Extreme Value Monte Carlo Tree Search for Classical Planning

Masataro Asai[*1], **Stephen Wissow**[*2]
[1]MIT-IBM Watson AI Lab
[2]University of New Hampshire
masataro.asai@ibm.com, sjw@cs.unh.edu

Abstract

Despite being successful in board games and reinforcement learning (RL), Monte Carlo Tree Search (MCTS) combined with Multi-Armed Bandits (MABs) has seen limited success in domain-independent classical planning until recently. Previous work (Wissow and Asai 2024) showed that UCB1, designed for bounded rewards, does not perform well as applied to cost-to-go estimates in classical planning, which are unbounded in $\mathbb{R}$, and showed improved performance using a Gaussian reward MAB instead. This paper further sharpens our understanding of ideal bandits for planning tasks. Existing work has two issues: first, Gaussian MABs under-specify the support of cost-to-go estimates as $(-\infty, \infty)$, which we can narrow down. Second, Full Bellman backup (Schulte and Keller 2014), which backpropagates sample max/min, lacks theoretical justification. We use *Peaks-Over-Threashold Extreme Value Theory* to resolve both issues at once, and propose a new bandit algorithm (UCB1-Uniform). We formally prove its regret bound and empirically demonstrate its performance in classical planning.

1 Introduction

A recent breakthrough (Wissow and Asai 2024) in Monte Carlo Tree Search (MCTS) combined with Multi-Armed Bandit (MAB) demonstrated that a better theoretical understanding of bandit-based algorithms can significantly improve search performance in classical planning (Fikes, Hart, and Nilsson 1972). Building upon the Trial-Based Heuristic Tree Search (THTS) framework (Schulte and Keller 2014), Wissow and Asai (2024) showed why the UCB1 bandit (Auer, Cesa-Bianchi, and Fischer 2002) does not perform well in classical planning: UCB1 assumes a reward distribution with a known, fixed, finite support (a mathematical term for a defined range such as $[0, 1]$) that is shared by all arms, incorrectly assuming that cost-to-go estimates (heuristic values) always fall in this particular range. They then proposed UCB1-Normal2 bandit that assumes a Gaussian reward distribution which has an infinite support $(-\infty, \infty)$ that is impossible to violate, and has a regret bound that can become constant when applied to deterministic state space search, as in classical planning.

[*]These authors contributed equally.

We build on these advances to further our understanding of the strengths and requirements of MABs as applied to heuristic search, and in particular to resolve two theoretical issues in previous work in this area. The first is UCB1-Normal2's assumption that cost-to-go estimates fall anywhere in $(-\infty, \infty)$, which is an under-specification that can be narrowed down to $[0, \infty)$ or even further. The second is the insufficient statistical characterization of *extrema* (maximum/minimum) in so-called *Full Bellman* backup (Schulte and Keller 2014) that backpropagates the smallest/largest mean among the arms. Schulte and Keller informally criticized the use of averages in UCT as "rather odd" for planning, but without bandit-theoretic justifications.

This paper introduces Extreme Value Theory (Beirlant et al. 2004; De Haan and Ferreira 2006, EVT) as the statistical foundation for understanding general optimization tasks. EVTs are designed to model the statistics of extrema of distributions using the *Extremal Limit Theorems*, unlike most statistical literature that models the *average* behavior based on the *Central Limit Theorem* (Laplace 1812, CLT). Among branches of EVTs, we identified *Peaks-Over-Threashold EVT* (Pickands III 1975; Balkema and De Haan 1974) as our primary tool for designing new algorithms, leading us to the Generalized Pareto (GP) distribution, which plays the same role in EVT as the Gaussian distribution does in the CLT. Based on this framework, we propose a novel MAB algorithm called UCB1-Uniform for heuristic search applied to classical planning, using the fact that the Uniform distribution is a special case of the GP distribution to avoid the numerical difficulty of estimating the latter's parameters. We propose a novel heuristic search algorithm for classical planning, GreedyUCT-Uniform (GUCT-Uniform), an MCTS that leverages UCB1-Uniform.

We compared GUCT-Uniform's performance against various existing bandit-based MCTS algorithms, traditional Greedy Best First Search (Bonet and Geffner 2001; Doran and Michie 1966, GBFS), and a state-of-the-art diversified search algorithm called Softmin-Type(h) (Kuroiwa and Beck 2022). The results showed that our algorithm outperforms existing state-of-the-art algorithms across diverse heuristics. For example, under the same evaluation budget of 10^4 nodes with the h^{FF} heuristic (Bonet and Geffner 2001), GUCT-Uniform solved 67.8, 23.4, and 33.2 more instances than GBFS, GUCT-Normal2, and Softmin-Type(h), respectively.

GUCT-Uniform also significantly outperformed MCTS variants combined with Max-k bandits (Cicirello and Smith 2004), a bandit paradigm whose objective differs significantly from those of classical planning. MCTS combined with Max-k bandits (MaxSearch (Streeter and Smith 2006a), RobustUCT (Bubeck, Cesa-Bianchi, and Lugosi 2013), and Threshold Ascent (Kikkawa and Ohno 2022)) performed poorly in the classical planning task, outperformed by GUCT-Uniform by more than 300 instances. Our code is published at github.com/guicho271828/pyperplan-mcts. A full version of the paper with appendix is on arxiv:2405.18248.

2 Preliminaries

We define a propositional STRIPS Planning problem as a 4-tuple $\langle P, A, I, G\rangle$ where P is a set of propositional variables, A is a set of actions, $I \subseteq P$ is the initial state, and $G \subseteq P$ is a goal condition. We omit the details of action applications as they are not important in this paper. It suffices to say an action $a \in A$ transitions from a state $s \subseteq P$ to a successor $s' = a(s) \subseteq P$. The task of classical planning is to find a sequence of actions called a *plan* $(a_1, \cdots, a_n)$ where, for $1 \leq t \leq n$, $s_0 = I$, $s_{t+1} = a_{t+1}(s_t)$, and $s_n \supseteq G$. A plan is *optimal* if there is no shorter plan. A plan is otherwise called *satisficing*. A problem setting that completely ignores the solution quality is called an *agile* setting, while a *satisficing* setting implies that the solver still attempts to find a shorter plan. This paper focuses on the *agile* setting.

A domain-independent heuristic function h in classical planning is a function of a state s and the problem $\langle P, A, I, G\rangle$, though the notation $h(s)$ usually omits the latter, that returns an estimate of the cumulative cost of a sequence of actions transitioning from s to a goal state $s_g \supseteq G$. Details of specific heuristic functions are beyond the scope of this paper, and are included in the appendix.

2.1 Multi-Armed Bandit (MAB)

MAB (Thompson 1933; Robbins 1952; Bush and Mosteller 1953) is the problem of finding the best strategy to choose from multiple unknown reward distributions. It is typically depicted by a row of K slot machines each with a lever or 'arm.' Each time the player pulls an arm (a *trial*), they receive a reward sampled from that arm's reward distribution. Through multiple trials, the player discovers the arms' distributions and selects arms to maximize the reward.

The most common optimization objective of MAB is *Cumulative Regret* (CR) minimization. Let r_{it} $(1 \leq i \leq K)$ be a random variable (RV) for the reward received from the t-th pull of an arm i. r_{it} follows an unknown *reward distribution* $p(\mathrm{r}_i)$ which stays the same over t. Let t_i be the number of pulls on arm i when $T = \sum_i t_i$ pulls are performed in total.

Definition 1. *Let I_t be the arm pulled at t. The* cumulative regret Δ *is the gap between the optimal and the actual expected cumulative reward:* $\Delta = \max_i \mathbb{E}[\sum_{t=1}^{T} \mathrm{r}_{it}] - \mathbb{E}[\sum_{t=1}^{T} \mathrm{r}_{I_t t}]$.

A regret bound indicates the *speed* of convergence. Algorithms with a logarithmically upper-bounded regret, $O(\log T)$, are called *asymptotically optimal* because this is the theoretical optimum achievable by any algorithm (Lai, Robbins et al. 1985).

Upper Confidence Bound 1 (Auer, Cesa-Bianchi, and Fischer 2002, UCB1) is a logarithmic CR MAB for rewards $\mathrm{r}_i \in [0, c]$ with a known c. Let $r_{i1}, \ldots, r_{it_i} \sim p(\mathrm{r}_i)$ be t_i i.i.d. samples obtained from an arm i. Let $\hat{\mu}_i = \frac{1}{t_i}\sum_{j=1}^{t_i} r_{ij}$. To minimize CR, UCB1 selects i with the largest Upper Confidence Bound value $\mathrm{UCB1}_i$:

$$\begin{aligned} \mathrm{UCB1}_i &= \hat{\mu}_i + c\sqrt{(2\log T)/t_i} \\ \mathrm{LCB1}_i &= \hat{\mu}_i - c\sqrt{(2\log T)/t_i} \end{aligned} \tag{1}$$

For reward (cost) minimization, we can select i with the smallest $\mathrm{LCB1}_i$ value defined above (e.g., in Kishimoto et al. (2022)), but we may use the terms U/LCB1 interchangeably.

U/LCB1's second term is often called an *exploration term*. In practice, c is often set heuristically as a hyperparameter and referred to as the *exploration rate*, ignoring the original theoretical meaning as the upper limit of support $[0, c]$.

U/LCB1 refers to a specific algorithm proposed by Auer, Cesa-Bianchi, and Fischer (2002), while U/LCB refers to general upper/lower confidence bounds of random variables. Often an LCB subtracts the exploration term instead of adding it as in a UCB.

2.2 Forward Heuristic Best-First Search

Classical planning problems are typically solved as a path finding problem defined over a state space graph induced by the transition rules, and the current dominant approach is based on *forward search*. Forward search maintains a set of search nodes called an *open list*, and repeatedly (1) (*selection*) selects a node from the open list, (2) (*expansion*) generates its successor nodes, (3) (*evaluation*) evaluates the successor nodes, and (4) (*queueing*) reinserts them into the open list. Termination typically occurs when the node selected for expansion satisfies a goal condition, but a satisficing/agile algorithm can perform *early goal detection*, which immediately checks whether any successor node generated in step (2) satisfies the goal condition. Since this paper focuses on agile search, we use early goal detection for all algorithms.

Within forward search, forward *best-first* search defines a particular ordering in the open list by defining *node evaluation criteria* (NEC) f for selecting the best node in each iteration. Let us denote a node by n and the state represented by n as s_n. As NEC, Dijkstra search (Dijkstra 1959) uses $f_{\mathrm{Dijkstra}}(n) = g(n)$ (g-value), the minimum cost from the initial state I to the state s_n found so far. A* (Hart, Nilsson, and Raphael 1968) uses $f_{A^*}(n) = g(n) + h(s_n)$, the sum of g-value and the value returned by a heuristic function h (h-value). Greedy Best First Search (Bonet and Geffner 2001, GBFS) uses $f_{\mathrm{GBFS}}(n) = h(s_n)$. Forward best-first search that uses h is called forward *heuristic* best-first search. Dijkstra search is a special case of A* with $h(s) = 0$.

Typically, an open list is implemented as a priority queue ordered by NEC. Since the NEC can be stateful, e.g., $g(s_n)$ can update its value, a priority queue-based open list, depending on implementation, may have unfavorable time complexity for removals and thus may assume monotonic updates to the NEC. A*, Dijkstra, and GBFS satisfy this condition because $g(n)$ decreases monotonically and $h(s_n)$ is constant.

MCTS is a class of forward heuristic best-first search that represents the open list as the leaves of a tree. We call such a tree a *tree-based open list*. Our MCTS is based on the description in Keller and Helmert (2013) and Schulte and Keller (2014), whose implementation details are available in the appendix. Overall, MCTS works in the same manner as other best-first searches with a few key differences. (1) (*selection*) To select a node from the tree-based open list, it recursively selects an action at each depth level of the tree, starting from the root, using the NEC to select a successor node, descending until reaching a leaf node. (Sometimes the action selection rule is also called a *tree policy*.) At the leaf, it (2) (*expansion*) generates successor nodes, (3) (*evaluation*) evaluates the new successor nodes, (4) (*queueing*) attaches them to the leaf, and *backpropagates* (or *backs-up*) the information to the leaf's ancestors, all the way up to the root.

The evaluation obtains a heuristic value $h(s_n)$ of a leaf node n. In adversarial games like Backgammon or Go, it is obtained either by (1) hand-crafted heuristics, (2) *playouts* (or *rollouts*) where the behaviors of both players are simulated by (e.g. uniformly) random actions (*default policy*) until the game terminates, or (3) a hybrid *truncated simulation*, which returns a hand-crafted heuristic after performing a short simulation (Gelly and Silver 2011). In recent work, the default policy is replaced by a learned policy (Silver et al. 2016).

Trial-based Heuristic Tree Search (Keller and Helmert 2013; Schulte and Keller 2014, THTS), an MCTS for classical planning, is based on two key observations: (1) the rollout is unlikely to terminate in classical planning due to sparse goals, unlike adversarial games, like Go, which are guaranteed to finish in a known number of steps with a clear outcome (win/loss); and (2) a tree-based open list can efficiently reorder entire subtrees of nodes, and thus is more flexible than a priority queue-based open list, and can readily implement traditional algorithms such as A* and GBFS without significant performance penalty. In this paper, we use THTS and MCTS interchangeably.

Finally, Upper Confidence Bound applied to trees (Kocsis and Szepesvári 2006, UCT) is an MCTS that uses the UCB1 Multi-Armed Bandit algorithm for action selection and became widely popular in adversarial games. Schulte and Keller (2014) proposed several variants of UCT including GreedyUCT (GUCT), which differs from UCT in that the NEC assigned to the node is simply its heuristic value $h(s_n)$ just like in GBFS, rather than the f-value ($f = g(n)+h(s_n)$). This paper only discusses the greedy variants due to our focus on agile planning.

