Monte-Carlo Tree Search: To MC or to DP?

Zohar Feldman and **Carmel Domshlak**[1]

Abstract. State-of-the-art Monte-Carlo tree search algorithms can be parametrized with any of the two information updating procedures: MC-backup and DP-backup. The dynamics of these two procedures is very different, and so far, their relative pros and cons have been poorly understood. Formally analyzing the dependency of MC- and DP-backups on various MDP parameters, we reveal numerous important issues that get hidden by the worst-case bounds on the algorithm performance, and reconfirm these findings by a systematic experimental test.

1 INTRODUCTION

Markov decision processes (MDPs) is a standard model for planning under uncertainty [17]. An MDP $\langle S, A, \mathbb{P}, R\rangle$ is defined by a set of states S, a set of state transforming actions A, a stochastic transition function $\mathbb{P} : S \times A \times S \to [0, 1]$, and a reward function $R : S \times A \times S \to \mathbb{R}$. The states are fully observable and, in the finite horizon setting considered here, the rewards are accumulated over some predefined number of steps H. The objective of planning in MDPs is to sequentially choose actions so as to maximize the accumulated reward. The representation of large-scale MDPs can be either declarative or generative, but anyway concise, and allowing for simulated execution of all feasible action sequences, from any state of the MDP. In *online MDP planning*, the agent focuses on its current state s_0 only, deliberates about the set of possible policies from that state onwards and, when interrupted, chooses what action to perform next. In formal analysis of algorithms for online MDP planning, the quality of the action a, chosen for s_0 with H steps-to-go, is assessed in terms of the induced "simple regret", capturing the performance loss that results from taking a and then following an optimal policy π^* for the remaining $H - 1$ steps, instead of following π^* from the beginning [4].

Many popular algorithms for online MDP planning constitute what is called *Monte-Carlo tree search (MCTS)* [21, 16, 15, 7, 6, 19, 22, 14, 10, 12], and adaptations of some of these algorithms are also popular in other settings of sequential decision making, including those with partial state observability and adversarial effects [13, 20, 2, 8, 3]. At a high level, all MCTS algorithms explore the state-space region around s_0 by iteratively (i) simulating an action/state trajectory from s_0, and (ii) using the outcome of that trajectory to update various action-value estimates related to the state-space region of interest, as well as to update the estimate of what action should be best applied at state s_0. In that respect, specific MCTS algorithms differ both in their trajectory rollout strategies, as well as in their rollout-based update strategies.

Recent work substantially advanced our understanding of how the performance of MCTS depends on the specifics of the rollout strategy, as well as on the choice of what pieces of information should be updated based on a given rollout [15, 7, 9]. Recently, however, Keller & Helmert [14] demonstrated empirically that the performance of MCTS also depends to a large extent on *how* the respective updates are being performed. Prior to the work of Keller & Helmert, updates in MCTS algorithms were all based on *MC-backups*, that is, sample averaging updates of the selected action-value estimates. Keller & Helmert showed that modifying a standard MCTS algorithm (such as UCT [15]), by replacing its MC-backups with dynamic programming estimates propagation *a la* Bellman backups in value iteration [1], can substantially affect the performance, and, at least in their experiments, *typically in favor of DP-backups*. Later on, Feldman & Domshlak [12] showed that switching from MC-backups to such *DP-backups* preserves the order-of-magnitude convergence rates of MCTS instances that guarantee exponential rate performance improvement (such as BRUE [9]), and can even allow for proving *somewhat better convergence bounds*. Still, the relative pros and cons of MC- and DP-backups have not been systematically studied so far, and thus are still poorly understood.

This is precisely our contribution in this paper: Using BRUE and MaxBRUE, a pair of state-of-the-art MCTS algorithms that, *ceteris paribus*, use MC-backups and DP-backups, respectively, we study the dynamics of MC- and DP-backups both formally and empirically. Starting with establishing a pair of comparable worst-case bounds on the convergence rates of these two algorithms, we use the analysis behind these bounds to examine specific dependencies of the two algorithms on various MDP parameters, namely the state and action branching factors, the shape of the reward function, and the entropy of the transition function. To our knowledge, our analysis is first of its kind, and it reveals numerous important issues that get hidden by the worst-case bounds due to certain deficiencies in the formal worst-case analysis of MC-backups. In particular, it suggests that MC-backups are less sensitive than DP-backups to the state branching factor, especially when the transition function concentrates on a small number of outcomes, as it is typically the case in practical applications. The various aspects of the analysis are then put on a systematic experimental test, which reconfirms its key findings.

2 BACKGROUND

In what follows, we adopt the notation and pseudo-code convention from [12]. In particular, when considering an MDP $\langle S, A, \mathbb{P}, R\rangle$, its state and action branching factors are respectively denoted by $K = \max_s |A(s)|$ and $B = \max_{s,a} |\{s' \mid \mathbb{P}(s'|s, a) > 0\}|$, $s\langle h\rangle$ denotes state $s \in S$ with h steps-to-go, and $A(s) \subseteq A$ denotes the actions applicable in state s. Some auxiliary notation: The operation of drawing a sample from a distribution $\mathcal{D}$ over set $\aleph$ is denoted by $\sim \mathcal{D}[\aleph]$, $\mathcal{U}$ denotes uniform distribution, and $[\![n]\!]$ for $n \in \mathbb{N}$ denotes the set $\{1, \ldots, n\}$. For a sequence of tuples ρ, $\rho[i]$ denotes the i-th tuple along ρ, and $\rho[i].x$ denotes the value of the field x in that tuple.

