Monte Carlo Tree Search

(and a bit of MDP)

Branislav Bošanský

PUI 2017/2018

MDP – VI/PI improvements

- value iteration is very simple
  - updates all states during each iteration
  - curse of dimensionality (huge state space)
  - **asynchronous VI**
    - select a single state to be updated in each iteration separately
    - each state must be updated infinitely often to guarantee convergence
    - lower memory requirements

- **Q: Can we use some heuristics to improve the convergence?**

MDP – VI/PI heuristics

- initial values can be assigned better
  - we can use a heuristic function instead of 0
- **Q: Can you think of any heuristic function?**
  - e.g., remember FFReplan/Robust FF?
  - we can use a single run of a planner on the determinized version
- **Q: What if the values V are initialized incorrectly?**

MDP – VI/PI with priority

- initialize V and a priority queue q
- select state s from the top of q and perform a Bellman backup
- add all possible predecessors of s to q
- repeat until convergence
  - priorities: changes in utility, position in the graph, …

- but, values are still updated regardless on the current values
- consider a typical probabilistic planning problem
  - finite-horizon MDP with some goal states

MDPs – Find and Revise

- we can further combine selective updates with heuristic search
  - starts with admissible $V(s) \geq V^*(s)$ for all states
  - select next state s' that is:
    - reachable from s_0 using current greedy policy π_V, and
    - residual $r(s') > \varepsilon$
  - update s'
  - repeat until such states exist

MDPs – Real-Time Dynamic Programming

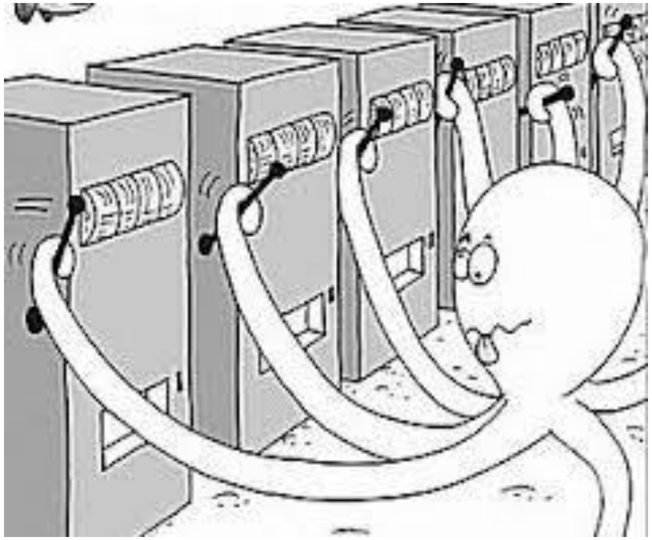
- updates the values only on the path from the starting state to the goal
- during one iteration updates one rollout/trial:
  - start with $s = s_0$
  - evaluate all actions using Bellman's Q-functions $Q(s, a)$
  - select action that maximizes current value: $\arg\max_{a \in A} Q(s, a)$
  - set $V(s) \leftarrow Q(s, a)$
  - get resulting state s'
  - if s' is not goal, then $s \leftarrow s'$ and go to step 2
- can be further improved with labeling (LRTDP) to identify solved states

MDPs – Using Monte Carlo Methods

- Monte Carlo Simulation: a technique that can be used to solve a mathematical or statistical problem using repeated sampling to determine the properties of some phenomenon (or behavior)
- Monte-Carlo Planning: compute a good policy for an MDP by interacting with an MDP simulator
- when simulator of a planning domain is available
  or can be learned from data
  - even if not described
    as a MDP
  - queries has to be cheap
    (relatively)

MDPs – Using Monte Carlo Methods

- sequential decision problem (over a single state)
- $k \geq 2$ stochastic actions (arms a_i)
  - each parameterized with an unknown probability distribution ν_i
  - each with a stored expectation μ_i
  - if executed (pulled) rewarded at random from ν_i
- objective
  - get maximal reward after N pulls
  - minimize regret for pulling wrong arm(s)



MCTS – UCB1

Upper Confidence Bounds

- UCB1 arm selection:
  - select arm a_i maximizing UCB1 formula:

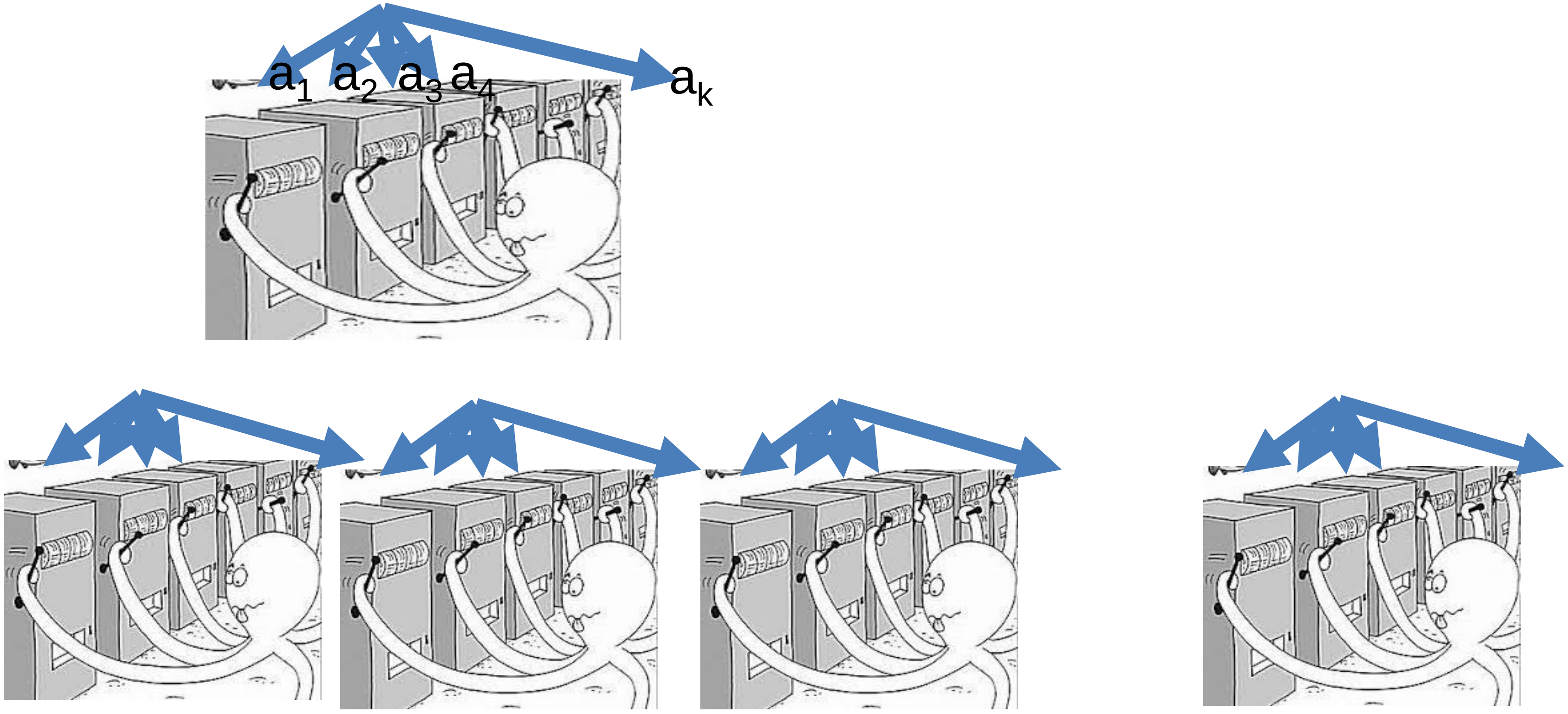
$$\mu_i + c\sqrt{\frac{\ln n}{n_i}}$$

  and update μ_i

  - n – times the state is visited; n_i – times the action is visited
  - μ_i – average reward from the previous plays
- exploration factor c ensures to evaluate actions that are evaluated rarely