3 Heuristic Search with MABs

We first revisit GBFS and GUCT from an MAB perspective. While Keller and Helmert (2013) generalized various algorithms focusing on the procedural aspects (e.g., recursive backup), we focus on their mathematical meaning.

Definition 2 (NECs). *Let $S(n)$ be the successors of a node n, $L(n)$ be the leaf nodes in the subtree under n. The NEC of GBFS and GUCT are shown below, where p is n's parent, and thus $|L(p)|$ and $|L(n)|$ correspond to T and t_i in Eq. 1.*

$$f_{\mathrm{GBFS}}(n) = h_{\mathrm{GBFS}}(n)$$

$$f_{\mathrm{GUCT}}(n) = h_{\mathrm{GUCT}}(n) - c\sqrt{(2\log|L(p)|)/|L(n)|}$$

MCTS/THTS computes $h_{\mathrm{GBFS}}(n)$, $h_{\mathrm{GUCT}}(n)$ using backpropagation. Below, we expand the definitions of two backup functions presented by Keller and Helmert (2013) recursively down to the leaves, assuming $h_{\mathrm{GBFS}}(n) = h_{\mathrm{GUCT}}(n) = h(s_n)$ if n is a leaf.

Definition 3 (Full Bellman Backup)**.**

$$\begin{aligned} h_{\mathrm{GBFS}}(n) &= \min\nolimits_{n'\in S(n)}[h_{\mathrm{GBFS}}(n')] \\ &= \min\nolimits_{n'\in S(n)}[\min\nolimits_{n''\in S(n')}[h_{\mathrm{GBFS}}(n'')]] \\ &= \ldots = \min\nolimits_{n'\in L(n)}[h(s_{n'})]. \end{aligned}$$

Definition 4 (Monte Carlo Backup)**.**

$$\begin{aligned} h_{\mathrm{GUCT}}(n) &= \textstyle\sum_{n'\in S(n)} \frac{|L(n')|}{|L(n)|} h_{\mathrm{GUCT}}(n') \\ &= \textstyle\sum_{n'\in S(n)} \frac{\cancel{|L(n')|}}{|L(n)|} \sum_{n''\in S(n')} \frac{|L(n'')|}{\cancel{|L(n')|}} h_{\mathrm{GUCT}}(n'') \\ &= \ldots = \textstyle\frac{1}{|L(n)|}\sum_{n'\in L(n)} h(s_{n'}). \end{aligned}$$

Notice that each backup is equivalent to simply computing the minimum or the weighted mean over all leaves in the subtree, where each leaf n' has $|L(n')| = 1$ in classical planning. In other words, the set $\{h(s_{n'}) \mid n' \in L(n)\}$ is a *reward dataset*, the heuristic $h(s_{n'})$ at each leaf n' is a *reward sample* in the dataset, and the NECs use their *statistics*, such as the mean and the minimum, estimated by Maximum Likelihood Estimation (MLE). Backpropagation is just an effective way to update and cache the statistics.

Theorem 1. *Given i.i.d. $x_1, \ldots, x_N \sim \mathcal{N}(\mathrm{x}|\mu,\sigma)$ (i.e., $\mathrm{x} \sim \mathcal{N}(\mu,\sigma)$), the MLEs of μ and σ are the empirical mean $\hat{\mu} = \frac{1}{N}\sum_i x_i$ and variance $\hat{\sigma}^2 = \frac{1}{N-1}\sum_i(x_i - \hat{\mu})$. (Well-known result. Educational proof in appendix.)*

Understanding each $h(s)$ as a sample of a random variable representing a reward for MABs makes it clear that existing MCTS/THTS for classical planning fails to leverage the theoretical efficiency guarantees from the rich MAB literature. For example, if we apply UCB1 to heuristic values in classical planning, UCB1 no longer guarantees asymptotically optimal convergence toward the best arm because it incorrectly assumes $h \in [0, c]$ for a fixed hyperparameter c, i.e., that h has an *a priori* known constant range $[0, c]$, which in fact does not exist (h varies significantly across states, and can be ∞). This cannot be fixed by simply making c larger.

Wissow and Asai (2024) proposed GUCT-Normal and GUCT-Normal2, MCTS algorithms for classical planning that use Gaussian bandits UCB1-Normal (Eq. 2) (Auer, Cesa-Bianchi, and Fischer 2002) and UCB1-Normal2 (Eq. 3), motivated by the Gaussian distribution's inclusive support range $\mathbb{R}$, and in effect these bandits use the h sample variance to dynamically estimate UCB1's c, i.e., the 'exploration rate'. Given $\mathrm{r}_i \sim \mathcal{N}(\mu_i, \sigma_i)$, let $\hat{\mu}_i$ and $\hat{\sigma}_i$ be the MLEs of μ_i, σ_i of arm i.

$$\text{U/LCB1-Normal}_i = \hat{\mu}_i \pm \hat{\sigma}_i\sqrt{(16\log T)/t_i}. \quad (2)$$

$$\text{U/LCB1-Normal2}_i = \hat{\mu}_i \pm \hat{\sigma}_i\sqrt{2\log T}. \quad (3)$$

Each $(\hat{\mu}_i, \hat{\sigma}_i)$ corresponds to the average and the standard deviation of the dataset $\{h(s_{n'}) \mid n' \in L(n)\}$ of a node

n. GUCT-Normal2 outperformed GBFS, GUCT, GUCT-Normal, and other variance-aware bandits.

Although GUCT-Normal2 explored better than existing algorithms while not violating assumptions about the reward range, it still does not fully characterize the nature of heuristic functions, as we describe below.

Under-Specification GUCT over-specifies the rewards to be in a fixed range $[0, c]$. While $\mathcal{N}(\mu, \sigma)$ does not have this issue, its support $\mathbb{R} = (-\infty, \infty)$ is an under-specification because heuristic values are *non-negative*, $\mathbb{R}^+ = [0, \infty)$.

Moreover, the range can be narrowed down further. For example, the FF heuristic (Hoffmann and Nebel 2001) satisfies $h^{\mathrm{FF}} \in [h^+, \infty)$, i.e., lower bounded by optimal delete relaxation heuristic h^+, though this value is **NP**-complete to compute (Bylander 1994) and thus in practice *unknown*. Similarly, the $h^{\max}$ heuristic is bounded by $[0, h^+]$. Finally, h^+ can be ∞ when the state is at a dead-end. This indicates that the support of a heuristic function is generally *unknown and half-bounded*. Choosing an appropriate distribution, and a corresponding MAB that correctly leverages its properties, should make MCTS faster. A similar statistical modeling flaw was recently discussed in supervised heuristic learning (Núñez-molina et al. 2024).

Estimating the Minimum Another issue in existing work is the use of the minimum (Full Bellman backup) in the GUCT*-family of algorithms (Schulte and Keller 2014), which lacks statistical justification, in particular a theoretical explanation of *why* using the minimum is allowed. Regardless of whether rewards have finite-support or follow a Gaussian distribution, the mean μ_i, not the minimum, is inextricable from the design of and regret bound proofs for UCB1/-Normal/2. In contrast, the theoretical framework we present in the next section addresses this conflict.

One candidate for addressing the extrema of reward distributions was proposed by Tesauro, Rajan, and Segal (2010). Given two Gaussian RVs $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, the backup uses their maximum $\max(X_1, X_2) \sim \mathcal{N}(\mu_3, \sigma_3)$ where $\mu_3 = \mu_1\Phi(\alpha)+\mu_2\Phi(-\alpha)+(\sigma_1^2+\sigma_2^2)\phi(\alpha)$ and $\sigma_3 = (\mu_1^2+\sigma_1^2)\Phi(\alpha)+(\mu_2^2+\sigma_2^2)\Phi(-\alpha)+(\mu_1+\mu_2)(\sigma_1^2+\sigma_2^2)\phi(\alpha)$, where Φ and ϕ are the CDF and the PDF of $\mathcal{N}(0, 1)$ (Clark 1961). Unfortunately, this is merely an approximation if we combine the estimates iteratively for more than two arms, as noted in (2010). In our experiments, we implemented this backup and call it GUCT$^+$ variants.

Dead-End Removal h^{FF} lacks an upper bound and could return ∞ when a node is a dead-end, which is problematic for MABs that use the average of rewards, including both UCB1 and UCB1-Normal2. Imagine that an arm returned rewards 3, 5, 4 and ∞ in order. Once an arm receives the fourth reward ∞, then the estimated mean suddenly becomes ∞ regardless of those observed previously—i.e., $\frac{3+5+4+\infty}{4} = \infty$—falsely masking potential solutions in the subtree. Schulte and Keller (2014) recognized this issue and decided to exclude ∞ from rewards by removing dead-end nodes from the tree, but this approach lacks statistical, MAB-theoretic justification as to why such a removal of a value from the sample is allowed.

4 Extreme Value Theory (EVT)

To address these theoretical issues, we use Peaks-Over-Threshold Extreme Value Theory (POT EVT). Regular statistics are typically built around the Central Limit Theorem (Laplace 1812, CLT), which deals with the limit behavior of the average of samples. In contrast, a branch of statistics called *Extreme Value Theory* (Beirlant et al. 2004; De Haan and Ferreira 2006, EVT) describes the limit behavior of the maximum of samples. EVT has been historically used for safety-critical applications whose worst case behaviors matter. For example, in hydrology, the estimated annual maximum water level of a river is used to decide the height for an embankment. We explain the EVT by way of first reviewing the CLT.

Definition 5. *A series of functions* f_n converges pointwise *to* f *(*$f_n \xrightarrow{n\to\infty} f$*) when* $\forall x; \forall \epsilon; \exists n; |f_n(x) - f(x)| < \epsilon$.

Definition 6. *A series of RVs* x_n converges in distribution *to a RV* x *if* $p(\mathrm{x}_n) = f_n \xrightarrow{n\to\infty} f = p(\mathrm{x})$*, denoted as* $\mathrm{x}_n \xrightarrow{D} \mathrm{x}$.

Theorem 2 (CLT). *Let* $\mathrm{x}_1, \dots \mathrm{x}_n$ *be i.i.d.,* $\forall i; \mathbb{E}[\mathrm{x}_i] = \mu$, $\mathrm{Var}[\mathrm{x}_i] = \sigma^2$*. Then* $\sqrt{n}\left(\frac{\sum_i \mathrm{x}_i}{n} - \mu\right) \xrightarrow{D} \mathrm{y} \sim \mathcal{N}(0, \sigma)$*. I.e., if* x_i*'s distribution* $p(\mathrm{x}_i)$ *has a finite mean and variance, the average of* $\mathrm{x}_1 \dots \mathrm{x}_n$ *converges (*$n \to \infty$*) in distribution to a Gaussian with the same mean/variance, regardless of other details of* $p(\mathrm{x}_i)$*, e.g., shape or support.*

A common misunderstanding is that CLT assumes each RV x_i to follow a Gaussian (untrue). CLT's strength comes from its minimal assumption that x_i are i.i.d. and share a finite μ and σ, and nothing else. x_i can follow Laplace, but not Cauchy (mean and variance are undefined). In heuristic search, x_i is a random choice from $\{h(s_{n'})\}$ in a subtree of a node n. Its mean and variance must be finite; therefore $\{h(s_{n'})\}$ should not contain ∞ as it makes the average ∞. But it does not require each h to follow a Gaussian (each h is indeed a Dirac delta $\delta(\mathrm{x} = h)$, a deterministic value of a state), nor the histogram of $\{h(s_{n'})\}$ to resemble a Gaussian; Only the mean and the variance matter.

EVT has two limit theorems similar to the CLT, called the Extremal Limit Theorems (ELT). The first kind, the *Block Maxima* ELT (Fisher and Tippett 1928; Gnedenko 1943), states that the maximum of i.i.d. RVs converges in distribution to an *Extreme Value Distribution*. Given multiple subsets of data points, it models the maximum of the next subset (*block maxima*), e.g., it predicts the maximum of the next month from the maxima of past several months. However, what we use is the second kind, *Peaks-Over-Threshold* (POT) ELT (Pickands III 1975; Balkema and De Haan 1974), which states that the excesses of i.i.d. RVs over a sufficiently high threshold θ converge in distribution to a Generalized Pareto (GP) distribution (Fig. 1), predicting *future excesses over* θ.

Definition 7 (Generalized Pareto Distribution)**.**

$$\mathrm{GP}(\mathrm{x} \mid \theta, \sigma, \xi) = \begin{cases} \frac{1}{\sigma}\left(1 + \xi\frac{\mathrm{x}-\theta}{\sigma}\right)^{-\frac{\xi+1}{\xi}} & (\xi \neq 0) \\ \frac{1}{\sigma}\exp\left(-\frac{\mathrm{x}-\theta}{\sigma}\right) & (\xi = 0) \end{cases} \quad (\mathrm{x} > \theta)$$

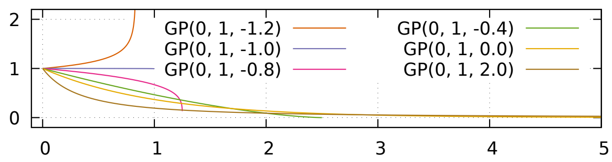

Figure 1: Generalized pareto distribution $\mathrm{GP}(0, 1, \xi)$.

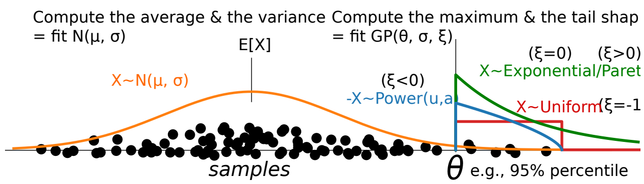

Figure 2: Computing the average and the variance is seen as fitting $\mathcal{N}(\mu, \sigma)$; Computing the maximum and the shape of the tail distribution is seen as fitting $\mathrm{GP}(\mu, \sigma, \xi)$ with $\xi < 0$.

