Algorithm Selection using Reinforcement Learning

Michail G. Lagoudakis and Michael L. Littman

In Proceeding ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning

Markov Decision Processes (MDP)

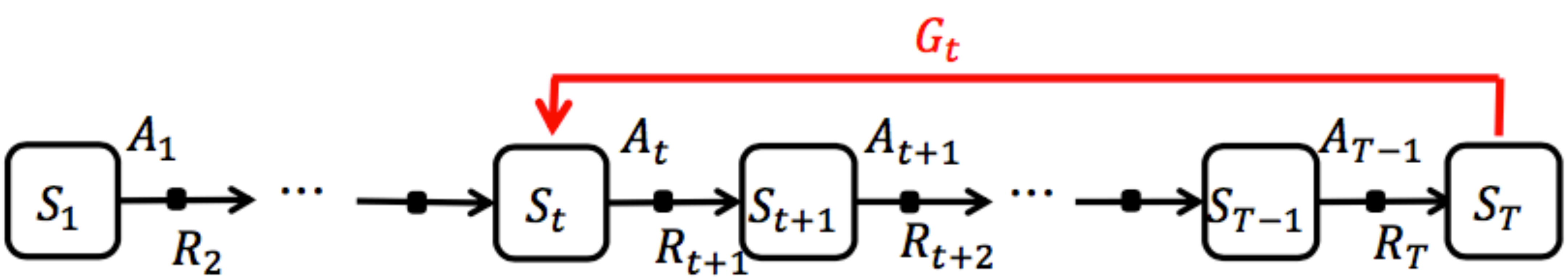
- A Markov Decision Process is a tuple
  $<\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma>$
  - $\mathcal{S}$ is a finite set of states
  - $\mathcal{A}$ is a finite set of actions
  - $\mathcal{P}$ is a state transition probability matrix (part of the environment),
    $\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
  - $\mathcal{R}$ is a reward function,
    $\mathcal{R}_{s}^{a} = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
  - γ is a discount factor $\gamma \in [0, 1]$.

Model-free Reinforcement Learning

- Temporal Difference (TD) Learning
  - TD methods learn directly from episodes of experience
  - TD is model-free: no knowledge of MDP transitions / rewards
  - TD learns from incomplete episodes, by bootstrapping
  - TD updates a guess towards a guess
- Monte-Carlo (MC) Learning
  - MC methods learn directly from episodes of experience
  - MC is model-free: no knowledge of MDP transitions / rewards
  - MC learns from complete episodes: no bootstrapping
  - MC uses the simplest possible idea: value = mean return
  - Caveat: can only apply MC to episodic MDPs
    - All episodes must terminate
  - Monte-Carlo Tree Search (MCTS) is a successful one based on MC learning.