[1] Technion, Israel, email: {zoharf@tx,dcarmel@ie}.technion.ac.il

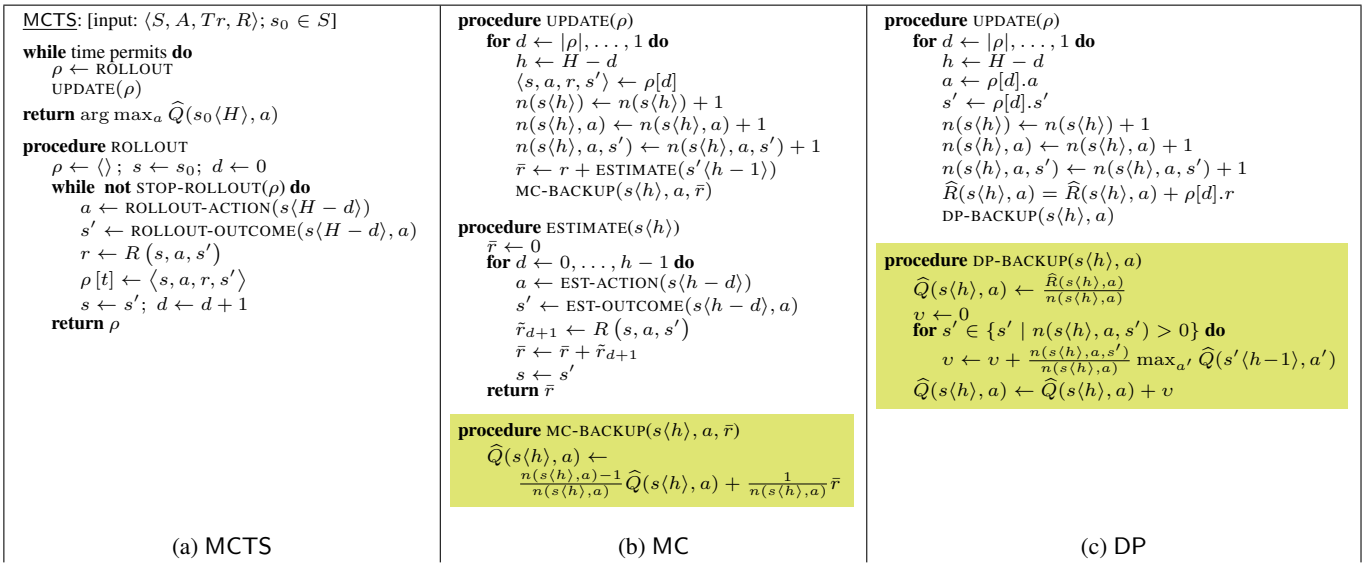

Figure 1. (a) Monte-Carlo tree search general scheme, with "separation of concerns" versions of (b) MC-backup and (c) DP-backup updates

MCTS, a canonical scheme underlying various MCTS algorithms for online MDP planning, is depicted in Figure 1. MCTS explores the state space in the radius of H steps from the initial state s_0 by iteratively issuing simulated ROLLOUTs from s_0. Each such rollout ρ comprises a sequence of simulated steps $\langle s, a, r, s'\rangle$, where s is a state, a is an action applicable in s, r is an immediate reward collected from issuing the action a, and s' is the resulting state. Once generated, the rollout is used to UPDATE some variables of interest, typically including at least the action value estimators $\widehat{Q}(s\langle h\rangle, a)$ and the counters $n(s\langle h\rangle, a)$ that record the number of times the corresponding estimators $\widehat{Q}(s\langle h\rangle, a)$ have been updated. Once interrupted, MCTS uses the information collected throughout the exploration to recommend an action to perform at state s_0.

Instances of MCTS vary mostly along their ROLLOUT-ACTION policies, prescribing the action to apply in the current state of the rollout; and their UPDATE strategies, specifying (i) which of the maintained variables should be updated based on the rollout, as well as (ii) how those variables should be updated. The "how" aspect of the MCTS UPDATE procedures is of our focus here. By decoupling between the decisions of what to update and how to update, the emphasized text in Figures 1b and 1c shows the respective subroutines for MC-backup and DP-backup, the two alternatives for "how to update" that are in use these days by various MCTS algorithms.

- MC-backups are based on the principle of averaging random variable samples: Given a new value sample $\bar{r}$ for an action a at $s\langle h\rangle$, $\bar{r}$ updates the running sample average $\widehat{Q}(s\langle h\rangle, a)$, either knowing or just assuming that this way $\widehat{Q}(s\langle h\rangle, a)$ will eventually converge to the true Q-value of a at $s\langle h\rangle$.
- DP-backups implement dynamic programming style estimates propagation, resembling Bellman backups in value iteration. With DP-backups, action value estimates $\widehat{Q}(s\langle h\rangle, a)$ are updated by the weighted sum of the value estimates of the empirically best actions at the outcomes of a (discovered so far), with the weights being induced by the gradually learned parameters of the MDP's stochastic transition function.