MCTS – from UCB1 to UCT

- UCB1 applied on trees – UCT



MCTS – from UCB1 to UCT

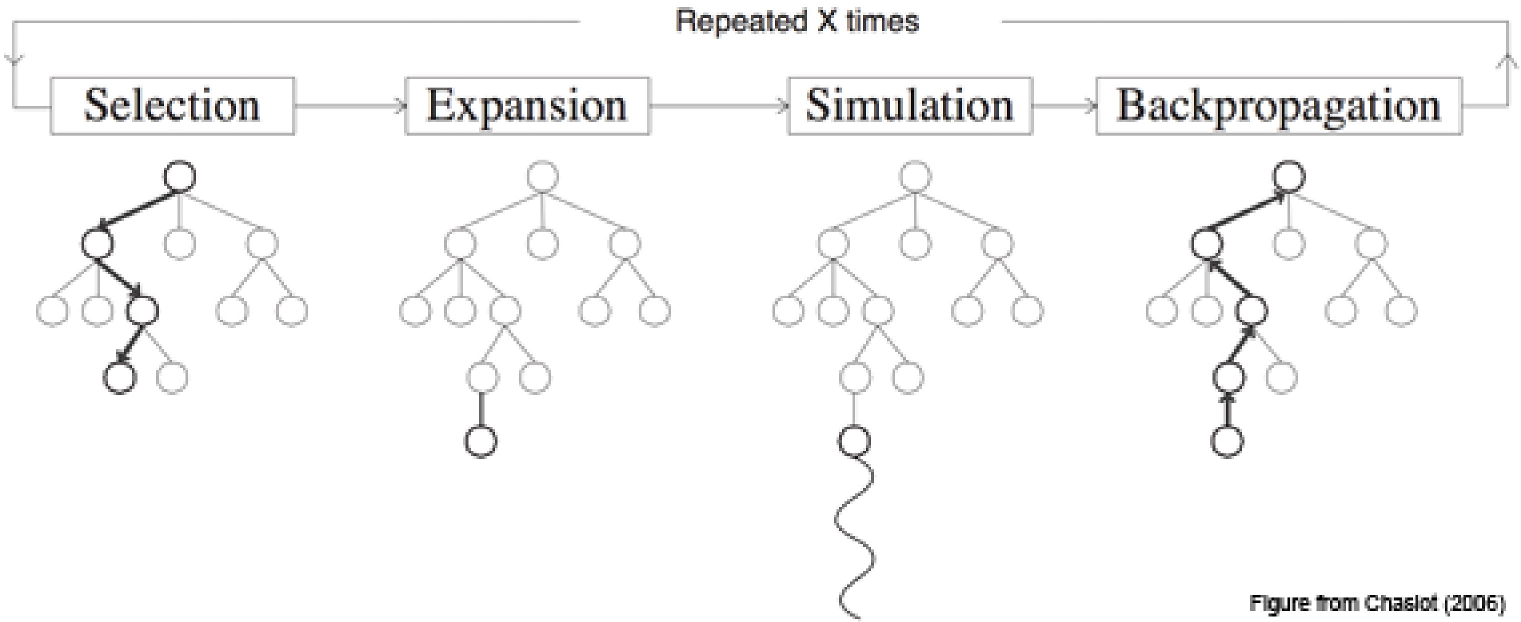
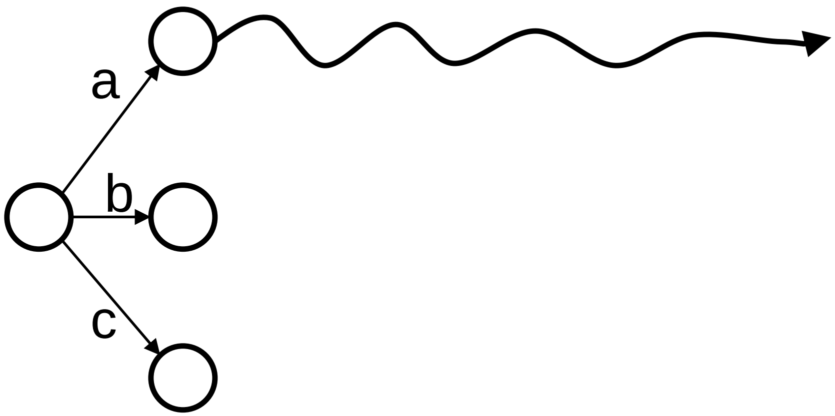