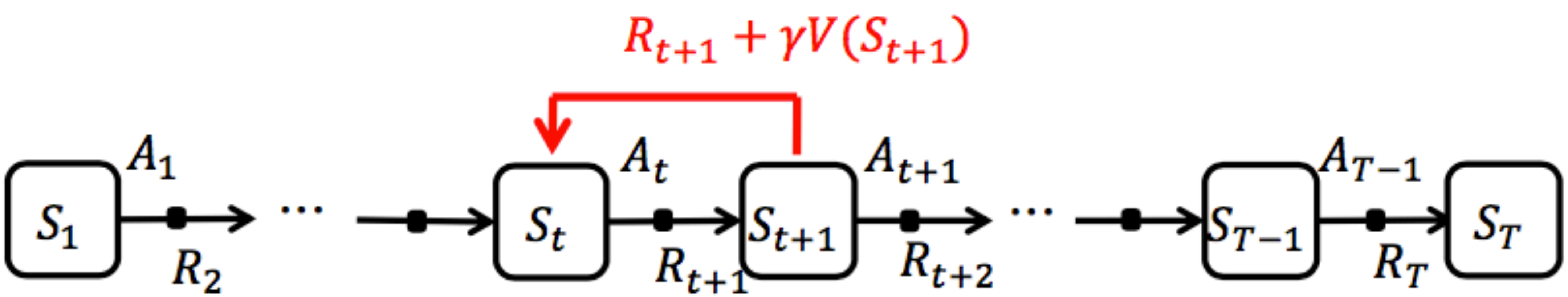
Monte-Carlo Learning

- Incremental Monte-Carlo
  - Update value $V(S_t)$ toward actual return G_t
    $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$
  - α: learning rate, or called step size.
- Unbiased, but high variance.



Temporal-Difference Learning

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$
    $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
  - TD target: $R_{t+1} + \gamma V(S_{t+1})$
  - TD error: $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$
  - α: learning rate, or called step size.
- Biased, but lower variance



