

Monte Carlo Tree Search

Branislav Božanský

PAH 2015/2016

Markov Decision Processes (MDPs)

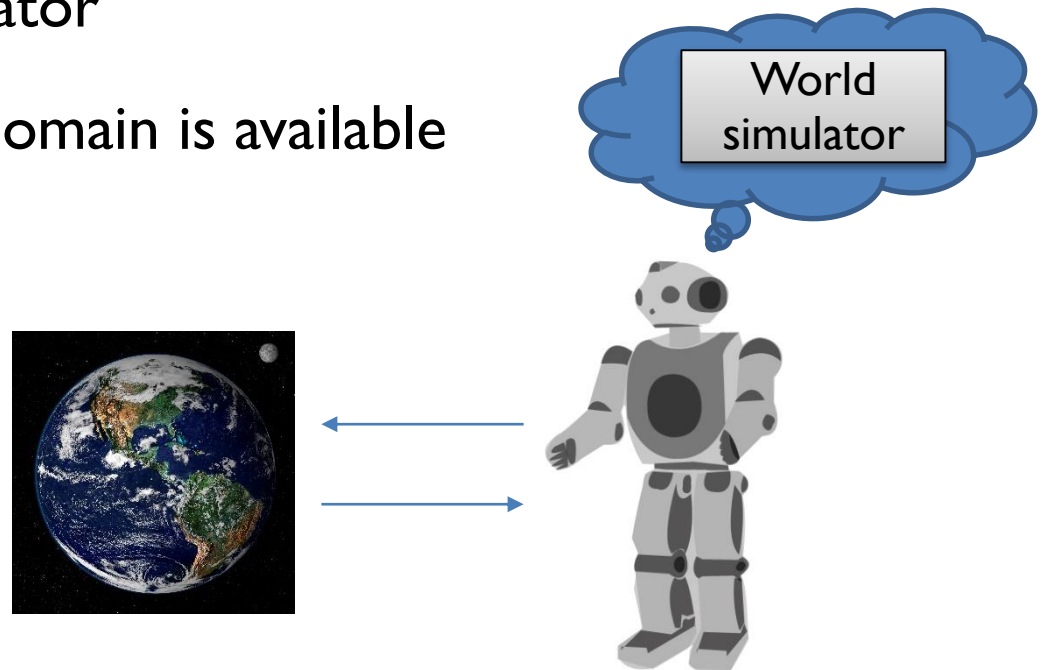
- online planning ~ any-time algorithm
 - in a limited time decide on best move
 - execute the move and repeat
- reward on final states (often win or lose)
- implicit (and compact) representation of large MDPs
 - cannot grow the full tree
 - cannot safely cut branches
 - cannot be greedy
- .

MDPs – Using Monte Carlo Methods

- Monte Carlo sampling is a well known method for searching through large state space
- exploiting MC in sequential decision making has first been successfully designed in (Kocsis & Szepesvari, 2006)
- foundations in mathematical theory
 - Multi-Armed Bandit (MAB) Problem
 - Upper Confidence Bounds (UCB)
 - exploration/exploitation dilemma
- .

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)



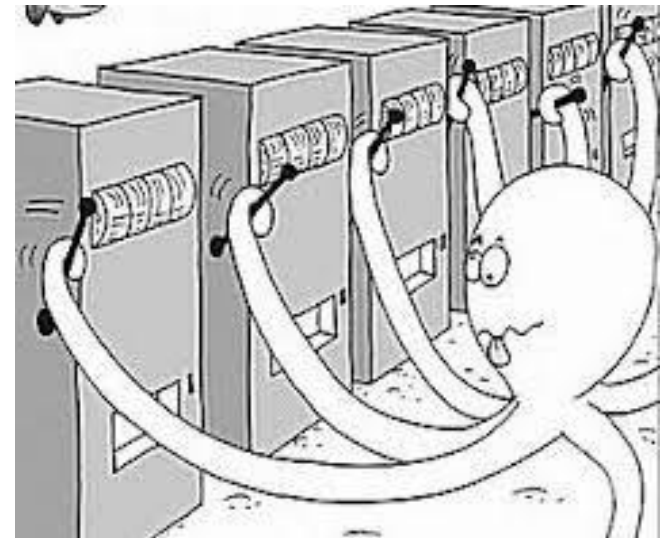
- .