Earlier MCTS algorithms, such as flat MC, ε-greedy, UCT, and their numerous variations [3], all reflected rather directly the algorithms for reinforcement learning-while-acting in multi-armed bandit problems (MAB) [18]: Given a rollout ρ, update (the selected) action value estimates "by ρ", that is, by the actual rewards obtained along the rollout. Recent works on MCTS algorithms for online MDP planning examined the important differences between the (single state) MABs and (multi-state) general MDPs, leading to what was baptized as the principle of separation of concerns [9]: Instead of updating "by ρ", update (the selected) action value estimates "along the trajectory of ρ" by some information that goes beyond, and possibly even has nothing to do with, the specific rewards achieved by ρ. In particular, one can use one of the following.

1. MC-updates along additional rollouts, issued from the states along ρ according to a special, update-oriented, "estimation" policy. Such an UPDATE procedure in particular gives rise to the BRUE algorithm [9], and it is depicted in Figure 1b, together with a general template for its ESTIMATE policy.
2. DP-updates along the trajectory of ρ, as depicted in Figure 1c. This procedure in particular gives rise to the MaxUCT [14] and MaxBRUE [12] algorithms.

3 WORST-CASE GUARANTEES VS. REALISTIC EXPECTATIONS

Our comparison between MC-backups and DP-backups in MCTS is carried through two particular MCTS algorithms, BRUE [9] and MaxBRUE [12], which guarantee exponential-rate reduction of simple regret. The only difference between BRUE and MaxBRUE is that the former employs MC-backups while the latter employs DP-backups. Hence, for ease of presentation, in what follows we refer to these two algorithms as MC and DP, respectively. Both MC and DP use uniform sampling for ROLLOUT-ACTION, and both use the same ROLLOUT-OUTCOME that samples the provided generative model of the action's transition function. With UPDATE as in Figure 1c, that basically concludes the definition of DP. The UPDATE procedure of MC in Figure 1b, and in particular, its ESTIMATE subroutine, needs one more choice to be made.

In analogy to DP-backup that propagates the value of the empirically best actions, MC in ESTIMATE makes the estimation rollouts along the empirically best actions (selected by EST-ACTION). Thus,

in particular, no implementation choices are left open here. In contrast, for outcome selection along the estimation rollouts, two options are plausible, and both are viable in terms of consistency and performance guarantees. One option is to use the generative model of the action's transition function, same as in ROLLOUT-OUTCOME, while another option is to estimate the transition probabilities, similarly to DP, and draw samples from that empirical distribution.

The advantage of the second scheme is that the number of "oracle calls" to the generative model is similar to that of DP. In contrast, the first scheme performs a factor of $\frac{H}{2}$ more calls to generative model. In applications where such oracle calls are expensive, due to, e.g., a need to simulate a complex physical model, this can be an important argument for using the second scheme. However, the advantage of the first scheme is that it is not affected by the errors in the estimation of the transition probabilities. In what follows, whenever we need to distinguish between the two options, we will use $\mathsf{MC_M}$ to refer to MC with EST-OUTCOME using the generative model, and $\mathsf{MC_P}$ to refer to MC with EST-OUTCOME using the estimated transition probabilities.

As we already mentioned, both DP and MC have been recently proven to reduce simple regret at exponential rate. However, the corresponding statements of the formal bounds in [9] and [12] are somewhat involved, and this complicates the comparative analysis and discussion of DP and MC that we want to make here. Propositions 1 and 2 below provide much more accessible formal bounds on the performance of DP and MC, simplified by assuming $B, K \gg 0$, which allows keeping track only of the highest order factors of B and K, and replacing uniform ROLLOUT-ACTION with round-robin, which is equivalent in expectation but allows for simplifying the bounds further. We note that, while the bound for DP in Proposition 1 is qualitatively similar to this in [12], the bound for MC in Proposition 2 actually improves over the result in [9]. The proofs are delegated to a technical report [11].

Proposition 1 *Let $\pi^B(s_0\langle H\rangle)$ be the action recommendation of* DP *after applying n iterations. Then,*

$$\mathbb{P}\left\{Q(s_0\langle H\rangle, \pi^*(s_0\langle H\rangle)) - Q(s_0\langle H\rangle, \pi^B(s_0\langle H\rangle)) \geq \delta\right\} \leq K(2BK)^{H-1} e^{-\frac{\delta^2 n}{4K(BK)^{H-1}H^2}}$$

Proposition 2 *Let $\pi^B(s_0\langle H\rangle)$ be the action recommendation of* MC *after applying n iterations. Then,*

$$\mathbb{P}\left\{Q(s_0\langle H\rangle, \pi^*(s_0\langle H\rangle)) - Q(s_0\langle H\rangle, \pi^B(s_0\langle H\rangle)) \geq \delta\right\} \leq \left(\frac{6B^2K}{\Delta^2}\right)^{H-1} (9BK)^{\frac{1}{2}(H-1)^2} (H-1)!^2 e^{-\frac{\Delta^2 n}{2K(9BK)^{H-1}H^2}}$$