Algorithm Selection

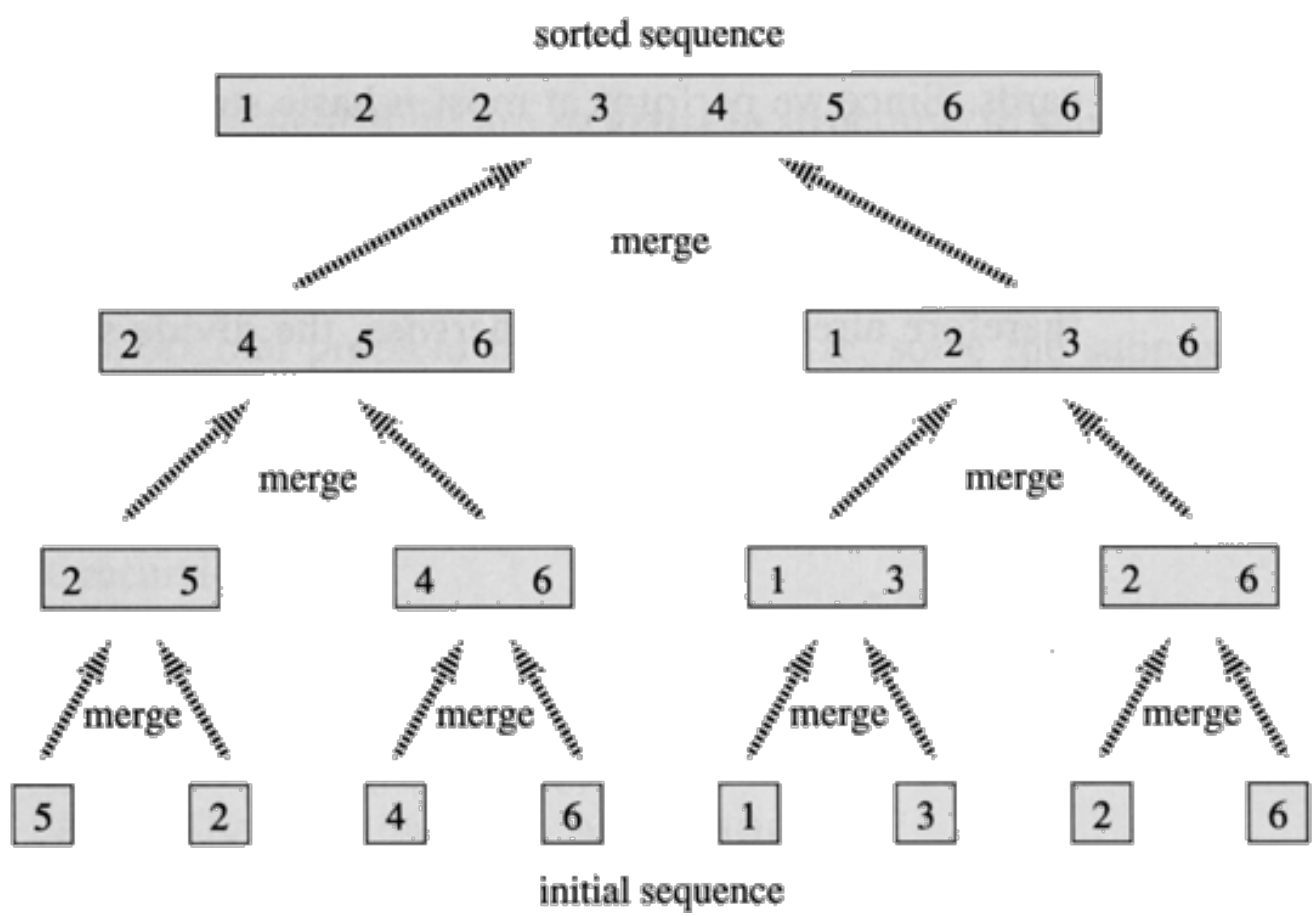
Assume two different sorting algorithm

- Shell Sort (O(n^1.5))
- Bubble Sort (O(n^2))

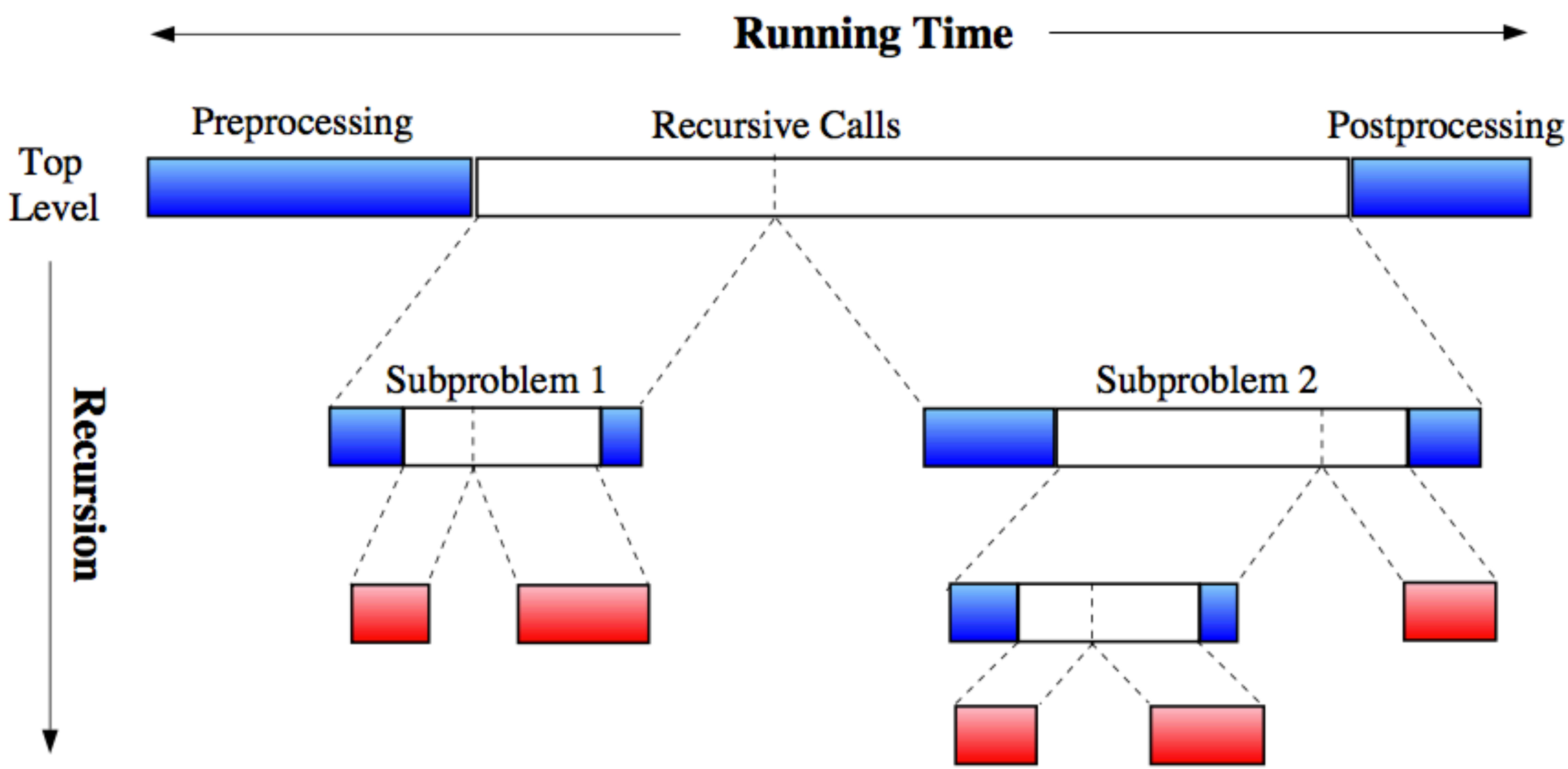
If we use only problem size, n, to decide which algorithm to run, the algorithm selection problem reduces to finding an optimal cutoff n' such that we sort lists of fewer than n' items with bubble sort and longer lists with shell sort.