Theorem 3 (Pickands–Balkema–de Haan theorem)**.** *Let* $\mathrm{x}_1, \ldots, \mathrm{x}_n \sim p(\mathrm{x})$ *be i.i.d. RVs and* $\mathrm{x}_{k,n} = \theta$ *be their top-k elements. As* $n \to \infty$*,* $k \to \infty$*,* $\frac{k}{n} \to 0$ *(*$k \ll n$*), then* $p(\mathrm{x} \mid \mathrm{x} > \theta) \xrightarrow{D} \mathrm{GP}(\mathrm{x} \mid \theta, \sigma, \xi)$ *for some* $\sigma \in \mathbb{R}^+, \xi \in \mathbb{R}$*, regardless of other details of* $p(\mathrm{x}_i)$*, e.g., shape or support.*

θ, σ, and ξ are called the location, scale, and shape parameters. GP has support $\mathrm{x} \in [\theta, \theta - \frac{\sigma}{\xi}]$ when $\xi < 0$, otherwise $\mathrm{x} \in [\theta, \infty)$. The shape dictates the tail behavior: $\xi > 0$ corresponds to a heavy-tailed distribution, $\xi < 0$ corresponds to a short-tailed distribution (i.e., has an upper limit), and $\xi = 0$ corresponds to an Exponential distribution. Pareto, Exponential, Reverted Power, and Uniform distributions are special cases of GP.

Fig. 2 shows a conceptual illustration of POT. In the standard statistical modeling, practitioners often compute the average and the standard deviation of the data to fit $\mathcal{N}(\mu, \sigma)$, which models the "normative" behavior of samples that appears at the center of the distribution. In contrast, POT models rare events occuring in the *tail distribution*. Practitioners first extract a top-k subset of samples in various ways, e.g., setting a threshold θ, selecting the top 5%, or directly specifying k, then fit the parameters σ, ξ of $\mathrm{GP}(\theta, \sigma, \xi)$ on this subset, which predicts future excesses. GP is accurate when we retain (extract) enough data $k \to \infty$ as $n \to \infty$ while ignoring almost all data ($\frac{k}{n} \to 0$). For example, estimates from top-1% examples ($\frac{k}{n} = 0.01$) tend to be more accurate than those from top-5% ($\frac{k}{n} = 0.05$), if k is the same.

EVTs are appealing because state-of-the-art search algorithms such as GBFS are based on the minimum. It is also worth noting that the short-tailed GP ($\xi < 0$) resolves the shortcomings discussed in Sec. 3. Consider a maximization scenario (as GP models the maxima), where we negate the heuristic values into rewards $-h^{\mathrm{FF}} \in (-\infty, -h^+]$. By fitting σ and ξ to the data, a short-tailed GP gives us an upper support $\theta - \frac{\sigma}{\xi}$, which works as an estimate of $-h^+$. GP also *justifies* discarding $-h^{\mathrm{FF}}$ below θ, including the dead-ends $-h^{\mathrm{FF}} = -\infty$, because GP is conditioned (only) by $\mathrm{x} > \theta$.

One difficulty of GP is its parameter estimation, which has been extensively studied with varying success (Smith 1987; Hill 1975; Resnick and Stărică 1997; Hosking and Wallis 1987; Diebolt et al. 2005; Sharpe and Juárez 2021). To avoid this, we focus on the uniform distribution $U(l, u)$ with an unknown support $[l, u]$, sacrificing one degree of freedom: It is a special case with $\xi = -1$. Note that POT does not assume any distribution (just like CLT); i.e., we do not assume heuristic values follow $U(l, u)$ or GP.

Definition 8. *The Uniform distribution is defined as follows.*

$$U(\mathrm{x}|l, u) = \tfrac{1}{u-l}. \quad (l \le x \le u) \qquad \mathbb{E}[\mathrm{x}] = \tfrac{l+u}{2}.$$

Theorem 4. $\mathrm{GP}(\theta, \sigma, -1) = U(\theta, \theta + \sigma)$. *(proof omitted)*

One last bit of the detail is how to set θ, but we actually do not use any explicit threshold. Observe that in heuristic search, the search is already heavily focused toward the goal. The search space (n nodes) is exponentially large and mostly unexplored, so the observed nodes (k nodes) are the tiny fraction ($k \ll n$) with small heuristic values relative to the entire state space. Visiting a state s with $h(s) > h(I)$ of initial state I tends to be rare. We confirmed this with GBFS+h^{FF} in the benchmark (Sec. 6): Only 3.3% of the evaluated nodes had $h(s) > h(I)$, thus bad nodes are already rare in the reward set. As a result, our implementation omits explicit filtering of samples, relying on implicit filtering from not expanding such bad states. The only explicit rule is the dead-end removal. Future work could theoretically justify the pruning with the cost of an incumbent solution in iterated anytime search (Richter, Thayer, and Ruml 2010).

5 Bandit for Uniform Distributions

To define our POT-based search algorithm, we first review the Maximum Likelihood Estimates (MLEs) of Uniform distributions, then propose a bandit that uses these estimates, which is then used by MCTS as its NEC.

Theorem 5 (MLE of Uniform)**.** *Given i.i.d.* $x_1, \ldots, x_N \sim U(\mathrm{x}|l, u)$*, the MLEs are* $\hat{u} = \max_i x_i$ *and* $\hat{l} = \min_i x_i$*. (Well-known result. Educational proof in appendix.)*

In MCTS, we backpropagate these estimates from the leaves to the root: i.e., for $\hat{l}$ and $\hat{u}$ we use Full Bellman backup (use the minimum/maximum among the children). This provides theoretical guidance on when Full Bellman backup is appropriate: while Full Bellman backup is a method for efficiently estimating $U(u, l)$ of each node from its subtree, GUCT* uses Full Bellman Backup with an MAB designed for the wrong distribution (a distribution with a known fixed support $[0, c]$), and therefore does not perform well. To address this shortcoming, we propose a new MAB for $U(u, l)$:

Theorem 6 (Main results)**.** *In each trial t, assuming the reward* r_i *of arm i follows a Uniform distribution with an unknown support* $U(l_i, u_i)$*, the* U/LCB1-Uniform *policy respectively pulls the arm i that maximizes/minimizes*

$$\text{U/LCB1-Uniform}_i \quad = \tfrac{\hat{u}_i + \hat{l}_i}{2} \pm (\hat{u}_i - \hat{l}_i)\sqrt{6 t_i \log T}$$

where $\hat{l}_i = \min_j r_{ij}, \hat{u}_i = \max_j r_{ij}$ *are the MLEs of* l_i, u_i*. Let* $\alpha \in [0, 1]$ *and* $C \in \mathbb{R}^+$ *be unknown problem-dependent*

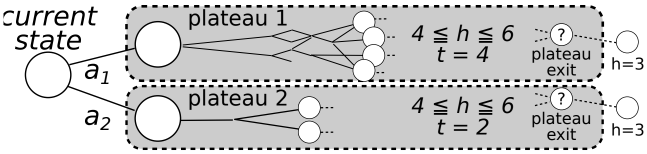

Figure 3: Given equally informative plateaus, UCB1-Uniform focuses on one plateau to find an exit quickly.

constants. When $t_i \geq 2$, U/LCB1-Uniform has a worst-case polynomial, best-case constant cumulative regret upper bound per arm, which converges to $1 + 2C$ when $\alpha \to 1$:

$$\frac{24(u_i-l_i)^2(1-\alpha)^2\log T}{\Delta_i^2}+1+2C+\frac{(1-\alpha)T(T+1)(2T+1)}{3}.$$

Proof. (Sketch of proof in appendix.) We apply *bounded difference inequality* (Boucheron, Lugosi, and Massart 2013) to derive a confidence bound of $\frac{\hat{u}_i+\hat{l}_i}{2}$. It contains an unknown value $u_i - l_i$, which is an issue. Therefore, we use lemmas about the critical value $\alpha = P(r_i < X)$ of $r_i = \frac{\hat{u}_i-\hat{l}_i}{u_i-l_i}$, its CDF, and union-bound to derive a looser upper-bound. The coefficient 6 and $t_i \geq 2$ are derived from the condition that makes C finite. □

Just like UCB1-Normal2, UCB1-Uniform is spread-aware. The second term is scaled by the support range $\hat{u}_i - \hat{l}_i$, similar to the empirical variance $\hat{\sigma}_i$ in UCB1-Normal2. A larger spread indicates more chance that the next pull results in a wildly different, smaller h, while a smaller spread indicates a plateau, a region of flat h landscape (Coles and Smith 2007) that hinders the search progress, particularly when $\hat{u}_i - \hat{l}_i = 0$. Penalizing a small spread gives UCB1-Uniform/Normal2 an ability to avoid plateaus.

However, UCB1-Uniform can not only avoid, but also escape plateaus quickly. For example, in Fig. 3, the two plateaus are equally informed ($u_1 = u_2 = 6, l_1 = l_2 = 4$) and $t_1 > t_2$, thus it keeps searching plateau 1 in a depth-first manner, rather than distributing the effort and failing to explore either one sufficiently.

6 Experimental Evaluation

We first evaluated the proposed algorithm implemented on Pyperplan (Alkhazraji et al. 2020) by counting the number of instances solved under 10,000 node evaluations over a subset of the International Planning Competition benchmark domains, selected for compatibility with the set of PDDL extensions supported by Pyperplan (772 problem instances across 24 domains in total). We focus on node evaluations to improve the reproducibility by removing the effect of low-level implementation detail. See the appendix for the results controlled by expansions and the runtime.

We evaluated various algorithms with h^{FF}, h^{add}, $h^{\max}$, and h^{GC} (goal count) heuristics (Fikes, Hart, and Nilsson 1972), and our analysis focuses on h^{FF}. We included h^{GC} despite its uninformativeness because it can be used in environments without domain descriptions, e.g., in the planning-based approach (Lipovetzky, Ramírez, and Geffner 2015) to the Atari environment (Bellemare et al. 2015). We ran each configuration with 5 random seeds and report the average number of problem instances solved. We do not evaluate A*, UCT, and UCT* as we focus on agile search settings. We included the evaluations with Deferred Evaluation (DE) and Preferred Operators (PO) (see appendix), following Schulte and Keller (2014).

We then evaluated some of the algorithms reimplemented in Fast Downward (Helmert 2006) on IPC2018 instances with h^{FF} under the agile IPC setting (5 minute, 8GB memory), using Intel Xeon 6258R CPU @ 2.70GHz.

Queue-based We first evaluated state-of-the-art queue-based search algorithms and compared them against our proposed GUCT-Uniform. In Table 1, GBFS (Pyperplan/FD) shows the results of GBFS implemented in Pyperplan and FastDownward, respectively. We evaluated them both to confirm the effect of implementation difference. We next evaluated Softmin-Type(h) (Kuroiwa and Beck 2022), a recent state-of-the-art diversified search algorithm for classical planning, from the original C++ implementation available online. GUCT-Uniform outperformed GBFS and Softmin-Type(h) by 67.8 and 33.2 instances, respectively.

$*$-Variants We next compared various bandit algorithms with Monte Carlo backup and Full Bellman backup to analyze the effect of backup differences. In Table 1, GUCT is a GUCT that uses the original UCB1 bandit for action selection. Note that this does not have the "normalization" feature (Schulte and Keller 2014) that turned out to be harmful (Wissow and Asai 2024). GUCT-Normal uses UCB1-Normal (Auer, Cesa-Bianchi, and Fischer 2002), and GUCT-Normal2 uses UCB1-Normal2 (Wissow and Asai 2024). The $*$-variants (GUCT*-Normal, etc.) use Full Bellman backup instead of Monte Carlo backup.

While $*$-variants tend to improve the performance over the base Monte Carlo variants, it happens only when the base algorithm is non-performant. GUCT*-Normal2 performs significantly worse than GUCT-Normal2, and GUCT/* and GUCT/*-Normal are vastly inferior to GUCT/*-Normal2. Our proposed GUCT-Uniform outperformed both $*$- and base variants: GUCT*, *-Normal, *-Normal2, GUCT, -Normal, and -Normal2 by +194.4, +323, +23.4, +147, +287.6, and +39.2 respectively.

These results demonstrate the benefit of selecting a backup method that is theoretically consistent with the given bandit. The Full Bellman Backup estimates Uniform distributions with unknown support, and negatively affects the performance of GUCT*-Normal2's Gaussian bandit with its conflicting assumptions. UCB1-Uniform, which does not have such theoretical dissonance, managed to extract the best performance while using Full Bellman backup.