Roughly speaking, the exponents in the bounds in Propositions 1 and 2 capture the reduction rate of the simple regret, while the multiplicative factors capture the length of the "cooling periods" after which the respective bounds become meaningful. In that respect, the convergence rates of DP and MC appear to be rather comparable, excluding numerical constants, while the "cooling period" of MC appears to be much longer than that of DP. The latter suggests, even if only informally, that the empirical performance of DP should be expected to be more attractive than the empirical performance of MC. However, a deeper inspection of MC and DP below suggests a different perspective on the relative attractiveness of these two algorithms, and more generally, on the relative attractiveness of MC-backups and DP-backups in MCTS.

First, in [9] it was shown that the formal guarantees of MC can be improved by basing the action value estimators only on a fraction α of the most recent samples. This enhancement was referred in [9] as "learning by forgetting". For some specific values of α convenient for our discussion here, the bound for MC from Proposition 1 translates to a bound for the "learning by forgetting" $\mathsf{MC}(\alpha)$ as in Proposition 3 below.

Proposition 3 *Let $\pi^B(s_0\langle H\rangle)$ be the action recommendation of $\mathsf{MC}(\alpha)$ after applying n iterations with a steps-to-go-dependent averaging fraction $\alpha_h \sim \frac{1}{(9BK)^{h-1}}$. Then, we have*

$$\mathbb{P}\left\{Q(s_0\langle H\rangle, \pi^*(s_0\langle H\rangle)) - Q(s_0\langle H\rangle, \pi^B(s_0\langle H\rangle)) \geq \delta\right\} \leq \left(\frac{40BK}{\delta^2}\right)^{H-1} (H-1)!^2 e^{-\frac{\delta^2 n}{2K(9BK)^{H-1}H^2}}$$

As it appears, the worst-case cooling period of $\mathsf{MC}(\alpha)$ improves on that of MC, and this seems to come at no cost in terms of the convergence rate, expressed by the exponent. However, as we explain below, it seems that this improvement of the bound in Proposition 3 should be attributed mostly to the looseness of the bound for the standard setup of MC, and much less to the actual improvement of the performance measures. Indeed, adopting "learning by forgetting" leads to only minor empirical improvement, if at all.[2] Nevertheless, in comparison to DP, the cooling period length of $\mathsf{MC}(\alpha)$ has stronger dependency on the accuracy level δ and the horizon H.

At this point, two things should be noted with regards to the above formal bounds. First, by definition, formal bounds capture the worst-case settings of the MDP parameters, that is, uniform transition probability functions, tree-structured state space, etc. As such, the bounds tend to blur certain advantages of one algorithm over another in solving MDPs with some specific (and possibly expected in practice) characteristics. Second, due to conceptual differences between the dynamics of MC- and DP-backups, derivation of the bounds in Propositions 1 and 2 is based on two very different types of analysis. Hence, unlike what often happens for conceptually close techniques [5, 9], the value of formal bounds as indicators for the relative attractiveness of MC and DP is questionable. Having these two reservations in mind, in what follows we provide a more conceptual (aka less mathematically specific) comparative analysis of MC and DP by exploring several key features of MDP models, and reasoning about the (possibly different) effects of each of these features on the performance of the two algorithms.

Branching factors B and K (The size of the problem) In both Proposition 1 and Proposition 2, the basis of the analysis that gives rise to the formal guarantees is the fact that identifying the optimal action a^* at the root node $s_0\langle H\rangle$ requires that

(1) the value of a^* at $s_0\langle H\rangle$ is not too underestimated, and that
(2) the values of all the other, sub-optimal actions at $s_0\langle H\rangle$ are not too overestimated.

Both MC and DP ensure that these accuracy requirements are met, yet they differ in the way that these requirements recursively translate into requirements from the descendants of $s_0\langle H\rangle$.

In DP, to ensure that a sub-optimal action a is not too overestimated, all the applicable actions in all of the outcome states of a must not be too overestimated. Thus, the accuracy of estimating a

[2] This was observed both in our experiments for this work, as well as in [9].

sub-optimal action a in DP translates to accuracy requirements being posed to all the possible BK immediate action successors of a. In contrast, in MC, the likelihood that a sub-optimal action a will be too overestimated is negligible, and this because the expected value of the samples that induce the estimate of a is upper-bounded by the true value of a, regardless of the estimates of the action successors of a. Thus, the accuracy of estimating a sub-optimal action a in MC translates to no accuracy requirements from the successors of a. In sum, in terms of "not overestimating sub-optimal actions", MC-backups seem to be clearly preferred to DP-backups.

Examining now the requirement of not underestimating the optimal action a^* too much, meeting this requirement in DP requires that the optimal actions at all of the outcome states of a^* are not too underestimated. Indeed, if the latter holds, then the maximal action values propagated to a^* from its outcome states are also not too underestimating. Thus, the requirement of "not too underestimating the optimal action" a^* at $s_0\langle H\rangle$ translates in DP into B similar requirements being posed to all of the state successors of $s_0\langle H\rangle$ via a^*.