Algorithm Selection

- Merge Sort (O(n log n))



Algorithm Selection as an MDP



Algorithm Selection as an MDP

$$T(n) = 2T(n/2) + \Theta(n), \quad T(1) = \Theta(1)$$

$$V(s_n) = 2V(s_{n/2}) + R(s_n, a_m), \quad V(s_1) = 0$$

$$T(n) = E\left[\sum_{j=1}^{k} T(n_j) + t(n)\right] \qquad V(s_n) = E\left[\sum_{j=1}^{k_a} V(s_{n_j}) + R(s_n, a)\right]$$

$$Q(s_n, a) = E\left[\sum_{j=1}^{k_a} \min_{a'}\{Q(s_{n_j}, a')\} + R(s_n, a)\right]$$

Learning Mechanism

General

$$Q^{(t+1)}(s_t, a_t) = (1-\alpha)Q^{(t)}(s_t, a_t) + \alpha\left[R_{t+1} + \min_a \left\{Q^{(t)}(s_{t+1}, a)\right\}\right]$$

Non-recursive

$$Q^{(t+1)}(s_t, a_t) = (1-\alpha)Q^{(t)}(s_t, a_t) + \alpha R(s_t, a_t)$$

Recursive

$$Q^{(t+1)}(s_t, a_t) = (1-\alpha)Q^{(t)}(s_t, a_t) + \alpha\left[R(s_t, a_t) + \min_a \left\{Q^{(t)}(s_1, a)\right\} + \min_a \left\{Q^{(t)}(s_2, a)\right\}\right]$$