CHK-Uniform We added CHK-Uniform (Cowan and Katehakis 2015), an asymptotically optimal bandit for Uniform distributions. To our knowledge, CHK-Uniform is the only bandit that works on uniform distributions with unknown supports and is asymptotically optimal, providing a baseline for UCB1-Uniform. Our UCB1-Uniform significantly outperformed CHK-Uniform. This is another interesting case of a non-asymptotically-optimal bandit outperform-

$h =$	h^{FF}	h^{add}	$h^{\max}$	h^{GC}	h^{FF}+PO	h^{FF}+DE+PO
GBFS (Pyperplan/FD)	538/539	518/517	224/226	354/349	†/539	†/‡
Softmin-Type(h)	576	542.6	297.2	357.6	575.8	‡
GUCT	412	397.8	228.4	285.2	454.2	440.4
GUCT*	459.4	480.8	242.2	312.2	496.2	471.8
GUCT-Normal	283.4	265	212	233.4	372.4	381.6
GUCT*-Normal	318.8	300	215.2	246.2	378.05	386.9
GUCT-Normal2	582.95	538	316.6	380.6	623.2	581.8
GUCT*-Normal2	567.2	533.8	263	341.2	619.8	570.6
GUCT-Uniform (ours)	**606.4**	**563.4**	**455.6**	**492.2**	**635.6**	**600.8**
CHK-Uniform	375.4	338.8	224.8	296.6	454.8	458.2
GUCT+-Normal2	578	550.4	442.4	490.6	630.6	582.2
MaxSearch	253.75	243.4	260	255.2	368.6	355.6
RobustUCT	267.8	270.8	234	231.8	403	435.2
ThresholdAscent	162.4	163.8	170.4	164.4	165.8	172.2

	domain	GBFS	SM	N2	Uni
Instances solved	agricola	9.0	10.2	9.4	**11.6**
	caldera	4.0	**7.0**	6.4	5.8
	data-net	4.0	**8.4**	8.2	7.0
	flashfill	**9.0**	8.8	7.2	6.8
	nurikabe	7.0	6.2	**8.4**	7.6
	org-syn	9.0	8.8	**9.2**	6.0
	settlers		**5.4**	2.4	2.6
	snake	5.0	5.0	15.4	**19.0**
	spider	8.0	8.2	**9.2**	8.6
	termes	**12.0**	11.6	5.8	5.0
	total	67.0	79.6	**81.6**	80.0
IPC score	agricola	1.9	3.4	2.0	**6.1**
	caldera	2.9	5.0	5.1	**5.3**
	data-net	3.5	4.5	**5.6**	4.9
	flashfill	6.2	**6.8**	5.4	4.8
	nurikabe	6.4	5.4	**6.9**	6.5
	org-syn	**7.2**	6.8	6.7	5.0
	settlers		**2.6**	1.6	2.3
	snake	2.9	3.3	9.1	**12.5**
	spider	2.2	3.1	**3.4**	**3.4**
	termes	**6.4**	**6.2**	2.6	2.5
	total	39.5	47.0	48.6	**53.2**

Table 1: Best algorithms in **bold**. **(left)** The number of problem instances solved with less than 10,000 node evaluations; each number represents an average over 5 seeds. PO/DE stand for Preferred Operators/Deferred Evaluation. Pyperplan supports PO only for h^{FF}. †: Data missing due to the lack of support of PO for GBFS in Pyperplan. ‡: Data missing because DE in Fast Downward measures node evaluations differently. **(right)** Number of instances solved and IPC scores on IPC 2018 instances, using h^{FF} under 5 min time limit and 8GB memory limit, averaged over 3 seeds. For caldera and organic-synthesis, we used their action-splitting variants (Areces et al. 2014) provided by the organizers. 'SM' stands for Softmin-Type(h), 'N2' stands for GUCT-Normal2, 'Uni' stands for GUCT-Uniform.

ing an asymptotically optimal one, such as UCB1-Uniform vs. CHK-Uniform and UCB1-Normal2 vs. UCB1-Normal. A deeper theoretical investigation into this phenomena is an important avenue of future work.

+-Variants GUCT^{+}-Normal2 uses the backup method explained in Sec. 3 that estimates the maximum of Gaussian RVs, modified for minimization. While this backup sometimes improved the results from GUCT/*-Normal2, the improvement depends on the heuristics and they were overall outperformed by GUCT-Uniform. The likely explanation is that the Maximum-of-Gaussians method is not accurate for combining the estimates for more than 2 arms.

Max-k Bandits Our work can be confused with the *Max k-Armed Bandit* framework (Cicirello and Smith 2004, 2005; Streeter and Smith 2006b,a; Carpentier and Valko 2014; Achab et al. 2017) that optimizes *extreme regret* $\max_i \mathbb{E}[\max_{t=1}^{T} \mathrm{r}_{it}] - \mathbb{E}[\max_{t=1}^{T} \mathrm{r}_{I_t t}]$, where I_t is the arm pulled at t. While both approaches use EVTs, UCB1-Uniform is not a Max-k bandit algorithm and has many theoretical/practical/conceptual differences. Existing Max-k bandits target long-tail distributions while we target short-tail distributions, they primarily use block maxima EVTs, and they fail to align conceptually with classical planning (due to space, this discussion continues in the appendix). We focus here on the experimental results.

We evaluated GUCT variants that use three Max-k bandits for action selection: *Threshold Ascent* (Streeter and Smith 2006a), *RobustUCB* (Bubeck, Cesa-Bianchi, and Lugosi 2013), *MaxSearch* (Kikkawa and Ohno 2022). All hyperparameters are based on the values suggested by their authors. Table 1 shows that these algorithms significantly underperformed.

Agile Experiments with C++ Table 1 (right) shows the results comparing C++ implementations of GUCT-Uniform and other algorithms on Fast Downward. UCB1-Uniform was on par with Softmin-Type(h) and GUCT-Normal2 in terms of the number of solved instances, and outperform them on the IPC score $\sum_i \min(1, 1 - \frac{\log t_i}{\log 300})$, where t_i is the runtime of each algorithm solving an instance i.

7 Conclusion

Previously, statistical estimates for guiding MCTS in classical planning did not respect the natural properties of heuristic functions, i.e. that they have unknown, half-bounded support, leading to overspecification (UCB1: known finite support) or underspecification (Gaussian bandit: entire $\mathbb{R}$). Also, why Monte Carlo backup (averaging) was used for minimization/maximization tasks was unclear. In searching for a theoretically justified backup for agile planning, we modeled the rewards with Peaks-Over-Threshold Extreme Value Theory (POT EVT), which captures the finer details of heuristic search. This led to our new bandit, UCB1-Uniform, which uses the MLE of the Uniform distribution to guide action selection. The resulting algorithm outperformed GBFS, GUCT-Normal2, asymptotically optimal uniform bandit CHK-Uniform, a state-of-the-art diversified search algorithm Softmin-Type(h), and Max-k bandits.

Acknowledgments

This work was supported through DTIC contract FA8075-18-D-0008, Task Order FA807520F0060, Task 4 - Autonomous Defensive Cyber Operations (DCO) Research & Development (R&D).

References

Achab, M.; Clémençon, S.; Garivier, A.; Sabourin, A.; and Vernade, C. 2017. Max K-Armed Bandit: On the Extreme-Hunter Algorithm and Beyond. In *Proc. of ECMLKDD*, 389–404. Springer.

Alkhazraji, Y.; Frorath, M.; Grützner, M.; Helmert, M.; Liebetraut, T.; Mattmüller, R.; Ortlieb, M.; Seipp, J.; Springenberg, T.; Stahl, P.; and Wülfing, J. 2020. Pyperplan.

Areces, C.; Bustos, F.; Dominguez, M. A.; and Hoffmann, J. 2014. Optimizing Planning Domains by Automatic Action Schema Splitting. In *Proc. of ICAPS*, volume 24.

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

Balkema, A. A.; and De Haan, L. 1974. Residual Life Time at Great Age. *Annals of Probability*, 2(5): 792–804.

Beirlant, J.; Goegebeur, Y.; Segers, J.; and Teugels, J. L. 2004. *Statistics of Extremes: Theory and Applications*, volume 558. John Wiley & Sons.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2015. The Arcade Learning Environment: An Evaluation Platform for General Agents (Extended Abstract). In Yang, Q.; and Wooldridge, M. J., eds., *Proc. of IJCAI*, 4148–4152. AAAI Press.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1): 5–33.

Boucheron, S.; Lugosi, G.; and Massart, P. 2013. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press.

Bubeck, S.; Cesa-Bianchi, N.; and Lugosi, G. 2013. Bandits with heavy tail. *IEEE Transactions on Information Theory*, 59(11): 7711–7717.

Bush, R. R.; and Mosteller, F. 1953. A Stochastic Model with Applications to Learning. *The Annals of Mathematical Statistics*, 559–585.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69(1): 165–204.

Carpentier, A.; and Valko, M. 2014. Extreme Bandits. *Advances in Neural Information Processing Systems*, 27.

Cicirello, V. A.; and Smith, S. F. 2004. Heuristic Selection for Stochastic Search Optimization: Modeling Solution Quality by Extreme Value Theory. In *Proc. of CP*, 197–211. Springer.

Cicirello, V. A.; and Smith, S. F. 2005. The MAX K-Armed Bandit: A New Model of Exploration Applied to Search Heuristic Selection. In *Proc. of AAAI*, volume 3, 1355–1361.

Clark, C. E. 1961. The Greatest of a Finite Set of Random Variables. *Operations Research*, 9(2): 145–162.

Coles, A.; and Smith, A. 2007. Marvin: A Heuristic Search Planner with Online Macro-Action Learning. *J. Artif. Intell. Res.(JAIR)*, 28: 119–156.

Cowan, W.; and Katehakis, M. N. 2015. An Asymptotically Optimal Policy for Uniform Bandits of Unknown Support. *arXiv preprint arXiv:1505.01918*.

De Haan, L.; and Ferreira, A. 2006. *Extreme Value Theory: An Introduction*. Springer Science & Business Media.

Diebolt, J.; El-Aroui, M.-A.; Garrido, M.; and Girard, S. 2005. Quasi-Conjugate Bayes Estimates for GPD Parameters and Application to Heavy Tails Modelling. *Extremes*, 8(1): 57–78.

Dijkstra, E. W. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 1(1): 269–271.

Doran, J. E.; and Michie, D. 1966. Experiments with the Graph Traverser Program. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 294(1437): 235–259.

Fikes, R. E.; Hart, P. E.; and Nilsson, N. J. 1972. Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3(1-3): 251–288.

Fisher, R. A.; and Tippett, L. H. C. 1928. Limiting Forms of the Frequency Distribution of the Largest or Smallest Member of a Sample. *Mathematical Proceedings of the Cambridge Philosophical Society*, 24(2): 180–190.

Gelly, S.; and Silver, D. 2011. Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go. *Artificial Intelligence*, 175(11): 1856–1875.

Gnedenko, B. 1943. Sur La Distribution Limite Du Terme Maximum D'Une Serie Aleatoire. *Annals of Mathematics*, 44(3): 423–453.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2): 100–107.

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.(JAIR)*, 26: 191–246.

Hill, B. M. 1975. A Simple General Approach to Inference about the Tail of a Distribution. *Annals of Statistics*, 1163–1174.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation through Heuristic Search. *J. Artif. Intell. Res.(JAIR)*, 14: 253–302.

Hosking, J. R.; and Wallis, J. R. 1987. Parameter and Quantile Estimation for the Generalized Pareto Distribution. *Technometrics*, 29(3): 339–349.

Keller, T.; and Helmert, M. 2013. Trial-Based Heuristic Tree Search for Finite Horizon MDPs. In *Proc. of ICAPS*.

Kikkawa, N.; and Ohno, H. 2022. Materials Discovery using Max K-Armed Bandit. *arXiv preprint arXiv:2212.08225*.

Kishimoto, A.; Bouneffouf, D.; Marinescu, R.; Ram, P.; Rawat, A.; Wistuba, M.; Palmes, P.; and Botea, A. 2022. Bandit Limited Discrepancy Search and Application to Machine Learning Pipeline Optimization. In *Proc. of AAAI*, volume 36, 10228–10237.

Kocsis, L.; and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In *Proc. of ECML*, 282–293. Springer.

Kuroiwa, R.; and Beck, J. C. 2022. Biased Exploration for Satisficing Heuristic Search. In *Proc. of ICAPS*.

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

Laplace, P.-S. 1812. *Théorie analytique des probabilités*.

Lipovetzky, N.; Ramírez, M.; and Geffner, H. 2015. Classical Planning with Simulators: Results on the Atari Video Games. In *Proc. of IJCAI*.

Núñez-molina, C.; Asai, M.; Mesejo, P.; and Fernandez-olivares, J. 2024. On Using Admissible Bounds for Learning Forward Search Heuristics. In *Proc. of IJCAI*.

Pickands III, J. 1975. Statistical Inference using Extreme Order Statistics. *Annals of Statistics*, 119–131.

Resnick, S.; and Stărică, C. 1997. Smoothing the Hill estimator. *Advances in Applied Probability*, 29(1): 271–293.

Richter, S.; Thayer, J. T.; and Ruml, W. 2010. The Joy of Forgetting: Faster Anytime Search via Restarting. In *Proc. of ICAPS*.

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

Schulte, T.; and Keller, T. 2014. Balancing Exploration and Exploitation in Classical Planning. In *Proc. of SOCS*.

Sharpe, J.; and Juárez, M. A. 2021. Estimation of the Pareto and Related Distributions–A Reference-Intrinsic Approach. *Communications in Statistics-Theory and Methods*, 1–23.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587): 484–489.

Smith, R. L. 1987. Estimating Tails of Probability Distributions. *Annals of Statistics*, 1174–1207.

Streeter, M. J.; and Smith, S. F. 2006a. A Simple Distribution-Free Approach to the Max K-Armed Bandit Problem. In *Proc. of CP*, 560–574. Springer.

Streeter, M. J.; and Smith, S. F. 2006b. An Asymptotically Optimal Algorithm for the Max K-Armed Bandit Problem. In *Proc. of AAAI*, 135–142.

Tesauro, G.; Rajan, V.; and Segal, R. 2010. Bayesian Inference in Monte-Carlo Tree Search. In *Proc. of UAI*, 580–588.

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.

Wissow, S.; and Asai, M. 2024. Scale-Adaptive Balancing of Exploration and Exploitation in Classical Planning. In *Proc. of ECAI*.

Extreme Value Monte Carlo Tree Search for Classical Planning

Masataro Asai[*1], **Stephen Wissow**[*2]
[1]MIT-IBM Watson AI Lab
[2]University of New Hampshire
masataro.asai@ibm.com, sjw@cs.unh.edu

The python code for MCTS is attached at the end of this supplement.

A1 Domain-Independent Heuristics in Classical Planning

A domain-independent heuristic function h in classical planning is a function of a state s and the problem $\langle P, A, I, G\rangle$, but the notation $h(s)$ usually omits the latter. In addition to what we discussed in the main article, this section also uses a notation $h(s, G)$. It returns an estimate of the cumulative cost from s to one of the goal states (states that satisfy G), typically through a symbolic, non-statistical means including problem relaxation and abstraction. Notable state-of-the-art functions that appear in this paper includes $h^{\text{FF}}, h^{\max}, h^{\text{add}}, h^{\text{GC}}$ (Hoffmann and Nebel 2001; Bonet and Geffner 2001; Fikes, Hart, and Nilsson 1972).

