Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

HEURISTICS FOR PLANNING WITH RARE CATASTROPHIC EVENTS

Youngjun Kim

Department of Aeronautics and Astronautics
Stanford University
496 Lomita Mall
Stanford, CA 94305, USA

Yonatan Gur

Graduate School of Business
Stanford University
655 Knight Way
Stanford, CA 94305, USA

Mykel J. Kochenderfer

Department of Aeronautics and Astronautics
Stanford University
496 Lomita Mall
Stanford, CA 94305, USA

ABSTRACT

High-dimensional Markov decision processes are often solved using online sampling-based methods such as Monte Carlo tree search. Problems that involve rare catastrophic events with large negative rewards can be very challenging for such approaches. The difficulty arises because the common policies for expanding a search tree are not able to handle rare large negative rewards appropriately, leading to high variance estimates of value. This paper studies the impact of reward structure with rare catastrophic events on the performance of common algorithms and proposes enhanced tree policy strategies for Monte Carlo tree search. Numerical experiments suggest that our proposed methods can significantly improve performance on a variety of problem domains ranging from stochastic multi-armed bandits to aircraft rerouting problems.

1 INTRODUCTION

Sequential decision problems with uncertainty in the outcomes of actions are often modeled as Markov decision processes (MDPs) (Bellman 1957, Howard 1960). When there is also uncertainty in the current state of the environment, such problems can be modeled as partially observable Markov decision processes (POMDPs) (Åström 1965, Sondik 1971). Both models have been extensively studied (Kochenderfer 2015). Solving for optimal decision strategies that maximize the expected accumulation of reward can quickly become computationally intractable as the dimension of the state space increases. Hence, there has been interest in approximate solution techniques, including online methods that involve planning over only a subset of the state space that is potentially reachable from the current state.

A category of online approaches that has attracted tremendous interest in the artificial intelligence community is Monte Carlo tree search (MCTS) (Coulom 2006). It finds approximately optimal actions by iteratively building a search tree using Monte Carlo simulations. A commonly used MCTS algorithm is UCT (Kocsis and Szepesvári 2006), which applies the Upper Confidence Bound (UCB) policy (Auer et al. 2002) to guide the construction of the tree. The policy involves an exploration constant as a parameter. With a proper exploration constant, UCT will consistently provide near-optimal solutions (Kocsis and Szepesvári 2006). However, tuning the constant can be challenging.

978-1-5386-3428-8/17/$31.00 ©2017 IEEE

Choosing a proper exploration constant for UCT is particularly difficult for problems with rare catastrophic events, such as a system failure of a nuclear plant or an aircraft collision. In MDP or POMDP formulations, large negative rewards are associated with catastrophic events. Since the possibility of large negative rewards increases the range of reward, UCB requires a large exploration constant to provide sufficient exploration. On the other hand, a large exploration constant may not be required in points of the UCT search tree that do not involve rare catastrophic events and may result in an unnecessarily slow convergence to optimal long-run average performance. In addition, the rare occurrence of catastrophic events causes high variance estimates of value in points of the UCT search tree that involve events and degrades the performance.

Main contribution. This paper presents modifications to Monte Carlo tree search to better account for rare catastrophic events. Three heuristics are proposed. First, we introduce an Adaptive UCB (A-UCB) heuristic that considers two possible settings: one in which rare catastrophic events occur ("hard" setting), and another one in which they do not ("easy setting"). A-UCB introduces an additional level of exploration over UCB policies to identify the more appropriate exploration constant among two constants, where each constant is pre-specified for each setting based on the associated reward structure. Second, we propose Thompson sampling with a Gaussian mixture model over rewards. We apply Thompson sampling over the reward model to balance exploration and exploitation without the need of an exploration parameter. Third, a splitting method borrowed from the rare event simulation literature (Rubino and Tuffin 2009, Garvels 2000) is applied to the construction of the tree. This method biases Monte Carlo tree search towards rare catastrophic events with the aim to reduce the variance of the estimate of the reward accumulation. We evaluate these strategies on three problem domains.

2 METHODS

This section briefly reviews prior work and describes our extensions to address rare catastrophic events in multi-armed bandit problems, MDPs, and POMDPs.

2.1 Multi-Armed Bandit Policies

A multi-armed bandit problem is originally proposed in the context of drug tests by Thompson (1933) and formulated for a general setting by Robbins (1952). In the problem, a decision maker needs to choose an arm from a set of arms at each round to maximize cumulative reward over a decision horizon (Berry and Fristedt 1985, Gittins 1989). In the stochastic setting of this class of problems (Lai and Robbins 1985), each arm is associated with an unknown, typically independent, reward distribution, and reward realizations are only observed when an arm is selected. To maximize cumulative expected reward, one needs to balance between acquiring information about arms through exploration and exploiting information about immediate rewards. One way to measure the success of this trade-off is cumulative regret, which is the sum of the expected reward difference between the best arm and played arms. The objective of the multi-armed bandit is to minimize cumulative regret.

UCB1 (Auer et al. 2002) is a common multi-armed bandit policy. One first pulls each arm once, and from that point on, plays the arm i that maximizes $\bar{X}_i + c\sqrt{\ln n/n_i}$ where $\bar{X}_i$ is the empirical mean of rewards for playing arm i, c is an exploration parameter, n is the total number of plays so far, and n_i is the number of times arm i was played. The first term in the UCB1 formulation, $\bar{X}_i$, encourages exploitation by playing what appears to be better arms. The second term, $c\sqrt{\ln n/n_i}$, encourages exploration by playing less-explored arms. Auer et al. (2002) show that UCB1 uniformly achieves expected cumulative regret that is logarithmic in the length of the decision horizon.