Monte Carlo Simulation

- Domains with Simulators
 - traffic
 - robotics
 - military missions
 - computer network
 - disaster relief and emergency planning
 - sports
 - board and video games
 - board (Go, Hex, Settlers of Catan, ...), card (poker, Magic: The Gathering, ...), RTS (Total War: Rome II, ...)
- .

Multi-Armed Bandit Problem

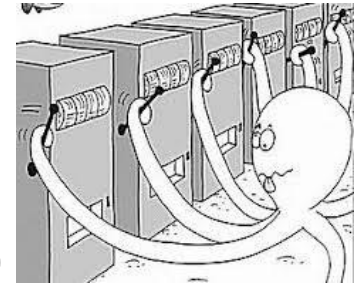
- sequential decision problem (over a single state)
- $k \geq 2$ stochastic actions (arms a_i)
 - each parameterized with an unknown probability distribution ν_i
 - each with a stored expectation μ_i
 - if executed (pulled) rewarded at random from ν_i
- objective
 - get maximal reward after N pulls
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



- .

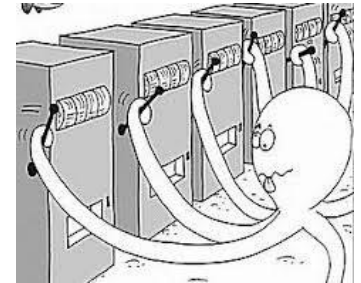
Multi-Armed Bandit Problem (variants)

- learning-while-acting
 - reward for each action
 - **cumulative regret** (exploration/exploitation dilemma)
 - algorithms: ϵ -greedy, UCB1
 - used in: Monte Carlo Tree Search, UCB1 applied to trees (UCT)
- online planning/learning-while-planning
 - reward only for final decision (N “free action tries” by simulator)
 - **simple regret** (only exploration)
 - algorithms: uniform sampling, ϵ -greedy, Sequential Halving
 - used in: Trial-based Heuristic Tree Search (THTS)
- .



ϵ -greedy

- parameterized by ϵ
- flip a ϵ -biased coin
 - (ϵ): select arm a_i randomly with uniform probability and update μ_i
 - ($1 - \epsilon$): select estimated best arm a^* and update μ^*
- typically $\epsilon \approx 0, 1$ (but this can vary depending on circumstances)
- exponential convergence to the optimal arm

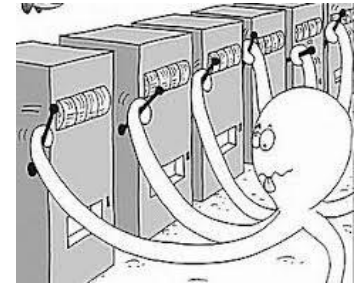


- .

Upper Confidence Bounds

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

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

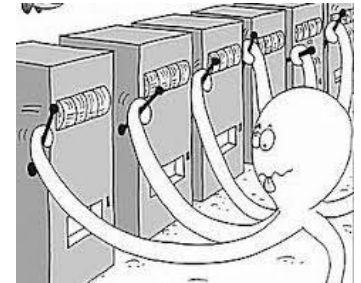


and update μ_i

- n – times the state is visited; n_i – times the action is visited
- μ_i – average reward from the previous plays
- exploration factor ensures to evaluate actions that are evaluated rarely
- only polynomial (but empirically fast) convergence to optimal arm
- .