Figure from Chaslot (2006)

MCTS – UCT

Example



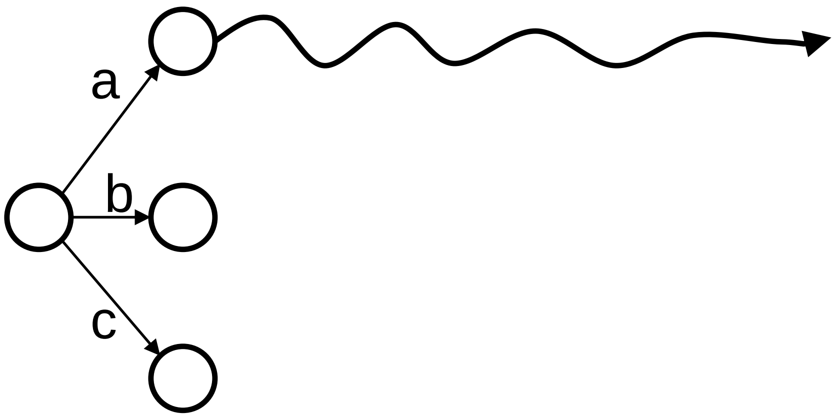
Random simulation results:
1. 2
2. 3
3. 1
4. 1
5. …

Questions (assume lexicographic tie braking):
1. Which action will be selected in the first iteration?
2. Which action will be selected in the fourth iteration?
3. Which action will be selected in the fifth iteration?

https://goo.gl/PQxm31

MCTS – UCT

Example



Random simulation results:
1. 2
2. 3
3. 1
4. 1
5. …

Questions (assume lexicographic tie braking):
1. Which action will be selected in the first iteration? **a**
2. Which action will be selected in the fourth iteration? **b**
3. Which action will be selected in the fifth iteration? **a**

https://goo.gl/PQxm31

MCTS – PROST

IPPC 2011 winner

- Vanilla UCT does not work very well in practice
  - huge branching factor
  - long (infinite) horizon
  - very difficult to find the correct plan by random rollouts

- these issues were addressed by PROST
  - search depth limitation
  - pruning out unreasonable actions
  - heuristic value initialization

MCTS – PROST

IPPC 2011 winner

- PROST – search depth limitation
  - we can limit search depth to L instead of solving to full depth
  - we need to do that if we have an infinite horizon
  - there can be a problem in re-using statistics from previous searches with limited depth (an optimal plan for horizon L does not have to be optimal for the full problem)

- PROST - pruning out unreasonable actions
  - we can heuristically identify unnecessary actions that do not yield any positive reward
  - compare to a NOOP action

MCTS – PROST

IPPC 2011 winner

- PROST – initialization of values
  - vanilla UCT first evaluates an action, if this action has not been evaluated before in state s
  - in case of a large branching factor, our search tree is very shallow
  - we can set some heuristic values to actions/children
  - we can set an artificial number of iterations
- we can set the values using some relaxation/determinization of the problem
  - Q-value initialization based on most probable outcome
  - the algorithm performs an iterative deepening search and checks whether the values are **informative** ($I(s,a) > I(s,a_\emptyset)$)

MCTS – PROST

IPPC 2011 winner

	CROSSING	ELEVATORS	GAME	NAVIGATION	RECON	SKILL	SYSADMIN	TRAFFIC	TOTAL
$\mathbf{P}^0$	0.46	0.01	0.86	0.14	0.00	0.89	0.86	0.98	0.53 ± 0.09
P	0.51	0.04	0.91	0.27	0.40	0.90	0.86	0.96	0.61 ± 0.08
$\mathbf{P}_{15}$	0.56	0.01	**0.95**	0.30	0.46	0.91	**0.91**	**0.99**	0.63 ± 0.08
$\mathbf{P}^I$	0.84	0.86	0.88	0.65	**0.98**	0.94	0.82	0.84	0.85 ± 0.05
$\mathbf{P}^I_{15}$	0.83	0.93	0.91	0.57	**0.98**	**0.95**	0.88	0.93	0.87 ± 0.05
$\mathbf{P}^{I,R}$	**0.98**	0.85	0.86	0.71	**0.98**	0.89	0.80	0.83	0.86 ± 0.04
$\mathbf{P}^{I,R}_{15}$	0.91	**0.94**	0.92	0.67	0.97	0.92	0.86	0.94	$\mathbf{0.89} \pm 0.04$
Glutton	0.80	0.90	0.67	**0.97**	0.76	0.86	0.34	0.67	0.75 ± 0.06