This paper introduces a new method, Adaptive UCB (A-UCB) shown in Algorithm 1. This heuristic considers two possible problem settings: one in which rare catastrophic events occur, and another one in which they do not. Given a suitable exploration constant in UCB1 for each setting, A-UCB increasingly

plays with the proper UCB1 policy. A control policy in A-UCB adaptively chooses which subpolicy to use based on expected rewards from subpolicies. The reward from each subpolicy is assumed to follow a Gaussian distribution, and the Normal-Gamma (NG) distribution is chosen as the conjugate prior. The control policy executes the subpolicy having the highest expected reward.

Algorithm 1 Adaptive UCB.

Initialize reward models for both subpolicies
loop
  for each subpolicy π_i
    $(\theta_i, \tau_i) \sim \text{NG}(\mu_i, \lambda_i, \alpha_i, \beta_i)$
  Set $i^* = \arg\max_i \theta_i$
  Play with subpolicy π_{i^*} and observe reward r
  Update reward model for subpolicy π_{i^*} with reward r

A-UCB is related to other policies that have been proposed in the context of adaptation to unknown environments. Sani et al. (2014) present an online optimization method that uses decisions proposed by two online optimization algorithms, where both online optimization algorithms need to be observed. This requirement does not fit in a multi-armed bandit context in which only one arm is played and observed in each period. Seldin and Slivkins (2014) present an algorithm for multi-armed bandits that achieves near-optimal performance in both stochastic and adversarial bandits. The algorithm detects which regime the environment is in and plays accordingly. Chaslot et al. (2008) uses the cross-entropy method (Rubinstein and Kroese 2004) to find parameters for planning problems including the exploration constant of UCB in Monte Carlo tree search. This approach finds one tuned exploration constant and applies it to all points in MCTS.

Recently, Thompson sampling (TS) has attracted attention because of its excellent empirical performance. TS is a randomized algorithm originally proposed by Thompson (1933). It assumes a Bayesian prior on parameters of the reward distribution for each arm. At each time step, it draws the posterior probability of being the best arm for each arm and plays an arm having the highest probability. Studies have demonstrated the merits of this approach empirically (Chapelle and Li 2011, May and Leslie 2011) as well as theoretically (Agrawal and Goyal 2012).

This paper presents a modified version of TS with a Gaussian mixture reward model (TSM), where the reward model is assumed to be a mixture of Gaussian distributions. Reward is divided into multiple groups, and each group of reward is assumed to be a Gaussian distribution. The estimate of a reward is obtained by weighting rewards from reward groups. For example, if a multi-armed bandit involves a rare catastrophic event, reward can be divided into two groups: one group includes large negative rewards and another group includes smaller rewards. The idea is to exploit prior knowledge of the reward structure.

TSM is similar to DNG-MCTS (Bai et al. 2013) and D^{2}NG-POMCP (Bai et al. 2014) in that Thompson sampling is applied to MCTS and rewards are Gaussian mixtures. However, the underlying reward modeling is different. In TSM, each group of rewards, depending on whether a rare catastrophic event exists or not, is modeled as a Gaussian and weights are the probability of the rare catastrophic event. DNG-MCTS and D^{2}NG-POMCP modeled accumulated reward with Gaussians given states and weights, which were transition probabilities.

Algorithm 2 outlines TSM. In this algorithm, γ_{ij}, μ_{ij}, λ_{ij}, α_{ij}, and β_{ij} are the parameters of the conjugate priors of the reward model. It is assumed that each reward group follows a Normal distribution. The Normal-Gamma distribution is chosen as the conjugate prior of parameters for the Gaussian distribution.

The weights for the reward groups follow a Dirichlet distribution. The expected reward for an arm is estimated by summing the weighted expected rewards of reward groups.

Algorithm 2 Thompson sampling with Gaussian mixture.

For each arm $i = 1, \ldots, K$, set parameters of the reward model for arm i with initial values
loop
  for each arm $i = 1, \ldots, K$
    $(w_1, w_2, \ldots, w_n) \sim \text{Dir}(\gamma_{i,1}, \gamma_{i,2}, \ldots, \gamma_{i,n})$
    for each component $j = 1, \ldots, n$
      $(\mu_j, \tau_j) \sim \text{NG}(\mu_{ij}, \lambda_{ij}, \alpha_{ij}, \beta_{ij})$
    Set $\theta_i = w_1 \times \mu_1 + w_2 \times \mu_2 + \cdots + w_n \times \mu_n$
  Play arm $i^* = \arg\max_i \theta_i$ and observe reward r
  Update reward model for arm i^* with reward r

2.2 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a family of algorithms that has been applied to both MDPs and POMDPs. The basic algorithm iteratively builds a search tree from the current state in attempt to find the best possible action. Nodes in the search tree represents states and links between nodes represents actions. Each state holds statistics for choosing an action such as expected rewards. The search tree represents sequences of states and actions. Each search iteration consists of four stages (Browne et al. 2012).