When it comes to MC, the picture is somewhat more complicated. Not underestimating the optimal action a^* too much requires that, in expectation, each of the samples inducing the estimate $\widehat{Q}(s_0\langle H\rangle, a^*)$ does not underestimate too much the true value of a^*. This implies that all of the outcome states of a^* should "identify" their optimal actions, which in turn translates into accuracy requirements posed to all (both optimal and sub-optimal) actions applicable at these outcome states of a^*. As we just mentioned, the accuracy requirements from sub-optimal actions in MC are negligible, and thus the effective burden is only with the accuracy requirements from the optimal actions at the outcome states.

In sum, the requirement of not underestimating the optimal action translates to accuracy requirements from the optimal actions at the outcome states, *at different stages of planning*. In the lack of more effective proof methods, the accuracy of each estimation sample in the proof of Proposition 2 is considered in isolation, and this is precisely the point where the difference between the bound and the actual performance may inflate. Indeed, the accuracy of an action estimate correlates with the accuracy of the same action estimate at subsequent points in time, yet this correlation is not factored into the bounds.

Importantly, the analysis of MC(α) in that respect is not any different: partial averaging does not offer a way to factor this correlation, but only reduces the bound by imposing accuracy requirements on fewer samples. Clearly, as the number of iterations increases, the correlation increases as well. The question, however, is when can we expect to have higher correlation at earlier stages. For instance, when the probability mass of the transition function concentrates on a small number of outcomes, the effective action branching becomes smaller than the nominal action branching B. In such cases, one should expect to have more samples based on overlapping rollouts, and thus to have a higher correlation. In any case, when the correlation is high, MC becomes equivalent to DP in terms of the accuracy requirement on the optimal action.

In summary, the dependency of DP on B and K is of order $(BK)^H$, whereas, depending on the correlation, the dependency of MC on B and K can be of order as small as B^H. Therefore, MC can be expected to be less sensitive than DP to K, especially when the effective action branching is relatively low, and thus the correlation is relatively high.

The shape of the reward function If right from the first steps, the immediate rewards of the optimal actions appear more attractive than these of the suboptimal actions, then identifying the optimal action a^* at $s_0\langle H\rangle$ is somewhat a simpler problem. The more challenging cases in that respect are when the discriminative rewards are pushed down the search tree, similarly to what happens in goal-driven MDPs. In such cases, identifying the optimal action a^* at $s_0\langle H\rangle$ requires properly identifying optimal actions far from the root, where samples are much sparser. Relating this point to the previous discussion on the size of the problem, it can be expected that the advantage of MC in the more challenging cases becomes more dependent on the effective action branching being relatively low.

The entropy of the transition function For all the bounds, the factor $\frac{1}{B}^H$ in the exponent results from the worst-case transition probability function, which, for each action, induces a uniform distribution over its outcomes. Clearly, as the entropy of the transition functions decreases, the better the bounds and performance of both DP and MC would be. However, here as well, some differences are expected depending on the update scheme. Since DP and MC$_\mathsf{P}$ use in their UPDATE the estimated transition function, their value estimations would be skewed towards the value of the more probable outcomes. Although this skew decreases with the number of samples, this decrease is slower at the deeper nodes since they are sampled less frequently. MC$_\mathsf{M}$, on the other hand, is free from this type of inaccuracy and therefore has certain advantage in that respect.

4 EXPERIMENTAL STUDY

In what follows, we put the qualitative comparison above into an empirical test. In previous work, the empirical effectiveness of online MDP planning algorithms was typically examined on a set of specific MDP problems, such as the benchmark suites of planning competitions (IPPC). These benchmarks, however, are problematic to use if one wants to examine the marginal impact of various parameters of the MDPs on the effectiveness of the algorithms, because these parameters simply cannot be controlled. In fact, almost all of these benchmarks are too large to compute the actual value of different actions at a state, and without that, assessing simple regret of different algorithms is impossible. Taking that on board, we devised a parametric MDP model from which one can select MDP instances with (i) arbitrary set of action values at the initial state, and (ii) arbitrary setting of the parameters discussed in the previous section. This allows us to experiment with large MDPs, for which otherwise it would be impossible (in reasonable time and computational resources) to compute the value function, and based on it, assess simple regret of different algorithms.

For ease of presentation, in what follows we refer to nodes $s\langle h\rangle$ simply by s; the steps-to-go component of the nodes remains clear from the text. In our base MDP setup, the horizon is set to $H = 10$, there are exactly $K = 20$ actions applicable at every node, and each node/action pair induces exactly B equiprobable outcome nodes, exclusive to that node/action pair, i.e, the induced state-space is tree-structured, and the transition probability functions are all uniform. For a node s, an applicable action a, and a possible outcome s', the immediate reward is set to $R(s, a, s') = \frac{Q(s,a)}{h}$, and the value of the outcome node receives the remainder $V(s') = Q(s, a) - R(s, a, s')$. At any node s, there is exactly one optimal action, the value of which equals the value $V(s)$ of the node, whereas all other actions have identical values of $\epsilon V(s)$, for some $\epsilon \in (0, 1)$. The choice of ϵ plays an important role here. If, for instance, ϵ is set equally for all nodes, then basing the value updates in both MC and DP on random (and not empirically best) action successors will surface the optimal action at s_0, and this because all the actions will be underestimated by

