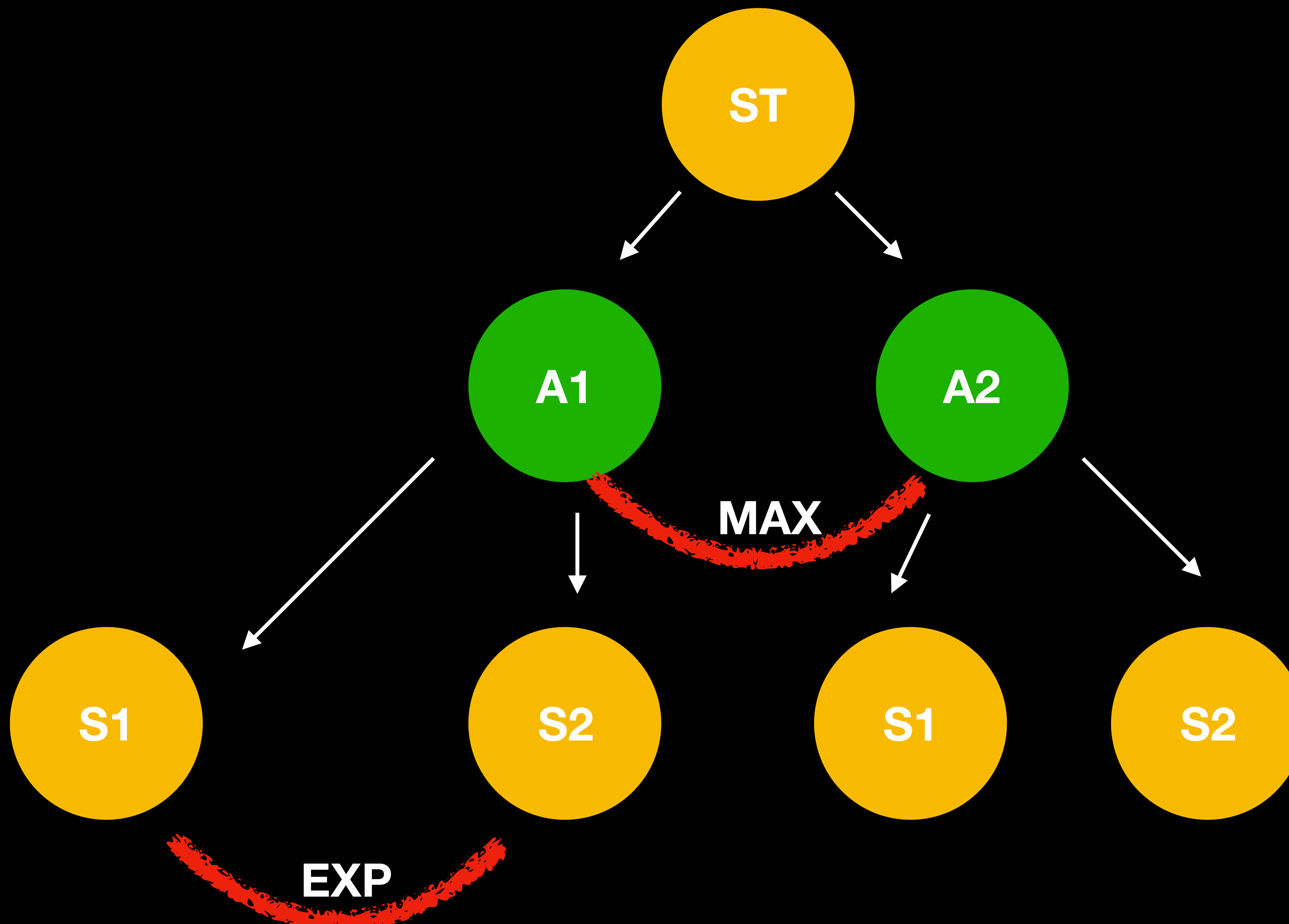
Memory-Augmented Monte Carlo Tree Search (M-MCTS)

Fan, Jett, and Audrina

Xiao, C., Mei, J., & Müller, M. (2018, April). Memory-augmented monte carlo tree search. In Thirty-Second AAAI Conference on Artificial Intelligence.