1. *Selection*: Starting from the root node, a tree policy is recursively applied for descending the search tree until a leaf node is reached.
2. *Expansion*: At the leaf node, an action is selected by the tree policy and a new child node is linked to the leaf node.
3. *Evaluation*: A simulation is performed from the new node according to a default policy.
4. *Backup*: An outcome from the simulation is backpropagated through the search path from the new node to the root node. Outcomes from nodes in the path are also backpropagated. Statistics of states in the path are updated accordingly.

The tree policy selects an action based on statistics that are updated each iteration. The tree policy must balance exploration with exploitation at each level in the search tree. UCT is a kind of MCTS that uses UCB as a tree policy. Pseudocode for a version of UCT applied to MDPs and POMDPs is provided by Kochenderfer (2015). When rewards are in the range $[0, 1]$, UCT consistently finds the optimal solution using an exploration constant of $\sqrt{2}$ multiplied by depth for the UCB (Kocsis et al. 2006). Otherwise, the exploration constant must be adjusted appropriately according to the range of rewards.

MCTS involves collecting statistics about the expected accumulation of reward after a series of actions. If catastrophic events occur in a problem, sampling may result in large negative rewards. Rare large negative rewards can make the reward estimates converge slowly, leading to poor performance. In the context of MCTS, there has been work on general variance reduction techniques such as control variates, common random numbers, and antithetic variates (Veness et al. 2011). We will focus on a particular kind of variance reduction technique from the rare event simulation literature, known as the splitting method (Rubino and Tuffin 2009, Garvels 2000).

The splitting method accelerates the convergence of Monte Carlo simulations by increasing the occurrence of rare events. The method generates more samples of interest by selecting and replicating sample paths. It

divides paths into multiple levels based on proximity to the event of interest. If a sample path starting from a level reaches to the next level, multiple copies of the path are reproduced. These copies are resimulated from the current level. If a copy reaches the next level, the replicating procedure is repeated. If not, the simulation of the copy ends without replication. While the replication procedure is repeated, generated sample paths are biased toward rare events. The splitting method has been successfully applied to various domains (Prandini et al. 2011, Kim et al. 2015).

The idea of selection and replication in the splitting method can be applied to UCT. Levels are defined by a series of distances to an event of interest. While traversing the search tree, if a state and an action lead to a next state hitting a level, a specified number of simulations at the level are replicated from the state hitting the level. The average of outcomes from replicated simulations is backpropagated. If a replicated simulation hits the next level, the replication procedure is repeated. Throughout the process, UCT simulates more states close to rare catastrophic events and obtains better estimates of expected rewards involving the events.

3 EXPERIMENTS

This section demonstrates the issues that can arise when decision problems involve rare catastrophic events. The methods presented in the previous section are evaluated empirically on a series of problems.

3.1 Stochastic Two-Armed Bandit

We begin with a two-armed bandit problem with rewards that follow a mixture of two truncated Normal (TN) distributions:

$$\begin{cases} \text{TN}(\mu_1, \sigma_1) & \text{with probability } 1-p \\ \text{TN}(\mu_2, \sigma_2) & \text{with probability } p \end{cases} \tag{1}$$

where $\mu_1, \mu_2 < 0$, $\mu_2 \ll \mu_1$, $0 < p < 1$, and $p \ll 1$. We assume a 5-σ truncated Normal, keeping the reward bounded. With this distribution, playing an arm returns a small negative reward most of the time, but sometimes it returns a large negative reward.

Figure 1 shows the impact of a rare catastrophic event on UCB1. The bandit used for the experiment has two arms. The reward distribution for each arm is defined as follows:

$$r_1 \sim \begin{cases} \text{TN}(-30, 4) & \text{with probability } 1-p_1 \\ -1000 & \text{with probability } p_1 \end{cases}$$

$$r_2 \sim \begin{cases} \text{TN}(-70, 10) & \text{with probability } 1-p_2 \\ -1000 & \text{with probability } p_2 \end{cases}$$

where $0 \leq p_1, p_2 \leq 0.1$. The figure shows two extreme settings that the event does not occur in easy setting and there is 10% chance of the event for each arm in hard setting. A proper exploration constant is chosen for each case empirically and both constants satisfy the Chernoff-Hoeffding inequality for UCB1 (Auer et al. 2002). Plots in the figure are averages of a thousand simulations. The figure shows that an exploration constant working well for one case does not work well for another. There are large cumulative regret differences between two UCB1 policies. Figure 1 (d) shows that setting an improper exploration constant can negatively impact the convergence of UCB1. UCB1(50), which is UCB1 with the exploration constant 50, selects the best arm only 60% of the time. It is close to 50% of random selection. It is an issue to choose a proper exploration constant for UCB1 in the case that the probability of the event is unknown to a player. The experiment shows that UCB1 might not be a good policy for a multi-armed bandit involving a rare catastrophic event.