Learning Mechanism

Monte-Carlo Return $\mathfrak{R}_\pi(s) = \sum_t R(s_t, a_t)$

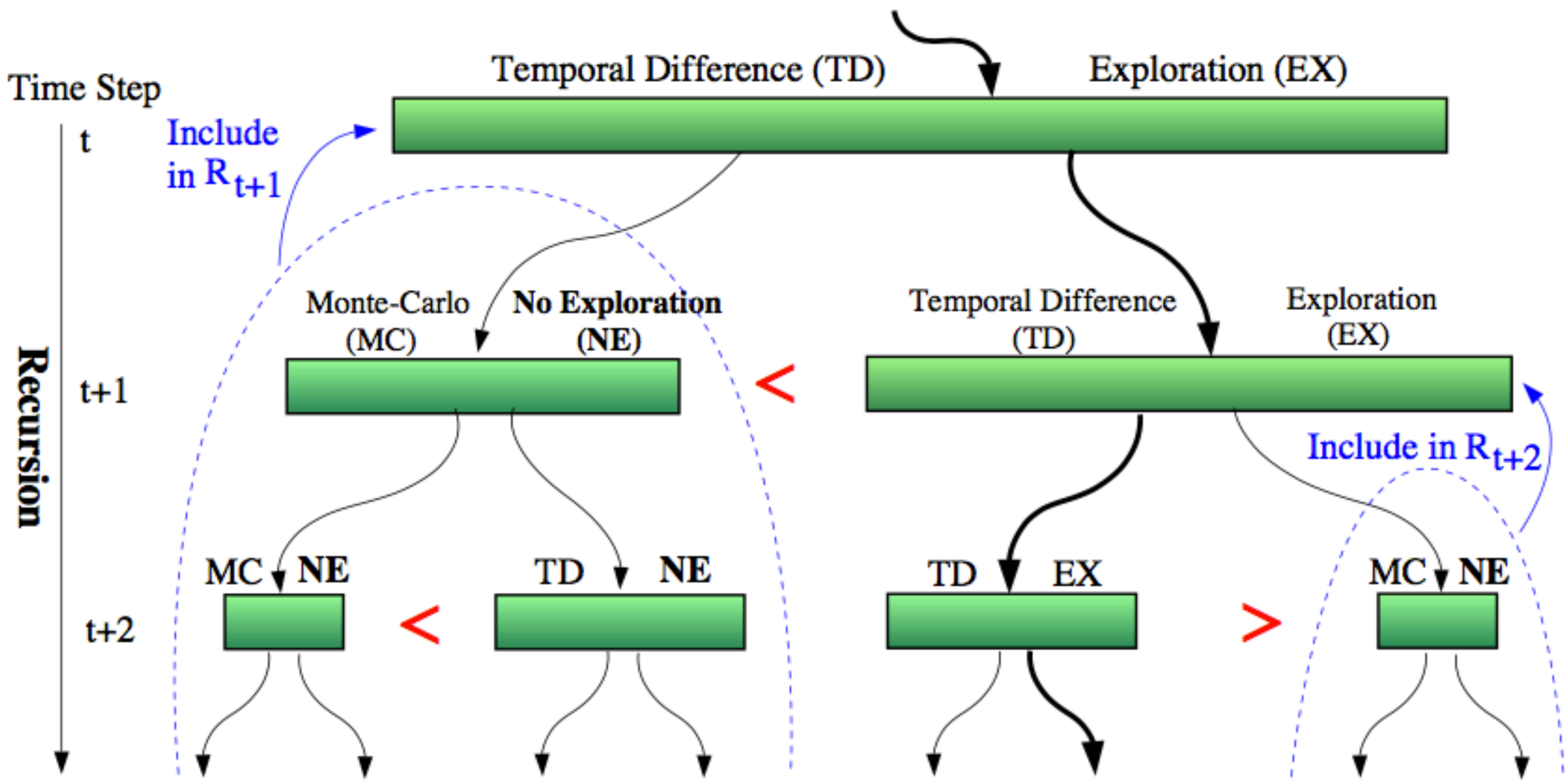
Pure Monte-Carlo

$$Q^{(t+1)}(s_t, a_t) = (1-\alpha)Q^{(t)}(s_t, a_t) + \alpha\left[R(s_t, a_t) + \mathfrak{R}_\pi(s_1) + \mathfrak{R}_\pi(s_2)\right]$$

Final Form

$$Q^{(t+1)}(s_t, a_t) = (1-\alpha)Q^{(t)}(s_t, a_t) + \alpha\left[\underbrace{R(s_t, a_t) + \mathfrak{R}_\pi(s_1)}_{R_{t+1}} + \min_a\left\{Q^{(t)}(s_2, a)\right\}\right]$$

