mathbf{x}^n_{\tau_s}$.
  end
```

Example 1 (Shortest Path with Random Edge Costs). In this example, we consider applying the primal-Dual MCTS to a shortest-path problem with random edge costs (note that the algorithm is stated for maximization, while the shortest path is a minimization problem). The graph used for this example is shown in Figure 2(a). An agent starts at vertex 1 and aims to reach vertex 6 at the minimum expected cumulative cost. The cost for edge e_{ij} (from vertex i to j) is distributed $\mathcal{N}(\mu_{ij}, \sigma^2_{ij})$ and independent from the costs of other edges and independent across time. At every decision epoch, the agent chooses an edge to traverse out of the current vertex without knowing the actual costs. After the decision is made, a realization of edge costs is revealed, and the agent incurs the one-stage cost associated with the traversed edge. Vertex 6 is "absorbing" in the sense that all subsequent transitions are costless, and the agent stays at vertex 6. For this particular shortest-path problem, the number of steps needed to reach vertex 6 is either two, three, or four; thus, the problem can be formulated as a finite-horizon MDP with $T = 4$.

The means of the cost distributions are also shown in Figure 2(a), and we assume that $\sigma_{ij} = 0.25$. The optimal path is $1 \to 4 \to 6$, which achieves an expected cumulative cost of 3.5. Consider applying primal-dual MCTS at vertex 1, meaning that we are choosing between traversing edges e_{12}, e_{13}, e_{14}, and e_{15}. The shortest paths starting with e_{12}, e_{13}, e_{14}, and e_{15} are $1 \to 2 \to 4 \to 6$ (cost of 4), $1 \to 3 \to 5 \to 6$ (cost of 5), $1 \to 4 \to 6$ (cost of 3.5), and $1 \to 5 \to 6$ (cost of 5.5), respectively. Hence, $Q^*(1,e_{12})=4$, $Q^*(1,e_{13})=5$, $Q^*(1,e_{14})=3.5$, and $Q^*(1,e_{15})=5.5$.

In this example, let us assume for simplicity that the sampling policy π^a for the expansion step always returns the set of all actions. We now illustrate several consecutive expansion steps (this means that there are nonexpansion steps in between that are not shown) from the point of view of vertex 1, where there are four possible actions, e_{1i} for $i = 2, 3, 4, 5$. On every expansion step, we use one sample of exogenous information (costs) to perform the information relaxation step and compute a standard dual (lower) bound. For simplicity, suppose that on every expansion step, we see the same sample of costs that are shown in Figure 2(b). By finding the shortest paths in the graph with sampled costs, the sampled dual bounds are thus given by $\bar{u}^n(1, e_{12}) = 3.58$, $\bar{u}^n(1, e_{13}) = 5.34$, $\bar{u}^n(1, e_{14}) = 3.81$, and $\bar{u}^n(1, e_{15}) = 5.28$ (assuming that the initial step size for averaging

Figure 2. Shortest-Path Problem with Random Edge Costs



Note. (a) Mean costs; (b) sample costs.

the upper bounds α^n is 1). Figure 3 illustrates the expansion process.

1. In the first expansion, nothing has been expanded, so we simply expand edge e_{12} because it has the lowest dual bound. Note that this is not the optimal action; the optimistic dual bound is the result of noise.

2. After some iterations, learning has occurred for $\bar{Q}^n(1, e_{12})$, and it is currently estimated to be 3.97. We expand e_{14} because it is the only unexpanded action with a dual bound that is better than 3.97. This is the optimal action.

3. In the last step of Figure 3, no actions are expanded because their dual bounds indicate that they are no better than the currently expanded actions using solid lines to indicate expanded actions and dotted lines to indicate unexpanded actions.

Before we move on to the convergence analysis, we point out that, in principle, our algorithm could have used any valid upper bound on the value function. For example, under the assumption that a stochastic model of the system is known, Chen and Farias (2013) derive a *certainty equivalent* upper bound for a dynamic pricing problem based on Jensen's inequality. Unlike the information relaxation method, this particular bound is not easily applicable if the decision maker does not have access to the true distributions of random quantities in the system. Hence, our paper focuses specifically on integrating the noisy sampled information relaxation bounds into MCTS and proving the consistency of the new algorithm.

4. Analysis of Convergence

Let $\mathcal{T}^\infty$ be the *limiting partial decision tree* as iterations $n \to \infty$. Similarly, we use the notation $\mathcal{X}^\infty$, $\mathcal{X}^\infty(\mathbf{y})$, $\tilde{\mathcal{X}}^\infty(\mathbf{y})$, $\mathcal{Y}^\infty$, $\mathcal{Y}^\infty(\mathbf{x})$, and $\tilde{\mathcal{Y}}^\infty(\mathbf{x})$ to describe the random sets of expanded and unexpanded nodes of the tree in the limit, analogous to the notation for a finite iteration n. Given that there are a finite number of nodes and that the cardinality of these sets is monotonic with respect to n, it is clear that these limiting sets are well defined.

Recall that each iteration of the algorithm generates a leaf node $\mathbf{x}^n_{\tau_s}$, which also represents the path down the tree for iteration n. Before we begin the convergence analysis, let us state a few assumptions.

Assumption 1. *Assume that the following hold:*

a. *There exists an $\epsilon^s > 0$ such that given any tree $\mathcal{T}$ containing a state node $\mathbf{x}_t \in \mathcal{X}$ and a state–action node $\mathbf{y}_t = (\mathbf{x}_t, a_t)$ with $a_t \in \mathcal{A}_{\mathbf{x}_t}$, it holds that $\mathbf{P}(\pi^s(\mathbf{x}_t, \mathcal{T}) = \mathbf{y}_t) \geq \epsilon^s$.*

b. *Given a tree $\mathcal{T}$ containing a state–action node $\mathbf{y}_t$, if all child state nodes of $\mathbf{y}_t$ have been expanded, then*

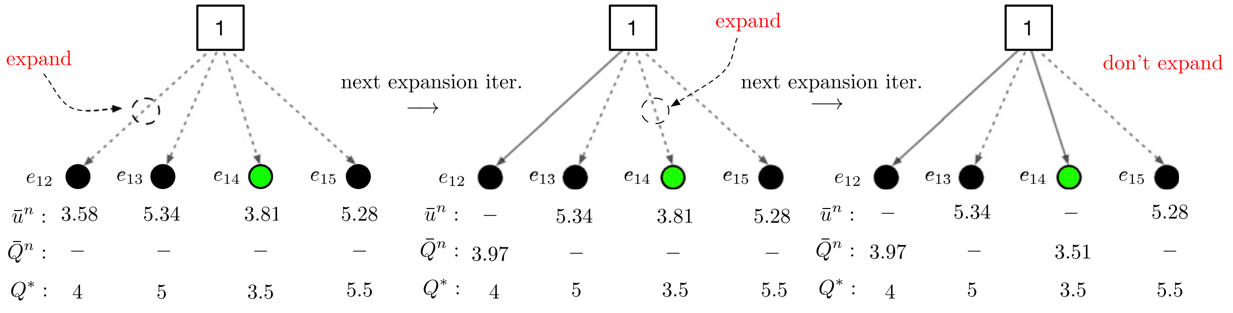
$$\mathbf{P}(\pi^s(\mathbf{y}_t, \mathcal{T}) = \mathbf{x}_{t+1}) = \mathbf{P}\big(S_{t+1} = s_{t+1} \,|\, \mathbf{y}_t\big),$$

where $\mathbf{x}_{t+1} = (\mathbf{y}_t, s_{t+1})$. This means that sampling eventually occurs according to the true distribution of S_{t+1}.