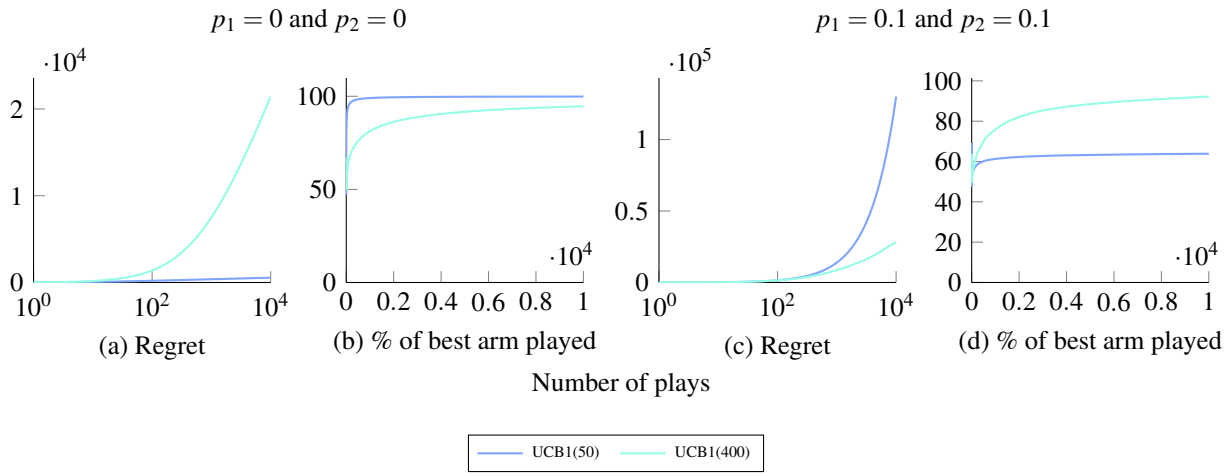

Figure 1: Impact of a rare catastrophic event on UCB1.

Figure 2 shows the average cumulative regret of a thousand simulations when using A-UCB and TSM compared to UCB1 and TS. The reward distribution for the experiment is the same as the reward distribution used in previous experiment. A-UCB selects UCB1(50) and UCB1(400) as subpolicies. For TSM, the reward is modeled with two Gaussian distributions, one for rewards from a rare catastrophic event and another for rewards from other. The uniform prior is set as follows: $\gamma_{i,1} = 0.01, \gamma_{i,2} = 0.01, \mu_{i,1} = -50, \lambda_{i,1} = 1, \alpha_{i,1} = 0.5, \beta_{i,1} = 12.5, \mu_{i,2} = -1000, \lambda_{i,2} = 1, \alpha_{i,2} = 0.5, \beta_{i,2} = 0.5$. Experiment has been performed with various set of probabilities of the rare catastrophic event for arms. UCB1(CE) shows average cumulative regret when using UCB1 with the optimal exploration constant. For each set of probabilities, the optimal exploration constant for UCB1 is computed using the cross-entropy method (Rubinstein and Kroese 2004, De Boer et al. 2005). The cross-entropy method is an adaptive optimization algorithm that searches for the optimal parameters optimizing a performance metric. The method is used for finding the optimal exploration constant minimizing cumulative regret. Few featured results from the experiment are presented in the figure.

In Figure 2, TSM performs better than other policies. TSM generally performs better than TS because TSM exploits the reward structure while TS maps rewards into [0, 1] regardless of the possibility of rare catastrophic events. In Figure 2 (a) and (b), the regret of A-UCB is close to the regret of UCB1(50) and the regret of UCB1(400), respectively. A-UCB performs close to the best UCB1 for each setting with some overhead. In A-UCB there is a cost for learning which subpolicy is best.

Figure 2 (c) shows an interesting characteristic of A-UCB. In the figure, A-UCB performs better than both UCB1(50) and UCB1(400), which are its subpolicies, and even better than UCB1(CE). For the reward distribution used in Figure 2 (c), one subpolicy works well with some of simulations while another subpolicy works well with others depending on when the rare catastrophic event occurs in simulations. Since A-UCB is designed to choose the best subpolicy based on empirical outcomes, it switches to play another subpolicy if a playing subpolicy performs badly during a simulation.

Extensive experiment with various reward settings is performed for validating the usefulness of A-UCB and TSM empirically. In the experiment, the mean of nominal reward distribution has been changed from -10 to -100, the probability of the rare catastrophic event has been changed from 0 to 0.1, and the reward for the event is fixed at -1000. To choose proper exploration constants for A-UCB, the optimal exploration constant is computed for every reward setting using the cross-entropy method and two extremes of the exploration constants are chosen, which are 10 and 350. Other experiment settings are the same as settings used in the previous experiment. From all reward settings, A-UCB and TSM performs better than both UCB1 and TS in 73% and 78% of all settings, respectively.