Sequential Halving

- parameterized by sampling budget T
- (1) begins with all arms as candidate arms S
- (2) sample/play all candidate arms in S t -times

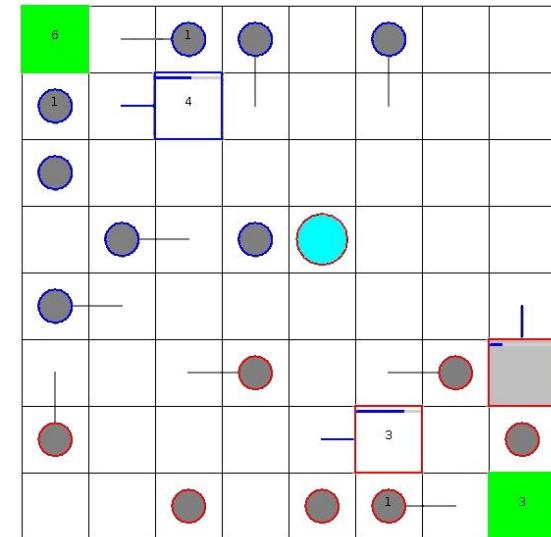


$$t = \left\lceil \frac{T}{|S| \lceil \log_2 k \rceil} \right\rceil$$

and update their μ_i

- (3) remove [half] of the candidate arms with lowest μ_i
- (4) until there is only one (resulting) candidate arm: goto (2)
- exponential convergence to the optimal arm (provided the budget is going to ∞ ; not any-time)
- .