MCTS – Online planning

- Anytime algorithm
- A typical use case for MCTS-like approach is online planning – i.e. selecting the best action in the current situation in a limited time
- This corresponds to a simple regret – we do not want to regret not selecting a different action in the current state
- However, UCB1 optimizes the cumulative regret (selecting the best arm over all attempts)
- But these attempts are fictitious in our case!
- While MAB approach works in practice, it does not exactly correspond to the online planning

MCTS – BRUE

- There are two conflicting tasks
  - selecting the best action in state s (reaching s’)
  - exploring and finding the best continuation after s’ is reached

- In order to satisfy Task 2 – we need to select the best action sufficiently often
- To do that, we need to know the optimal continuation
- BRUE algorithm uses two different action selection methods
  - the action in the selection phase is selected uniformly
  - the action in the update phase is selected using greedy strategy

MCTS – BRUE

procedure $\textsc{Update}(\rho)$
  for $d \leftarrow |\rho|, \ldots, 1$ **do**
    $h \leftarrow H - d$
    $\langle s, a, r, s' \rangle \leftarrow \rho[d]$
    $n(s\langle h \rangle) \leftarrow n(s\langle h \rangle) + 1$
    $n(s\langle h \rangle, a) \leftarrow n(s\langle h \rangle, a) + 1$
    $n(s\langle h \rangle, a, s') \leftarrow n(s\langle h \rangle, a, s') + 1$
    $\bar{r} \leftarrow r + \textsc{Estimate}(s'\langle h-1 \rangle)$
    $\textsc{MC-backup}(s\langle h \rangle, a, \bar{r})$

procedure $\textsc{Estimate}(s\langle h \rangle)$
  $\bar{r} \leftarrow 0$
  for $d \leftarrow 0, \ldots, h-1$ **do**
    $a \leftarrow \textsc{EstAction}(s\langle h-d \rangle)$
    $s' \leftarrow \textsc{EstOutcome}(s\langle h-d \rangle, a)$
    $\tilde{r}_{d+1} \leftarrow R(s, a, s')$
    $\bar{r} \leftarrow \bar{r} + \tilde{r}_{d+1}$
    $s \leftarrow s'$
  return $\bar{r}$

procedure $\textsc{StopRollout}(\rho)$
  $d \leftarrow |\rho|$
  return $d = H$ **or** $A(\rho[d].s') = \emptyset$

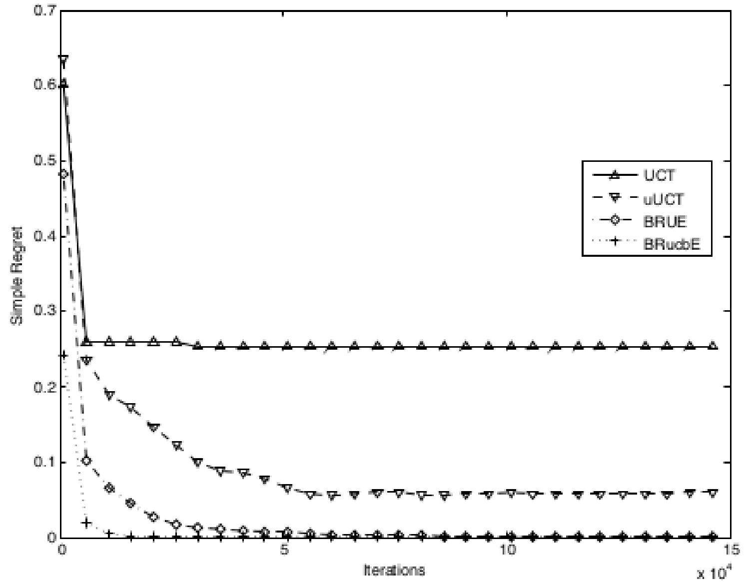
procedure $\textsc{RolloutAction}(s\langle h \rangle)$ *// uniform*
  return $a \sim \mathcal{U}[A(s)]$

