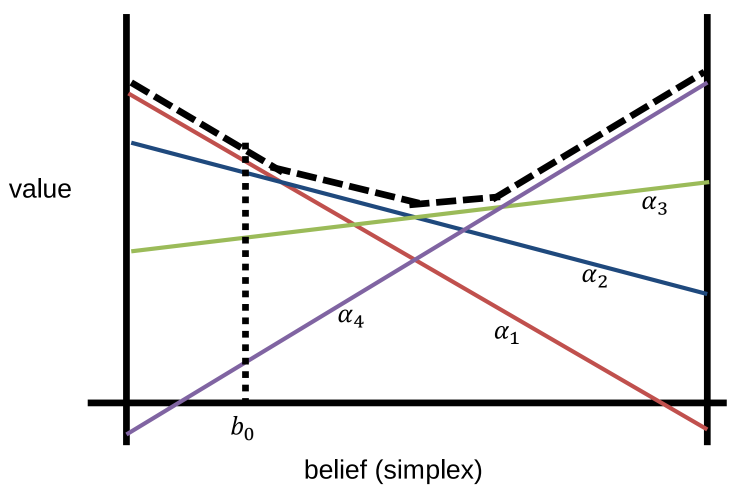
Partially Observable Markov Decision Processes - Scalable Approximate Algorithms

Branislav Bošanský

PAH/PUI 2016/2017

Point Based Value Iteration for POMDPs

- instead of the complete belief space we use a limited set
  - $B = \{b_0, \dots, b_q\}$
- the algorithm keeps only a single alpha vector for one belief point
- anytime algorithm altering 2 main steps
  - belief point value update
  - belief point set expansion



Point Based Value Iteration for POMDPs

- belief value update
  - $V_b^a = \alpha^{a,*} + \gamma \sum_{o \in O} \arg \max_{\alpha \in \alpha_i^{a,o}} (\alpha . b)$
  - $V \leftarrow \arg \max_{V_b^a, \forall a \in A} V_b^a . b \quad \forall b \in B$
- removes the exponential complexity
- we calculate |A|x|O| α-vectors
- new belief points are added that are the most distant in forward search
- $b' = \max_{a,o} |b^{a,o} - B|_L$, where $|\,|_L$ is a distance metric
  - $|b' - B|_L = \min_{b \in B} |b - b'|_L$,

Point Based Value Iteration for POMDPs

Algorithm 3 PBVI

Function PBVI

$B \leftarrow \{b_0\}$
while V has not converged to V^* **do**
    $Improve(V, B)$
    $B \leftarrow Expand(B)$

Function Improve(V,B)

repeat
    for each $b \in B$ **do**
        $\alpha \leftarrow backup(b, V)$ *//execute a backup operation on all points in B in arbitrary order*
        $V \leftarrow V \cup \{\alpha\}$
until V has converged *//repeat the above until V stops improving for all points in B*

Function Expand(B)