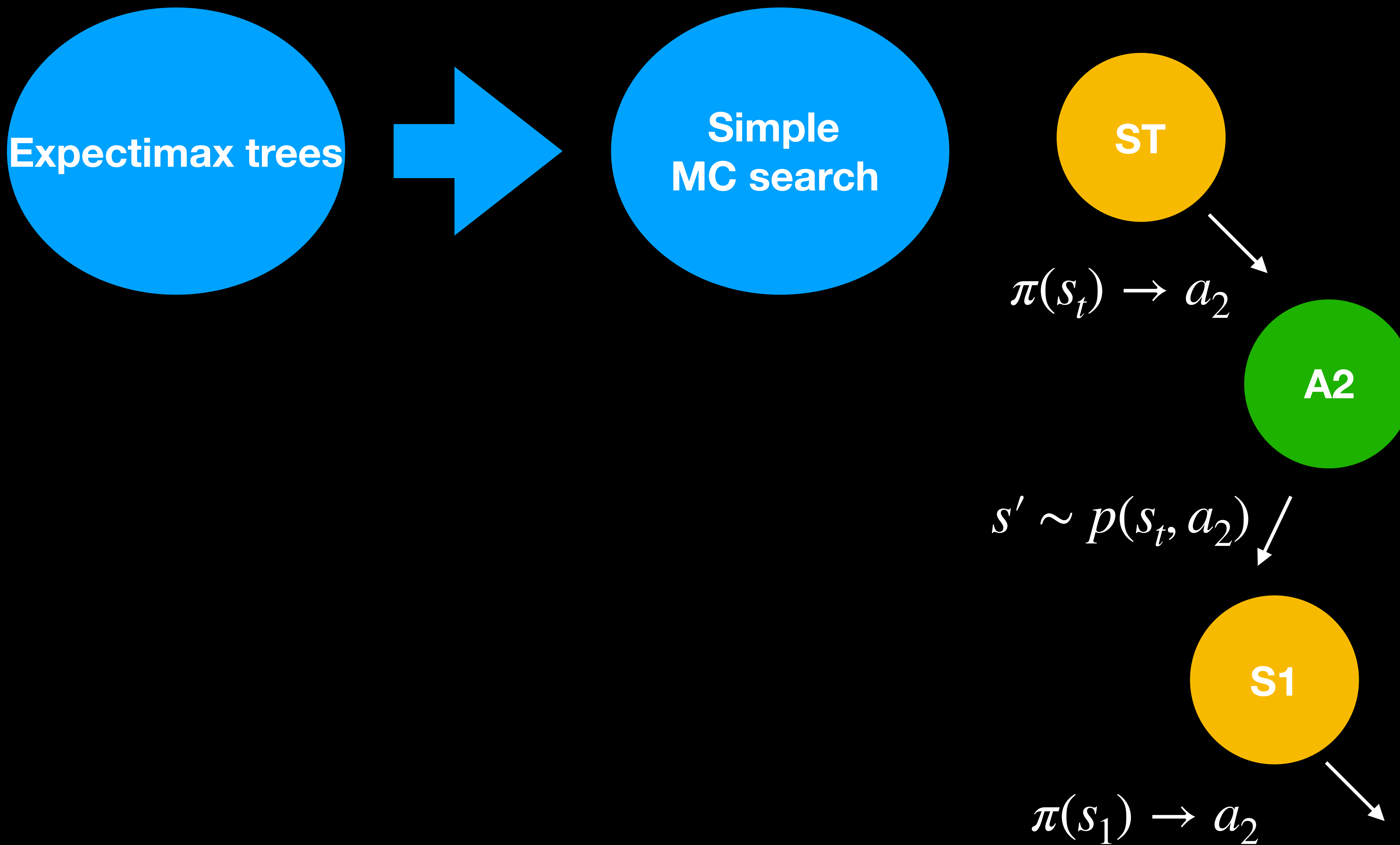
Intro and background

Expectimax trees

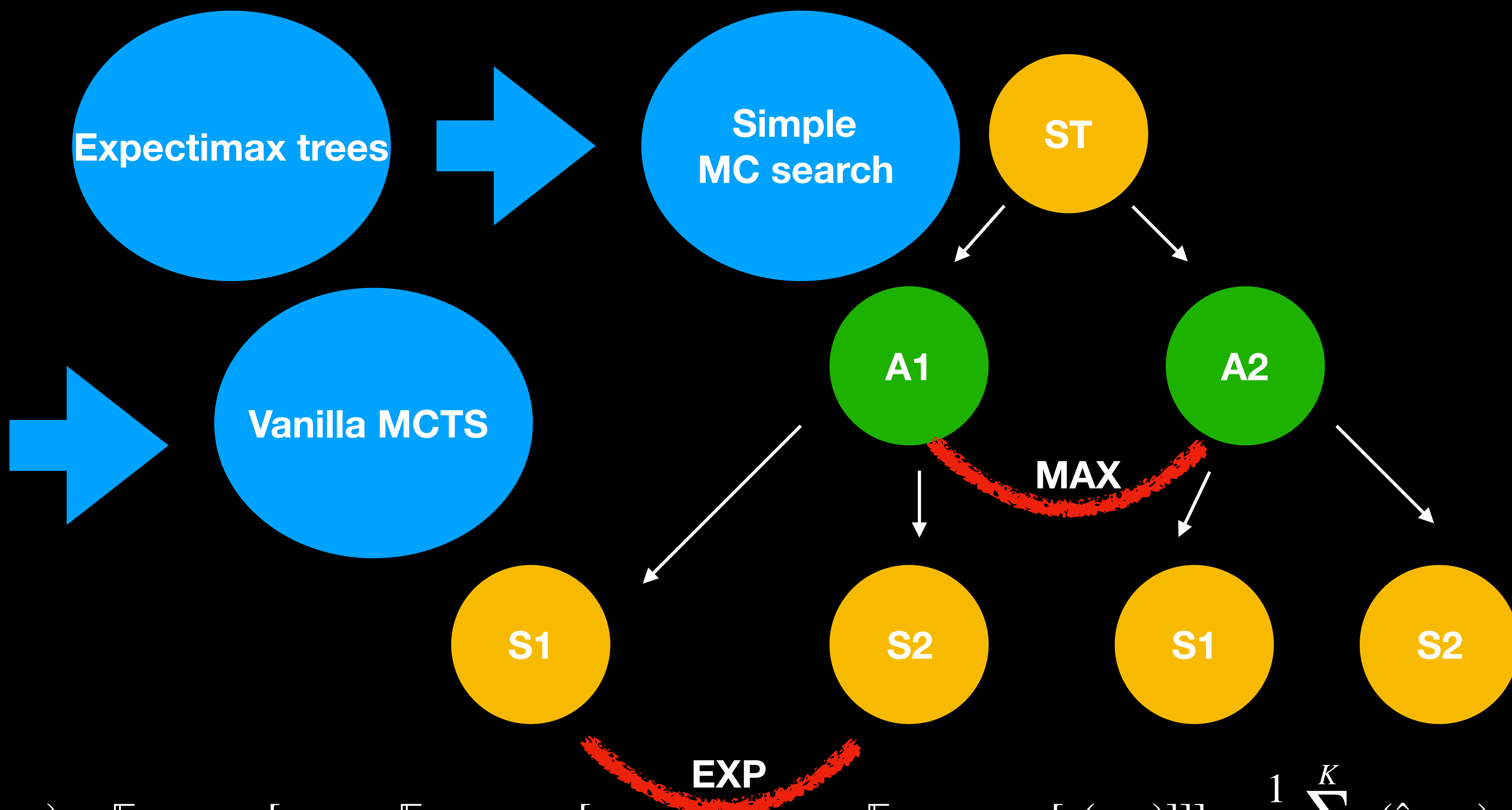


$$P(s_1 \mid s_t, a_1) * V(s_1) + P(s_2 \mid s_t, a_2) * V(s_2)$$

Intro and background

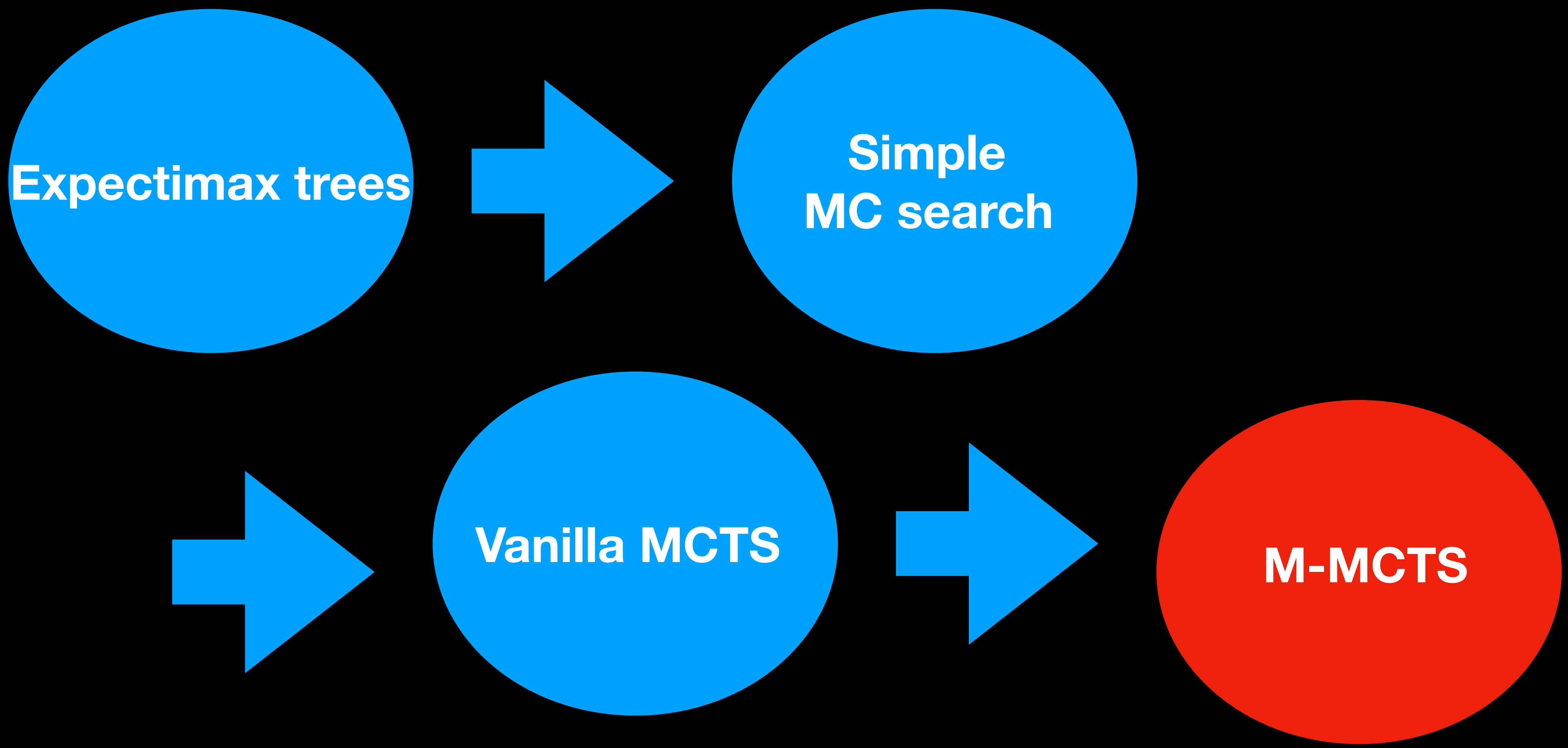


Intro and background