Combinatorial Multi-Armed Bandit Problem

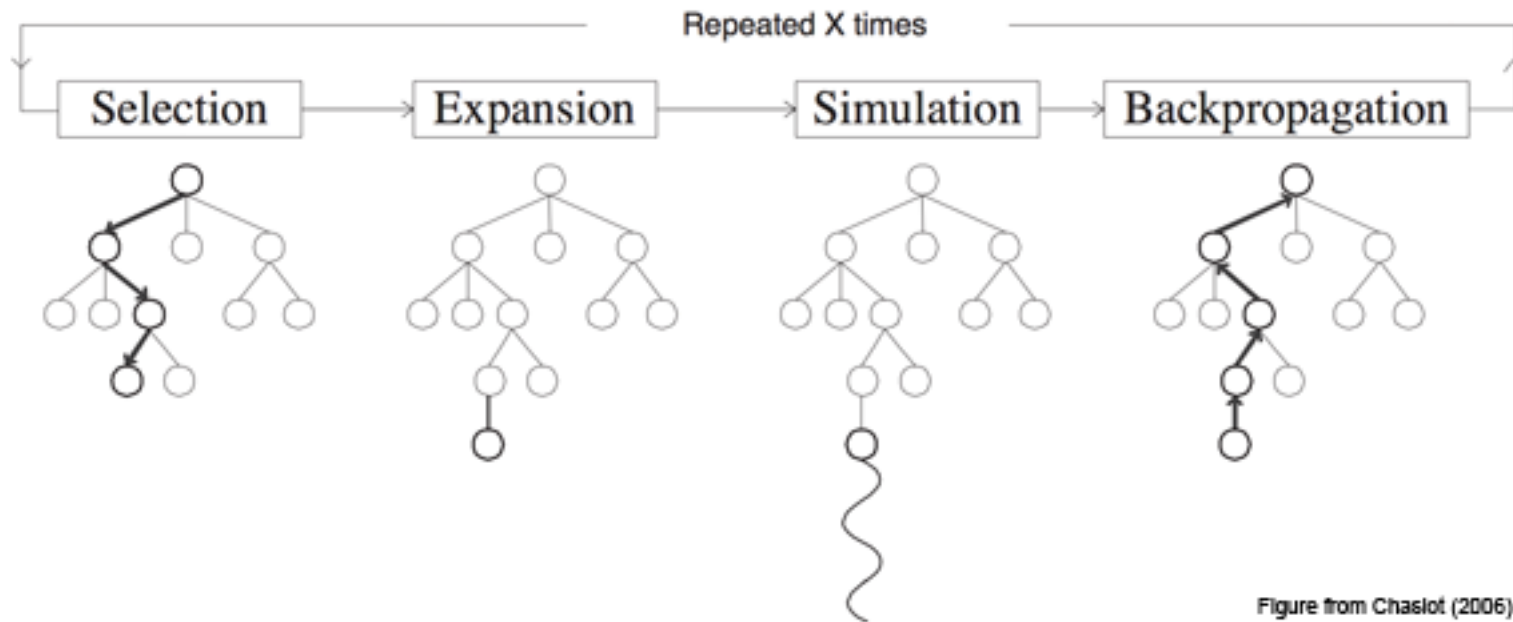


- combination of actions (arms) has to be selected (some forbidden)
- reward defined over combinations of actions (c-actions)
- expectation of reward per c-action
- ☹️ curse of dimensionality (action combinations), $O(\exp(|\Pi|))$
- 😊 we can approximate
 - randomly generate candidate c-actions, pick the best one (NMC)
 - assume additive rewards for one c-action; linear-side inform. (LSI)
- de-coupled dynamics

Monte Carlo Tree Search (MCTS)

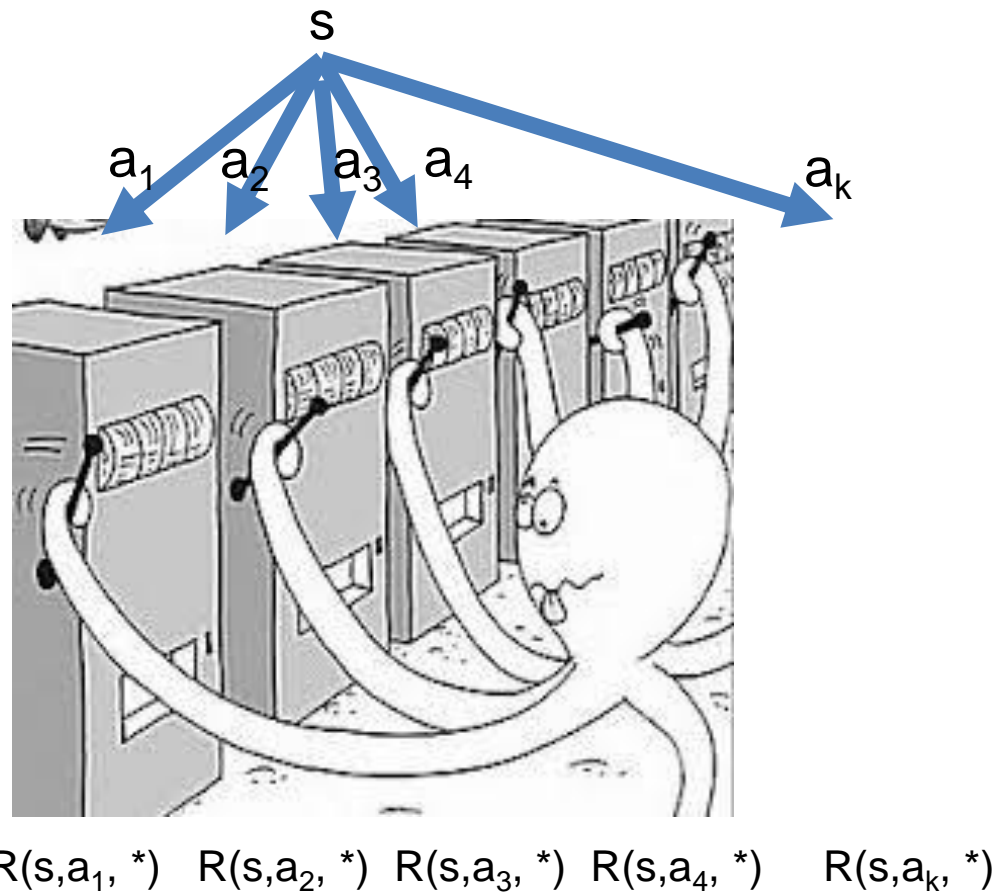
- sequential decision making (over different states)
- gradually grow the search tree
- two types of tree nodes
 - decision nodes (action selection) – the algorithm selects
 - chance nodes (world selection) – the world selects the outcome (in case of MDP model based on known probabilities)
- returned solution: path (action from root) visited the most often
- .