c. *There exists an $\epsilon^a > 0$ such that, given any tree $\mathcal{T}$ containing a state node $\mathbf{x}_t \in \mathcal{X}$ and action $a_t \in \mathcal{A}_{\mathbf{x}_t}$, it holds that $\mathbf{P}(a_t \in \pi^a(\mathbf{x}_t, \mathcal{T})) \geq \epsilon^a$.*

d. *There are an infinite number of progressive widening iterations: $|\mathcal{N}_x| = |\mathcal{N}_y| = \infty$.*

Figure 3. (Color online) Expansion Steps for the Example Problem



e. *For any state node* $\mathbf{x}_t \in \mathcal{X}$ *and action* a_t, *the step size* $\alpha^n(\mathbf{x}_t, a_t)$ *takes the form*

$$\alpha^n(\mathbf{x}_t, a_t) = \tilde{\alpha}^n \, \mathbf{1}_{\{\mathbf{x}_t \in \mathbf{x}^n_{\tau_s}\}} \, \mathbf{1}_{\{v^n(\mathbf{x}_t) \in \mathcal{N}_x\}} \, \mathbf{1}_{\{a_t \in \pi^a(\mathbf{x}_t, \mathcal{T}^{n-1})\}},$$

for some possibly random sequence $\{\tilde{\alpha}^n\}$, *with the notation* $\mathbf{x}_t \in \mathbf{x}^n_{\tau_s}$ *used to indicate that* $\mathbf{x}_t$ *is a path contained in* $\mathbf{x}^n_{\tau_s}$. *Thus, whenever the dual look-ahead update* (10) *is not performed, the step size is zero. In addition, the step-size sequence satisfies*

$$\sum_{n=0}^{\infty} \alpha^n(\mathbf{x}_t, a_t) = \infty \quad \textit{a.s. and}$$
$$\sum_{n=0}^{\infty} \alpha^n(\mathbf{x}_t, a_t)^2 < \infty \quad \textit{a.s.,}$$

the standard stochastic approximation assumptions.

f. *As* $n \to \infty$, *the backpropagation mixture parameter* $\lambda^n \to 1$.

An example of a step-size sequence that satisfies Assumption 1(e) is $1/l^n(\mathbf{x}_t, a_t)$. We now use various aspects of Assumption 1 to demonstrate that expanded nodes within the decision tree are visited infinitely often. This is, of course, crucial in proving convergence, but because of the use of dual bounds, we only require that the limiting *partial decision tree* be visited infinitely often. Previous results in the literature require this property on the fully expanded tree.

Lemma 1. *Under Assumption* 1, *the following statements hold:*

a. *Let* $\mathbf{x} \in \mathcal{X}$ *be a state node such that* $\mathbf{P}(\mathbf{x} \in \mathcal{X}^\infty) > 0$. *Then* $v^n(\mathbf{x}) \to \infty$ *a.e. on* $\{\mathbf{x} \in \mathcal{X}^\infty\}$.

b. *Let* $\mathbf{y} \in \mathcal{Y}$ *be a state–action node such that* $\mathbf{P}(\mathbf{y} \in \mathcal{Y}^\infty) > 0$. *Then* $v^n(\mathbf{y}) \to \infty$ *a.e. on* $\{\mathbf{y} \in \mathcal{Y}^\infty\}$.

c. *Also, let* $\mathbf{y}' \in \mathcal{Y}$ *be such that* $\mathbf{P}(\mathbf{y}' \in \tilde{\mathcal{Y}}^\infty) > 0$. *Then* $l^n(\mathbf{y}') \to \infty$ *a.e. on* $\{\mathbf{y}' \in \tilde{\mathcal{Y}}^\infty\}$; *that is, the dual look-ahead for the unexpanded state–action node* $\mathbf{y}' \in \tilde{\mathcal{Y}}^\infty(\mathbf{x}')$ *is performed infinitely often.*

Proof. See Appendix A.

The next lemma reveals the central property of primal-dual MCTS (under the assumption that all relevant values converge appropriately): for any expanded state node, its corresponding optimal state–action node is expanded. In other words, if a particular action is never expanded, then it must be suboptimal.

Lemma 2. *Consider a state node* $\mathbf{x}_t \in \mathcal{X}$. *Consider the event on which* $\mathbf{x}_t \in \mathcal{X}^\infty$ *and the following hold:*

a. $\bar{Q}^n(\mathbf{y}_t) \to Q^*(\mathbf{y}_t)$ *for each expanded* $\mathbf{y}_t \in \mathcal{Y}^\infty(\mathbf{x}_t)$.

b. $\bar{u}^n(\mathbf{y}'_t) \to u^\nu(\mathbf{y}'_t)$ *for each unexpanded* $\mathbf{y}'_t \in \tilde{\mathcal{Y}}^\infty(\mathbf{x}_t)$.

Then, on this event, there is a state–action node $\mathbf{y}^*_t = (\mathbf{x}_t, a^*_t) \in \mathcal{Y}^\infty(\mathbf{x}_t)$ *associated with an optimal action* $a^*_t \in \arg\max_{a \in \mathcal{A}} Q^*_t(s_t, a)$.

Sketch of Proof. The essential idea of the proof is as follows. If all optimal actions are unexpanded and the assumptions of the lemma hold, then, eventually, the dual bound associated with an unexpanded optimal action must upper-bound the values associated with the expanded actions (all of which are suboptimal). Thus, given the design of our expansion strategy to explore actions with high dual upper bounds, it follows that an optimal action must eventually be expanded. Appendix A gives the technical details of the proof.

We are now ready to state the main theorem, which shows the consistency of the proposed procedure. We remark that it is never required that $\mathcal{X}^\infty_t = \mathcal{X}_t$ or $\mathcal{Y}^\infty_t = \mathcal{Y}_t$. In other words, an important feature of primal-dual MCTS is that the tree does not need to be fully expanded in the limit, as we alluded to earlier in Figure 1.

Theorem 2. *Under Assumption* 1, *the primal-dual MCTS procedure converges at the root node (initial state) in two ways:*

$$\bar{V}^n(\mathbf{x}_0) \to V^*(\mathbf{x}_0) \;\; \textit{a.s.} \quad \textit{and} \quad \limsup_{n\to\infty} \arg\max_{\mathbf{y} \in \mathcal{Y}^n(\mathbf{x}_0)} \bar{Q}^n(\mathbf{y}) \subseteq \arg\max_{\mathbf{y}_0 = (\mathbf{x}_0, a)} Q^*(\mathbf{y}_0) \;\; \textit{a.s.,}$$

meaning that the value of the node $\mathbf{x}_0$ *converges to the optimal value and that an optimal action is both expanded and identified.*

Sketch of Proof. The proof of the main theorem places the results established in the previous lemmas in an induction framework that moves up the tree, starting from state nodes $\mathbf{x}_T \in \mathcal{X}_T$. We first establish the following convergence results:

$$\bar{Q}^n(\mathbf{y}_t) \to Q^*(\mathbf{y}_t)\, \mathbf{1}_{\{\mathbf{y}_t \in \mathcal{Y}^\infty_t\}} \quad \text{a.s.,}$$
$$\bar{u}^n(\mathbf{y}_t) \to u^\nu(\mathbf{y}_t)\, \mathbf{1}_{\{\mathbf{y}_t \in \tilde{\mathcal{Y}}^\infty_t\}} \quad \text{a.s.,}$$

after which Lemma 2 can be invoked to conclude $\bar{V}^n(\mathbf{x}_t) \to V^*(\mathbf{x}_t)\, \mathbf{1}_{\{\mathbf{x}_t \in \mathcal{X}^\infty_t\}}$ a.s. The full details are given in Appendix A.

In addition, let us comment that Assumption 1(e) could also be replaced with an alternative condition on the selection policy; for example, if the visits concentrate on the optimal action asymptotically, then the average over the state–action values would converge to the optimal value. Chang et al. (2005) take this approach by using results from the multiarmed bandit literature (Bubeck and Cesa-Bianchi 2012). The Chang et al. (2005) method, although similar to MCTS (and indeed serving as inspiration for MCTS), differs from our description of MCTS in a crucial way: the levels or stages of the tree are never updated together. The algorithm is a one-stage method that calls itself in a recursive fashion starting from $t = T$. When nodes at a particular stage t are updated, the value function approximation for stage $t+1$ has already been fixed; hence, results from the multiarmed bandit literature can be directly applied. Unfortunately, this is not the case for MCTS, where updates to every level of the tree are made at each iteration (because all nodes along a specific path to a leaf node are updated).

5. Driver Behavior on a Ride-Sharing Platform

In this section, we show numerical results of applying primal-dual MCTS on a model of driver behavior on a ride-sharing platform (e.g., Uber and Lyft). Our motivation for studying this problem is because of the importance of incorporating the aspect of driver decisions into fleet simulation models. Such large-scale models of the entire system operations can aid in making *platform-level* decisions, including (1) spatial dynamic pricing for riders (Uber's "surge pricing"), (2) dynamic wages/incentives for drivers (Uber's "earnings boost"), and (3) the integration of autonomous vehicles with traditional human drivers (e.g., in Pittsburgh, Pennsylvania). Because optimal decisions from the driver's point of view intimately depend on parameters (e.g., prices) determined by the platform, we envision that the problem studied here is a crucial building block within a higher-level simulation model. Experimental testing suggests that the new version of MCTS produces deeper trees and reduced sensitivity to the size of the action space.