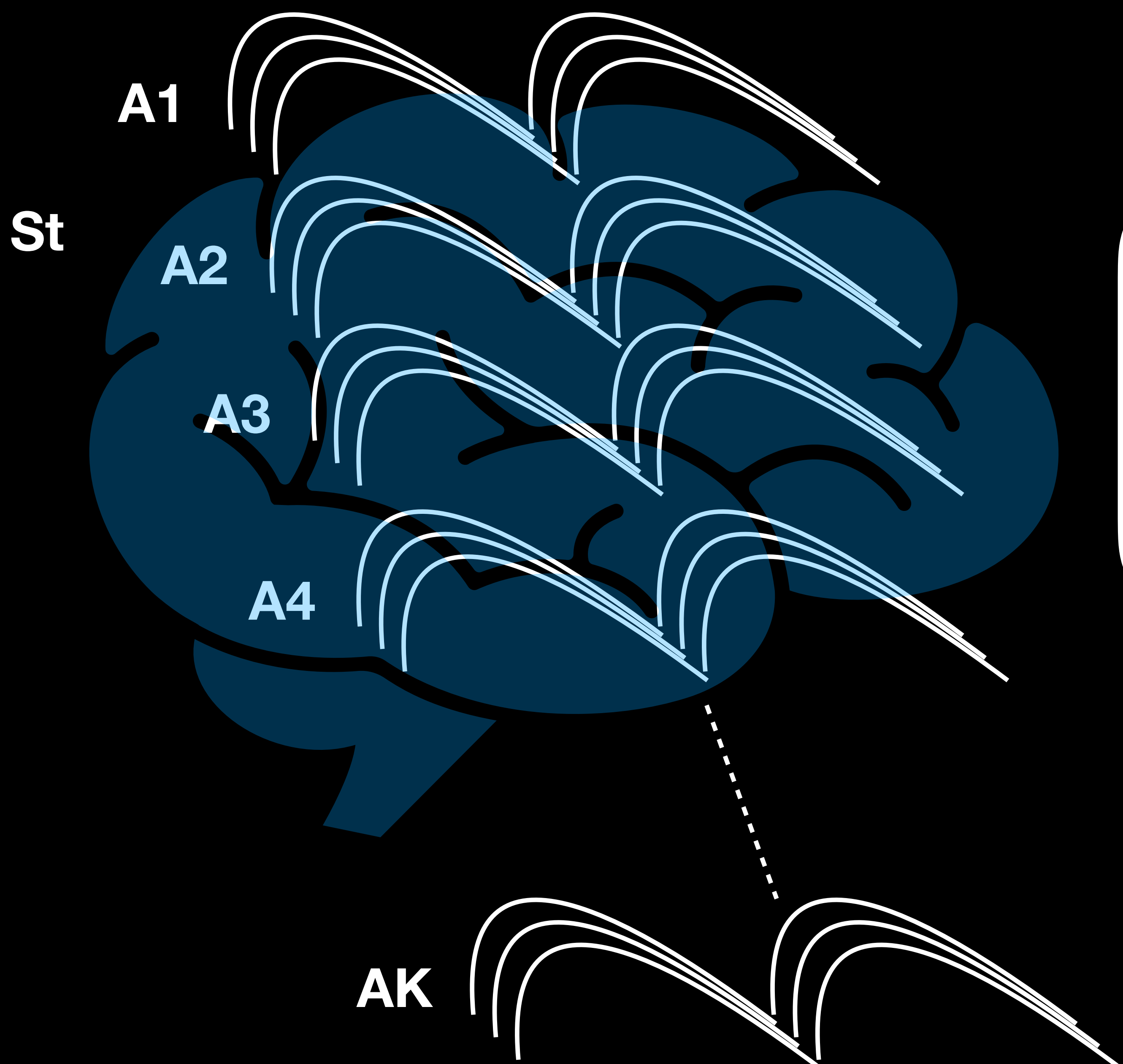


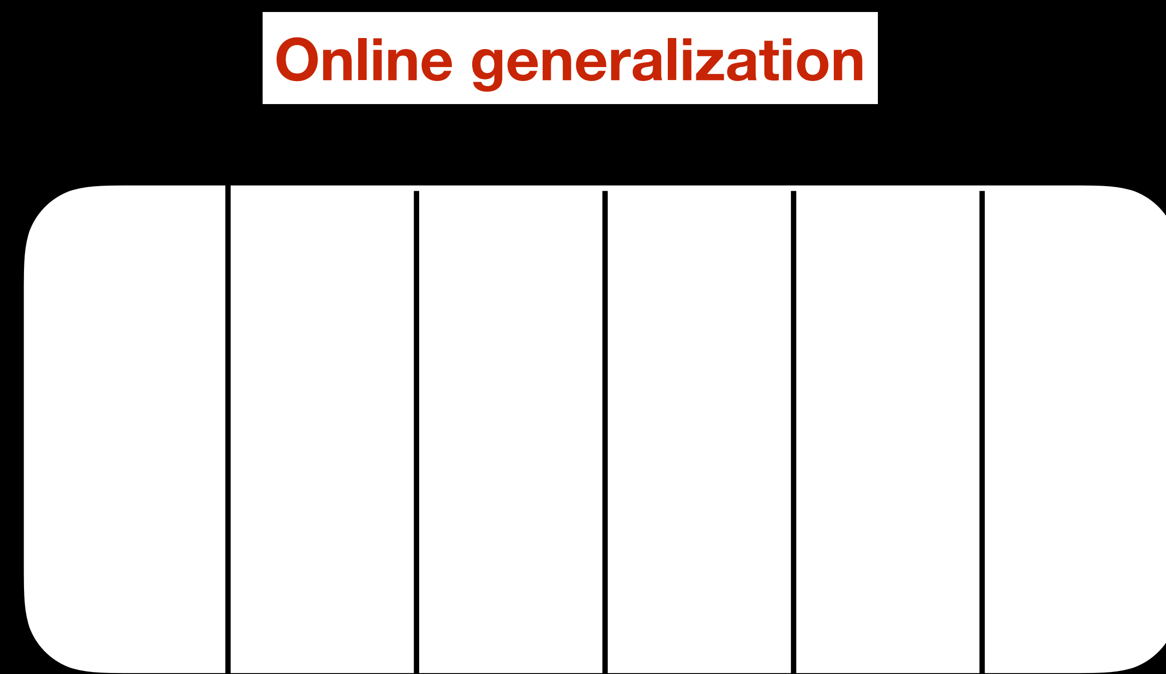
$$q(a_t, s_t) = \mathbb{E}_{p(s_{t+1}|a_t,s_t)}[max_{a_{t+1}}\mathbb{E}_{p(s_{t+1}|s_t,a_{t+1})}[max_{a_{t+2}}\ldots max_{a_T}\mathbb{E}_{p(s_T|a_T,s_T-1)}[r(s_{1:T})]]] \approx \frac{1}{K}\sum_{i=1}^{K} r(\hat{s}_{1:T,K})$$