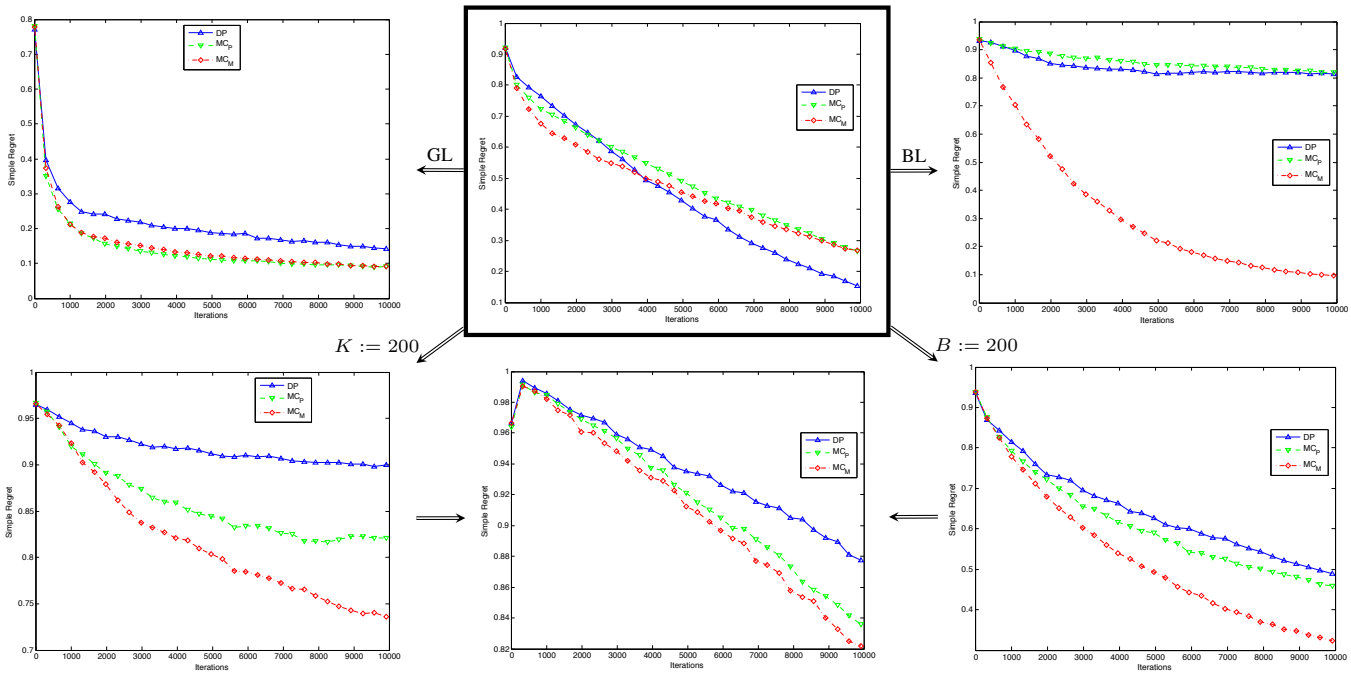

Figure 2. Experimental results on the base setup with $K = 20$, $B = 20$, and all action outcomes being equiprobable (top-center), as well as on the variants with ($\swarrow$) $K = 200$, ($\searrow$) $B = 200$, ($\downarrow$) $K = 200$ and $B = 200$, ($\leftarrow$) "good likely" transition functions, and ($\rightarrow$) "bad likely" transition functions.

a similar magnitude. Therefore, in our setup, for all nodes reachable by the optimal policy from s_0, we set $\epsilon = 0.6$, while for all nodes off the optimal policy, we set $\epsilon = 0.8$. The only node that is not properly covered by this categorization is the actual initial node s_0, and there we also set $\epsilon = 0.8$. In this setup, the estimates induced by random updates would not preserve the right order of the action values, imposing a harder challenge on the algorithms.

The emphasized plot in the top-center of Figure 2 depicts the simple regret obtained by the three examined algorithms, DP, $\mathsf{MC_P}$, and $\mathsf{MC_M}$, on the base setup as above, as a function of the number of iterations. While MC here appear slightly better at the start, DP quickly catches up and gradually outperforms MC. However, the picture changes when the base setup is modified in several different ways. First, when scaling up the problem by increasing K to 200 (bottom-left), or by increasing B to 200 (bottom-right), or both (bottom-center), MC performs much better than DP at all times, with $\mathsf{MC_M}$ being the clear dominator. Second, the top-left and top-right plots in Figure 2 depict the results for two setups that deviate from the base only by altering the entropy of the transition probability functions. In both setups, for each node s and each applicable action a, one outcome s' is substantially more likely than all other, with $\mathbb{P}(s' \mid s, a) = 0.9$ and, for all outcomes $s'' \neq s$, $\mathbb{P}(s'' \mid s, a) = \frac{0.1}{B-1}$. The difference between the two setups is the relative value of the more likely outcome s': In the "good likely" setup (GL), the more likely outcome is also more valuable, i.e., $R(s, a, s') + V(s') > R(s, a, s'') + V(s'')$ for all $s'' \neq s'$, and in the "bad likely" setup (BL), it is the other way around. In either case, all action-outcome values are set such that (1) they reflect the value of the action, i.e. $\sum_{s'} \mathbb{P}(s' \mid s, a) \left(R(s, a, s') + V(s')\right) = Q(s, a)$, and (2) each action-outcome value is neither smaller than half of the action value, nor higher than the maximal immediate reward ($= 1$, in our experiments), times the number of steps-to-go.