5.1. MDP Model

The operating region is represented as a connected graph consisting of a set of locations $\mathcal{L} = \{1, 2, \ldots, M\}$. Adjacent locations $i, j \in \mathcal{L}$ are connected by an edge. Let $L_t \in \mathcal{L}$ be the location of the driver at time t. The units of time (decision epoch) are in increments of a "minimum trip length," where a trip between adjacent nodes requires one time increment. A *trip request* is an ordered pair $r = (i, j)$, where i is the starting location and j is the destination location. The status σ_t of the driver can take several forms: "idle," "empty car moving toward destination j," "en route to trip $r = (i, j)$," and "with passenger, moving toward destination j." Respectively, these statuses are encoded as $\sigma_t = (0, \varnothing)$, $\sigma_t = (0, j)$, $\sigma_t = (1, (i, j))$, and $\sigma_t = (2, j)$. The second element of σ_t is interpreted as the current "goal" of the driver.

An idle driver advances through the following sequence of events: (1) at the beginning of period t, the driver observes a random set of requested trips $\mathcal{R}_t$; (2) the decision is made to either accept one of the trips in $\mathcal{R}_t$ *or* reject all of them and move to a new location in a set of locations $\mathcal{L}(L_t)$ surrounding L_t; and (3) the driver's status is updated. If the driver is not idle (i.e., $\sigma_t \neq (0, \varnothing)$), then there is no decision to be made, and the current course is maintained. We assume that the stochastic process describing the sets of trip requests $\{\mathcal{R}_t\}$ is independent across time and that $|\mathcal{R}_t| \leq \mathcal{R}_{\max}$. Thus, the state variable of the MDP is $S_t = (L_t, \sigma_t, \mathcal{R}_t)$, which takes values in a state space $\mathcal{S}$. The set of available decisions is given by

$$\mathcal{A}_{S_t} = \begin{cases} \mathcal{L}(L_t) \cup \mathcal{R}_t, & \text{if} \quad \sigma_t = (0, \varnothing), \\ \varnothing, & \text{otherwise.} \end{cases}$$

Suppose that there is a well-defined choice of a shortest path between any two locations i and j given by a sequence of locations along the path $p(i,j) = (p_k(i,j))_{k=1}^{d(i,j)} \in \mathcal{L}^{d(i,j)}$, where $p_k(i,j) \in \mathcal{L}$ is the kth location along the path and $d(i,j)$ is the distance of the shortest path between i and j. Naturally, $p_{d(i,j)}(i,j) = j$. Let $a_t \in \mathcal{A}_{S_t}$ be the decision made at time t. The transition model of the driver's location is

$$L_{t+1} = f_L(S_t, a_t) = \begin{cases} p_1(L_t, a_t), & \text{if} \quad \sigma_t = (0, \varnothing),\ a_t \in \mathcal{L}(L_t), \\ p_1(L_t, i), & \text{if} \quad \sigma_t = (0, \varnothing),\ a_t = (i,j) \in \mathcal{R}_t \text{ or } \sigma_t = (1, (i,j)), \\ p_1(L_t, j), & \text{if} \quad \sigma_t = (0, j) \text{ or } \sigma_t = (2, j). \end{cases}$$

Similarly, we can write the transition model for the driver's status as

$$\sigma_{t+1} = f_\sigma(S_t, a_t) = \begin{cases} (0, \varnothing), & \text{if} \quad [\sigma_t = (0, j) \text{ or } \sigma_t = (2, j)] \text{ and } d(L_t, j) = 1, \\ (0, \varnothing), & \text{if} \quad \sigma_t = (0, \varnothing),\ a_t \in \mathcal{L}(L_t),\ d(L_t, j) = 1, \\ (0, j), & \text{if} \quad \sigma_t = (0, \varnothing),\ a_t \in \mathcal{L}(L_t),\ d(L_t, j) > 1, \\ (1, (i,j)), & \text{if} \quad [\sigma_t = (0, \varnothing),\ a_t = (i,j) \in \mathcal{R}_t \text{ or } \sigma_t = (1, (i,j))] \text{ and } d(L_t, i) > 1, \\ (2, j), & \text{if} \quad [\sigma_t = (0, \varnothing),\ a_t = (i,j) \in \mathcal{R}_t \text{ or } \sigma_t = (1, (i,j))] \text{ and } d(L_t, i) = 1, \\ (2, j), & \text{if} \quad \sigma_t = (2, j),\ d(L_t, j) > 1. \end{cases}$$

Suppose that the base fare is w_{base} and that the customer pays a random location-dependent $w_{\text{dist}}(i)$ per unit distance traveled. The driver is profit maximizing and the contribution function is revenue with per mile travel costs c (e.g., gas, vehicle depreciation) subtracted whenever the driver moves locations:

$$c_t(S_t, a_t) = \left[w_{\text{base}} + w_{\text{dist}}(i)\, d(i,j)\right] \cdot \mathbf{1}_{\{\sigma_t = 0,\, a_t = (i,j) \in \mathcal{R}_t\}} - c \cdot \mathbf{1}_{\{L_t \neq f_L(L_t, a_t)\}}.$$

The objective function is the previously stated (1), with $c_t(S_t, \pi(S_t))$ replacing the contribution function

$c_t(S_t, \pi(S_t), W_{t+1})$. Let $s = (l, \sigma, \mathcal{R})$, and the corresponding Bellman optimality condition is

$$V_t^*(s) = \max_{a \in \mathcal{A}_s} \mathbf{E}\big[c_t(s,a) + V_{t+1}^*(f_L(s,a), f_\sigma(s,a), \mathcal{R}_{t+1})\big], \quad \text{for all } s \in \mathcal{S},\ t \in \mathcal{T},$$
$$V_T^*(s) = 0, \quad \text{for all } s \in \mathcal{S}.$$

Intuitively, the aim of the driver is to position the vehicle in the city and accept trip requests so that revenues can be collected without significant travel costs.

5.2. Numerical Results

Our problem setting is a region in New Jersey, and the experiments are performed on a data set of all trips taken in one day throughout the state on a particular taxi service. We consider the situation where each unit of time corresponds to 15 minutes, that is, the time it takes to travel the distance between any two adjacent locations. The driver is assumed to work for 5 hours a day, giving us $T = 20$. Over this time horizon, the data set contains a total of 850 trips. The graph is built using realistic geography; each location in our model represents a 0.5×0.5 square mile area over a region of approximately 40×60 square miles in New Jersey. At each time step, the requests shown to the driver in the next time period $\mathcal{R}_{t+1}$ are sampled uniformly from the data set. The fare parameters are $w_{\text{base}} = 2.40$ and $c = 0.05$. For most locations i, we set $w_{\text{dist}}(i) = 0.25$, but for some high-density locations, we mimic surge pricing by letting $w_{\text{dist}}(i)$ be randomly drawn between 0.25 and 5.0. We created a range of MDP problem instances of varying difficulty by changing the size of the action space; these are labeled `D10`, `D15`, ..., `D100`, where `D`x is a problem with x actions per period.[2] For instances `D`x with $x \leq 50$, we used an action space of $R_{\max} = x$ sampled trips, whereas for $x > 50$, we used an action space comprising $R_{\max} = 50$ sampled trips and $x - 50$ nearby locations for relocation with an empty car.

Given the size of the problem, standard MDP techniques for computing the optimal policy are intractable, even with the availability of a simulator. On each of the 19 problem instances `D`x, we compare the performances (total profit) of four policies:

- *Closest-E.* The first policy, called *closest-explore*, is designed to be simple so that it can be used as the default policy in tree search. A naive strategy is for the driver to simply accept the *closest trip* in order to minimize the cost of driving to the starting location of the trip. This is motivated by the *closest driver* policy for the case where drivers are fully controlled by the platform (Ozkan and Ward 2017). We added an exploration component to this policy so that, 10% of the time, a random trip is selected. We did not allow the policy to reject all trips and move to a nearby location; thus, even for $x > 50$, the policy will always return a trip.
- *S-RH.* The next policy we consider is a *sampled rolling horizon* (S-RH) approach, much in the spirit of the resolving/reoptimization policies found in Chand et al. (2002), Jasin and Kumar (2012), Chen and Farias (2013), and Bertsimas et al. (2017), where a deterministic approximation of the future is solved in each period, and the first decision is implemented. However, in many of these models, certain aspects of the transition model are used in the deterministic model (e.g., stochastic elements are replaced with their means). Our approach, within the black-box simulator assumption, where the transition model is not known, solves a deterministic dynamic program given a *sample* of future ride requests and then implements the first decision.