Intro and background



Main idea + contribution





Proving M-MCTS is better

Proof. We show that under condition (6), it can be guaranteed that $\Pr\left(-F_\tau(-\mathbf{c}) + \tau \log M \le \delta_x\right) \ge 1 - \beta$.

$$\Pr\left(-\tau \log\left(\sum_{i=1}^{M} \exp(-c_i/\tau)\right) \le \delta_x - \tau \log M\right)$$
$$= \Pr\left(\sum_{i=1}^{M} \exp(-c_i/\tau) \ge \exp(-(\delta_x - \tau \log M)/\tau)\right)$$
$$\ge \Pr\left(\sum_{i=1}^{M} \exp(-\delta_i/\tau) \ge \exp(-(\delta_x - \varepsilon - \tau \log M)/\tau)\right)$$
$$\ge \Pr(\exists\, i, \exp(\delta_i/\tau) \le \exp((\delta_x - \varepsilon - \tau \log M)/\tau)$$
$$= 1 - \prod_{i=1}^{M} \Pr\left(\delta_i \ge \alpha - \tau \log M\right)$$
$$\ge 1 - \prod_{i=1}^{M} \exp\left(-\frac{(\alpha_x - \tau \log M)^2 N_i}{2\sigma^2}\right)$$
$$= 1 - \exp\left(-\frac{(\alpha_x - \tau \log M)^2 n}{2\sigma^2}\right)$$

ERROR

$$\vec{c} = [c_1, \ldots, c_M]^T, c_i = \delta_i + \epsilon_{i,x}$$

Value estimation E **True value diff**

OPT OBJ

$$max_{w \in \triangle}\{w \cdot (-c)\}$$

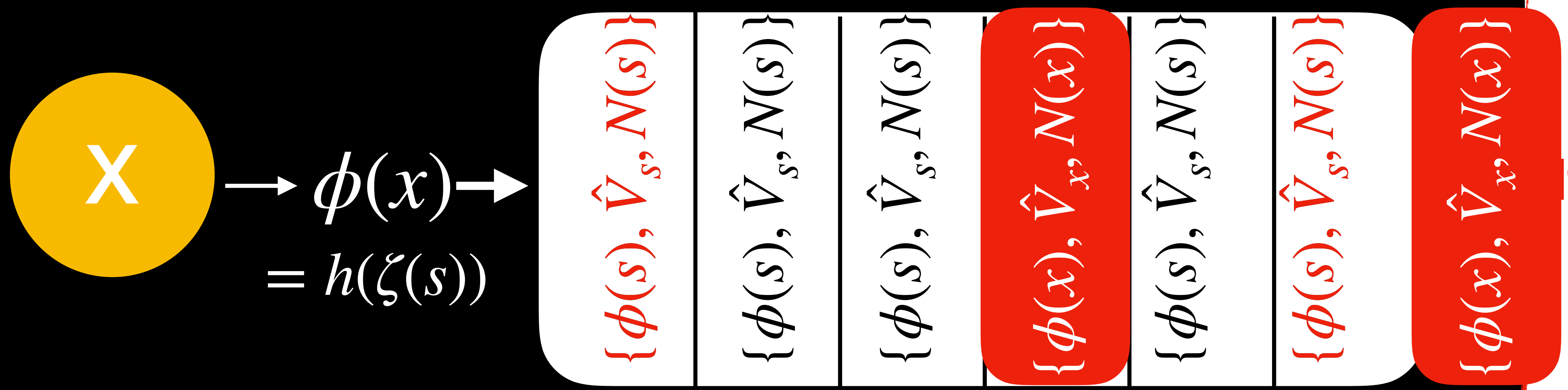
OPT OBJ (entropy regularized)

Eqn (1) $$max_{w \in \triangle}\{w \cdot (-c) + \tau * H(w)\}$$

MAIN POINT

$$Pr(eqn(1) \le \delta_x) \ge 1 - \beta$$

Memory



$Update(x)$

If s is stored in the memory, only update $\hat{V}_x, N(x)$.

$Query(x)$

Find top M similar states in M, based on the distance function

$d(\cdot, x) = -cos(\phi(\cdot), \phi(x))$

and compute the approximated memory value.