A significant class of heuristics is called delete relaxation heuristics, which solve a relaxed problem which does not contain delete effects, and then returns the cost of the solution of the relaxed problem as an output. The cost of the optimal solution of a delete relaxed planning problem from a state s is denoted by $h^+(s)$, but this is too expensive to compute in practice (NP-complete) (Bylander 1996). Therefore, practical heuristics typically try to obtain its further relaxations that can be computed in polynomial time.

One such admissible heuristic based on delete-relaxation is called $h^{\max}$ (Bonet and Geffner 2001) that is recursively defined as follows:

$$h^{\max}(s,G) = \max_{p\in G} \begin{cases} 0 \text{ if } p \in s. \text{ Otherwise,} \\ \min_{\{a\in A \mid p\in \text{ADD}(a)\}} \\ \quad \left[\text{COST}(a) + h^{\text{add}}(s, \text{PRE}(a))\right]. \end{cases} \tag{1}$$

Its inadmissible variant is called additive heuristics h^{add} (Bonet and Geffner 2001) that is recursively defined as follows:

$$h^{\text{add}}(s,G) = \sum_{p\in G} \begin{cases} 0 \text{ if } p \in s. \text{ Otherwise,} \\ \min_{\{a\in A \mid p\in \text{ADD}(a)\}} \\ \quad \left[\text{COST}(a) + h^{\text{add}}(s, \text{PRE}(a))\right]. \end{cases} \tag{2}$$

Another inadmissible delete-relaxation heuristics called h^{FF} (Hoffmann and Nebel 2001) is defined based on another heuristics h, such as $h = h^{\text{add}}$, as a subprocedure. For each unachieved subgoal $p \in G \setminus s$, the action a that adds p with the minimal $[\text{COST}(a) + h(s, \text{PRE}(a))]$ is conceptually "the cheapest action that achieves a subgoal p for the first time under delete relaxation", called the *cheapest achiever / best supporter* $\text{bs}(p, s, h)$ of p. h^{FF} is defined as the sum of actions in a relaxed plan Π^+ constructed as follows:

$$h^{\text{FF}}(s,G,h) = \sum_{a\in \Pi^+(s,G,h)} \text{COST}(a) \tag{3}$$

$$\Pi^+(s,G,h) = \bigcup_{p\in G} \begin{cases} \emptyset \text{ if } p \in s. \text{ Otherwise,} \\ \{a\} \cup \Pi^+(s, \text{PRE}(a)) \\ \qquad \text{where } a = \text{bs}(p,s,h). \end{cases} \tag{4}$$

$$\text{bs}(p,s,h) = \operatorname*{arg\,min}_{\{a\in A \mid p\in \text{ADD}(a)\}} \left[\text{COST}(a) + h(s, \text{PRE}(a))\right]. \tag{5}$$

Goal Count heuristics h^{GC} is a simple heuristic proposed in (Fikes, Hart, and Nilsson 1972) that counts the number of propositions that are not satisfied yet. $\langle\text{condition}\rangle$ is a cronecker's delta / indicator function that returns 1 when the condition is satisfied.

$$h^{\text{GC}}(s,G) = \sum_{p\in G} [\![p \notin s]\!]. \tag{6}$$

A2 Deferred Evaluation (DE)

DE is a technique that reduces the runtime of search algorithm while sacrificing the memory efficiency. Standard forward-search algorithms evaluate the value of a heuristic function $h(n)$ of each node n, which is called *Eager Evaluation* in comparison to DE. In DE (also sometimes called *Lazy Evaluation*), the search algorithm instead computes and caches the heuristic value of a node's parent p and uses it in order to sort the nodes in the OPEN list. Since the computation

[*]These authors contributed equally.

of heuristic values is the largest bottleneck in the entire search algorithm compared to other operations (e.g., inserting a node into and retrieving the current best node from the OPEN list / priority queue), this saves a large amount of computation. However, this operation also makes the heuristic function less informative because the cost-to-go estimate by the heuristic is not calculated based on the nodes' own state information. Therefore, the search algorithm tends to visit more irrelevant nodes, which results in an increased number of nodes inserted into the OPEN list, thus more memory usage. DE affects the optimality guarantee of the solution, therefore can only be used in a satisficing/agile search scenario.

In the Pyperplan-based experiment performed in this paper, DE should perform worse than eager evaluations because DE trades the number of calls to heuristics with the number of nodes inserted to the tree, which is limited to 10,000. When CPU time is the limiting resource, DE can sometimes solve more instances (Richter and Helmert 2009), assuming the implementation is optimized for speed (e.g., using C++). However, our Python implementation (typically 100–1,000 times slower than C++) is not able to measure this effect because this low-level bottleneck could hide the effect of speed improvements.

A3 Preferred Operators (PO)

In addition to the heuristic value of a state, some heuristic functions are able to return a list of actions called "helpful actions" (Hoffmann and Nebel 2001) or "preferred operators" (Richter and Helmert 2009) and are used by a planner in a variety of ways (e.g., initial incomplete search of FF planner (Hoffmann and Nebel 2001) and alternating open list in LAMA planner (Richter, Westphal, and Helmert 2011)). We reimplemented Schulte and Keller (2014)'s strategy which limits the action selection to the preferred operators and falling back to the normal behavior if there are none. In MCTS terminology (Sec. 2.2), this is a way to modify the tree policy by re-weighting it with a mask.

A4 Detailed Explanation for the Base MCTS for Graph Search

Alg. 1 shows the pseudocode of MCTS adjusted for graph search (Schulte and Keller 2014). Aside from what was described from the main section, it has a node-locking mechanism that avoids duplicate search effort.

Following THTS, our MCTS has a hash table that implements a *CLOSE* list and a *Transposition Table* (TT). A CLOSE list stores the generated states and avoids instantiating nodes with duplicate states. A TT stores various information about the states such as the parent information and the action used at the parent. The close list is implemented by a lock mechanism.

Since an efficient graph search algorithm must avoid visiting the same state multiple times, MCTS for graph search marks certain nodes as *locked*, and excludes them from the selection candidates. A node is locked either (1) when a node is a dead-end that will never reach a goal (detected by having no applicable actions, by a heuristic function, or other facilities), (2) when there is a node with the same state in the search tree

Algorithm 1 High-level general MCTS. **Input**: Root node r, successor function S, NEC f, heuristic function h, priority queue Q sorted by g. Initialize $\forall n; g(n) \leftarrow \infty$.

```
while True do
  Parent p ← r
  while not leaf p do                                  # Selection
    p ← arg min_{n∈S(p)} f(n)
  Q ← {p}
  for n ∈ S(p) do                                      # Expansion
    return n if n is goal.                  # Early goal detection
    if ∃n' already in tree with same state s_{n'} = s_n then
      if g(n) > g(n') then
        continue
      Lock n', S(n) ← S(n'), Q ← Q ∪ {n, n'}
    else
      Compute h(s_n)                                   # Evaluation
      Q ← Q ∪ {n}
  while n ← Q.POPMAX() do                        # Backpropagation
    Update n's statistics and lock status
    Q ← Q ∪ {n's parent}
```

with a smaller g-value, (3) when all of its children are locked, or (4) when a node is a goal (relevant in an anytime iterated search setting (Richter, Thayer, and Ruml 2010; Richter, Westphal, and Helmert 2011), but not in this paper). Thus, in the expansion step, when a generated node n has the same state as a node n' already in the search tree, MCTS discards n if $g(n) > g(n')$, else moves the subtree of n' to n and marks n' as locked. It also implicitly detects a cycle, as this is identical to the duplicate detection in Dijkstra/A*/GBFS.

The queueing step backpropagates necessary information from the leaf to the root. Efficient backpropagation uses a priority queue ordered by descending g-value. The queue is initialized with the expanded node p; each newly generated node n that is not discarded is inserted into the queue, and if a node n' for the same state was already present in the tree it is also inserted into the queue. In each backpropagation iteration, (1) the enqueued node with the highest g-value is popped, (2) its information is updated by aggregating its children's information (including the lock status), (3) and its parent is queued.

A5 Proof of MLE

We show the maximum likelihood estimators of Gaussian and Uniform distribution (Thm. 1-5). First, let us restate the definition of MLE:

Definition 1. *Given i.i.d. samples $x_1, \ldots, x_N \sim p(\mathrm{x}|\theta)$, the Maximum Likelihood Estimate (MLE) of the parameter θ is:*

$$
\begin{aligned}
\hat{\theta} &= \arg\max_\theta p(x_1, \ldots, x_N|\theta) \\
&= \arg\max_\theta \log p(x_1, \ldots, x_N|\theta) \quad (\log \textit{ is monotinic.}) \\
&= \arg\max_\theta \sum_i \log p(x_i|\theta). \quad \because i.i.d.
\end{aligned}
$$

Sometimes the MLE is solved by the following equation for the target parameter θ.

$$
\begin{aligned}
0 &= \frac{\partial}{\partial\theta} p(x_1, \ldots, x_N|\theta) \\
&= \frac{\partial}{\partial\theta} \log p(x_1, \ldots, x_N|\theta) \\
&= \frac{\partial}{\partial\theta} \log \prod_i p(x_i|\theta) = \frac{\partial}{\partial\theta} \sum_i \log p(x_i|\theta).
\end{aligned}
$$

An estimator that is obtained by maximizing some quantity is called an *M-estimator*. MLE is an M-estimator. Some M-estimators can be solved by setting the derivatives to zero, in which case they are called *Z-estimator*. Not all M-estimators are solved in this way, as seen below (l and u).

Theorem 1 (MLE of Gaussian)**.** *Given i.i.d. $x_1, \ldots, x_N \sim \mathcal{N}(\mathrm{x}|\mu, \sigma)$, the Maximum Likelihood Estimate of μ is the sample average $\hat{\mu} = \frac{1}{N}\sum_i x_i$. Similarly, the Maximum Likelihood Estimate of σ^2 is the sample variance $\frac{1}{N-1}\sum_i (x_i - \hat{\mu})$.*

Proof. For μ,

$$
\begin{aligned}
0 &= \frac{\partial}{\partial\mu} \sum_i \log \mathcal{N}(x_i|\mu, \sigma) \\
&= \frac{\partial}{\partial\mu} \sum_i \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{(x_i-\mu)^2}{2\sigma^2} \right) \\
&= \frac{\partial}{\partial\mu} \sum_i \left(-\frac{(x_i-\mu)^2}{2\sigma^2} - \log\sqrt{2\pi\sigma^2} \right) \\
&= \frac{\partial}{\partial\mu} \sum_i \left(\frac{-x_i^2 + 2\mu x_i - \mu^2}{2\sigma^2} - \log\sqrt{2\pi\sigma^2} \right) \\
&= \sum_i \frac{2x_i - 2\mu}{2\sigma^2}. \quad \therefore \hat{\mu} = \frac{\sum_i x_i}{N}.
\end{aligned}
$$

For σ^2,

$$
\begin{aligned}
0 &= \frac{\partial}{\partial\sigma^2} \sum_i \log \mathcal{N}(x_i|\mu, \sigma) \\
&= \frac{\partial}{\partial\sigma^2} \sum_i \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{(x_i-\mu)^2}{2\sigma^2} \right) \\
&= \frac{\partial}{\partial\sigma^2} \sum_i \left(-\frac{(x_i-\mu)^2}{2\sigma^2} - \frac{\log 2\pi + \log\sigma^2}{2} \right) \\
&= \sum_i \left(\frac{(x_i-\mu)^2}{2(\sigma^2)^2} - \frac{1}{2\sigma^2} \right). \\
\therefore \hat{\sigma}^2 &= \frac{\sum_i (x_i-\mu)^2}{N} = \frac{\sum_i (x_i-\hat{\mu})^2}{N-1}.
\end{aligned}
$$

□

Theorem 2 (MLE of Uniform)**.** *Given i.i.d. $x_1, \ldots, x_N \sim U(\mathrm{x}|l, u)$, $\hat{u} = \max_i x_i$ and $\hat{l} = \min_i x_i$.*

Proof. For u, we cannot use the differentiation-based proof. Instead, we maximize the log likelihood directly.

$$
\begin{aligned}
&\sum_i \log U(x_i|l, u) \\
&= \sum_i \log \frac{1}{u-l} \\
&= \sum_i -\log(u-l).
\end{aligned}
$$

The value of u that maximizes this formula is the infimum of possible values u can take. Since $\forall i; x_i \leq u$, $u = \max_i x_i$.

Similarly, the value of l that maximizes this formula is the supremum of possible values l can take. Since $\forall i; l \leq x_i$, $l = \min_i x_i$. □

A6 Proof of Bandit Algorithms (Overview)

This section serves as a tutorial for understanding our main proof in the later sections. To help understand the proof of various confidence bounds, we first describe the general procedure for proving the regret of bandit algorithms, demonstrate the proof of UCB1 using this scheme, then finally show the proof of other bandits.

The ingredients for proving an upper/lower confidence bound are as follows:

- **Ingredient 1: The main term and the exploration term.** For example, in the standard UCB1 (Auer, Cesa-Bianchi, and Fischer 2002), the *main term* is the empirical mean $\hat{\mu}$ while the *exploration term* is $c\sqrt{\frac{2\log T}{n_i}}$. Their forms are heavily affected by the proof of the upper bound on the regret, and an arbitrary exploration term is not guaranteed to have a provable bound.
- **Ingredient 2: A specification of reward distributions.** For example, in the standard UCB1 (Auer, Cesa-Bianchi, and Fischer 2002), one assumes a reward distribution bounded in $[0, c]$. Different algorithms assume different reward distributions, and in general, more information about the distribution gives a tighter bound (and faster convergence). For example, one can assume an unbounded distribution with known variance, etc.
- **Ingredient 3: A concentration inequality.** It is also called a tail probability bound. For example, in the standard UCB1, one uses Hoeffding's inequality. Different algorithms use different inequalities to prove the bound, based on what reward distribution it assumes and what main term it uses. Examples include the Chernoff bound, Chebishev's inequality, Bernstein's inequality, Bennett's inequality, etc. Note that the inequality may be two-sided or one-sided.