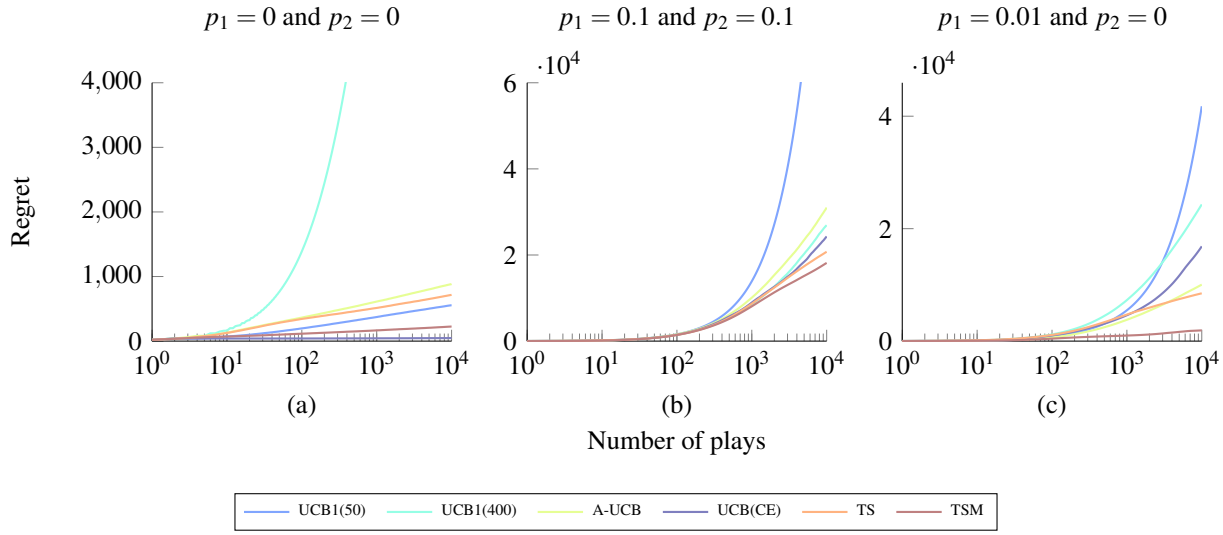

Figure 2: Regret from simulation with A-UCB and TSM.

Figure 3 shows some policy maps representing the best policy in terms of average cumulative regret for each reward setting. For each plot, x and y axes are means of nominal reward distributions for the first and the second arms, respectively. The mean varies from -10 to -100. In the figure, the first row compares A-UCB and its subpolicies. The second row compares UCB1, TS, and TSM. When the rare catastrophic event does not occur in both arms, as shown in Figure 3 (a) and (e), A-UCB and TSM perform better than UCB1 and TS in most settings, and UCB1 with the small exploration constant performs well in some settings. When the rare catastrophic event can occur in both arms, UCB1 with the large exploration constant can be seen in the policy maps. When the event can occur in one arm but cannot occur in another arm, UCB1 with the small exploration constant works better than A-UCB and TSM in many settings as shown in Figure 3 (b) and (f) because playing one arm is obviously better than playing another arm. However, the difference in average cumulative regret among UCB1 with the small exploration constant, A-UCB, and TSM is not large. If the event may occur in both arms, A-UCB and TSM tend to work better than UCB1 and TS as shown in Figure 3 (c), (d), (g), and (h).

In short, experiments demonstrate that A-UCB and TSM work well for a multi-armed bandit involving a rare catastrophic event whose probability is unknown. The source code of this experiment can be found at https://github.com/neosado/MultiArmedBandit.git. Additional policy maps from the experiment can be found at https://goo.gl/EZA3Tp.

3.2 MineField

MineField is a simple MDP involving rare catastrophic events. The world is modeled as a grid as shown in Figure 4. There is a rover at the bottom left corner and a goal at the top right corner. Mines are placed at random locations. The rover does not know where mines are located. The rover can move up or right. There is a random cost whenever the rover moves. If a mine is at the location of the rover, there is a chance that the mine explodes with a large cost. The rover can still move regardless of explosions. The objective is to find the optimal path to the goal that minimizes expected cost.

Scenarios for simulations are randomly generated. Each scenario involves a 7 by 5 grid world. Mines are placed randomly at approximately 10% of the grid cells. Each mine gives a large negative reward of -10000 when exploded. Otherwise, the scenario gives a random cost following a Gaussian distribution

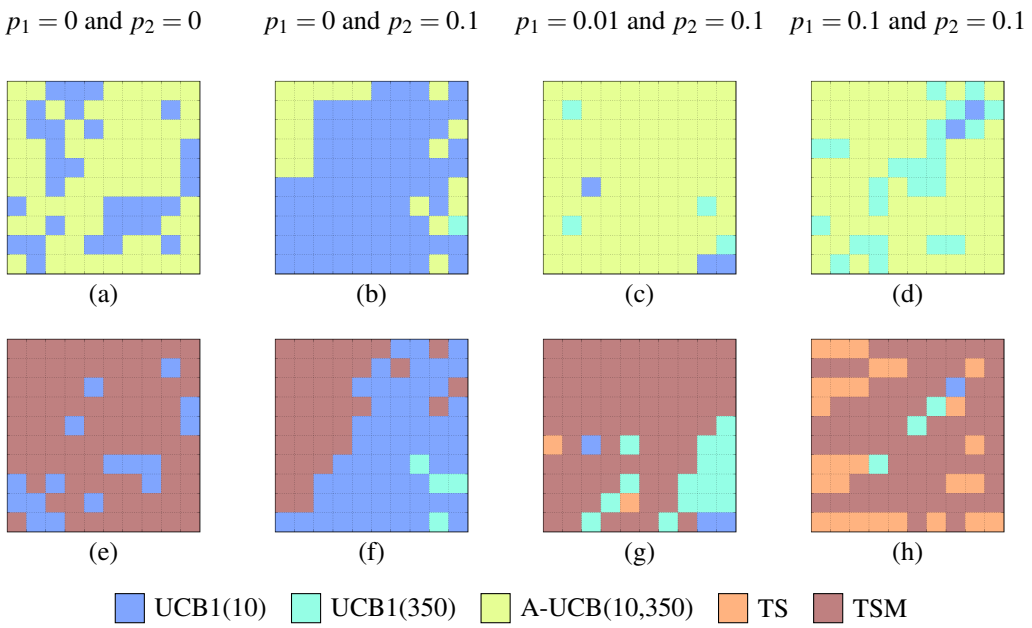

Figure 3: Policy maps from simulation of a stochastic two-armed bandit.

with a randomly selected mean in $[-110, -10]$ and standard deviation in $[2, 22]$. The probability of the mine explosion is chosen uniformly at random between 0 and 0.1. UCT is used for solving the problem.