Monte Carlo Tree Search



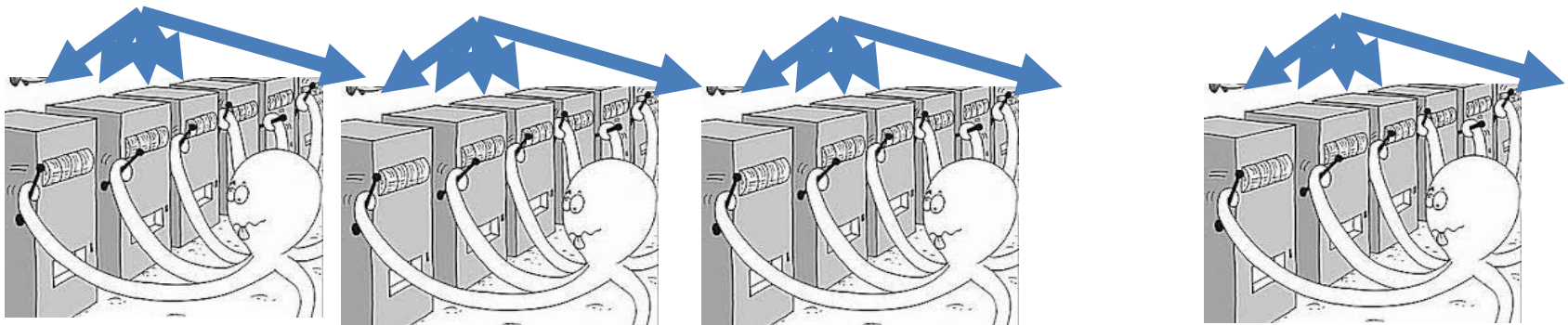
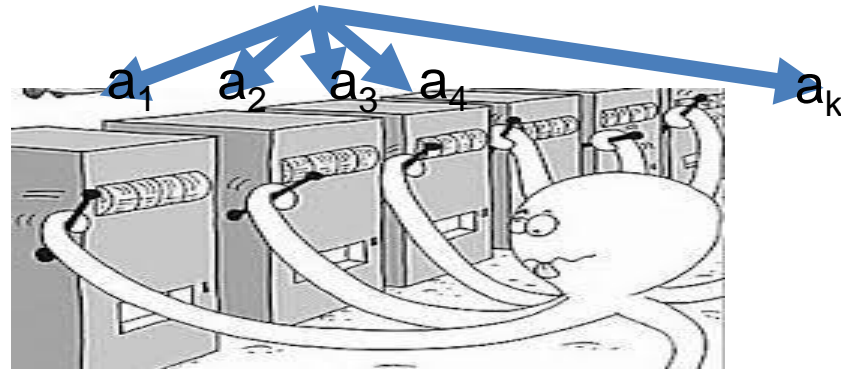
UCT – Principle

- UCBI applied on trees – UCT



UCT – Principle

- UCBI applied on trees – UCT



UCT – Selection

- selection (UCBI)
 - for each action a_i applicable in s UCB selects the one that maximizes

$$c \sqrt{\frac{\ln n}{n_i}} + \sum_{s' \in S} T(s, a_i, s') [R(s, a_i, s') + \gamma V(s')]$$

- n – times the state is visited; n_i – times the action is visited
- $V(s)$ – average reward from the previous iterations
- c - exploration constant (linear to expected utility)
- exploration factor ensures to evaluate actions that are evaluated rarely

UCT – Expansion, Simulation, Backup

- expansion (MCTS)
 - in a selection node where not all actions were yet sampled, expand (uniformly) randomly one of the new nodes
- simulation (MCTS)
 - (uniformly) randomly select actions in decision nodes
 - using the simulator based on the probabilities in the MDP simulate world behavior in the chance nodes MDP
- backup (MCTS)
 - updating μ_i^s for all search tree nodes along the trial based on the rewards (incl. the simulation)
- .