Learning Mechanism



Results

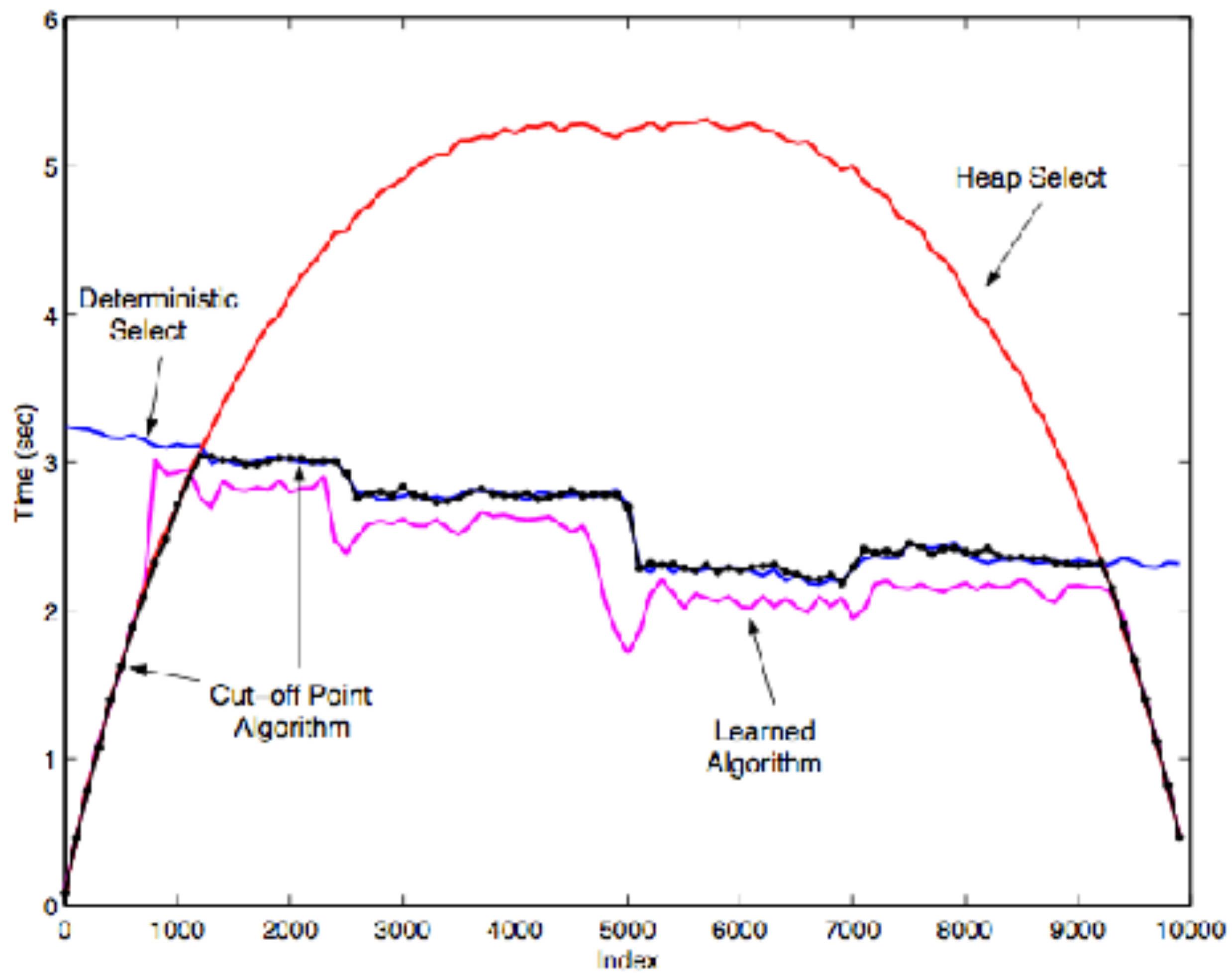


Figure 3. Results for order statistic selection (tabular case).