- *UCT.* The *upper confidence trees* (UCT) algorithm of Kocsis and Szepesvári (2006) is our "baseline" version of MCTS. The selection policy π^{s} is chosen to be UCB1, as given in (9). The default policy π^{d} is the closest-E policy discussed earlier. The UCT policy is run for 100 iterations per time period. Because we limited the number of iterations per period, we set $\lambda^n = 0$ and allowed expansions on every iteration: $\mathcal{N}_x = \mathcal{N}_y = \mathbb{N}$.
- *PD and PD-0.* Lastly, we show two versions of the primal-dual version of MCTS proposed in this paper, one with and one without the penalty, PD and PD-0, respectively. For both versions, any parameters that overlap with UCT's parameters are kept the same (see above). In particular, 100 iterations are used per time period, and we again use the closest-E default policy. For PD, the penalty function in the information relaxation is computed by approximating the value function associated with the closest-E policy: we build regression models to the results of many simulations of the default policy to obtain $\nu_\tau \approx V_\tau^{\pi^{\text{d}}}$. The expected value in (6) is estimated using an unbiased sample average approximation with five sampled sets of trip requests; see Brown et al. (2010, proposition 2.3(iv)), which shows the validity of this method. Full details of the regression are given in Appendix B. Note that these value function approximations are not necessarily realizable by any particular policy and are simply approximations of $V_t^{\pi^{\text{d}}}$, similar to the approach of Haugh and Kogan (2004). A more direct approach, along the lines of Andersen and Broadie (2004), is to run *online* simulations of the default policy whenever the penalty is computed; however, this would dramatically increase the computation time of the tree search. See Brown et al. (2010) and Desai et al. (2012) for further discussion on these approaches. Furthermore, π^{a} samples each unexpanded action with probability 0.1, and the step size $\alpha^n(\mathbf{x}_t, a_t)$ is chosen to be $1/l^n(\mathbf{x}_t, a_t)$.

Each of the four policies was run 50 times on each of the 19 problem instances. As discussed earlier, the three tree search methods are compared using a fixed number of iterations. Table 1 shows the average profits along with standard errors for each combination. We then estimated upper bounds based on the dual approach for each instance. Figure 4 shows the fraction of the upper bound achieved by each of the three look-ahead policies and the associated 95% confidence intervals for D10,D20,...,D100 (we exclude half the benchmark problems and the closest-E results from Table 1 from the plot for presentation purposes).

5.3. Discussion

The relatively good performance of the four look-ahead policies (PD, PD-0, S-RH, and UCT), as shown in Table 1 and Figure 4, demonstrates that a non-myopic approach is worthwhile for our class of MDPs. Although closest-E performs poorly on its own, it is clear that it can serve as a useful default policy for a tree search algorithm (PD, PD-0, or UCT). Figure 5 illustrates the percentage improvement of the three tree search methods over closest-E, the default policy for all three, via a simple ratio estimate (ratio of the means) across the problem instances. We notice that the tree search methods start with approximately 100% improvement for the smaller problems D10 and D15. However, we see that the penalized PD algorithm is best equipped to handle the larger problems.

The same trend can be seen when directly comparing PD versus either PD-0 or UCT in Figure 4. For problems with fewer than 30 actions, the methods behave similarly, with UCT even occasionally outperforming the other two. This is not necessarily surprising because all methods are able to explore the tree in some depth when the number of actions is small. When there are 60 actions or more, the ability to ignore actions via the dual bounds (and thus reduce the tree's branching factor) seems to become particularly important, and the difference in performance becomes noticeable. For most cases (with a few exceptions), PD-0 outperforms UCT, suggesting that even a loose upper bound can be beneficial.

Next, we comment on the performance of S-RH versus UCT seen in Figure 4. In smaller problems, UCT is superior, but its advantage over S-RH degrades quickly as the problem size increases. The crucial point seems to be that with limited iterations and a large action space, UCT is unable to explore deep decision trees, so the weaknesses of the default policy become pronounced. By contrast, the information relaxation approach of S-RH allows it to explore a decision tree until the end of the horizon (albeit a tree with an incomplete view of the future). These observations match the findings of Bertsimas et al. (2017), who also provided comparisons showing that a rolling horizon approach works better than an MCTS approach on problems with large action spaces. The takeaway here is that the MCTS method can be augmented with sampled dual bounds to become competitive (and even outperform) a particular rolling horizon method.

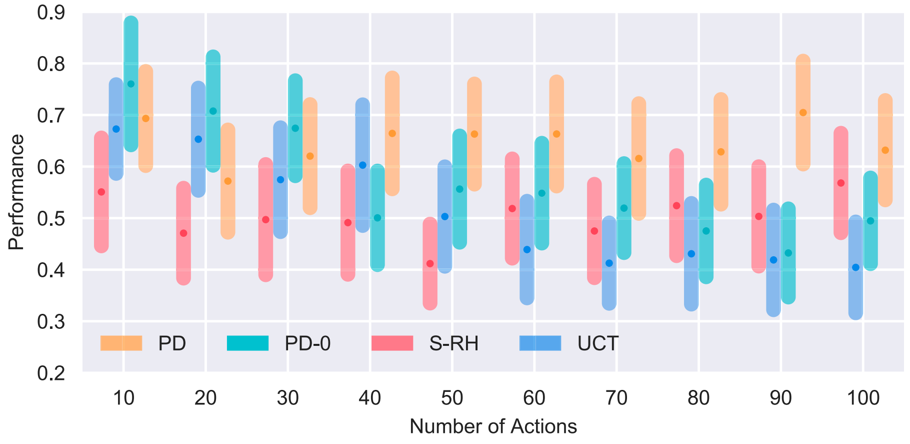
The central processing unit (CPU) time required for one iteration of UCT is very low, at around 0.005 second, whereas the deterministic dynamic programs solved for the information relaxation bound in PD

Table 1. Profit Statistics over 50 Independent Runs of Each Policy

	PD		PD-0		UCT		S-RH		Closest-E	
Instance	Mean	SE	Mean	SE	Mean	SE	Mean	SE	Mean	SE
D10	124.41	8.37	136.44	21.21	120.73	7.88	98.84	9.58	61.54	1.83
D15	128.64	9.31	115.35	19.60	131.57	10.48	101.19	9.20	61.84	1.74
D20	108.09	9.56	133.75	19.87	123.44	9.56	88.99	8.41	59.95	1.79
D25	122.77	9.51	127.28	20.48	112.99	9.38	108.39	10.39	58.40	1.06
D30	121.82	10.02	132.45	18.10	112.84	10.09	97.63	10.72	63.05	2.13
D35	135.88	10.86	123.95	19.28	107.13	9.88	90.03	8.53	62.40	3.10
D40	135.40	11.18	102.02	18.50	122.85	12.17	100.12	10.42	61.03	1.65
D45	139.66	11.13	115.66	20.72	106.73	9.60	110.46	9.76	60.74	1.61
D50	139.57	10.43	117.10	21.69	105.90	10.36	86.69	8.23	64.12	3.45
D55	123.40	10.31	120.27	21.40	109.07	10.90	98.20	8.89	61.00	1.92
D60	141.16	10.96	116.72	20.60	93.44	10.18	110.38	10.45	61.10	1.77
D65	143.54	11.18	98.41	18.27	98.51	9.16	100.76	9.81	60.67	1.62
D70	133.51	11.77	112.65	18.72	89.50	8.62	103.02	10.03	62.71	2.40
D75	141.71	11.87	108.62	20.84	91.24	10.22	118.41	11.75	63.08	2.21
D80	137.60	11.34	104.02	19.47	94.32	10.90	114.75	10.84	61.86	2.04
D85	118.68	11.59	118.17	19.07	106.94	10.77	95.75	9.69	61.35	1.79
D90	155.02	11.20	95.13	18.81	92.16	10.85	110.73	10.82	62.58	2.14
D95	141.02	11.38	109.53	19.94	86.74	8.78	114.61	10.30	60.16	1.37
D100	139.39	10.85	109.15	18.34	89.23	9.92	125.36	10.86	62.30	1.94

Note. SE, standard error.