Beyond UCT

- UCT is far from optimal algorithm
 - there exist simple examples where vanilla UCT performs bad
- number of reasons
 - learning the best action is different from learning the best (contingency) plan
 - situation that occur in states does not exactly correspond to multi-armed bandit (mathematically)
- there are modifications and improvements
 - RAVE (Gelly & Silver, 2007) → rapid action value estimate
 - THTS (Keller & Helmert, 2013) → MaxUCT, UCT*
 - many others ...
 - .

Beyond UCT many others

- MCTS/UCT requires large number of iterations to converge
- depth-limited rollouts / heuristic function
- reducing branching factor (some actions are dominated → remove)
- progressive widening
- different action selection principles
- improving rollout policy (biased simulators, “clever” decision nodes)
- incorporate prior knowledge
- parallelization
- extendable to more complex settings (partial observation, games, ...)
- .

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)

- 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
 - UCBI, ϵ -greedy, ...
- Outcome selection
 - Monte Carlo sampling; outcome based on biggest potential impact
- .

Trial-based Heuristic Tree Search (THTS)

- 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

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

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

Trial-based Heuristic Tree Search (THTS)

Algorithm 1: The THTS schema.

```

1 THTS(MDP  $M$ , timeout  $T$ ):
2    $n_0 \leftarrow \text{getRootNode}(M)$ 
3   while not solved( $n_0$ ) and time() <  $T$  do
4     visitDecisionNode( $n_0$ )
5   return greedyAction( $n_0$ )

6 visitDecisionNode(Node  $n_d$ ):
7   if  $n_d$  was never visited then initializeNode( $n_d$ )
8    $N \leftarrow \text{selectAction}(n_d)$ 
9   for  $n_c \in N$  do
10    visitChanceNode( $n_c$ )
11  backupDecisionNode( $n_d$ )

12 visitChanceNode(Node  $n_c$ ):
13   $N \leftarrow \text{selectOutcome}(n_c)$ 
14  for  $n_d \in N$  do
15    visitDecisionNode( $n_d$ )
16  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
 - added child node for each action
 - heuristics used to initialize the estimates
 - allows selection phase for new nodes
 - .

Algorithm 1: The THTS schema.

```

1 THTS(MDP  $M$ , timeout  $T$ ):
2    $n_0 \leftarrow \text{getRootNode}(M)$ 
3   while not solved( $n_0$ ) and time() <  $T$  do
4     visitDecisionNode( $n_0$ )
5   return greedyAction( $n_0$ )
6 visitDecisionNode(Node  $n_d$ ):
7   if  $n_d$  was never visited then initializeNode( $n_d$ )
8    $N \leftarrow \text{selectAction}(n_d)$ 
9   for  $n_c \in N$  do
10    visitChanceNode( $n_c$ )
11  backupDecisionNode( $n_d$ )
12 visitChanceNode(Node  $n_c$ ):
13   $N \leftarrow \text{selectOutcome}(n_c)$ 
14  for  $n_d \in N$  do
15    visitDecisionNode( $n_d$ )
16  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 \rightarrow 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.

```

1 THTS(MDP  $M$ , timeout  $T$ ):
2    $n_0 \leftarrow \text{getRootNode}(M)$ 
3   while not solved( $n_0$ ) and time() <  $T$  do
4     visitDecisionNode( $n_0$ )
5   return greedyAction( $n_0$ )
6 visitDecisionNode(Node  $n_d$ ):
7   if  $n_d$  was never visited then initializeNode( $n_d$ )
8    $N \leftarrow \text{selectAction}(n_d)$ 
9   for  $n_c \in N$  do
10    visitChanceNode( $n_c$ )
11    backupDecisionNode( $n_d$ )
12 visitChanceNode(Node  $n_c$ ):
13    $N \leftarrow \text{selectOutcome}(n_c)$ 
14   for  $n_d \in N$  do
15     visitDecisionNode( $n_d$ )
16   backupChanceNode( $n_c$ )

```

MaxUCT

- backup function
 - action-value by Monte Carlo backup ($Q^k(s)$)
 - state-value by Full Bellman backup ($V^*(s)$)
- action selection → UCBI
- 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 → UCBI
- 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)