procedure $\textsc{RolloutOutcome}(s\langle h \rangle, a)$
  return $s' \sim \mathcal{P}(S \mid s, a)$

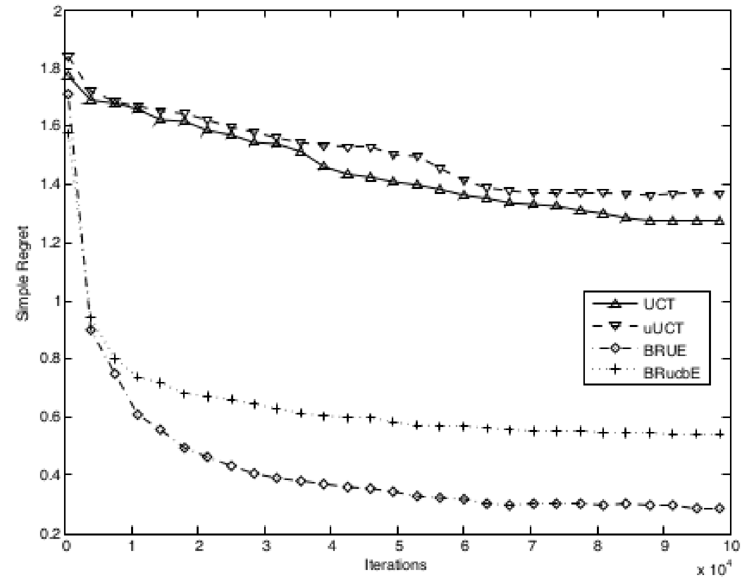
procedure $\textsc{EstAction}(s\langle h \rangle)$ *// best*
  return $\operatorname{argmax}_{a \in A(s)} \widehat{Q}(s\langle h \rangle, a)$

procedure $\textsc{EstOutcome}(s\langle h \rangle, a)$
  for $s' : n(s\langle h \rangle, a, s') > 0$ **do**
    $\widehat{\mathcal{P}}(S = s' \mid s, a) \leftarrow \frac{n(s\langle h \rangle, a, s')}{n(s\langle h \rangle, a)}$
  return $s' \sim \widehat{\mathcal{P}}(S \mid s, a)$