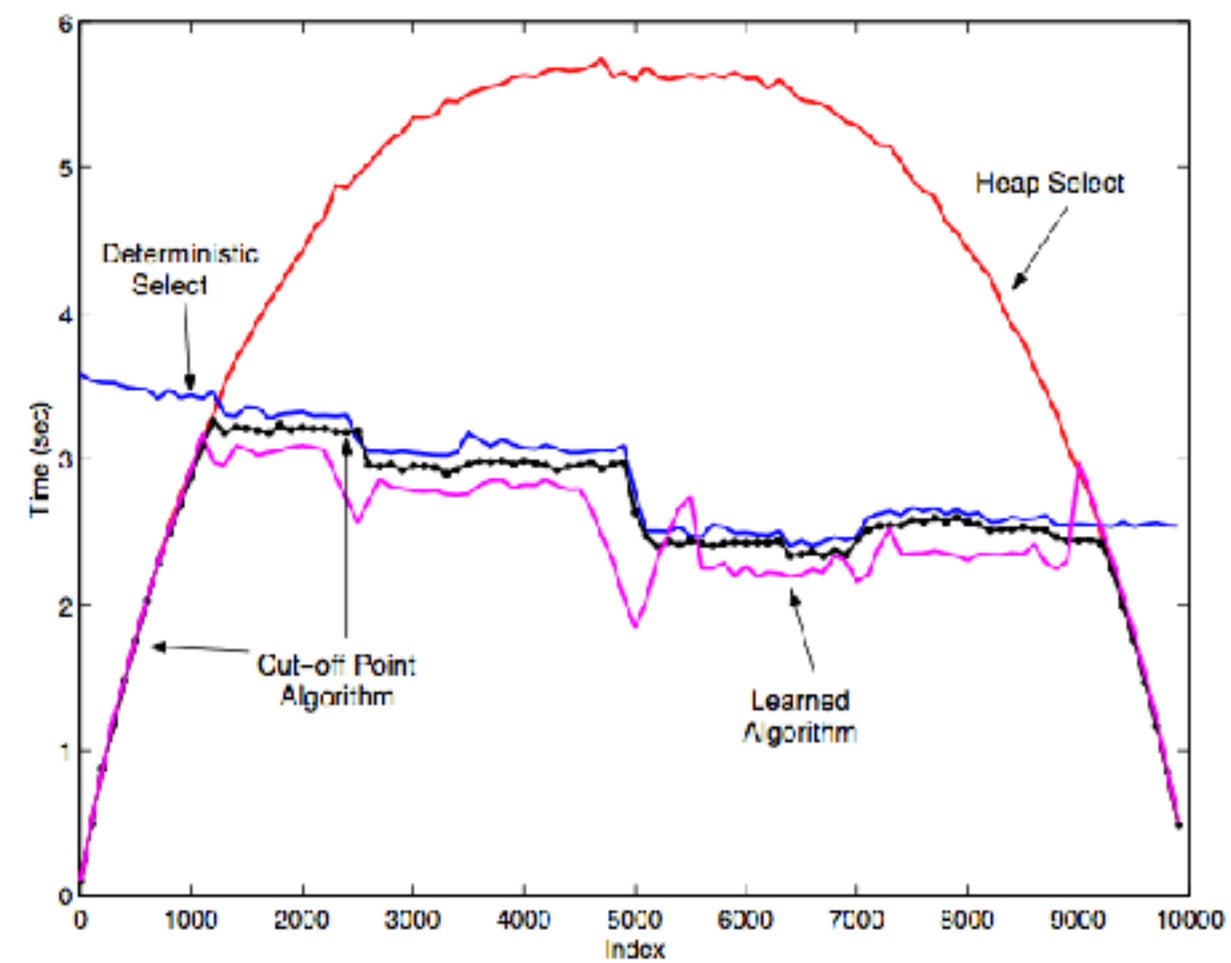


Figure 4. Order statistic selection (linear architecture).