The general procedure for proving the bound is as follows.

1. Let the main term be a random variable X_n and the exploration term be δ. Then write down the concentration inequality for X_n as follows.
   - $P(|\mathrm{X}_n - \mathbb{E}[\mathrm{X}_n]| \geq \delta) \leq F(\delta)$. (two-sided)

   F is an inequality-specific formula. If necessary, simplify the inequality based on the assumptions made in the reward distribution, e.g., bounds, mean, variance.
2. Expand $|\mathrm{X}_n - \mathbb{E}[\mathrm{X}_n]| \geq \delta$ into $\delta \geq \mathrm{X}_n - \mathbb{E}[\mathrm{X}_n] \geq -\delta$.
3. Change the notations to model the bandit problem because each concentration inequality is a general statement about RVs. Before this step, the notation was:
   - n (number of samples)
   - X_n is a function of i.i.d. random variables $(\mathrm{x}_1, \ldots, \mathrm{x}_n)$
   - $\mathbb{E}[\mathrm{X}_n] = \mathbb{E}[\mathrm{x}_1] = \ldots = \mathbb{E}[\mathrm{x}_n]$ is assumed.

   For example,
   - $\mu_n = \frac{1}{n}\sum_{i=1}^{n} \mathrm{x}_i$
   - $\mathbb{E}[\mu_n] = \mathbb{E}[\mathrm{x}_1] = \ldots = \mathbb{E}[\mathrm{x}_n]$

   After the change, they correspond to:
   - n_i (number of pulls of arm i).
   - $\hat{X}_i$ (empirical value of arm i from n_i pulls),
   - X_i (true value of arm i).

   For example,
   - $\hat{\mu}_i$ (sample mean of arm i from n_i pulls),
   - μ_i (true mean of arm i),
4. Let i be a suboptimal arm, $*$ be an optimal arm, $\mathrm{UCB}_i = \hat{\mu}_i + \delta$, and $\mathrm{LCB}_i = \hat{\mu}_i - \delta$. Derive the relationship between δ and the gap $\Delta_i = \mu_i - \mu_*$ so that the following conditions for the best arm holds:
   - $\mathrm{UCB}_i \leq \mathrm{UCB}_*$ (for maximization)
   - $\mathrm{LCB}_i \geq \mathrm{LCB}_*$ (for minimization)

   This results in $2\delta \leq \Delta_i$.
5. Replace the δ with the exploration term. For example, in UCB1, $\delta = \sqrt{\frac{2\log T}{n_i}}$.
6. Derive the lower bound L for n_i from $2\delta \leq \Delta_i$.
7. Find the upper-bound of the probability of selecting a suboptimal arm i. This is typically done by a union-bound argument.
8. Derive the upper bound of the expected number of pulls $\mathbb{E}[n_i]$ of a suboptimal arm i using a triple loop summation. This is typically the heaviest part that needs mathematical tricks. The tricks do not seem generally transferable between approaches.
9. Finally, derive an upper bound of the regret $TX_* - \sum_{i=1}^{K} X_i\mathbb{E}[n_i]$ by

$$TX_* - \sum_{i=1}^{K} X_i\mathbb{E}[n_i] = \sum_{i=1}^{K}(X_* - X_i)\mathbb{E}[n_i] = \sum_{i=1}^{K} \Delta_i\mathbb{E}[n_i].$$

A6.1 The Proof of UCB1 (Tutorial Example)

We prove the logarithmic upper bound of the cumulative regret of the UCB1 $\hat{\mu}_i - c\sqrt{\frac{2\log T}{n_i}}$ where $\hat{\mu}_i$ is the empirical mean of samples from arm i, n_i is the number of pulls from arm i, and $T = \sum_{i=1}^{K} n_i$ is the total pulls from all K arms.

1. UCB1 assumes a reward distribution with a known bound. For such a distribution, we can use Hoeffding's inequality. Given RVs $\mathrm{x}_1 \dots \mathrm{x}_n$, where $\mathrm{x}_i \in [l_i, u_i]$, and their sum $\mathrm{S}_n = \sum_{i=1}^{n} \mathrm{x}_i$,
$$P(|\mathrm{S}_n - \mathbb{E}[\mathrm{S}_n]| \geq \epsilon) \leq 2\exp - \frac{2\epsilon^2}{\sum_{i=1}^{n}(u_i - l_i)^2}.$$
Using $\delta = \frac{\epsilon}{n}$ and $\mu_n = \frac{\mathrm{S}_n}{n}$,
$$P(|\mu_n - \mathbb{E}[\mu_n] \geq \delta|) \leq 2\exp - \frac{2n^2\delta^2}{\sum_{i=1}^{n}(u_i - l_i)^2}.$$
UCB1 assumes x_i are i.i.d. copies, thus $\forall i; u_i - l_i = c$.
$$P(|\mu_n - \mathbb{E}[\mu_n] \geq \delta|) \leq 2\exp - \frac{2n^2\delta^2}{nc^2} = 2\exp - \frac{2n\delta^2}{c^2}.$$
2. Expanding the two-sided error:
$$\delta \geq \mu_n - \mathbb{E}[\mu_n] \geq -\delta.$$
3. Changing the notation:
$$\delta \geq \hat{\mu}_i - \mu_i \geq -\delta.$$
4. Adding $\mu_i - \delta$ to both sides,
$$\mu_i \geq \hat{\mu}_i - \delta = \mathrm{LCB}_i(T, n_i) \geq \mu_i - 2\delta.$$
Substituting $i = *$ (optimal arm), the first inequality is
$$\mu_* \geq \hat{\mu}_* - \delta = \mathrm{LCB}_*(T, n_*).$$
Assuming $2\delta \leq \Delta_i = \mu_i - \mu_*$, the second inequality is
$$\mathrm{LCB}_i(T, n_i) \geq \mu_i - 2\delta \geq \mu_i - \Delta_i = \mu_*.$$
Therefore
$$\mathrm{LCB}_i(T, n_i) \geq \mu_* \geq \mathrm{LCB}_*(T, n_*).$$
5. Let $\delta = c\sqrt{\frac{2\log T}{n_i}}$. Then
$$P(\mu_{n_i} - \mathbb{E}[\mu_{n_i}] \geq \delta) \leq \exp - \frac{2n_i c^2 \frac{2\log T}{n_i}}{c^2} = T^{-4}.$$
6. From $2\delta \leq \Delta_i$, considering n_i is an integer,
$$2c\sqrt{\frac{2\log T}{n_i}} \leq \Delta_i \Leftrightarrow 4c^2\frac{2\log T}{n_i} \leq \Delta_i^2$$
$$\Leftrightarrow \frac{8c^2\log T}{\Delta_i^2} \leq \left\lceil \frac{8c^2\log T}{\Delta_i^2} \right\rceil = L \leq n_i.$$
7. $\mathrm{LCB}_i(T, n_i) \geq \mu_* \geq \mathrm{LCB}_*(T, n_*)$ does not hold when either inequality does not hold. $\mathrm{LCB}_i(T, n_i) \geq \mu_*$ does not hold with probability less than T^{-4}. $\mu_* \geq \mathrm{LCB}_i(T, n_*)$ does not hold with probability less than T^{-4}. Thus, by union-bound (probability of disjunctions),
$$P(\mathrm{LCB}_i(T, n_i) \leq \mathrm{LCB}_*(T, n_*)) \leq 2T^{-4}.$$
8. Assume we followed the UCB1 strategy, i.e., we pulled the arm that minimizes the LCB. The expected number of pulls $\mathbb{E}[n_i]$ from a suboptimal arm i is as follows. Note that for K arms, every arm is at least pulled once.
$$\begin{aligned}
\mathbb{E}[n_i] &= 1 + \sum_{t=K+1}^{T} P(i \text{ is pulled at time } t) \\
&\leq L + \sum_{t=K+1}^{T} P(i \text{ is pulled at time } t \wedge n_i > L) \\
&= L + \sum_{t=K+1}^{T} P(\forall j; \mathrm{LCB}_j(t, n_j) \geq \mathrm{LCB}_i(t, n_i)) \\
&\leq L + \sum_{t=K+1}^{T} P(\mathrm{LCB}_*(t, n_*) \geq \mathrm{LCB}_i(t, n_i)) \\
&\leq L + \sum_{t=K+1}^{T} P(\exists u, v; \mathrm{LCB}_*(t, u) \geq \mathrm{LCB}_i(t, v)) \\
&\leq L + \sum_{t=K+1}^{T} \sum_{u=1}^{t-1} \sum_{v=L}^{t-1} P(\mathrm{LCB}_*(t, u) \geq \mathrm{LCB}_i(t, v)) \\
&\leq L + \sum_{t=K+1}^{T} \sum_{u=1}^{t-1} \sum_{v=L}^{t-1} 2t^{-4} \\
&\leq L + \sum_{t=1}^{\infty} \sum_{u=1}^{t} \sum_{v=1}^{t} 2t^{-4} = L + \sum_{t=1}^{\infty} t^2 \cdot 2t^{-4} \\
&= L + 2\sum_{t=1}^{\infty} t^{-2} = L + 2 \cdot \frac{\pi}{6} = L + \frac{\pi}{3} \\
&\leq c^2\frac{8\log T}{\Delta_i^2} + 1 + \frac{\pi}{3} \quad \because \lceil x \rceil \leq x + 1
\end{aligned}$$
9. The regret is
$$\begin{aligned}
T\mu_* - \sum_{i=1}^{K} \mu_i \mathbb{E}[n_i] &= \sum_{i=1}^{K} (\mu_* - \mu_i)\mathbb{E}[n_i] = \sum_{i=1}^{K} \Delta_i \mathbb{E}[n_i] \\
&\leq \sum_{i=1}^{K} \Delta_i \left(c^2\frac{8\log T}{\Delta_i^2} + 1 + \frac{\pi}{3} \right) \\
&\leq \sum_{i=1}^{K} \left(c^2\frac{8\log T}{\Delta_i} + \left(1 + \frac{\pi}{3}\right)\Delta_i \right).
\end{aligned}$$

A7 The Proof of UCB1-Uniform's Regret Bound

Recall the definitions of various LCBs.

$$\begin{aligned}
\text{LCB1}_i &= \hat{\mu} - c\sqrt{(2\log T)/n_i}.\\
\text{LCB1-Normal}_i &= \hat{\mu} - \hat{\sigma}\sqrt{(16\log T)/n_i}.\\
\text{LCB1-Normal2}_i &= \hat{\mu} - \hat{\sigma}\sqrt{2\log T}.\\
\text{LCB1-Uniform}_i &= \tfrac{\hat{u}+\hat{l}}{2} - (\hat{u}-\hat{l})\sqrt{6n_i\log T}.
\end{aligned}$$

In the definition of UCB1/-Normal/-Normal2, the main estimate $\hat{\mu}$ is defined by the sum of i.i.d. random variables for individual rewards, which made it possible to use Hoeffding's inequality in its proof for regret bounds. In contrast, the main estimate $\frac{\hat{u}+\hat{l}}{2}$ of UCB1-Uniform is not defined from the sum of data points, i.e., $\hat{l} = \min_i x_i$ uses a minimum, and $\hat{u} = \max_i x_i$ uses a maximum. To handle these cases, we need concentration inequalities for more general functions other than sums. Among many concentration inequalities for general functions (e.g., Efron-Stein inequality, Han's inequality, logarithmic Sobolev inequality), we found that *bounded differences inequality* (Boucheron, Lugosi, and Massart 2013) can be used as the basis of the proof.

A7.1 Preliminary: Bounded Difference Inequality

To use the inequality, we must assume a function with bounded differences, which we define first.

Definition 2 (Functions with Bounded Differences)**.** *A n-ary function $g : X^n \to \mathbb{R}$ for a set X has a* bounded difference *when, for all $1 \le i \le n$, there exists a constant $c_i \in \mathbb{R}$ such that*

$$\max_{y\in X}\left| \begin{array}{l} g(\dots, x_{i-1}, x_i, x_{i+1}, \dots) \\ -g(\dots, x_{i-1}, y, x_{i+1}, \dots) \end{array} \right| \le c_i.$$

Note that an MLE of a parameter obtained from n sample points $(x_1, \dots, x_n)$ are n-ary function. Moreover, quantities defined from the parameters obtained by MLE are also one of them. For example, the MLE is u in $U(l, u)$ is $\hat{u} = \max_i x_i$, which is a function of $(x_1, \dots, x_n)$, and the mean $\frac{\hat{l}+\hat{u}}{2}$ of the fitted distribution $U(\hat{l}, \hat{u})$ is also a function of $(x_1, \dots, x_n)$.

When the distribution function of a random variable has bounded differences, then we can bound its tail probability as follows. This inequality generalizes Hoeffding's inequality by $g(\mathrm{x}_1, \dots, \mathrm{x}_n) = \sum_{i=1}^{n} \mathrm{x}_i$.

Theorem 3 (Bounded Differences Inequality)**.** *When a n-ary function $g : X^n \to \mathbb{R}$ for a set X has a bounded difference $c_1, \dots c_n$ for each argument, let $C = \sum_{i=1}^{n} c_i^2$. Then a random variable $\mathrm{z}_n = g(\mathrm{x}_1, \dots, \mathrm{x}_n)$ satisfies*

$$P(|\mathrm{z}_n - \mathbb{E}[\mathrm{z}_n]| \ge \delta) \le \exp -2\delta^2/C.$$

Theorem 4 (Hoeffding's Inequality)**.** *Given random variables $\mathrm{x}_i \in [a, b]$, let their sum be $\mathrm{S}_n = \sum_{i=1}^{n} \mathrm{x}_i$ and $C = \sum_{i=1}^{n} |b_i - a_i|^2$. Then S_n satisfies*

$$P(|\mathrm{S}_n - \mathbb{E}[\mathrm{S}_n]| \ge \delta) \le \exp -2\delta^2/C.$$

Due to the similarity of the form, once we show that the quantities used in LCB-Uniform have bounded difference, it is straightforward to prove its regret.