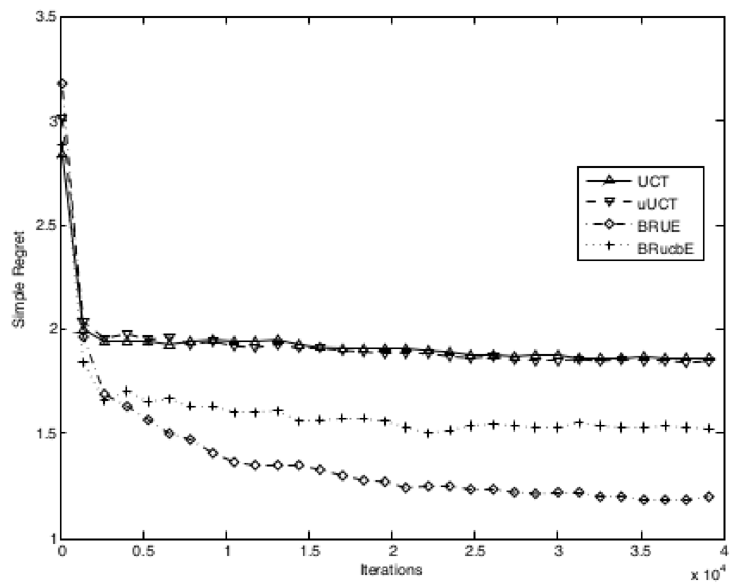
MCTS – BRUE



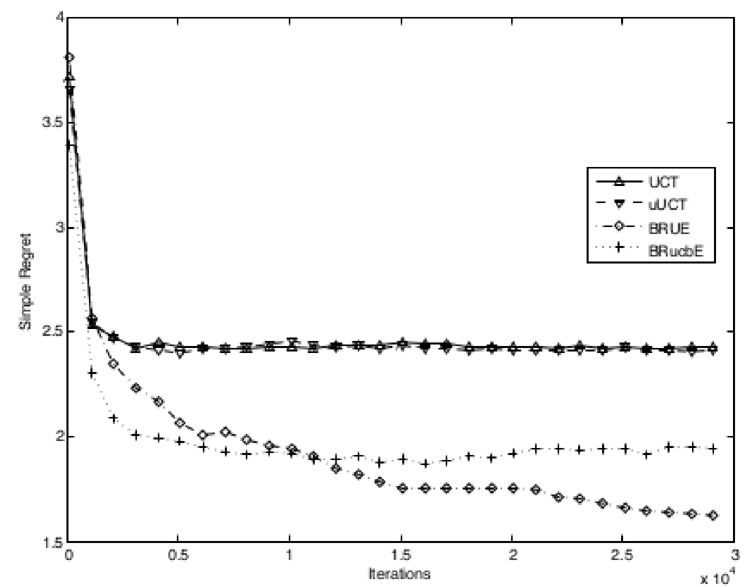
5×5



10×10



20×20



40×40

Trial-based Heuristic Tree Search (THTS)

- a common framework based on five ingredients:
  - heuristic function
  - backup function
  - action selection
  - outcome selection
  - trial length
- subsuming: MCTS, UCT, FIND-and-REVISE, AO* (AND/OR graph solver), Real-Time Dynamic Programming (RTDP), various heuristic functions (e.g., iterative deepening search)
- providing: MaxUCT, UCT*, …
- UCT* in PROST 2014 is currently best performing IPPC planner

Trial-based Heuristic Tree Search (THTS)

Algorithm 1: The THTS schema.

THTS(MDP M, timeout T)**:**
    $n_0 \leftarrow$ getRootNode(M)
    while not solved(n_0) and time() $< T$ **do**
        visitDecisionNode(n_0)
    return greedyAction(n_0)

visitDecisionNode(Node n_d)**:**
    if n_d was never visited **then** initializeNode(n_d)
    $N \leftarrow$ selectAction(n_d)
    for $n_c \in N$ **do**
        visitChanceNode(n_c)
    backupDecisionNode(n_d)

visitChanceNode(Node n_c)**:**
    $N \leftarrow$ selectOutcome(n_c)
    for $n_d \in N$ **do**
        visitDecisionNode(n_d)
    backupChanceNode(n_c)

Trial-based Heuristic Tree Search (THTS)

- maintains explicit tree of alternating decision and chance nodes
- selection phase
  - alternating **visitDecisionNode** and **visitChanceNode**
  - selection by **selectAction** and **selectOutcome**
  - tree traversing (down)
- expansion phase
  - when unvisited node encountered
  - add child node for each action
  - heuristics used to initialize the estimates
  - allows selection phase for new nodes

Algorithm 1: The THTS schema.

```
THTS(MDP M, timeout T):
    n_0 ← getRootNode(M)
    while not solved(n_0) and time() < T do
        visitDecisionNode(n_0)
    return greedyAction(n_0)
visitDecisionNode(Node n_d):
    if n_d was never visited then initializeNode(n_d)
    N ← selectAction(n_d)
    for n_c ∈ N do
        visitChanceNode(n_c)
    backupDecisionNode(n_d)
visitChanceNode(Node n_c):
    N ← selectOutcome(n_c)
    for n_d ∈ N do
        visitDecisionNode(n_d)
    backupChanceNode(n_c)
```

Trial-based Heuristic Tree Search (THTS)

- selection and expansion phases alternate until the **trial length**
- backup phase (**backupDecisionNode & backupChanceNode**)
  - all selected nodes are updated in reverse order
  - when another selected, but not yet visited → selection phase
  - a trial ends when the backup is called on the root node
  - tree backing (up)
- the process is repeated until the timeout T allows for another trial
- highest expectation action is returned **greedyAction**

Algorithm 1: The THTS schema.