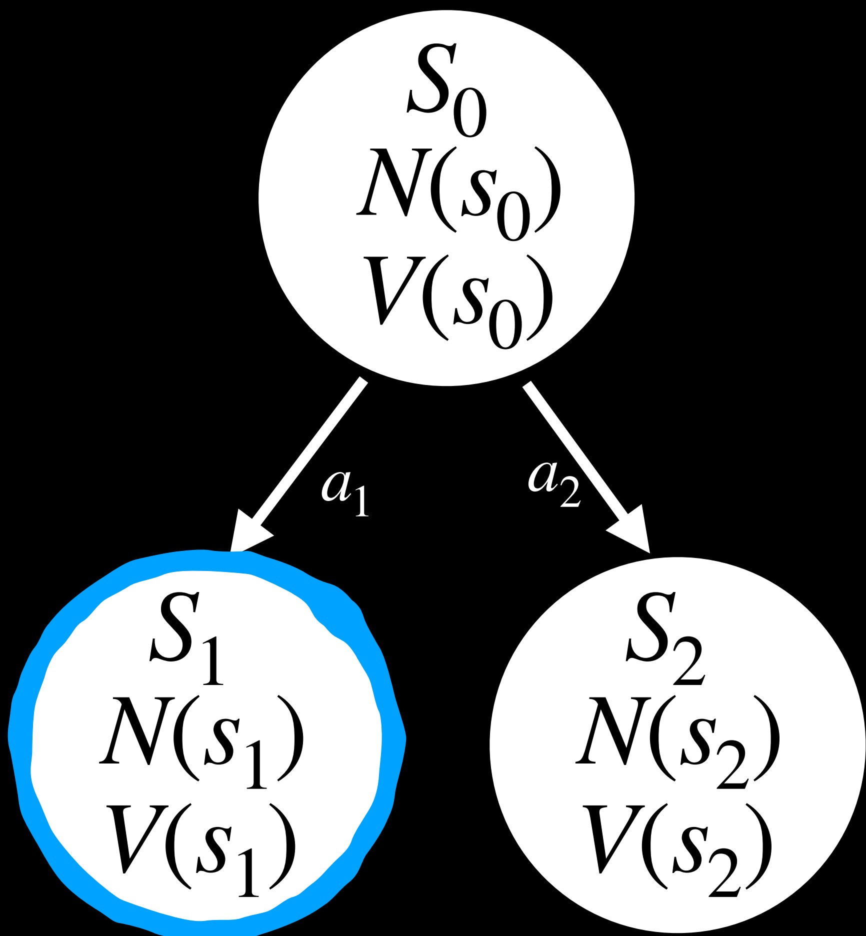
$Add(x)$

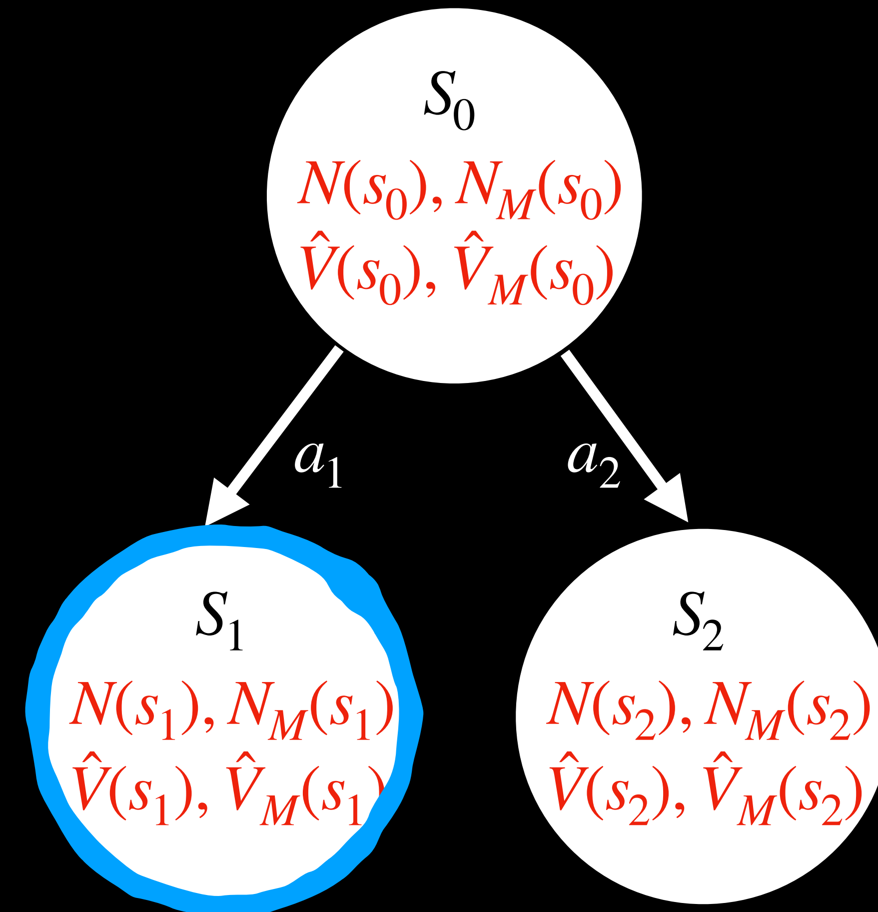
Add new state s by adding new memory entry

$\{\phi(x), \hat{V}_x, N(x)\}$

If the memory is full, then replace the least recently queried or updated memory entry with the new one.

Architecture - Selection & Expansion

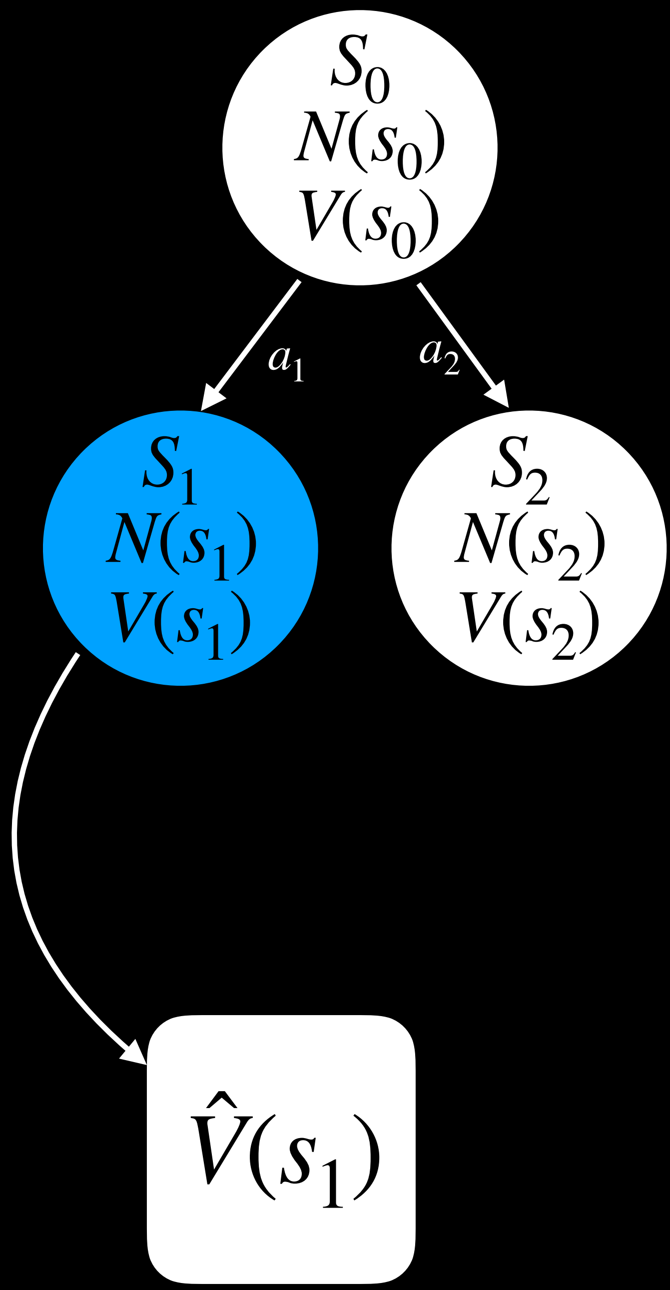




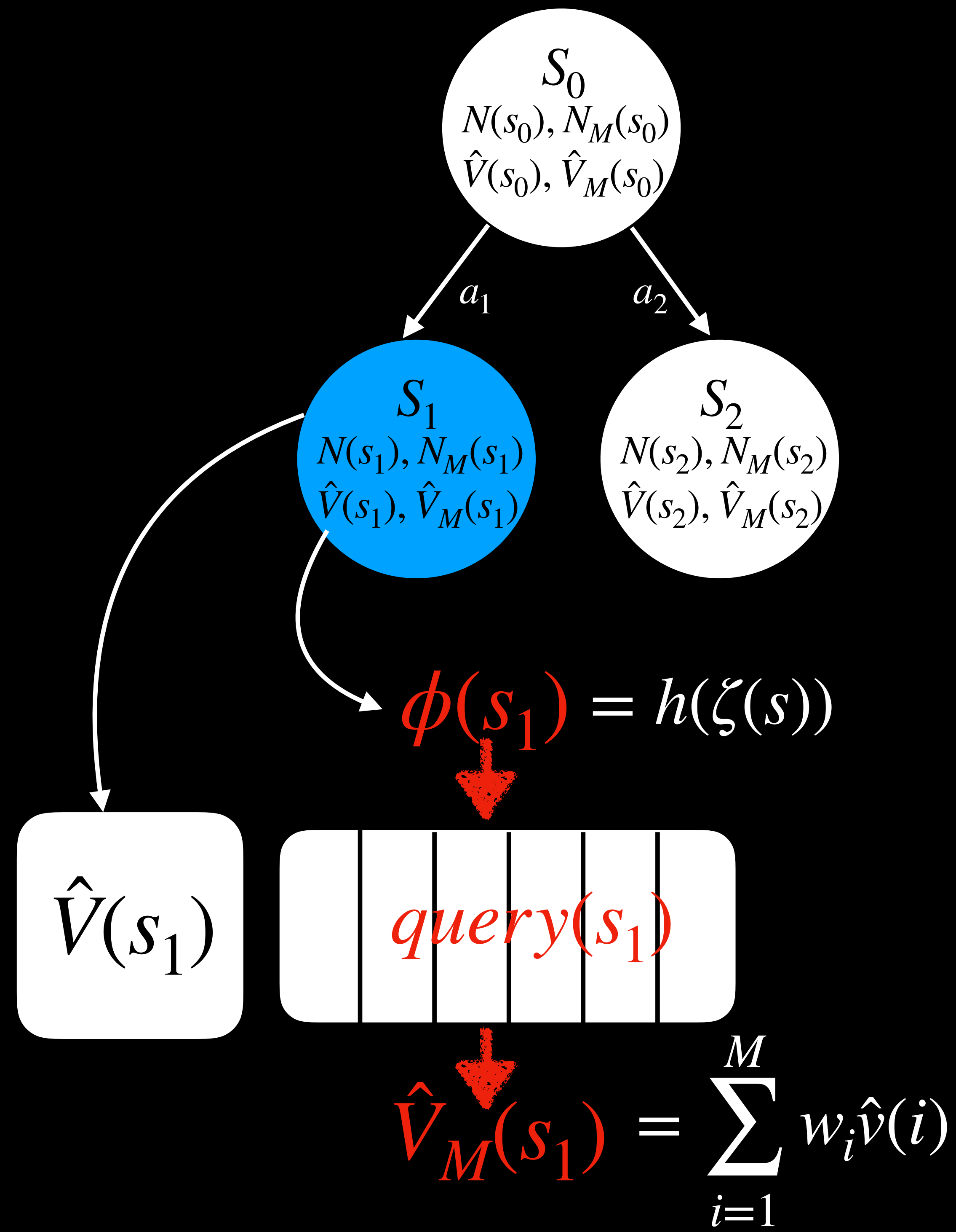
$$UCB = V(s_i) + c\sqrt{(ln(N)/N(s_i)}$$