Figure 4. (Color online) The 95% Confidence Intervals of % Optimality



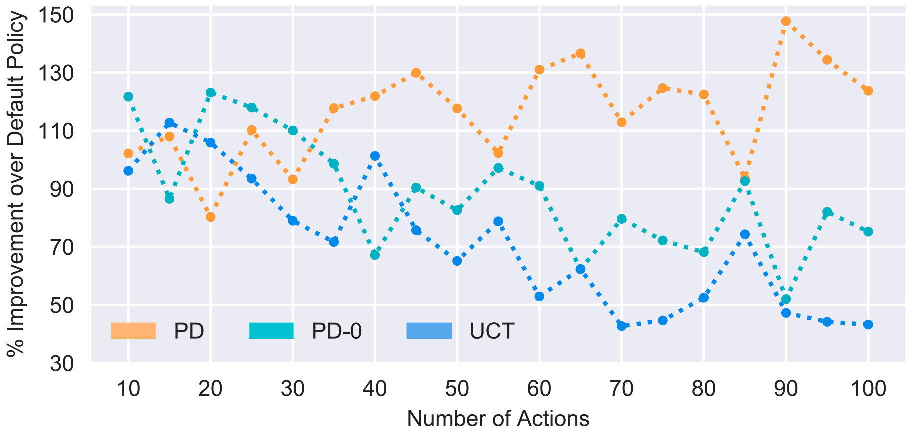
and PD-0 increase the CPU time to approximately 0.035 second per iteration (assuming that the penalty function has been precomputed offline).[3] Certainly, the proposed method is not competitive with UCT in terms of CPU time, and in general, its effectiveness when measured in terms of CPU time will be highly problem dependent. It is important to note that our experimental setup has a fast default policy along with a fast simulator for generating trip request trajectories (because it simply samples from a data set). In other words, the *fixed cost* per iteration of MCTS is low, so the time needed to compute the information relaxation computation dominates the total CPU time.

However, we anticipate that in real, practical scenarios where the "fixed cost per iteration of MCTS" might be large—say, either (1) the default policy is complex and expensive to evaluate or (2) sampling from the simulator is expensive—our proposed dual bound approach can be especially valuable because the overhead of computing dual bounds is relatively diminished. In these situations where the dual bound approach is warranted, the CPU time required for computing the upper bounds can be further mitigated via parallelization. For example, several search threads can be simultaneously executed on the same tree, as is done in Silver et al. (2016) (with special care taken for the backpropagation steps). Alternatively, we can use a standard tree search with the dual bounds updated in a separate *background* process. Because the primary focus of this paper is to introduce and analyze a new algorithmic strategy, we leave the investigation of more sophisticated implementation details to future work.

6. Conclusion

In this paper, we study a new algorithm called *primal-dual MCTS* that attempts to address (1) the fact that convergence of MCTS requires the full expansion of the decision tree and (2) the challenge of applying MCTS on problems with large action spaces (Bertsimas et al. 2017). To do so, we introduce the idea of using samples of an information relaxation upper bound to guide the tree expansion procedure of standard MCTS. It is shown that primal-dual MCTS converges to the optimal action at the root node, even if the entire tree is not fully expanded in the limit (an

Figure 5. (Color online) Improvement of Tree Search Policy over Default Policy



assumption typically required by MCTS convergence results). We also introduce the application of optimizing a single driver operating on a ride-sharing network subject to a real data set of taxi rides occurring over a day in New Jersey. The empirical results indicate that the primal-dual MCTS approach outperforms a UCT policy, a sampled rolling horizon policy, and a simple myopic policy based on the closest trip.

Acknowledgments
The authors are very grateful to the area editor, the associate editor, and the reviewers for their valuable comments throughout the review process. Their detailed readings of the paper have helped to improve its presentation, clarity, and notation in many ways; the review team also helped to identify several issues that were overlooked. The first two authors contributed equally to this work.

Appendix A. Proofs

Proof of Lemma 1. First, because all iterations of MCTS begin at the root node, it is clear that $\mathbf{x}_0 \in \mathcal{X}^\infty$ and $v^n(\mathbf{x}_0) \to \infty$ (the base case). We assume that for some t, part (a) holds for any $\mathbf{x}_t \in \mathcal{X}_t$ (induction hypothesis). To prove the result by induction, we must show the following: (1) for any $\mathbf{y}_t \in \mathcal{Y}_t$, parts (b) and (c) *of the lemma* hold, and (2) for any $\mathbf{x}_{t+1} \in \mathcal{X}_{t+1}$, part (a) holds. We show the proof of (1) for part (b) of the lemma.

Let $\mathbf{y}_t = (\mathbf{x}_t, a_t) \in \mathcal{Y}_t$, where $\mathbf{P}(\mathbf{y}_t \in \mathcal{Y}^\infty) > 0$. Define $\mathcal{V}(\mathbf{x}_t)$ to be the random set of iterations at which $\mathbf{x}_t$ is visited. We also let $\eta(\mathbf{y}_t)$ be the iteration number at which $\mathbf{y}_t$ is added to the tree. Note that a subset of the event $\{\mathbf{y}_t$ is visited on iteration $n\}$ is $\{n \in \mathcal{V}(\mathbf{x}_t), v^n(\mathbf{x}_t) \notin \mathcal{N}_x, \pi^{\mathrm{s}}(\mathbf{x}_t, \mathcal{T}^{n-1}) = \mathbf{y}_t\}$, which is interpreted as (1) a visit to $\mathbf{x}_t$ on an iteration that is not an expansion step and (2) the independent call to the selection policy π^{s} resulting in $\mathbf{y}_t$. The two events are not equal because another way to visit $\mathbf{y}_t$ is when $v^n(\mathbf{x}_t) \in \mathcal{N}_x$, but the potential expansion step is skipped after the dual bound update, and $\mathbf{y}_t$ is selected afterward.

Consider a fixed $j > 0$ at which $\mathbf{y}_t$ is added to the tree and an iteration $n > j$. Let $\mathcal{F}^{n-1} = \sigma(\mathcal{T}^0, \mathcal{T}^1, \ldots, \mathcal{T}^{n-1})$. By the fact that $\mathbf{x}_t$ exists in the tree by iteration n (if $\mathbf{y}_t$ is added at j, then $\mathbf{x}_t$ must have been added prior to j), we can repeatedly apply Assumption 1, (a) and (b), down the tree to conclude that the probability of $n \in \mathcal{V}(\mathbf{x}_t)$ can be lower-bounded by some positive constant (note that by Case 2 of the expansion step, we know that only states that can be reached with positive probability are added). Applying Assumption 1(a) and the independence of the sampling policy π^{s}, we can deduce that whenever $v^{n-1}(\mathbf{x}_t) + 1 \notin \mathcal{N}_x$, there exists some constant $\delta > 0$, where