In both "good likely" and "bad likely" variants, the reduction in the entropy of the transition function basically reduces the effective state branching of the actions, and thus the correlation between the successive samples in MC is expected to grow, getting more in line with the optimistic assumption on MC's dependence on B and K. The results depicted in Figure 2 support this expectation. Moving from the base setup, the performance of MC improves in both "good likely" and "bad likely" setups, and in fact, in both setups, MC outperforms DP. Likewise, importantly, while in "good likely" there is effectively no difference between $\mathsf{MC_M}$ and $\mathsf{MC_P}$, in "bad likely", $\mathsf{MC_M}$ is performing much better than $\mathsf{MC_P}$, with the latter meeting the very poor performance of DP. Basically, the "bad likely" setup demonstrates how dramatic can be the implications of establishing value estimation on the estimated transition probabilities. Here, the underestimation of the values by DP and $\mathsf{MC_P}$ results in their very poor performance. To recap Figure 2, it appears that MC outperforms DP except for on MDPs of relatively small size, and $\mathsf{MC_M}$ being justifiably more robust than $\mathsf{MC_P}$.

Another important aspect that we examined in our experiments pertains to the dependence of the algorithm performance on the shape of the rewards as a function of the node depth. In the base setup, *at any node*, the actions are rewarded proportionally to their actual value, and thus, in particular, optimal actions have higher immediate rewards than the sub-optimal actions.

Figure 3 shows the results for two setups that deviate from the base setup in that aspect as follows. (Both these setups are more challenging than the base, and thus the x-axis in Figure 3 goes up to 10^5 iterations, and not to 10^4 iterations like in Figure 2.) In "first equal" (top-center), the immediate rewards differ from the base setup only at the root, where, instead of rewarding the optimal action higher than the sub-optimal actions, all the actions have the same reward of 0.5, independently of the outcome. The results for the "good likely" and "bad likely" variants of "first equal" are depicted in top-left and top-right corners of Figure 3. In the even more challenging setup "first few equal" (bottom-center), the immediate rewards are set to the minimum between 0.5 and the action-outcome value, that is $R(s, a, s') = \min\{0.5, Q(s, a)\}$; in the "good likely" (bottom-left) and "bad likely" (bottom-right) variants, the appropriate factor