Figure 5 shows the relative return error as the computational budget is increased. The relative return error is defined by the difference between a return and the optimal return for a scenario divided by the optimal return. The return for a scenario is computed by averaging cumulative rewards from one hundred simulations. The figure shows the average and the 95% confidence interval of relative return errors from one hundred scenarios. The relative error goes to zero as the return is close to the optimum. The computational budget is the maximum number of iterations for UCT. Two exploration constants for UCB1 are chosen according to the range of rewards without or with the mine explosion, respectively. A-UCB uses two UCB1 policies with the exploration constants as subpolicies. In the experiments, TSM has the uniform prior set to $\gamma_{i,1} = 0.01, \gamma_{i,2} = 0.01, \mu_{i,1} = -100, \lambda_{i,1} = 1, \alpha_{i,1} = 0.5, \beta_{i,1} = 50, \mu_{i,2} = -10000, \lambda_{i,2} = 1, \alpha_{i,2} = 0.5, \beta_{i,2} = 0.5$.

As shown in Figure 5, A-UCB and TSM performs better than other policies, especially when the computational budget is limited. UCB1(10000), A-UCB, UCB1(CE), TS, and TSM converge to a similar relative return error with sufficient computational budget while the relative return error of UCB1(100) is far from others. A-UCB performs better than both UCB1(100) and UCB1(10000), which are its subpolicies, and even better than UCB1(CE), which applies one single optimal exploration constant to all decision nodes in UCT search tree. A-UCB adaptively applies a proper exploration constant to decision nodes in UCT search tree depending on whether a node faces large negative rewards or not. The experiment shows that A-UCB and TSM perform better than other policies in UCT for a MDP with rare catastrophic events.

The source code for this experiment is found at https://github.com/neosado/MineField.git.

3.3 Rerouting of Unmanned Aircraft

Unmanned aircraft often use GPS for navigation. It is not uncommon for the GPS signal to be lost due to occlusions, interference, or failures of GPS device. When the signal is lost, alternative localization methods can be used that rely on inertial measurement units or cellular networks. However, these methods can

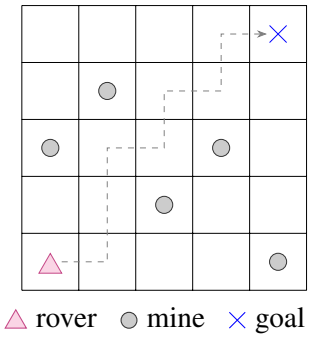

Figure 4: MineField.

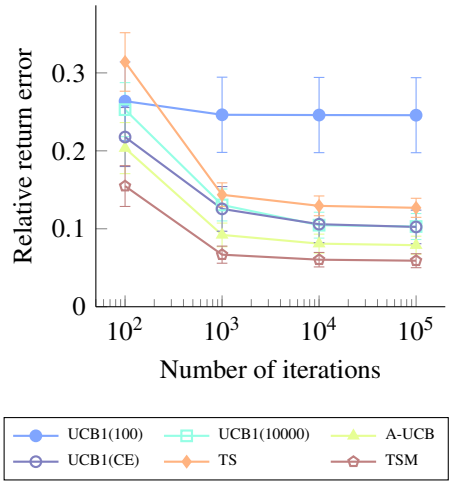

Figure 5: Relative return error from simulation of MineField.

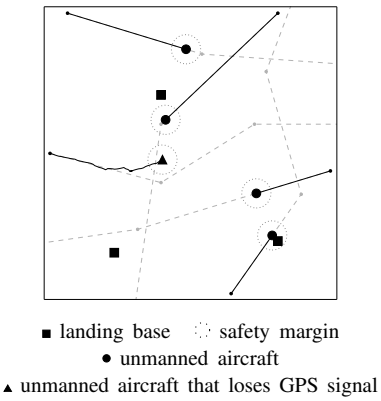

Figure 6: Rerouting of unmanned aircraft.

result in large errors in localization and navigation. If multiple unmanned aircraft are operated in the same airspace, a GPS loss can lead to mid-air collisions. To avoid collision with degraded position information, one must carefully reroute the paths of unmanned aircraft.

Figure 6 shows a scenario of the rerouting problem. There are five unmanned aircraft in 5010 ft by 5010 ft area and one of the aircraft loses GPS signal. Each aircraft follows a pre-planned path that consists of waypoints. A pre-planned path is shown as a dashed line and waypoints are shown as gray dots. There is a safety margin centered at the location of the unmanned aircraft. If an aircraft violates the safety margin of another aircraft, it is regarded as a near mid-air collision. There are emergency landing bases indicated as black squares. In the figure, trajectories of unmanned aircraft are shown as black solid lines and the unmanned aircraft that losing GPS signal deviates from its pre-planned path. The aircraft requires rerouting to avoid mid-air collision while it tries to follow its pre-planned path as far as it can.