$$\mathbf{P}\big(n \in \mathcal{V}(\mathbf{x}_t), v^n(\mathbf{x}_t) \notin \mathcal{N}_x, \pi^{\mathrm{s}}\big(\mathbf{x}_t, \mathcal{T}^{n-1}\big) = \mathbf{y}_t \,|\, \eta(\mathbf{y}_t) = j, \mathcal{F}^{n-1}\big) > \delta.$$

Because $v^{n-1}(\mathbf{x}_t) + 1 \notin \mathcal{N}_x$ occurs infinitely often by the induction hypothesis, it follows that

$$\sum_{n=j+1}^{\infty} \mathbf{P}\big(n \in \mathcal{V}(\mathbf{x}_t),\, v^n(\mathbf{x}_t) \notin \mathcal{N}_x,\, \pi^{\mathrm{s}}\big(\mathbf{x}_t, \mathcal{T}^{n-1}\big) = \mathbf{y}_t \,|\, \eta(\mathbf{y}_t) = j, \mathcal{F}^{n-1}\big) = \infty.$$

Applying the extended Borel–Cantelli lemma (see Breiman 1992, corollary 5.29) with the probability measure $\mathbf{P}(\cdot \,|\, \eta(\mathbf{y}_t) = j)$, we can conclude that $\mathbf{y}_t$ is visited infinity often; that is, $v^n(\mathbf{y}_t) \to \infty$ a.e. on the event $\{\eta(\mathbf{y}_t) = j\}$. The event that $\mathbf{y}_t$ is expanded can be written $\{\mathbf{y}_t \in \mathcal{Y}^\infty\} = \cup_j \{\eta(\mathbf{y}_t) = j\}$, so the preceding argument can be repeated for each j, and we conclude that $v^n(\mathbf{y}_t) \to \infty$ on $\{\mathbf{y}_t \in \mathcal{Y}^\infty\}$. We omit the proofs of (1) part (c) and (2) because they can be shown using analogous arguments and Assumption 1(c). □

Proof of Lemma 2. Consider any ω in the event defined in the statement of the lemma, and fix this ω throughout this proof. Suppose for this sample path ω that there is an optimal state–action node that is never expanded; that is, $\mathbf{y}_t^* = (\mathbf{x}_t, a_t^*) \in \tilde{\mathcal{Y}}^\infty$. By Lemma 1, we know that $l^n(\mathbf{y}_t^*) \to \infty$, and let us denote the set of iterations for which $\mathbf{y}_t^* \in \pi^{\mathrm{a}}(\mathbf{x}_t, \mathcal{T}^{n-1})$ by $\mathcal{N}_x(\mathbf{x}_t, \mathbf{y}_t^*) \subseteq \mathcal{N}_x$ (i.e., iterations for which it is a candidate action). Then because this optimal action was never expanded, it must hold true that for any iteration $n \in \mathcal{N}_x(\mathbf{x}_t, \mathbf{y}_t^*)$, the dual bound approximation does not exceed the current value approximation (i.e., current best):

$$\bar{u}^n(\mathbf{y}_t^*) \le \big(1 - \lambda^{n-1}\big)\, \tilde{V}^{n-1}(\mathbf{x}_t) + \lambda^{n-1} \max_{\mathbf{y}_t \in \mathcal{Y}^{n-1}(\mathbf{x}_t)} \bar{Q}^{n-1}(\mathbf{y}_t).$$

Because $|\mathcal{N}_x(\mathbf{x}_t, \mathbf{y}_t^*)| = \infty$, we may pass to the limit and use conditions (a) and (b) in the statement of the lemma along with Assumption 1(e) to obtain $u^\nu(\mathbf{y}_t^*) \le \max_{\mathbf{y} \in \mathcal{Y}^\infty(\mathbf{x}_t)} Q^*(\mathbf{y})$. Because we have made the assumption that $\mathcal{Y}^\infty(\mathbf{x}_t)$ does not contain an optimal action,

$$u^\nu(\mathbf{y}_t^*) \le \max_{\mathbf{y}_t \in \mathcal{Y}^\infty(\mathbf{x}_t)} Q^*(\mathbf{y}_t) < V^*(\mathbf{x}_t). \tag{A.1}$$

By contrast, applying Proposition 1 to the optimal state–action pair $\mathbf{y}_t^*$, we see that

$$u^\nu(\mathbf{y}_t^*) \ge Q^*(\mathbf{y}_t^*) = V^*(\mathbf{x}_t),$$

a contradiction with (A.1). Thus, we conclude that our original assumption was incorrect, and it must be the case that an optimal state–action node is expanded in the limit for our fixed ω. Because the sample path ω was arbitrary, the conclusion holds for any $\omega \in A$. □

Proof of Theorem 2. *Before moving on to the main proof, we state a result that will be useful later on. Let $\mathbf{y}_t \in \mathcal{Y}$. For any $k_0 \in \mathbb{N}$, let $\alpha_{k_0}^k(\mathbf{y}_t)$ be a step size where*

$$\sum_{k=k_0+1}^{\infty} \alpha_{k_0}^k(\mathbf{y}_t) = \infty \quad \textit{and} \quad \sum_{k=k_0+1}^{\infty} \Big(\alpha_{k_0}^k(\mathbf{y}_t)\Big)^2 < \infty \quad \textit{a.s.}$$

For any initial value $\bar{Q}_{k_0}^{k_0}(\mathbf{y}_t)$, consider the iteration

$$\bar{Q}_{k_0}^{k+1}(\mathbf{y}_t) = \bar{Q}_{k_0}^{k}(\mathbf{y}_t) + \alpha_{k_0}^{k+1}(\mathbf{y}_t)\Big[\bar{V}^k((\mathbf{y}_t, S_{t+1})) - \bar{Q}_{k_0}^{k}(\mathbf{y}_t)\Big], \quad \text{for } k \ge k_0, \tag{A.2}$$

where $S_{t+1} \sim \mathbf{P}(\cdot \mid \mathbf{y}_t)$ and $\bar{V}^k(\mathbf{y}) \to V^*(\mathbf{y})$ a.s. for every $\mathbf{y}$. We can write

$$\begin{aligned}\bar{V}^k((\mathbf{y}_t, S_{t+1})) - \bar{Q}^k_{k_0}(\mathbf{y}_t) &= Q^*(\mathbf{y}_t) - \bar{Q}^k_{k_0}(\mathbf{y}_t) + V^*((\mathbf{y}_t, S_{t+1})) - Q^*(\mathbf{y}_t) \\ &\quad + \bar{V}^k((\mathbf{y}_t, S_{t+1})) - V^*((\mathbf{y}_t, S_{t+1})).\end{aligned}$$

Because $\bar{V}^k((\mathbf{y}_t, S_{t+1})) - V^*((\mathbf{y}_t, S_{t+1})) \to 0$ a.s. and $\mathbf{E}[V^*((\mathbf{y}_t, S_{t+1})) - Q^*(\mathbf{y}_t) \mid \mathbf{y}_t] = 0$, it follows by Kushner and Yin (2003, theorem 2.4) that

$$\bar{Q}^k_{k_0}(\mathbf{y}_t) \to Q^*(\mathbf{y}_t) \quad \text{a.s.} \tag{A.3}$$

The conditions of Kushner and Yin (2003, theorem 2.4) are not difficult to check given our setting with bounded contributions.

With this auxiliary result in mind, we move on to the proof of Theorem 2. We aim to show via induction that, for any $\mathbf{x}_{t+1} \in \mathcal{X}_{t+1}$,

$$\bar{V}^n(\mathbf{x}_{t+1}) \to V^*(\mathbf{x}_{t+1})\, \mathbf{1}_{\{\mathbf{x}_{t+1} \in \mathcal{X}^\infty_{t+1}\}} \quad \text{a.s.} \tag{A.4}$$

This means that convergence to the optimal value function occurs on the event that the node is expanded. Otherwise, the value function approximation stays at its initial value of zero. We proceed by backward induction from $t+1 = T$. The base case is clear by (11) and because $V^*(\mathbf{x}_T) = 0$ for $\mathbf{x}_T \in \mathcal{X}_T$. The induction hypothesis is that (A.4) holds for an arbitrary $t+1$.