$$UCB_M = (1 - \lambda_{s_i})\hat{V}(s_i) + \lambda_{s_i}\hat{V}_M(s_i)$$

Architecture - Simulation

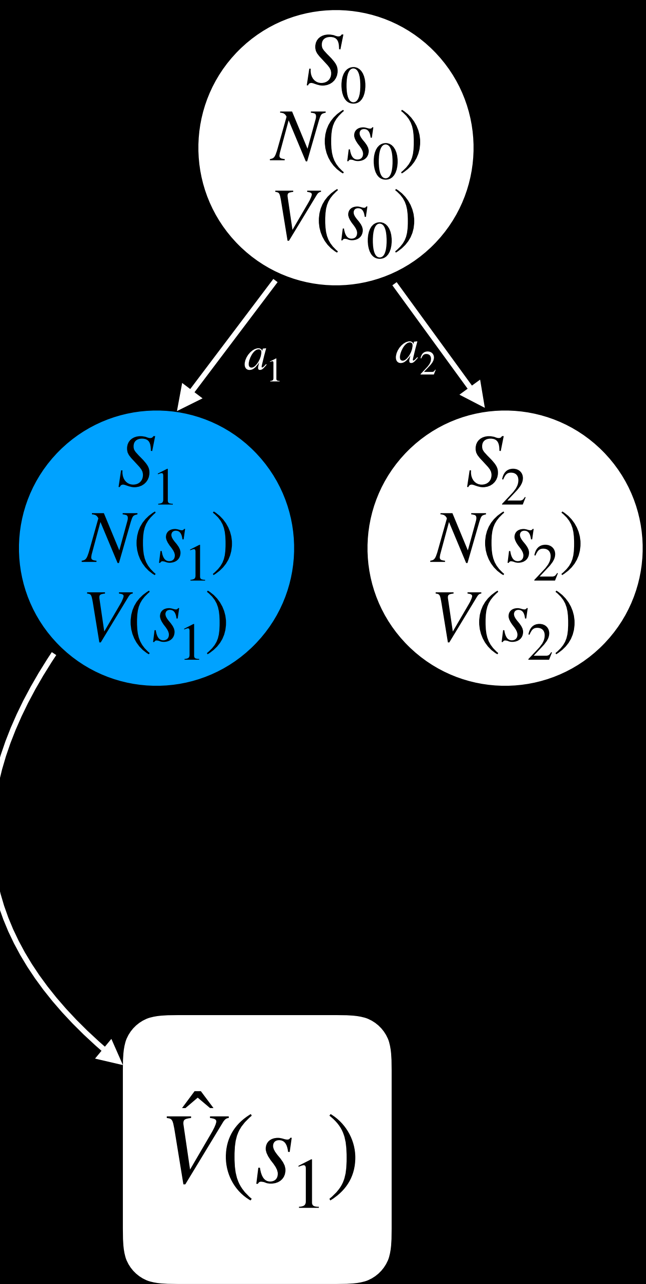


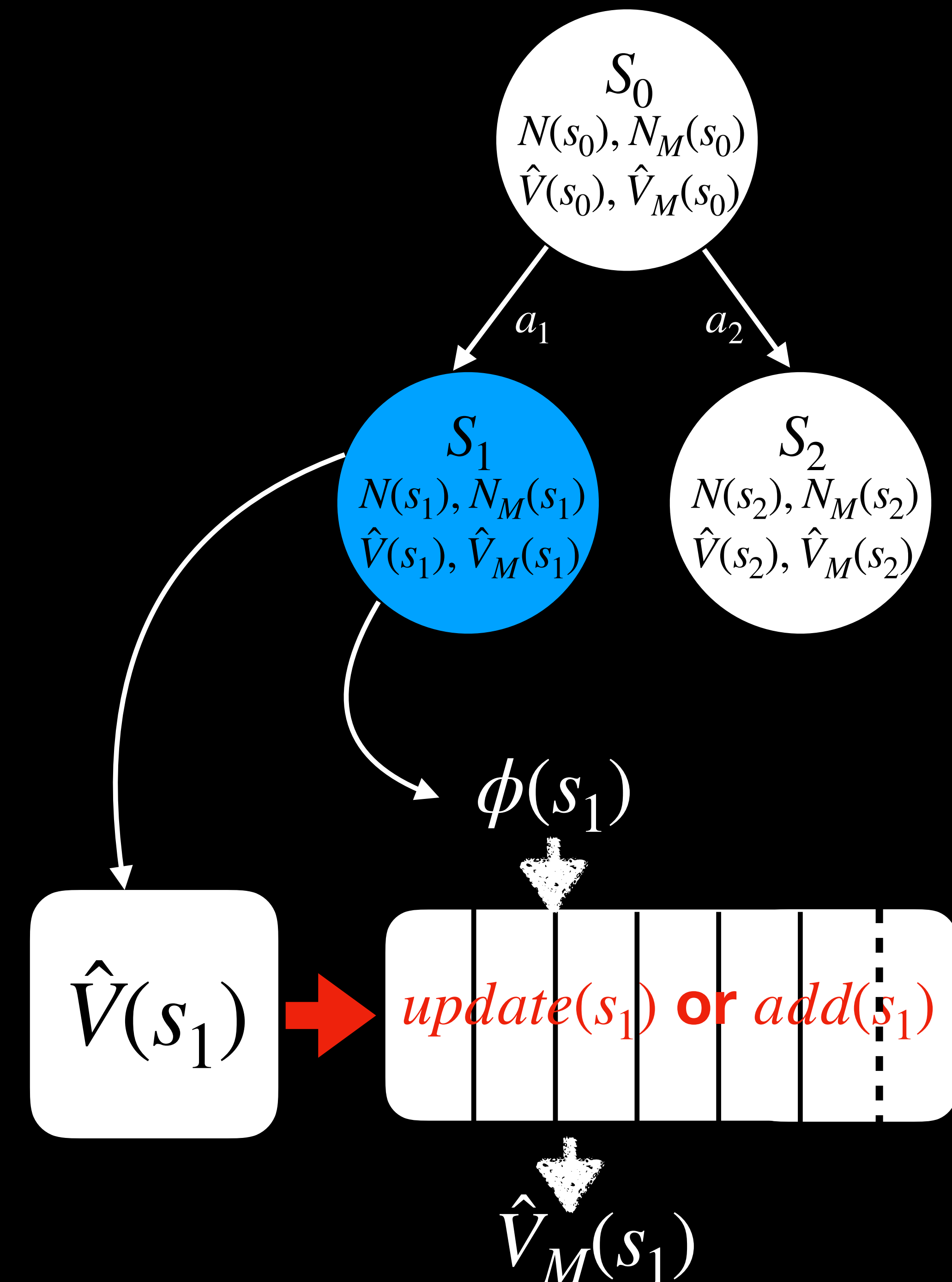
All the trajectory of visited states are obtained in each stimulation.