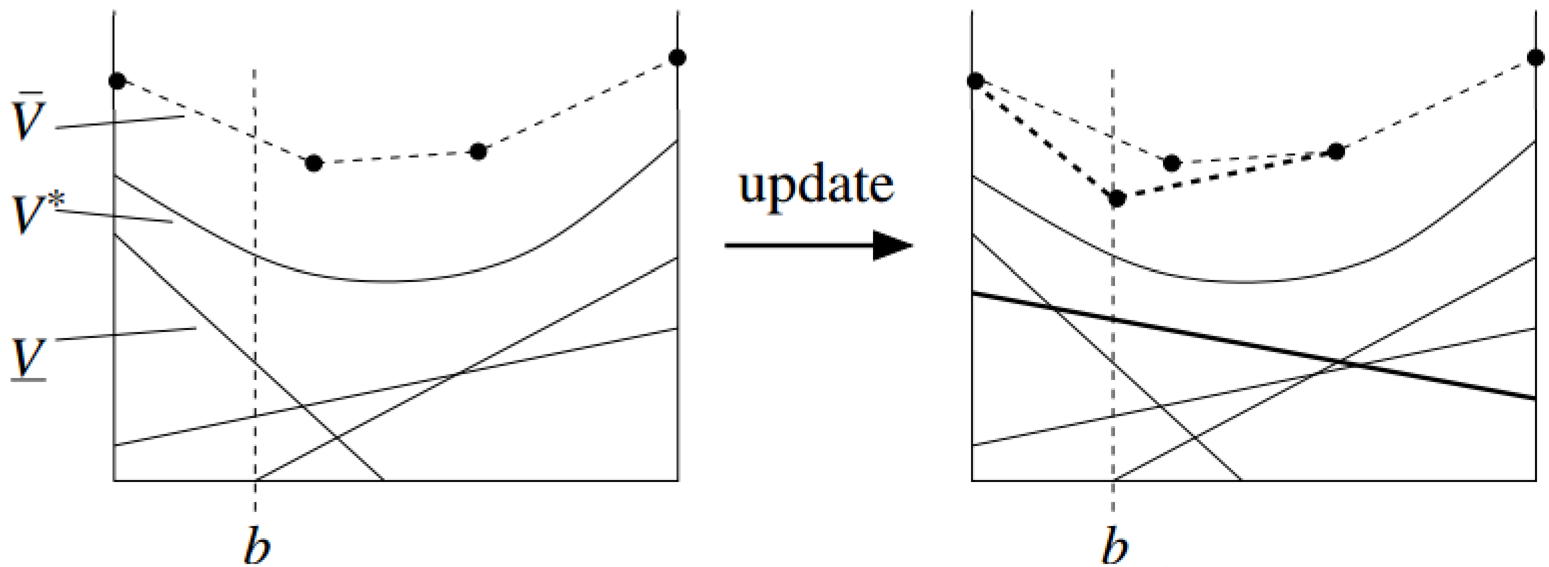
$B_{new} \leftarrow B$
for each $b \in B$ **do**
    $Successors(b) \leftarrow \{b^{a,o} | \Pr(o|b,a) > 0\}$
    $B_{new} \leftarrow B_{new} \cup \text{argmax}_{b' \in Successors(b)} ||B, b'||_L$ *//add the furthest successor of b*
return B_{new}

Heuristic Search Value Iteration (HSVI)

- Disadvantages of PBVI
  - PBVI updates each belief point from B
  - we do not know how close we are to the solution
  - we are missing an upper bound direction

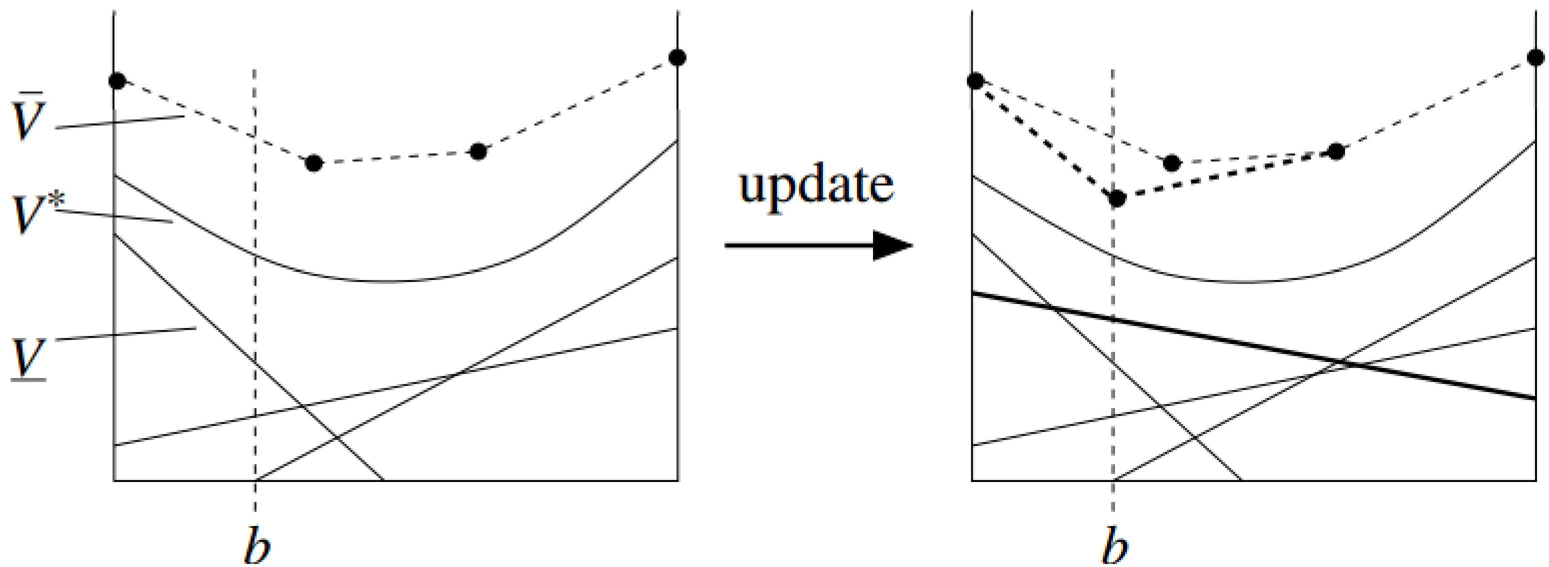
- HSVI
  - maintains two approximations – upper and lower bound
  - upper bound is a set of points
    - how do we get an UB on POMDP value? (MDP)
  - lower bound is a set of alpha vectors