In order to complete the inductive step, our first goal is to show that the state–action value function converges on the event that the node $\mathbf{y}_t = (\mathbf{x}_t, a_t) \in \mathcal{Y}_t$ is expanded:

$$\bar{Q}^n(\mathbf{y}_t) \to Q^*(\mathbf{y}_t)\, \mathbf{1}_{\{\mathbf{y}_t \in \mathcal{Y}^\infty_t\}} \quad \text{a.s.} \tag{A.5}$$

We will give an ω-wise argument. Let us fix $\omega \in \Omega$. If $\mathbf{y}_t \notin \mathcal{Y}^\infty_t$, then $\bar{Q}^n(\mathbf{y}_t)$ is never updated and thus converges to zero. Now suppose that the node is expanded; that is, $\mathbf{y}_t \in \mathcal{Y}^\infty_t$. Let $\alpha^n_v(\mathbf{y}_t) = 1/v^n(\mathbf{y}_t)\, \mathbf{1}_{\{\mathbf{y}_t \in \mathbf{x}^n_{\tau_s}\}}$, where the notation $\mathbf{y} \in \mathbf{x}^n_{\tau_s}$ is used to indicate that $\mathbf{y}$ is visited on iteration n. Thus, we can rewrite (12) and (15) in the form of a stochastic approximation step as follows:

$$\bar{Q}^n(\mathbf{y}_t) = \bar{Q}^{n-1}(\mathbf{y}_t) + \alpha^n_v(\mathbf{y}_t)\big[\bar{V}^n\big(\pi^{\mathrm{s}}(\mathbf{y}_t, \mathcal{T}^{n-1})\big) - \bar{Q}^n(\mathbf{y}_t)\big]. \tag{A.6}$$

We will analyze the tail of this iteration. Because $v^n(\mathbf{y}_t) \to \infty$ by Lemma 1 and $|\mathcal{S}|$ is finite, it is clear that there exists an iteration N^* (depending on ω) after which all state nodes $\mathbf{x}_{t+1} = (\mathbf{y}_t, s_{t+1})$, where s_{t+1} is reachable with positive probability, are expanded. By Assumption 1(c), the tail of the iteration (A.6) starting at N^* is equivalent to

$$\begin{aligned}&\bar{Q}^{n+1}_{N^*}(\mathbf{y}_t) \\ &\quad = \bar{Q}^n_{N^*}(\mathbf{y}_t) + \alpha^{n+1}_v(\mathbf{y}_t)\,\big[\bar{V}^n((\mathbf{y}_t, S_{t+1})) - \bar{Q}^n_{N^*}(\mathbf{y}_t)\big], \quad \text{for } n \ge N^*,\end{aligned}$$

where $\bar{Q}^{N^*}_{N^*}(\mathbf{y}_t) = \bar{Q}^{N^*}(\mathbf{y}_t)$. Define $\eta^i(\mathbf{y}_t)$ as the ith iteration for which $\mathbf{y}_t$ is visited. By Lemma 1, we know that $(\eta^1(\mathbf{y}_t), \eta^2(\mathbf{y}_t), \eta^3(\mathbf{y}_t), \ldots)$ is a subsequence of $(1, 2, 3, \ldots)$ that goes to ∞. Let i^* be the smallest i such that $\eta^i(\mathbf{y}_t) > N^*$. Hence,

$$\sum_{n=N^*+1}^{\infty} \alpha^n_v(\mathbf{y}_t) = \sum_{i=i^*}^{\infty} \alpha^{\eta^i(\mathbf{y}_t)}_v(\mathbf{y}_t) = \sum_{i=i^*}^{\infty} 1/i = \infty.$$

Similarly, $\sum_{n=N^*+1}^{\infty} (\alpha^n_v(\mathbf{y}_t))^2 < \infty$. Because $(\mathbf{y}_t, S_{t+1}) \in \mathcal{X}^\infty_{t+1}$ for every realization of S_{t+1}, by the induction hypothesis, it holds that $\bar{V}^n(\mathbf{y}_t, S_{t+1}) \to V^*(\mathbf{y}_t, S_{t+1})$. Thus, by the auxiliary result stated in (A.3), it follows that $\bar{Q}^n_{N^*}(\mathbf{y}_t) \to Q^*(\mathbf{y}_t)$, and so we can conclude that

$$\bar{Q}^n(\mathbf{y}_t) \to Q^*(y_t) \tag{A.7}$$

for when our choice of ω is in $\{\mathbf{y}_t \in \mathcal{Y}^\infty_t\}$, which proves (A.5).

The next step is to examine the dual upper bounds. Analogously, we would like to show that

$$\bar{u}^n(\mathbf{y}_t) \to u^\nu(\mathbf{y}_t)\, \mathbf{1}_{\{\mathbf{y}_t \in \tilde{\mathcal{Y}}^\infty_t\}} \quad \text{a.s.} \tag{A.8}$$

The averaging iteration (10) can be written as

$$\bar{u}^n(\mathbf{y}'_t) = \bar{u}^{n-1}(\mathbf{y}'_t) - \alpha^n(\mathbf{y}'_t)\big[\bar{u}^{n-1}(\mathbf{y}'_t) - \hat{u}^n(\mathbf{y}'_t)\big],$$

where $\mathbf{E}[\hat{u}^n(\mathbf{y}'_t)] = u^\nu(\mathbf{y}'_t)$. There is no bias term here. Under Lemma 1, Assumption 1(d), and our finite model and bounded contributions, an analysis similar to the case of (A.5) allows us to conclude (A.8).

Finally, we move on to completing the inductive step, that is, (A.4) with t replacing $t+1$. Consider $\mathbf{x}_t \in \mathcal{X}_t$, and again let us fix an $\omega \in \Omega$. If $\mathbf{x}_t \notin \mathcal{X}^\infty_t$, then there are no updates, and we are done, so consider the case where $\mathbf{x}_t \in \mathcal{X}^\infty_t$. The conditions of Lemma 2 are verified by (A.5) and (A.8), so the lemma implies that an optimal action $a^*_t \in \arg\max_{a \in \mathcal{A}} Q^*(\mathbf{x}_t, a)$ is expanded. Consequently, it follows by (A.5) that

$$\max_{\mathbf{y}_t \in \mathcal{Y}^\infty(\mathbf{x}_t)} \bar{Q}^n(\mathbf{y}_t) \to Q^*(\mathbf{y}^*_t) = V^*(\mathbf{x}_t).$$

By Assumption 1(e) and the backpropagation update (13), we see that $\bar{V}^n(\mathbf{x}_t) \to V^*(\mathbf{x}_t)$, which proves (A.4). We have shown that all value function approximations converge appropriately for expanded nodes in the random limiting tree $\mathcal{T}^\infty$.

Now we move on to the second part of the theorem, which concerns the action taken at the root node. If the optimal action is unique (i.e., there is separation between the best action and the second- best action), then (A.4) allows us to conclude that the limit of the set of maximizing actions $\arg\max_{\mathbf{y} \in \mathcal{Y}^n(\mathbf{x}_0)} \bar{Q}^n(\mathbf{y})$ is equal to $\arg\max_{\mathbf{y}_0 = (\mathbf{x}_0, a)} Q^*(\mathbf{y}_0)$. However, if uniqueness is not assumed (which we have not), then we may conclude that each accumulation point of $\arg\max_{\mathbf{y} \in \mathcal{Y}^n(\mathbf{x}_0)} \bar{Q}^n(\mathbf{y})$ is an optimal action at the root node:

$$\limsup_{n \to \infty} \arg\max_{\mathbf{y} \in \mathcal{Y}^n(\mathbf{x}_0)} \bar{Q}^n(\mathbf{y}) \subseteq \arg\max_{\mathbf{y}_0 = (\mathbf{x}_0, a)} Q^*(\mathbf{y}_0) \;\; \text{a.s.}$$

The proof is complete. □

Appendix B. Penalty Function

We follow the value function approximation approach of Brown et al. (2010) to design the dual penalty; see (6) and (7). Specifically, we let ν_τ be an approximation of the value function associated with the closest-E policy. The approach for computing this is relatively straightforward; however, choosing the feature vector for approximation required some experimentation because the dimension of the state variable depends on the number of available trips. The steps are as follows:

1. We randomly sample initial driver conditions $s_i = (t_i, l_i, \sigma_i, \mathcal{R}_i)$ (i.e., time, location, idle status, and available trip requests).

2. We run the closest-E starting from each s_i until the end of the problem horizon. Let the cumulative profit of the ith simulation be denoted y_i.

3. Let $\mathcal{R}_i = \{(a_{ij}, b_{ij}, c_{ij}, d_{ij})\}_j$ be the set of trips offered for s_i, where (a_{ij}, b_{ij}) are the coordinates of the starting location for trip j and (c_{ij}, d_{ij}) are the coordinates of the ending location for trip j. We compute the centroids of the starting locations and the ending locations: $(\bar{a}_i, \bar{b}_i)$ and $(\bar{c}_i, \bar{d}_i)$. Using the driver location l_i as a reference, we then compute the angle θ_i^{start} and distance r_i^{start} to $(\bar{a}_i, \bar{b}_i)$. Similarly, we can define $(\theta_i^{\text{end}}, r_i^{\text{end}})$ to be the angle and distance to $(\bar{c}_i, \bar{d}_i)$ starting from l_i. The compact state is defined as $s_i' = (t_i, l_i, \theta_i^{\text{start}}, r_i^{\text{start}}, \theta_i^{\text{end}}, r_i^{\text{end}})$.

4. The feature vector for each state $\phi(s_i)$ consists of all degree 2 (or less) monomials generated by the components of s_i'. The value function approximation is then generated by regressing the y_i values against the features $\phi(s_i)$.

Endnotes

[1] Because MCTS is a look-ahead technique, it is typically applied in large-scale planning settings, where a simulator is available and at least some aspects of the model are known.

[2] The smaller problems are certainly more realistic in the context of a ride-sharing platform, but the larger instances allow for comprehensive numerical testing of the various algorithms.

[3] On a machine with a 4-GHz Intel Core i7 processor and 8 GB of random-access memory for the `D100` instance.

References

Al-Kanj L, Powell WB, Bouzaiene-Ayari B (2016) The information-collecting vehicle routing problem: Stochastic optimization for emergency storm response. Preprint, submitted May 18, https://arxiv.org/abs/1605.05711.