A7.2 Preliminary: Order Statistics

The proof of UCB1-Uniform relies on several theorems in order statistics (Hryniv 2017).

Definition 3 (Order Variable)**.** *Assume n random variables $\mathrm{x}_1, \dots \mathrm{x}_n$. Let $\mathrm{x}_{(k)}$ denote a k-th* order variable *defined as*

$$\mathrm{x}_{(k)} = \text{the } k\text{-th largest of } \mathrm{x}_1, \dots \mathrm{x}_n.$$

In other words, $\mathrm{x}_{(1)} \le \mathrm{x}_{(2)} \le \dots \le \mathrm{x}_{(n)}$, where $\mathrm{x}_{(1)}$ is the smallest and $\mathrm{x}_{(k)}$ is the largest.

Definition 4 (Range Variable)**.** *Given order variables $\mathrm{x}_{(1)}, \dots \mathrm{x}_{(n)}$, a* range variable *is $\mathrm{R}_n = \mathrm{x}_{(n)} - \mathrm{x}_{(1)}$.*

Lemma 1 (CDF of a Range Variable)**.** *Assume n i.i.d. random variables $\mathrm{x}_1, \dots \mathrm{x}_n$ with a CDF F and a PDF f. The cumulative distribution function of R_n is given by*

$$P(\mathrm{R}_n < r) = n \int (F(x + r) - F(x))^{n-1} f(x)dx.$$

A7.3 The mean of Uniform has a bounded difference

To prove the regret bound for UCB1-Uniform, we first prove two lemmas to show that the mean $\frac{u+l}{2}$ of a Uniform distribution $U(l, u)$ has a bounded difference.

Lemma 2 (CDF of a Range Variable for $U(0, 1)$)**.** *Assume n i.i.d. random variables $\mathrm{x}_1, \dots \mathrm{x}_n \sim U(0, 1)$. The cumulative distribution function of R_n is $P(\mathrm{R}_n < r) = nr^{n-1}$.*

Proof. $P(\mathrm{R}_n < r)$

$$= n \int_0^1 ((x + r) - x)^{n-1} \cdot 1 \cdot dx$$

$$= n \int_0^1 r^{n-1}dx = n\langle r^{n-1}x\rangle_0^1 = nr^{n-1}(1 - 0) = nr^{n-1}.$$

□

Lemma 3. *Assume i.i.d. RVs $\mathrm{x}_1, \dots \mathrm{x}_n \sim U(l, u)$, $\mathrm{x}_{(1)} = \hat{l}$ and $\mathrm{x}_{(n)} = \hat{u}$. Then $\mathrm{z}_n = g(\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_n) = \frac{\hat{u}+\hat{l}}{2} = \frac{\mathrm{x}_{(n)}+\mathrm{x}_{(1)}}{2}$ has a bounded difference $u - l$.*

Proof. First, $g(\mathrm{x}_1, \dots \mathrm{x}_i, \dots \mathrm{x}_n) \in [l, u]$ because

$$l = \frac{l + l}{2} \le \frac{\mathrm{x}_{(n)} + \mathrm{x}_{(1)}}{2} \le \frac{u + u}{2} = u.$$

The same trivially holds for $g(\mathrm{x}_1, \dots y, \dots \mathrm{x}_n) \in [l, u]$ when $y \in [l, u]$. Their difference is largest when either one is u and another one is l. Thus

$$|\ g(\dots, \mathrm{x}_i, \dots) - g(\dots, y, \dots)\ | \le u - l = c_i.$$

□

This gives us a following concentration inequality based on $C = \sum_i c_i^2 = n(u - l)^2$:

$$P(A : |\mathrm{z}_n - \mathbb{E}\mathrm{z}_n| \ge \delta) \le \exp -\frac{2\delta^2}{n(u - l)^2}.$$

A7.4 The Proof of UCB1-Uniform

Now we prove the upper bound of the cumulative regret of the LCB1-Uniform parameterized by b, LCB1-Uniform$_i(b) = \frac{\hat{u}_i+\hat{l}_i}{2} - (\hat{u}_i - \hat{l}_i)\sqrt{bn_i \log T}$, where $\hat{u}_i, \hat{l}_i$ are the empirical max/min of samples from arm i, n_i is the number of pulls from arm i, and $T = \sum_{i=1}^{K} n_i$ is the total pulls from all K arms. Later, the best b is determined to be $b = 6$.

1. As shown above.
2. Same as UCB1.
3. Same as UCB1.
4. Same as UCB1.
5. Let $\delta = (\hat{u}_i - \hat{l}_i)\sqrt{bn_i \log T}$. and $r_i = \frac{\hat{u}_i-\hat{l}_i}{u_i-l_i} \in [0, 1]$. Then

$$P(A) \leq \exp -\frac{2(\hat{u}_i - \hat{l}_i)^2 \cdot bn_i \log T}{n_i(u_i - l_i)^2}$$
$$= \exp -2br_i^2 \log T = T^{-2br_i^2}.$$

The trick starts here. The value of $r_i = \frac{\hat{u}_i-\hat{l}_i}{u_i-l_i}$ is unavailable because we do not know the true bounds u_i and l_i. However, if event $B : r_i \geq X$ holds for some constant X, we would obtain a more convenient form T^{-2bX^2}:

$$P(A) \leq T^{-2br_i^2} \leq T^{-2bX^2}.$$

Assume $P(B) = \alpha$. Note that $r_i = \frac{\hat{u}_i-\hat{l}_i}{u_i-l_i}$ is a range variable for n_i i.i.d. RVs from $U(0, 1)$. Therefore, from its CDF (Lemma. 2), for some $X \in [0, 1]$,

$$P(r_i < X) = nX^{n-1}.$$

Solving X for $P(r_i < X) = P(\neg B) = 1 - \alpha$,

$$X = X_{n,1-\alpha} = \left(\frac{1-\alpha}{n}\right)^{\frac{1}{n-1}}.$$

A and B could be correlated. Using union-bound

$$P(\neg(A \wedge B)) = P(\neg A \vee \neg B) \leq P(\neg A) + P(\neg B)$$
$$1 - P(A \wedge B) \leq 1 - P(A) + P(\neg B)$$
$$\therefore P(A) \leq P(A \wedge B) + P(\neg B)$$
$$= T^{-2gX^2_{n,1-\alpha}} + 1 - \alpha.$$

6. From $2\delta \leq \Delta_i$ and using the fact that n_i is an integer,

$$\Delta_i \geq 2(\hat{u}_i - \hat{l}_i)\sqrt{bn_i \log T}$$
$$\Leftrightarrow \Delta_i^2 \geq 4b(\hat{u}_i - \hat{l}_i)^2 n_i \log T$$
$$= 4b(u_i - l_i)^2 r_i^2 n_i \log T$$
$$\geq 4b(u_i - l_i)^2 X^2_{n_i,1-\alpha} n_i \log T \ (\because B)$$
$$= 4b(u_i - l_i)^2 \left(\frac{1-\alpha}{n_i}\right)^{\frac{2}{n_i-1}} n_i \log T$$
$$\geq 4b(u_i - l_i)^2 \left(\frac{1-\alpha}{n_i}\right)^{2} n_i \log T$$
$$(\because 0 < \frac{1-\alpha}{n_i} < 1,\ \frac{2}{n_i-1} < 2)$$
$$\Leftrightarrow n_i \geq \frac{4b(u_i - l_i)^2(1-\alpha)^2 \log T}{\Delta_i^2}$$
$$\therefore n_i \geq L = \left\lceil \frac{4b(u_i - l_i)^2(1-\alpha)^2 \log T}{\Delta_i^2} \right\rceil$$

7. Using the same union-bound argument as UCB1, $P(\mathrm{LCB}_i(T, n_i) \leq \mathrm{LCB}_*(T, n_*)) \leq 2T^{-2bX_{n_i,1-\alpha}} + 2 - 2\alpha.$
8. For a reason explained later, we require each arm to be pulled at least M times. For a fixed α, the solution $X_{n_i,1-\alpha}$ of X for $P(r_i < X) = 1 - \alpha$ increases monotonically as n_i increases. Therefore, $X_{n_i,1-\alpha} \geq X_{M,1-\alpha}$ under $n_i \geq M$. Then, using the same argument as UCB1,

$$\mathbb{E}[n_i] \leq L + \sum_{t=1}^{T} t^2(2t^{-2bX_{M,1-\alpha}} + 2(1-\alpha))$$
$$\leq L + 2\sum_{t=1}^{\infty} t^{2-2bX_{M,1-\alpha}} + 2(1-\alpha)\sum_{t=1}^{T} t^2$$
$$\leq L + 2C + \frac{(1-\alpha)T(T+1)(2T+1)}{3}$$
$$\leq \frac{4b(u_i - l_i)^2(1-\alpha)^2 \log T}{\Delta_i^2} + 1$$
$$+ 2C + \frac{(1-\alpha)T(T+1)(2T+1)}{3}$$
$$(\because \lceil x \rceil \leq x + 1)$$

We introduced M because C is convergent only when the exponentiator of t is below -1. We consider the condition in which an integer M exists for a given α. Consider:

$$2 - 2bX_{x,1-\alpha} = 2 - 2b\left(\frac{1-\alpha}{x}\right)^{\frac{2}{x-1}} < -1$$
$$\Leftrightarrow f(x) = x^2\left(\frac{3}{2b}\right)^{x-1} < (1-\alpha)^2$$

Since $1 - \alpha \in [0, 1]$, this is satisfiable only when $0 \leq f(x) \leq 1$. Positivity is trivial. To satisfy $\forall x \geq M; f(x) \leq 1$, it is sufficient if $f(M) \leq 1$ and $\forall x \geq M; f'(x) < 0$. From the derivative below,

$$f'(x) = x\left(\frac{3}{2b}\right)^{x-1}\left(2 + x \log \frac{3}{2b}\right),$$

this condition is achieved when $3/2b < 1$. If $b = 2$, then $f(22) > 1 > f(23)$, thus it requires too many initialization pulls: $M = 23$. For MCTS where we evaluate each node at least once, we wish to achieve $M = 2$. $b = 6$ achieves $f(2) = 1$, thus is the ideal parameter for MCTS.
9. The regret is polynomial.

$$Tz_* - \sum_{i=1}^{K} z_i\mathbb{E}[n_i] = \sum_{i=1}^{K}(z_* - z_i)\mathbb{E}[n_i] = \sum_{i=1}^{K} \Delta_i\mathbb{E}[n_i]$$
$$\leq \sum_{i=1}^{K} \Delta_i \left(\begin{array}{c} \frac{4b(u_i-l_i)^2(1-\alpha)^2 \log T}{\Delta_i^2} + 1 + 2C \\ + \frac{(1-\alpha)T(T+1)(2T+1)}{3} \end{array} \right).$$

A8 Max-k Bandits vs. UCB1-Uniform

Our methods can be easily confused with a similarly-named bandit framework called the *Max k-Armed Bandit* (Cicirello and Smith 2004, 2005; Streeter and Smith 2006b,a; Achab et al. 2017), later rediscovered as *Extreme Bandit* (Carpentier and Valko 2014). While both UCB1-Uniform and Max-k bandits are based on Extreme Value Theory, **UCB1-Uniform is not a Max-k bandit algorithm**. Also, the Max-k bandit fails to conceptually align with the classical planning application. Moreover, existing Max-k bandits target long-tail distributions while we target short-tail distributions.

Our UCB1-Uniform maximizes the cumulative reward by estimating the reward distribution using the maximum over the reward samples collected in the subtree. This is formalized as optimizing the cumulative regret, which is the difference between the (unknown) *expected* rewards of a suboptimal arm and the best arm. In contrast, the Max-k bandit aims at *maximizing the single maximum reward sampled/obtained*, which is a significantly different objective. It is formalized as optimizing the *extreme regret*, which is the difference between the (unknown, as yet unseen) *maximum* rewards of a suboptimal arm and the best arm, *We merely use the extreme (max/min) information for the cumulative regret minimization, and do not minimize the extreme regret.*

In addition, *Max-k bandit does not make sense in classical planning*. If we use a Max-k bandit in classical planning, we would use the minimization version that estimates the (yet unseen / unknown) minimum heuristic value in a subtree. **However, we know as a fact that the minimum possible heuristic value is 0** for **any** state from which the goal is reachable, because goal-aware heuristics are guaranteed to be 0 at goal states, and otherwise ∞ because they are dead-ends. In other words, Max-k bandit in classical planning estimates whether the goal is reachable or not. Such an estimate (0 or ∞) does not help the search algorithm *approach* the goal. Indeed, we are not interested in whether the goal is reachable (as we by default assume it is). Approaching the goal requires knowing how far the leaves in a subtree tend to be from the goal on average (cumulative regret).

Moreover, existing Max-k bandits target *long-tail distributions*. For example, ExtremeHunter is designed for heavy-tailed reward distributons, and estimates the parameters of a Pareto distribution, which is a subclass of Generalized Pareto distribution with $\xi > 0$. On the contrary, our approach (UCB1-Uniform) targets short-tailed distribution with a finite bound, which is a subclass of Generalized Pareto distribution with $\xi < 0$.

A9 Additional Results

A9.1 Cumulative Histograms for All Heuristics and All Search Statistics

Fig. A1-A3 show the cumulative histogram of the number of instances solved by our Pyperplan implementation within a particular evaluation/expansion/runtime limit.

References

Achab, M.; Clémençon, S.; Garivier, A.; Sabourin, A.; and Vernade, C. 2017. Max K-Armed Bandit: On the ExtremeHunter Algorithm and Beyond. In *Proc. of ECMLKDD*, 389–404. Springer.

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

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1): 5–33.

Boucheron, S.; Lugosi, G.; and Massart, P. 2013. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press.

Bylander, T. 1996. A Probabilistic Analysis of Prepositional STRIPS Planning. *Artificial Intelligence*, 81(1): 241–271.