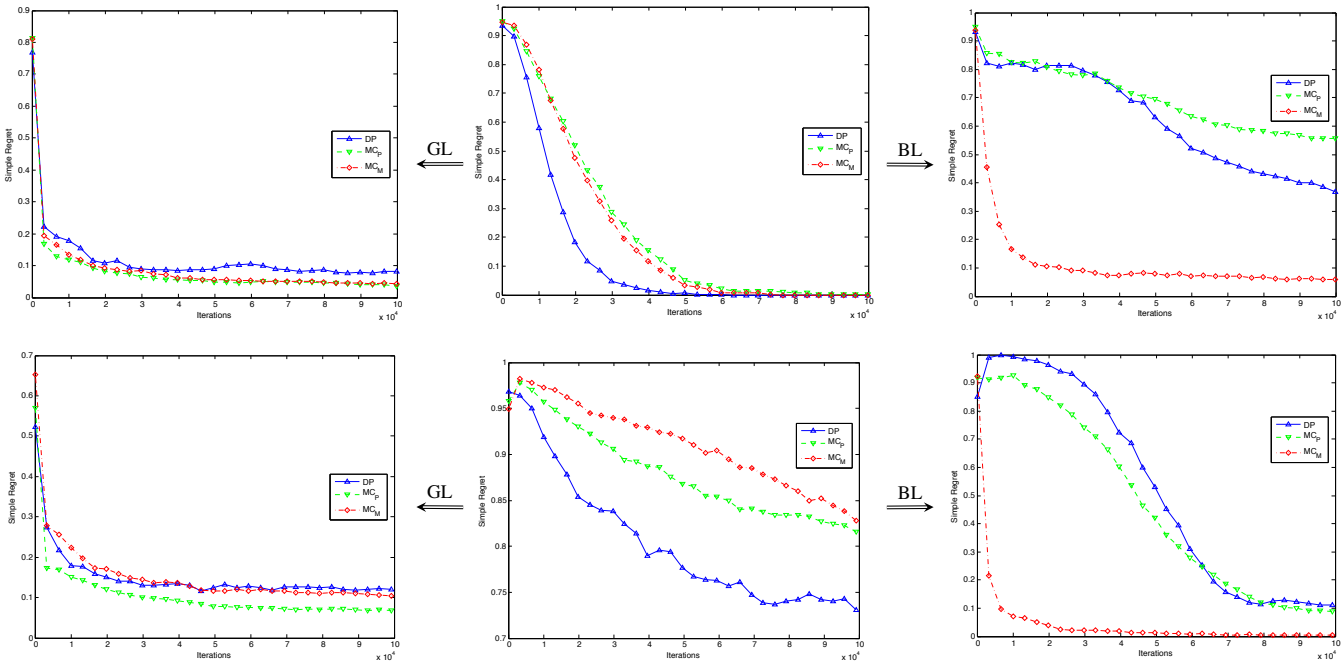

Figure 3. Experimental results on the "first equal" (top-center) and "first few equal" (bottom-center) modifications of the base setup, as well as on their variants with "good likely" (←) and "bad likely" (→) transition functions.

is added to $Q(s, a)$.

Comparing the results for the variants of the base setup in Figure 2 with the results for "first equal" and "first few equal" in Figure 3, the qualitative relative performance of DP, MC_P, and MC_M remains the same, with the absolute performance of all algorithm decreasing, as expected, from the base setup to "first equal", and from "first equal" to "first few equal". It should also be noted that here, in contrast to the base setup, the advantage of DP over MC under equiprobable action outcomes was observed also when K and B were higher than 20. This goes in line with the dependence of MC's performance on the correlation between the successive samples, because pushing the discriminative rewards down the tree delays the correlation. Finally, we also experimented with various graph-structured (in contrast to tree-structured) variants of our MDP model. As expected, the performance of all three algorithms improved with the degree of the multi-connectedness of the nodes, but the improvements were of the same magnitude for all three algorithms. In sum, based on our experiments and in line with the analysis in Section 3, DP appear more effective than MC as long as the size of the problem is sufficiently small, but otherwise, MC outperforms DP even under most challenging conditions, especially if the probability mass of the transition functions concentrates on very few outcomes.

ACKNOWLEDGEMENTS

This work was partially supported by the FA8655-12-1-2096, and the EOARD grant ISF grant 1045/12.

REFERENCES

[1] R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.
[2] R. Bjarnason, A. Fern, and P. Tadepalli, 'Lower bounding Klondike Solitaire with Monte-Carlo planning', in *ICAPS*, (2009).
[3] C. Browne, E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, 'A survey of Monte-Carlo tree search methods', *IEEE Trans. on Comp. Intell. and AI in Games*, 143, (2012).
[4] S. Bubeck and R. Munos, 'Open loop optimistic planning', in *COLT*, pp. 477–489, (2010).
[5] S. Bubeck, R. Munos, and G. Stoltz, 'Pure exploration in finitely-armed and continuous-armed bandits', *Theor. Comput. Sci.*, **412**(19), 1832–1852, (2011).
[6] T. Cazenave, 'Nested Monte-Carlo search', in *IJCAI*, pp. 456–461, (2009).
[7] P-A. Coquelin and R. Munos, 'Bandit algorithms for tree search', in *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 67–74, Vancouver, BC, Canada, (2007).
[8] P. Eyerich, T. Keller, and M. Helmert, 'High-quality policies for the Canadian Traveler's problem', in *AAAI*, (2010).
[9] Z. Feldman and C. Domshlak, 'Simple regret optimization in online planning for Markov decision processes', *CoRR*, `arXiv:1206.3382v2 [cs.AI]`, (2012).
[10] Z. Feldman and C. Domshlak, 'Monte-Carlo planning: Theoretically fast convergence meets practical efficiency', in *UAI*, (2013).
[11] Z. Feldman and C. Domshlak, 'Monte-Carlo tree search: To MC or to DP?', Technical Report IE/IS-2014-03, Technion, (2014).
[12] Z. Feldman and C. Domshlak, 'On MABs and separation of concerns in Monte-Carlo planning for MDPs', in *ICAPS*, (2014).
[13] S. Gelly and D. Silver, 'Monte-Carlo tree search and rapid action value estimation in computer Go', *AIJ*, **175**(11), 1856–1875, (2011).
[14] T. Keller and M. Helmert, 'Trial-based heuristic tree search for finite horizon MDPs', in *ICAPS*, pp. 135–143, (2013).
[15] L. Kocsis and C. Szepesvári, 'Bandit based Monte-Carlo planning', in *ECML*, pp. 282–293, (2006).
[16] L. Péret and F. Garcia, 'On-line search for solving Markov decision processes via heuristic sampling', in *ECAI*, pp. 530–534, (2004).
[17] M. Puterman, *Markov Decision Processes*, Wiley, 1994.
[18] H. Robbins, 'Some aspects of the sequential design of experiments', *Bull. Amer. Math. Soc.*, **58**(5), 527535, (1952).
[19] C. D. Rosin, 'Nested rollout policy adaptation for Monte Carlo tree search', in *IJCAI*, pp. 649–654, (2011).
[20] N. Sturtevant, 'An analysis of UCT in multi-player games', in *CCG*, p. 3749, (2008).
[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.
[22] D. Tolpin and S. E. Shimony, 'MCTS based on simple regret', in *AAAI*, (2012).