Andersen L, Broadie M (2004) Primal-dual simulation algorithm for pricing multidimensional American options. *Management Sci.* 50(9):1222–1234.

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

Auger D, Couëtoux A, Teytaud O (2013) Continuous upper confidence trees with polynomial exploration—consistency. *Proc. Joint Eur. Conf. Machine Learn. Knowledge Discovery Databases* (Springer, Berlin, Heidelberg), 194–209.

Bertsimas D, Griffith JD, Gupta V, Kochenderfer MJ, Mišić VV (2017) A comparison of Monte Carlo tree search and rolling horizon optimization for large-scale dynamic resource allocation problems. *Eur. J. Oper. Res.* 263(2):664–678.

Breiman L (1992) *Probability* (Society of Industrial and Applied Mathematics, Philadelphia, PA).

Broadie M, Glasserman P (1997) Pricing American-style securities using simulation. *J. Econom. Dynam. Control.* 21(8–9):1323–1352.

Brown DB, Smith JE (2011) Dynamic portfolio optimization with transaction costs: Heuristics and dual bounds. *Management Sci.* 57(10):1752–1770.

Brown DB, Smith JE (2014) Information relaxations, duality, and convex stochastic dynamic programs. *Oper. Res.* 62(6):1394–1415.

Brown DB, Smith JE, Sun P (2010) Information relaxations and duality in stochastic dynamic programs. *Oper. Res.* 58(4):785–801.

Browne C, Powley E, Whitehouse D, Lucas S, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S (2012) A survey of Monte Carlo tree search methods. *IEEE Trans. Intelligence AI Games*. 4(1):1–49.

Bubeck S, Cesa-Bianchi N (2012) Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations Trends Machine Learn.* 5(1):1–122.

Cazenave T (2009) Nested Monte-Carlo search. *Proc. 21st Internat. Joint Conf. Artificial Intelligence, Pasadena, CA*, 456–461.

Chand S, Hsu V, Sethi S (2002) Forecast, solution, and rolling horizons in operations management problems: A classified bibliography. *Manufacturing Service Oper. Management* 4(1):25–43.

Chang HS, Fu MC, Hu JQ, Marcus SI (2005) An adaptive sampling algorithm for solving Markov decision processes. *Oper. Res.* 53(1):126–139.

Chaslot G, Saito JT, Uiterwijk JWHM, Bouzy B, van den Herik HJ (2006) Monte-Carlo strategies for computer go. *Proc. 18th Belgian-Dutch Conf. Artificial Intelligence*, 83–90.

Chaslot G, Bakkes S, Szita I, Spronck P (2008) Monte-Carlo tree search: A new framework for Game AI. *Proc. 4th Artificial Intelligence Interactive Digital Entertainment Conf.*, 216–217.

Chen Y, Farias VF (2013) Simple policies for dynamic pricing with imperfect forecasts. *Oper. Res.* 61(3):612–624.

Couëtoux A, Hoock JB, Sokolovska N, Teytaud O, Bonnard N (2011) Continuous upper confidence trees. *Proc. Internat. Conf. Learn. Intelligent Optim.*, 433–445.

Coulom R (2007) Efficient selectivity and backup operators in Monte-Carlo tree search. *Comput. Games* 4630:72–83.

Desai VV, Farias VF, Moallemi CC (2012) Pathwise optimization for optimal stopping problems. *Management Sci.* 58(12): 2292–2308.

Gelly S, Silver D (2011) Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence* 175(11): 1856–1876.

Gelly S, Kocsis L, Schoenauer M, Sebag M, Silver D, Szepesvári C, Teytaud O (2012) The grand challenge of computer Go: Monte Carlo tree search and extensions. *Comm. ACM* 55(3):106–113.

Goodson JC, Thomas BW, Ohlmann JW (2016) Restocking-based rollout policies for the vehicle routing problem with stochastic demand and duration limits. *Transportation Sci.* 50(2):591–607.

Haugh MB, Kogan L (2004) Pricing American options: A duality approach. *Oper. Res.* 52(2):258–270.

Hingston P, Masek M (2007) Experiments with Monte Carlo Othello. *IEEE Congress Evolutionary Comput.*, Singapore, 4059–4064.

Jasin S, Kumar S (2012) A re-solving heuristic with bounded revenue loss for network revenue management with customer choice. *Math. Oper. Res.* 37(2):313–345.

Knuth DE, Moore RW (1975) An analysis of alpha-beta pruning. *Artificial Intelligence* 6(4):293–326.

Kocsis L, Szepesvári C (2006) Bandit based Monte-Carlo planning. *Proc. 17th Eur. Conf. Machine Learn.* (Springer, Berlin), 282–293.

Kushner HJ, Yin GG (2003) *Stochastic Approximation and Recursive Algorithms and Applications* (Springer, New York).

Lai G, Wang MX, Kekre S, Scheller-Wolf A, Secomandi N (2011) Valuation of storage at a liquefied natural gas terminal. *Oper. Res.* 59(3):602–616.

Land AH, Doig AG (1960) An automatic method of solving discrete programming problems. *Econometrica* 28(3):497–520.

Maitrepierre R, Mary J, Munos R (2008) Adaptive play in Texas Hold'em Poker. Ghallab M, Spyropoulos CD, Fakotakis N, Avouris N, eds. *Proc. 18th Eur. Conf. Artificial Intelligence (ECAI 2008)* (IOS Press, Amsterdam), 458–462.

Méhat J, Cazenave T (2010) Combining UCT and nested Monte Carlo search for single-player general game playing. *IEEE Trans. Comput. Intelligent AI Games* 2(4):271–277.

Nadarajah S, Margot F, Secomandi N (2015) Relaxations of approximate linear programs for the real option management of commodity storage. *Management Sci.* 61(12):3054–3076.

Nascimento JM, Powell WB (2009) An optimal approximate dynamic programming algorithm for the lagged asset acquisition problem. *Math. Oper. Res.* 34(1):210–237.

Nijssen JPAM (2007) Playing Othello using Monte Carlo. Working paper, Maastricht University, Maastricht, Netherlands.

Osaki Y, Shibahara K, Tajima Y, Kotani Y (2008) An Othello evaluation function based on temporal difference learning using probability of winning. *IEEE Sympos. Comput. Intelligence Games*, 205–211.

Ozkan E, Ward A (2017) Dynamic matching for real-time ridesharing. Working paper, Koç University, Istanbul, Turkey.

Ponsen M, Gerritsen G, Chaslot G (2010) Integrating opponent models with Monte-Carlo tree search in Poker. *Interactive Decision Theory and Game Theory* (Association for the Advancement of Artificial Intelligence, Menlo Park, CA), 37–42.

Robles D, Rohlfshagen P, Lucas SM (2011) Learning non-random moves for playing Othello: Improving Monte Carlo tree search. *IEEE Conf. on Computational Intelligence and Games, Seoul*, 305–312.

Rogers LC (2002) Monte Carlo valuation of American options. *Math. Finance* 12(3):271–286.

Silver D, Veness J (2010) Monte-Carlo planning in large POMDPs. *Adv. in Neural Inform. Processing Systems*, vol 23 (Curran Associates, Red Hook, NY), 2164–2172.

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, et al. (2016) Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Van den Broeck G, Driessens K, Ramon J (2009) Monte-Carlo tree search in Poker using expected reward distributions. *Adv. in Machine Learn.* (Springer, Berlin, Heidelberg), 367–381.

Van Lishout F, Chaslot G, Uiterwijk JWHM (2007) Monte-Carlo tree search in Backgammon. Working paper, University of Liège, Montefiore Institute, Liège, Belgium.

Daniel R. Jiang is an assistant professor in the Department of Industrial Engineering at the University of Pittsburgh. His research interests are in the areas of approximate dynamic programming, reinforcement learning, and Bayesian optimization, with applications in energy, the sharing economy, and public health.

Lina Al-Kanj is an associate research scholar in the Operations Research and Financial Engineering Department at Princeton University. Her research interests include sequential stochastic optimization, look-ahead approximations, and reinforcement learning with applications in energy, communication, and transportation systems.

Warren B. Powell is a professor in the Department of Operations Research and Financial Engineering at Princeton University, where he has taught since 1981. He is the founder and director of CASTLE Labs, which develops models and algorithms in stochastic optimization, with applications in energy systems, transportation, medical research, business analytics, and the laboratory sciences. He pioneered the use of approximate dynamic programming in freight transportation. He is an INFORMS Fellow.