Carpentier, A.; and Valko, M. 2014. Extreme Bandits. *Advances in Neural Information Processing Systems*, 27.

Cicirello, V. A.; and Smith, S. F. 2004. Heuristic Selection for Stochastic Search Optimization: Modeling Solution Quality by Extreme Value Theory. In *Proc. of CP*, 197–211. Springer.

Cicirello, V. A.; and Smith, S. F. 2005. The MAX K-Armed Bandit: A New Model of Exploration Applied to Search Heuristic Selection. In *Proc. of AAAI*, volume 3, 1355–1361.

Fikes, R. E.; Hart, P. E.; and Nilsson, N. J. 1972. Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3(1-3): 251–288.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation through Heuristic Search. *J. Artif. Intell. Res.(JAIR)*, 14: 253–302.

Hryniv, O. 2017. Durham University Lecture Note. Probability III-IV – MATH3211/4131.

Richter, S.; and Helmert, M. 2009. Preferred Operators and Deferred Evaluation in Satisficing Planning. In *Proc. of ICAPS*, 273–280.

Richter, S.; Thayer, J. T.; and Ruml, W. 2010. The Joy of Forgetting: Faster Anytime Search via Restarting. In *Proc. of ICAPS*.

Richter, S.; Westphal, M.; and Helmert, M. 2011. LAMA 2008 and 2011. In *Proc. of IPC*, 117–124.

Schulte, T.; and Keller, T. 2014. Balancing Exploration and Exploitation in Classical Planning. In *Proc. of SOCS*.

Streeter, M. J.; and Smith, S. F. 2006a. A Simple Distribution-Free Approach to the Max K-Armed Bandit Problem. In *Proc. of CP*, 560–574. Springer.

Streeter, M. J.; and Smith, S. F. 2006b. An Asymptotically Optimal Algorithm for the Max K-Armed Bandit Problem. In *Proc. of AAAI*, 135–142.

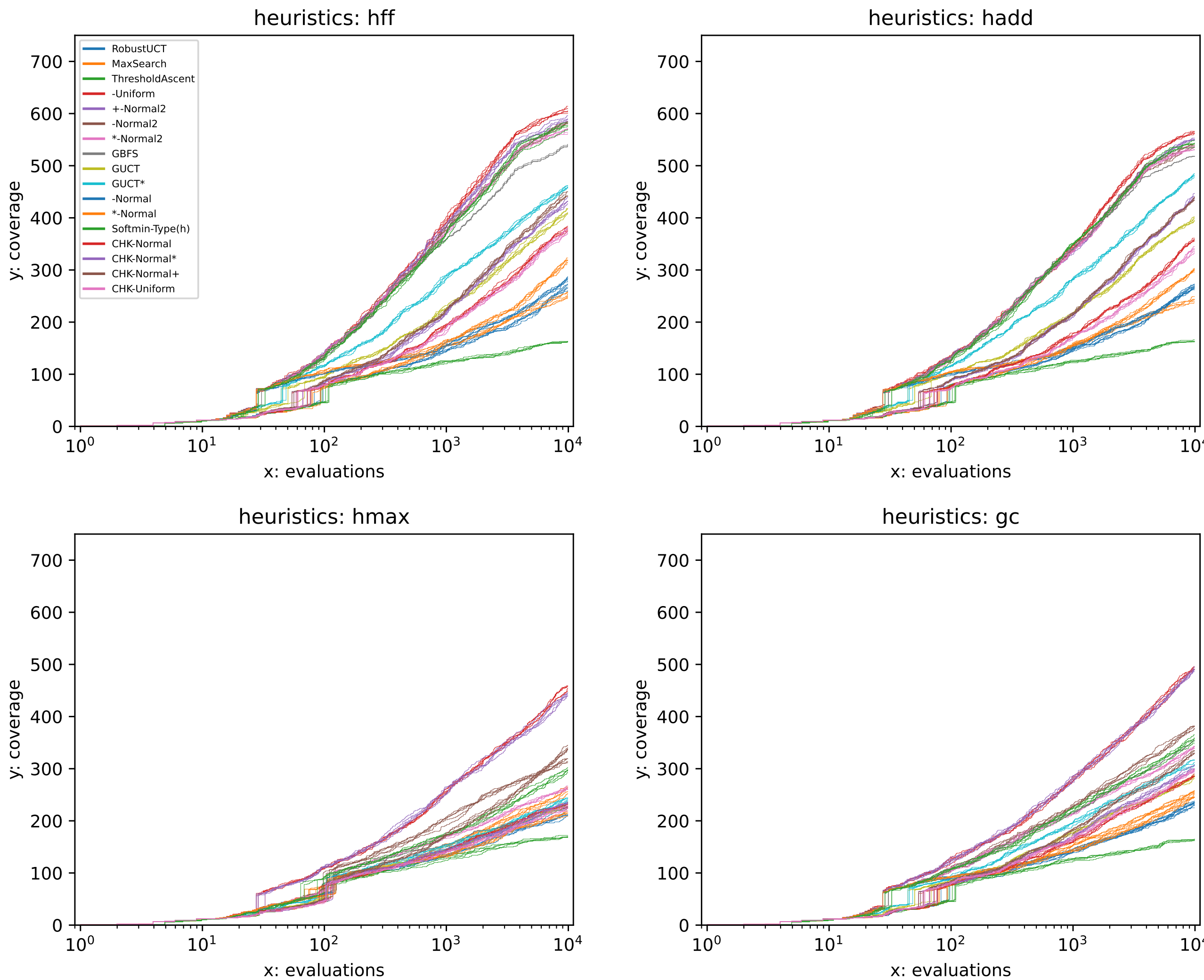

Figure A1: The cumulative histogram of the number of problem instances solved (y-axis) below a certain number of node evaluations (x-axis, 10,000 nodes maximum). Each line represents a random seed. The total numbers at the limit differ from those in other plots (this result does not limit the expansions or the runtime).

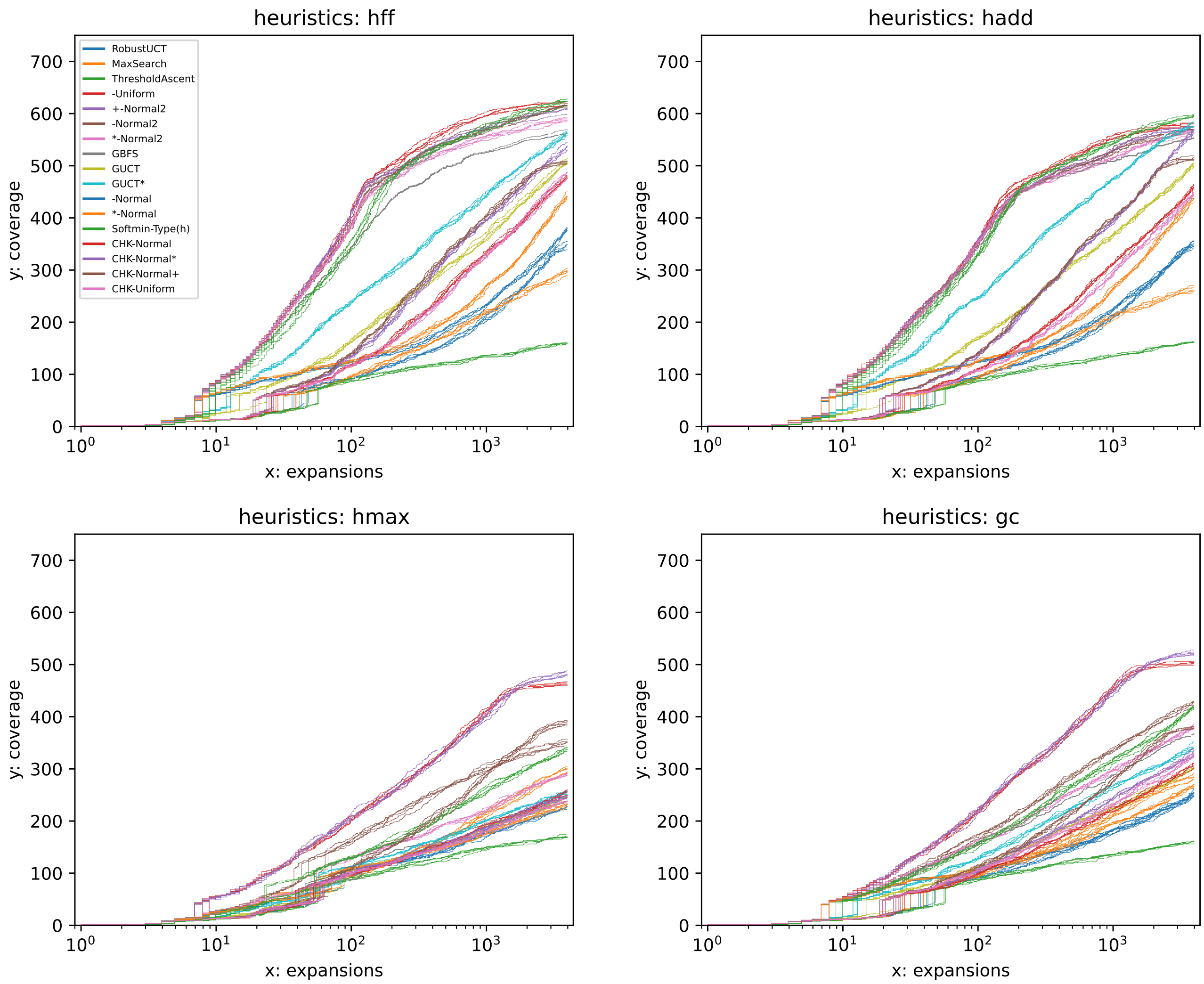

Figure A2: The cumulative histogram of the number of problem instances solved (y-axis) below a certain number of node expansions (x-axis, 4,000 nodes maximum). Each line represents a random seed. The total numbers at the limit differ from those in other plots (this result does not limit the evaluations or the runtime).

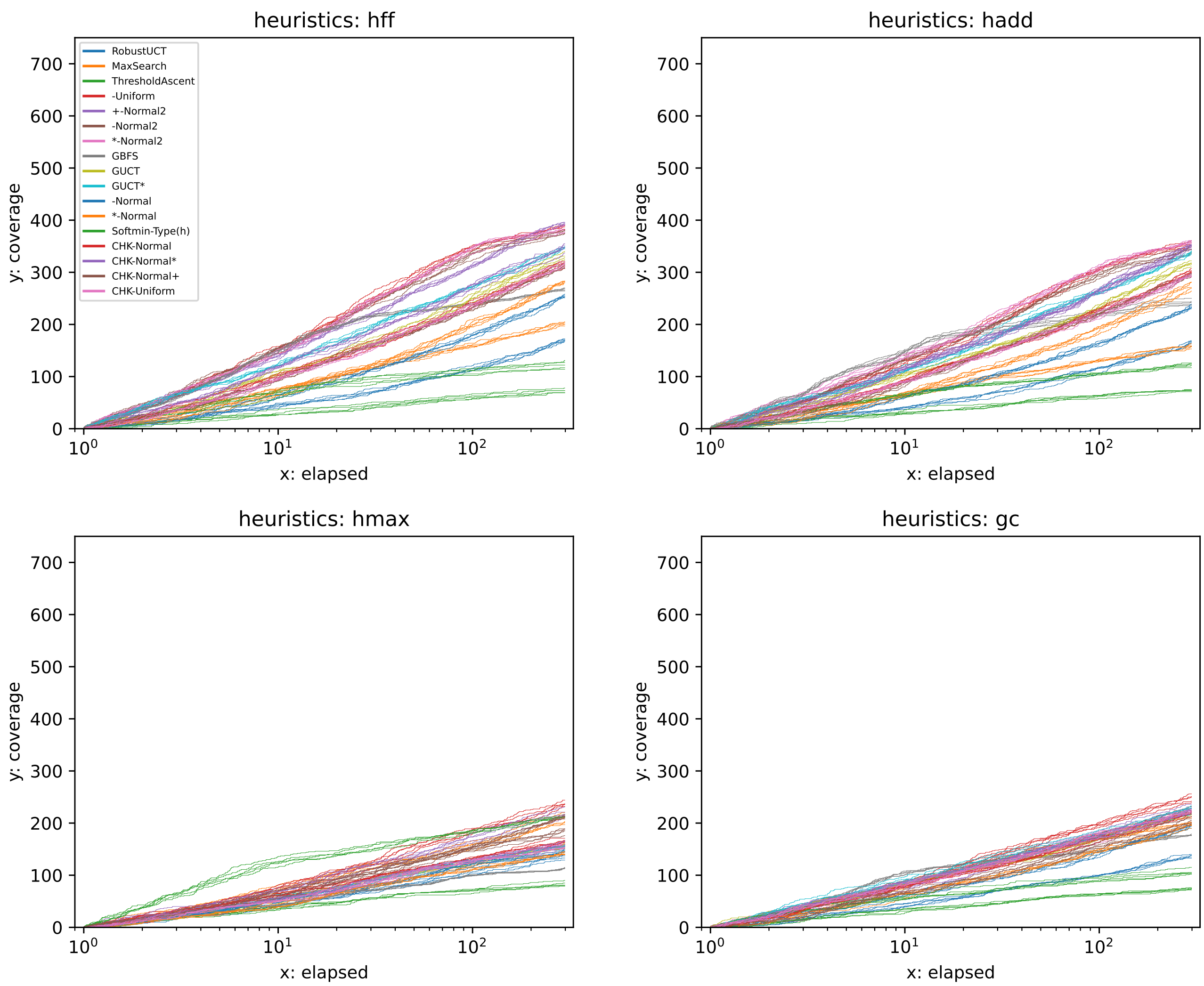

Figure A3: The cumulative histogram of the number of problem instances solved (y-axis) below a certain runtime (x-axis, 300 seconds maximum). Each line represents a random seed. The total numbers at the limit differ from those in other plots (this result does not limit the evaluations or the expansion).