The rerouting problem is formulated as a POMDP. States are tuples of an unmanned aircraft location, its status, last passed waypoint, and a time step. The unmanned aircraft that loses GPS signal can move toward one of its waypoints or a landing base. It observes its location with Gaussian noise and navigates based on its location estimate. It receives the reward 100 whenever it passes a waypoint and the reward -0.5 every time step while it moves. If the safety margin is violated, the large negative reward -10000 will be obtained. The unmanned aircraft losing GPS signal knows the locations of the other aircraft.

A MCTS algorithm, POMCP is used for solving the rerouting problem. POMCP (Silver and Veness 2010) is an online method for solving POMDPs that combines UCT and particle filtering for belief updating. Since the observation space of the problem is large, we use the Sparse UCT heuristic (Bjarnason et al. 2009) to limit the number of observations under each node of the search tree so that the search tree can explore more deeply.

The enhanced tree policy heuristics are applied to POMCP. Table 1 shows the relative return among different tree policies. The relative return for a tree policy in a scenario is defined by the difference between a return from the tree policy and a mean of returns from all tree policies divided by the mean. The return for a scenario is computed by averaging cumulative rewards from one hundred simulations. Computational time differences among tree policies are negligible. The table shows the average and the 95% confidence interval of relative returns from one hundred random scenarios. Two exploration constants for UCB1 are chosen according to the range of rewards without or with the near mid-air collision, respectively. A-UCB uses two UCB1 policies with the exploration constants as subpolicies. TSM sets the same prior used in MineField.

Table 1: Relative return from simulation of aircraft rerouting.

UCB1(100)	UCB1(10000)	A-UCB	UCB1(CE)	TS	TSM
-0.7230 ± 0.042	0.1723 ± 0.025	0.2378 ± 0.021	0.1812 ± 0.029	-0.1489 ± 0.026	0.2806 ± 0.023

Table 2: Relative return from simulation with splitting method.

UCB1(100)	UCB1(10000)	A-UCB	TS	TSM
0.07467 ± 0.0055	0.07560 ± 0.0056	0.1644 ± 0.0053	0.02467 ± 0.0043	0.1224 ± 0.0078

As shown in the table, A-UCB and TSM performs 24% and 28% better than the mean performance of all tree policies, respectively. A-UCB performs better than its subpolicies, UCB1(100) and UCB1(10000), and UCB1(CE).

Table 2 shows the relative return with the splitting method. The relative return for a tree policy is defined by the difference between returns from the tree policy with and without the splitting method divided by the return from the tree policy without the splitting method. Levels are defined by the distance between the unmanned aircraft losing GPS signal and an other unmanned aircraft. Among various levels and numbers of replications for levels that have been tested empirically, levels [500 ft, 200 ft] and the number of replications [2, 2] are chosen for the splitting method. The computational overhead of using the splitting method depends on the setting of the method. In the experiment, the computational time difference is within 10% between experiments with and without the splitting method. As shown in the table, returns are improved with the splitting method for all tree policies. A-UCB with the splitting method achieves 16% better return compared to the return of A-UCB without the splitting method.

The source code for this experiment is available at https://github.com/neosado/SAPR.git.

4 CONCLUSIONS

The paper presents three methods to improve Monte Carlo tree search involving rare catastrophic events. A-UCB improves the performance by adaptively applying a proper exploration constant according to the reward structure. A version of Thompson sampling with a Gaussian mixture model improves performance by exploiting the reward structure. The splitting method improves performance by reducing the variance of the estimate from Monte Carlo simulations. Empirical studies demonstrate improved performance.

Although A-UCB demonstrates its usefulness empirically, the overhead of learning which subpolicy is better than another is unknown. Theoretical analysis is required to understand the overhead and prove the performance bound of the method. In the splitting method, the performance is sensitive to the choice of splitting levels and the number of replications. Future work will involve exploring an adaptive splitting method that finds the optimal levels and the number of replications during simulation.

ACKNOWLEDGMENTS

This work is based on work supported by NASA Grant No. NNX14AI11G. The authors acknowledge the support of Guillaume P. Brat from the NASA Ames Research Center. The authors also thank David Wolpert, James Bono, and Justin Grana for helpful discussions.

REFERENCES

Agrawal, S., and N. Goyal. 2012. “Analysis of Thompson Sampling for the Multi-Armed Bandit Problem”. In *Conference on Learning Theory (COLT)*, 39.1–39.26.

Åström, K. J. 1965. "Optimal Control of Markov Processes with Incomplete State Information". *Journal of Mathematical Analysis and Applications* 10 (1): 174–205.

Auer, P., N. Cesa-Bianchi, and P. Fischer. 2002. "Finite-Time Analysis of the Multiarmed Bandit Problem". *Machine Learning* 47 (2-3): 235–256.

Bai, A., F. Wu, and X. Chen. 2013. "Bayesian Mixture Modelling and Inference based Thompson Sampling in Monte-Carlo Tree Search". In *Advances in Neural Information Processing Systems (NIPS)*, 1646–1654.

Bai, A., F. Wu, Z. Zhang, and X. Chen. 2014. "Thompson Sampling based Monte-Carlo Planning in POMDPs". In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Bellman, R. 1957. "A Markovian Decision Process". *Journal of Mathematics and Mechanics* 6:679–684.

Berry, D. A., and B. Fristedt. 1985. *Bandit Problems: Sequential Allocation of Experiments*. Springer.