THTS(MDP M, timeout T)**:**
  $n_0 \leftarrow$ getRootNode(M)
  while not solved(n_0) and time() $< T$ **do**
    visitDecisionNode(n_0)
  return greedyAction(n_0)

visitDecisionNode(Node n_d)**:**
  if n_d was never visited **then** initializeNode(n_d)
  $N \leftarrow$ selectAction(n_d)
  for $n_c \in N$ **do**
    visitChanceNode(n_c)
  backupDecisionNode(n_d)

visitChanceNode(Node n_c)**:**
  $N \leftarrow$ selectOutcome(n_c)
  for $n_d \in N$ **do**
    visitDecisionNode(n_d)
  backupChanceNode(n_c)

Trial-based Heuristic Tree Search (THTS)

- Heuristic function
  - action value initialization (Q-value)
    $$h: S \times A \mapsto \mathbb{R}$$
  - state value initialization (V-value)
    $$h: S \mapsto \mathbb{R}$$
- Action selection
  - UCB1, ϵ-greedy, …
- Outcome selection
  - Monte Carlo sampling; outcome based on biggest potential impact

Trial-based Heuristic Tree Search (THTS)

Backup

- optimal policy derived from the Bellman optimality equation:

$$V^*(s) = \begin{cases} 0 & \text{if } s \text{ is terminal} \\ \max_{a \in A} Q^*(a,s) & \text{otherwise} \end{cases}$$

$$Q^*(a,s) = R(a,s) + \sum_{s' \in S} P(s'|a,s) \cdot V^*(s')$$

- Full Bellman backup ~ Bellman optimality equation, *k* trials

- Monte Carlo backup

$$V^k(s) = \begin{cases} 0 & \text{if } s \text{ is terminal} \\ \frac{\sum_{a \in A} n_{a,s} \cdot Q^k(a,s)}{n_s} & \text{otherwise} \end{cases}$$

$$Q^k(a,s) = R(a,s) + \frac{\sum_{s' \in S} n_{s'} \cdot V^k(s')}{n_{a,s}}$$

MaxUCT

- backup function
  - action-value by Monte Carlo backup ($Q^k(s)$)
  - state-value by Full Bellman backup ($V^*(s)$)
- action selection → UCB1
- outcome selection → Monte Carlo sampling (MDP based)
- heuristic function → N/A
- trial length → UCT (horizon length, i.e. to leafs)

UCT*

- backup function
  - Partial Bellman backup
    (weighted proportionally to subtree probability)
- action selection → UCB1
- outcome selection → Monte Carlo sampling (MDP based)
- heuristic function → Iterative Deepening Search (depth: 15)
- trial length → explicit tree length +1
  (only initialized new nodes using heuristics)
- resembles classical heuristic Breadth-First-Search (rather than UCT Depth-First-Search)

UCT*

	Elevators	Sysadmin	Recon	Game	Traffic	Crossing	Skill	Navigation	Total
UCT	0.93	0.66	**0.99**	0.88	0.84	0.85	0.93	0.81	0.86
MaxUCT	**0.97**	0.71	0.88	0.9	0.86	**0.96**	0.95	0.66	0.86
DP-UCT	**0.97**	0.65	0.89	0.89	0.87	**0.96**	**0.98**	**0.98**	0.9
UCT*	**0.97**	**1.0**	0.88	**0.98**	**0.99**	**0.98**	**0.97**	**0.96**	**0.97**
Prost	0.93	0.82	**0.99**	0.93	0.93	0.82	**0.97**	0.55	0.87

References

- Keller & Eyerich "PROST: Probabilistic Planning Based on UCT" ICAPS 2012
- Feldman & Domshlak "Simple Regret Optimization in Online Planning for Markov Decision Processes" JAIR 2014
- Keller & Helmert "Trial-based Heuristic Tree Search for Finite Horizon MDPs" ICAPS 2013