Compute the memory value by $query(s_1)$ operation of the memory M.

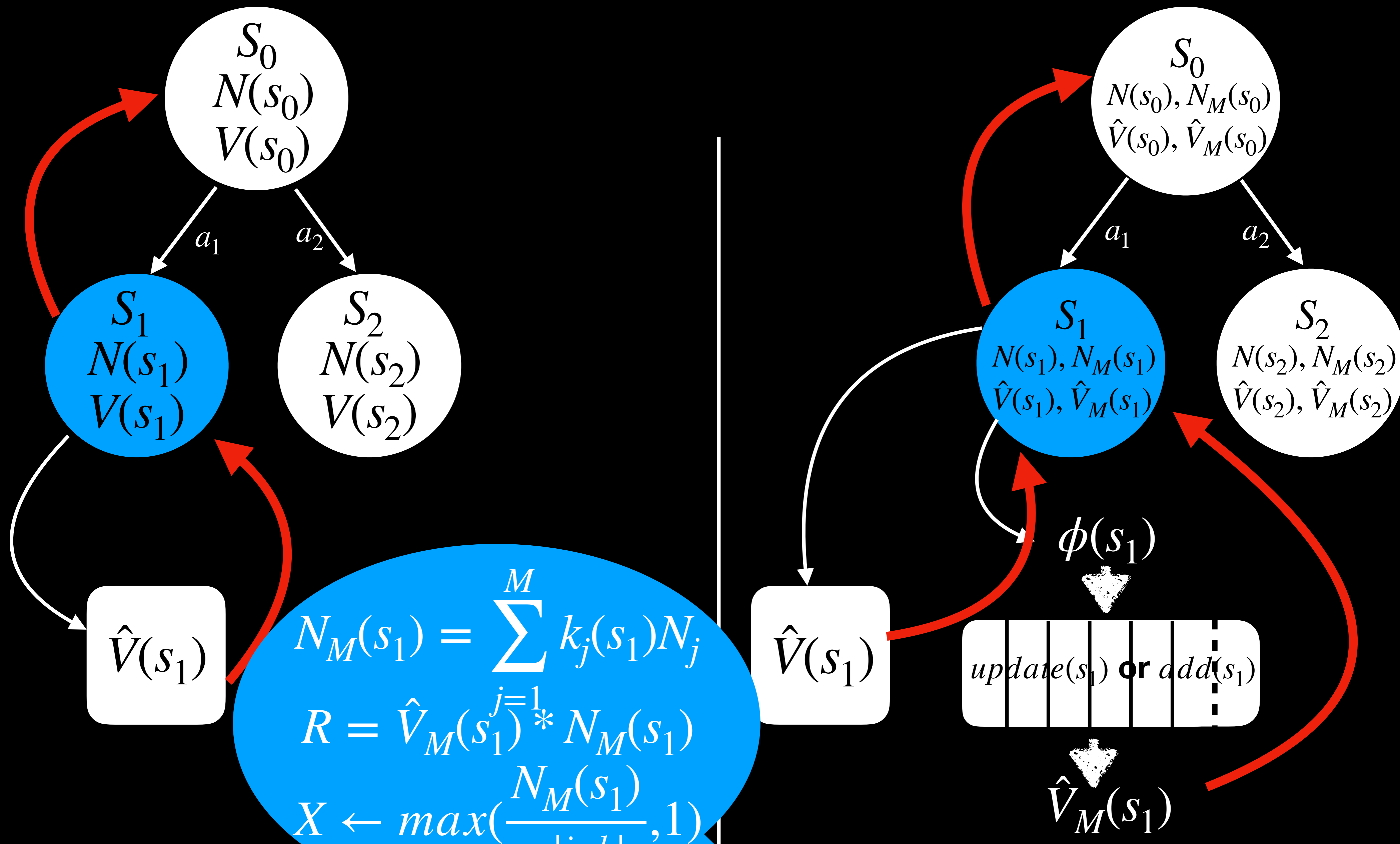
Architecture - Update





Update the in-memory statistics by performing the $update(s_1)$ or $add(s_1)$ operation in the memory M.

Architecture - Back-propagation



$$N_M(s_1) = \sum_{j=1}^{M} k_j(s_1) N_j$$

$$R = \hat{V}_M(s_1) * N_M(s_1)$$

$$X \leftarrow max(\frac{N_M(s_1)}{\eta^{|i-h|}}, 1)$$

$$N(s) \leftarrow N(s) + 1$$

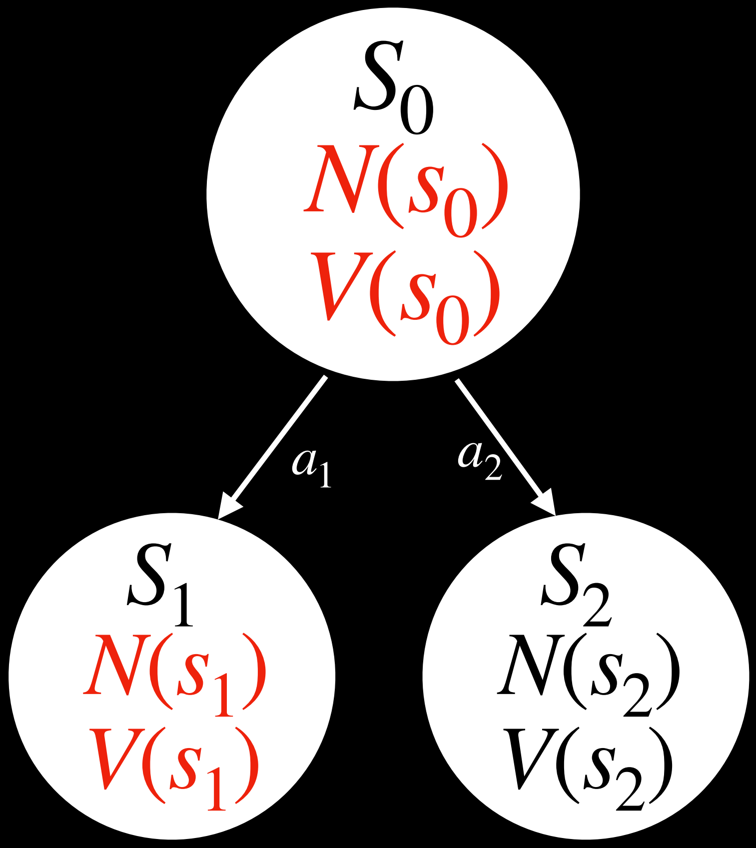
$$\hat{V}(s) \leftarrow \hat{V}(s) + \frac{R - \hat{V}(s)}{N(s)}$$