Heuristic Search Value Iteration (HSVI)



- Lower bound update
  - standard point based update
  - uses a set B' that corresponds to a subset of beliefs based on a heuristic forward search
  - adds new α vectors to $\underline{V}$

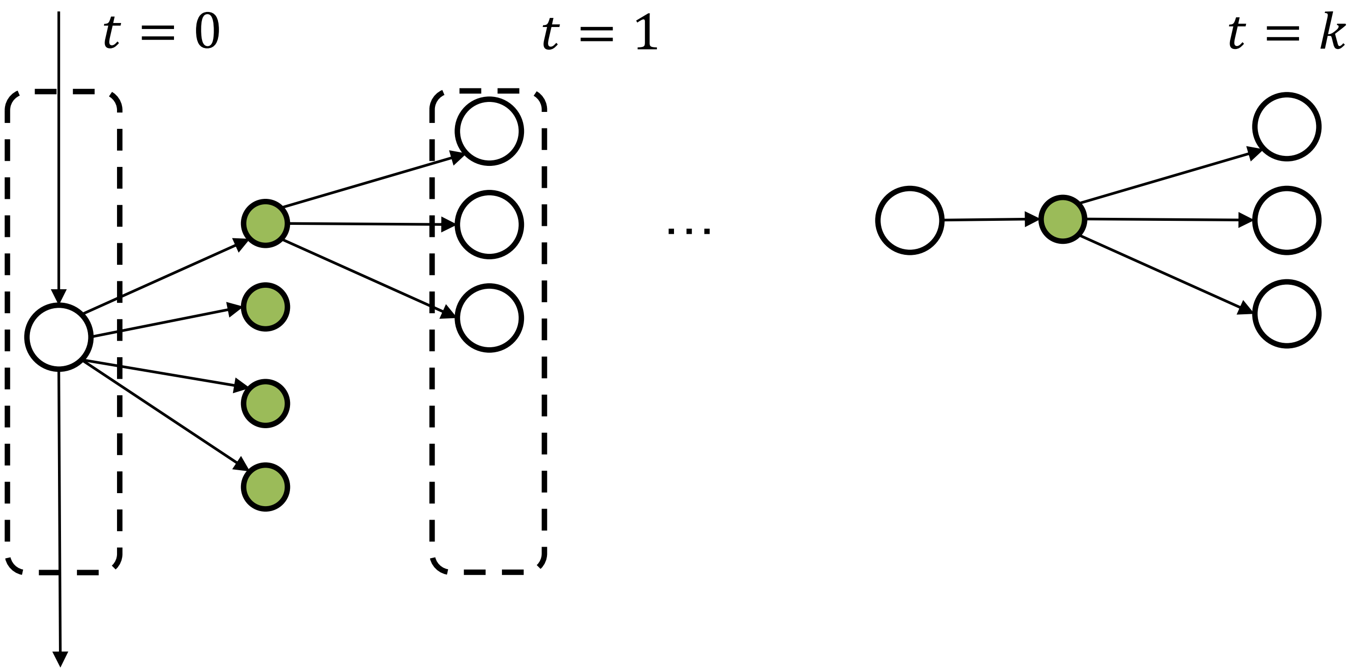
Heuristic Search Value Iteration (HSVI)