Bjarnason, R., A. Fern, and P. Tadepalli. 2009. "Lower Bounding Klondike Solitaire with Monte-Carlo Planning". In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Browne, C., E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. 2012. "A Survey of Monte Carlo Tree Search Methods". *IEEE Transactions on Computational Intelligence and AI in Games* 4 (1): 1–43.

Chapelle, O., and L. Li. 2011. "An Empirical Evaluation of Thompson Sampling". In *Advances in Neural Information Processing Systems (NIPS)*, 2249–2257.

Chaslot, G. M. J.-B., M. H. M. Winands, I. Szita, and H. J. van den Herik. 2008. "Cross-Entropy for Monte-Carlo Tree Search". *ICGA Journal* 31 (3): 145–156.

Coulom, R. 2006. "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search". In *Computers and Games (CG)*, 72–83.

De Boer, P.-T., D. P. Kroese, S. Mannor, and R. Y. Rubinstein. 2005. "A Tutorial on the Cross-Entropy Method". *Annals of Operations Research* 134 (1): 19–67.

Garvels, M. J. J. 2000. *The Splitting Method in Rare Event Simulation*. Ph. D. thesis, University of Twente.

Gittins, J. C. 1989. *Multi-Armed Bandit Allocation Indices*. John Wiley & Sons.

Howard, R. A. 1960. *Dynamic Programming and Markov Processes*. MIT Press.

Kim, Y., M. J. Kochenderfer, J. Grana, J. Bono, and D. Wolpert. 2015. "Optimal Lost-Link Policies for Unmanned Aircraft". In *Digital Avionics Systems Conference (DASC)*.

Kochenderfer, M. J. 2015. *Decision Making under Uncertainty: Theory and Application*. MIT Press.

Kocsis, L., and C. Szepesvári. 2006. "Bandit based Monte-Carlo Planning". In *European Conference on Machine Learning (ECML)*, 282–293.

Kocsis, L., C. Szepesvári, and J. Willemson. 2006. "Improved Monte-Carlo Search". Technical Report 1, University of Tartu.

Lai, T. L., and H. Robbins. 1985. "Asymptotically Efficient Adaptive Allocation Rules". *Advances in Applied Mathematics* 6 (1): 4–22.

May, B. C., and D. S. Leslie. 2011. "Simulation Studies in Optimistic Bayesian Sampling in Contextual-Bandit Problems". Technical report, Statistics Group, Department of Mathematics, University of Bristol.

Prandini, M., H. A. P. Blom, and G. J. B. Bakker. 2011. "Air Traffic Complexity and the Interacting Particle System Method: An Integrated Approach for Collision Risk Estimation". In *American Control Conference (ACC)*, 2154–2159.

Robbins, H. 1952. "Some Aspects of the Sequential Design of Experiments". *Bulletin of the American Mathematical Society* 58 (5): 527–535.

Rubino, G., and B. Tuffin. 2009. *Rare Event Simulation using Monte Carlo Methods*. John Wiley & Sons.

Rubinstein, R. Y., and D. P. Kroese. 2004. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer.

Sani, A., G. Neu, and A. Lazaric. 2014. "Exploiting Easy Data in Online Optimization". In *Advances in Neural Information Processing Systems (NIPS)*, 810–818.

Seldin, Y., and A. Slivkins. 2014. "One Practical Algorithm for Both Stochastic and Adversarial Bandits". In *International Conference on Machine Learning (ICML)*, 1287–1295.

Silver, D., and J. Veness. 2010. "Monte-Carlo Planning in Large POMDPs". In *Advances in Neural Information Processing Systems (NIPS)*, 2164–2172.

Sondik, E. J. 1971. *The Optimal Control of Partially Observable Markov Processes*. Ph. D. thesis, Stanford University.

Thompson, W. R. 1933. "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples". *Biometrika* 25 (3-4): 285–294.

Veness, J., M. Lanctot, and M. Bowling. 2011. "Variance Reduction in Monte-Carlo Tree Search". In *Advances in Neural Information Processing Systems (NIPS)*, 1836–1844.

AUTHOR BIOGRAPHIES

YOUNGJUN KIM received his Ph.D. in Aeronautics and Astronautics from Stanford University. His research interests include optimal planning, collision avoidance, and unmanned aircraft. His email address is youngjun@stanford.edu.

YONATAN GUR is an Assistant Professor of Operations, Information and Technology at Stanford University's Graduate School of Business. His primary research interests are in business analytics, pricing and revenue management, auction design, statistical and machine learning, and optimization, with a particular emphasis on data-driven decision making in e-commerce, online recommendation systems, online advertising, and other dynamic and competitive marketplaces that are characterized by uncertainty. Prior to joining Stanford in 2014, he received his Ph.D. in Decision, Risk, and Operations from Columbia Business School. He also holds a B.Sc. degree from the School of Physics and Astronomy and an M.Sc. from the School of Mathematical Sciences, Tel Aviv University. His email address is ygur@stanford.edu.

MYKEL J. KOCHENDERFER is an Assistant Professor of Aeronautics and Astronautics at Stanford University. His primary research interests are in models and algorithms for decision making under uncertainty with applications to areas such as automated driving and unmanned aircraft. Prior to joining Stanford in 2013, he was a member of the technical staff at MIT Lincoln Laboratory. He received his Ph.D. in Informatics from the University of Edinburgh. He also holds B.S. and M.S. degrees in Computer Science from Stanford University. His email address is mykel@stanford.edu.