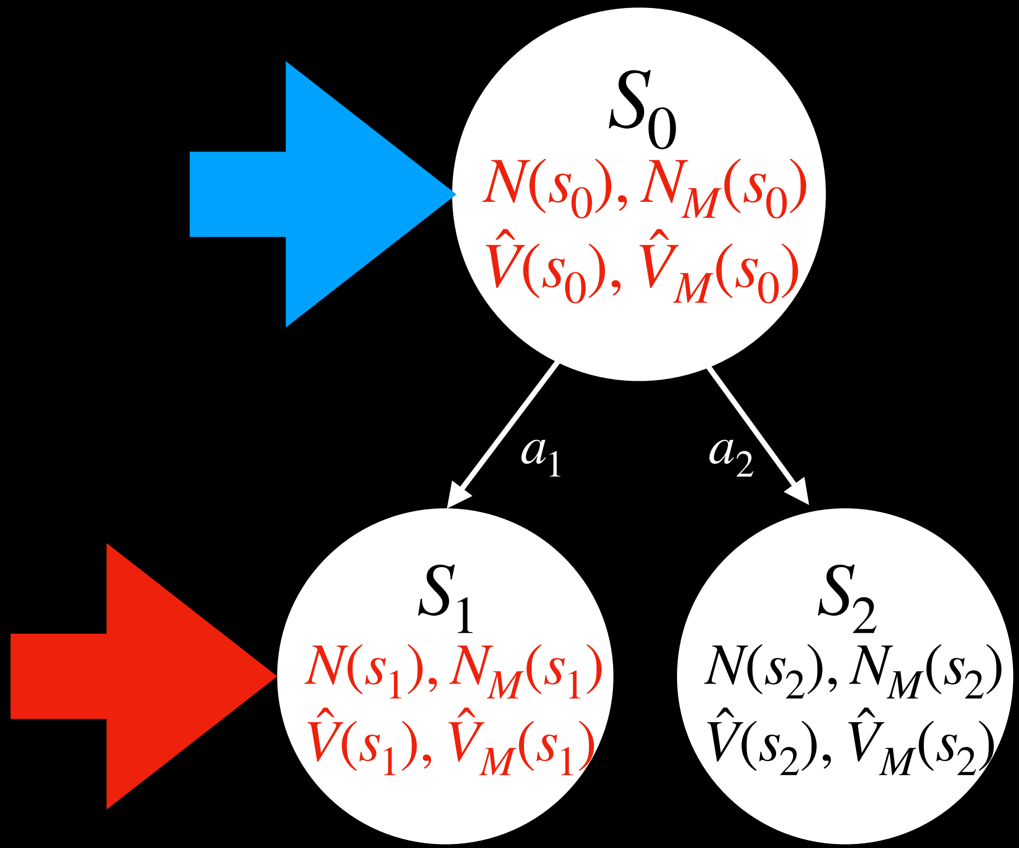
+

$$N_M(s) \leftarrow N_M(s) + X$$

$$\hat{V}_M(s) \leftarrow \hat{V}_M(s) + \frac{R - \hat{V}_M(s) * X}{N_M(s)}$$

Architecture





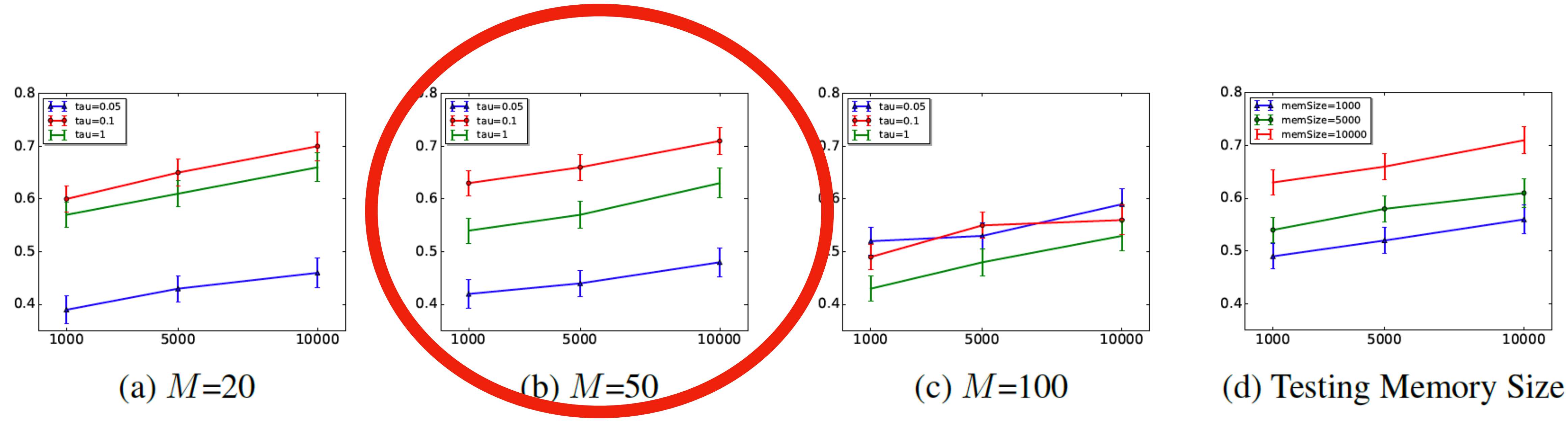
- **To not be time-consuming, Only compute the memory value at the leaf node**

M-MCTS Algo

```
# The Overall structure of M-MCTS and MCTS is similar except adding a memory data structure
Class node:
    V, V_m, N, N_m

Class Memory:
    {φ(x), V, N} * size of Memory
    Query()
    Update()
    Add()

def selection(node):
    while node is fully expanded:
        node = ucb_M(node)
    return pick unvisited node.children or node

def rollout(node):
    feature = h(ζ(node))
    M = closest states in memory by d( · , node)
    V_m = sum_{i=1}^{M} w_i v̂(i)

    while node does not return a result:
        node = find actions that max the value
    V = result(node)
    Update()

    return V and V_m

def backpropagate(node, result):
    Update N and N_m
    Update V and V_m
    node.stats =(N, N_m, V, V_m)
    if is_root(node) return
    backpropagate(node.parent)
```

$$V\_m = \sum_{i=1}^{M} w_i \hat{v}(i)$$

Result



(a) M=20 (b) M=50 (c) M=100 (d) Testing Memory Size

Figure 2: Experimental results. Figure (a)-(c) shows the results of testing different value of M. Figure (d) shows the results of testing different size of memory. In all figures, x-axis is the number of simulations per move, y-axis means the winning rate against the baseline.

Xiao, C., Mei, J., & Müller, M. (2018, April). Memory-augmented monte carlo tree search. In Thirty-Second AAAI Conference on Artificial Intelligence.

Related Work

- Utilizing information
  - Kawano, Y. 1996. Using similar positions to search game trees
    - Uses the priority scheme to extend nodes. Similar positions have same priority (i.e., static) score.
    - MCTS is better -> rollout policy can utilize offline training results.
- Memory Architectures
  - Pritzel, A. 2017. Neural episodic control
    - A buffer of past experience containing slowly changing state representations and rapidly updated estimates of the value function (memory framework).
    - Both M-MCTS and NEC stored more information in memory.
    - M-MCTS & NEC have shown experiment results better than MCTS.

- Generalization
  - Childs, B. E. 2008. Transpositions and move groups in Monte Carlo tree search
    - Nodes in the same state share the same simulation statistics (Transposition table -> τ ≈ τ ≈ 0 in M-MCTS).
    - M-MCTS with $\tau > 0$ the memory can provide more generalization.
  - Srinivasan, S. 2015. Improving exploration in UCT using local manifolds
    - Uses kernel regression to approximate a state value function.
    - Equivalent to M-MCTS's addressing scheme using $w = f_\tau(-c)$
    - M-MCTS provides theoretical justifications but their work did not.

Limitation

- Online Value Approximation
  - might not generalized feature representation (offline)
  - compute and lookup time can be expensive
- Memory augmented
  - might face scalability issues (exploration vs exploitation)
- Lack of incorporating feature representation learning with M-MCTS in an end-to-end fashion.

Thanks!