- Upper bound update
  - standard Bellman backup
  - uses $\bar{V}$ as a set of points of beliefs – values
  - adds new points to $\bar{V}$

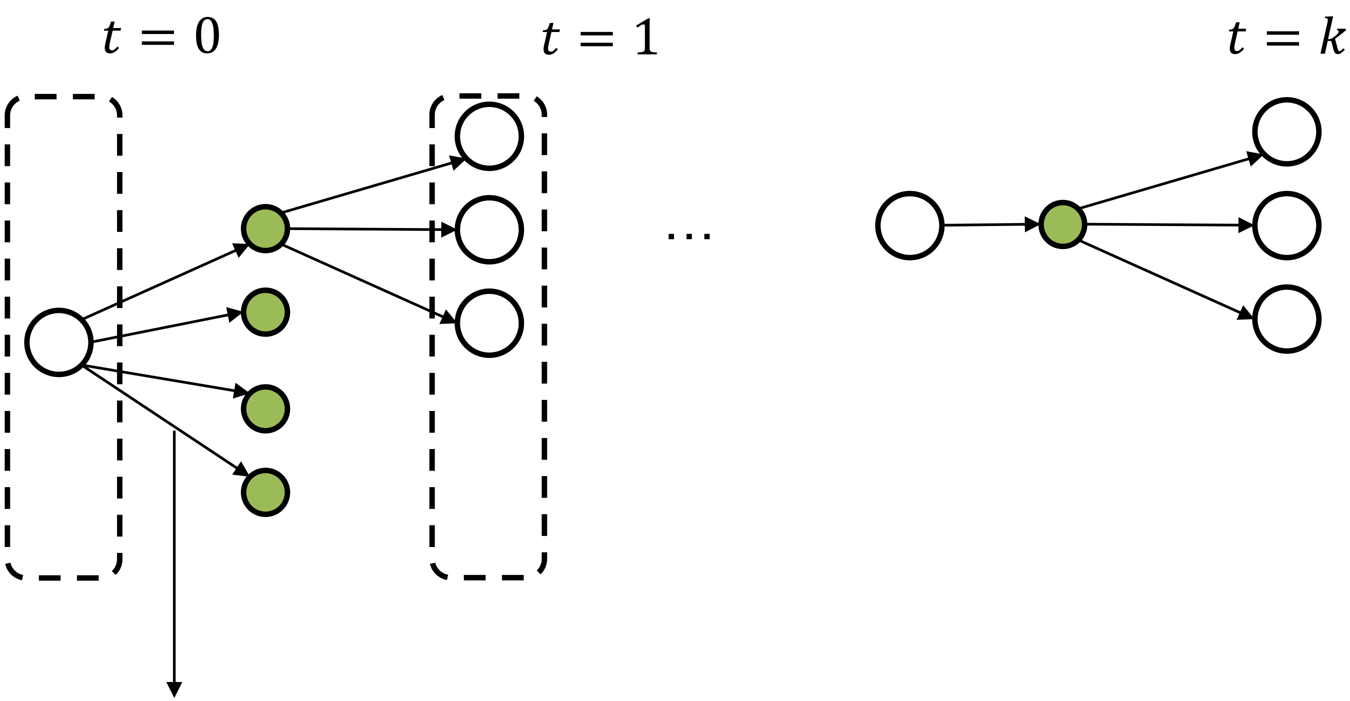
Heuristic Search Value Iteration (HSVI)

We want to minimize the gap between the upper and lower bounds in b_0



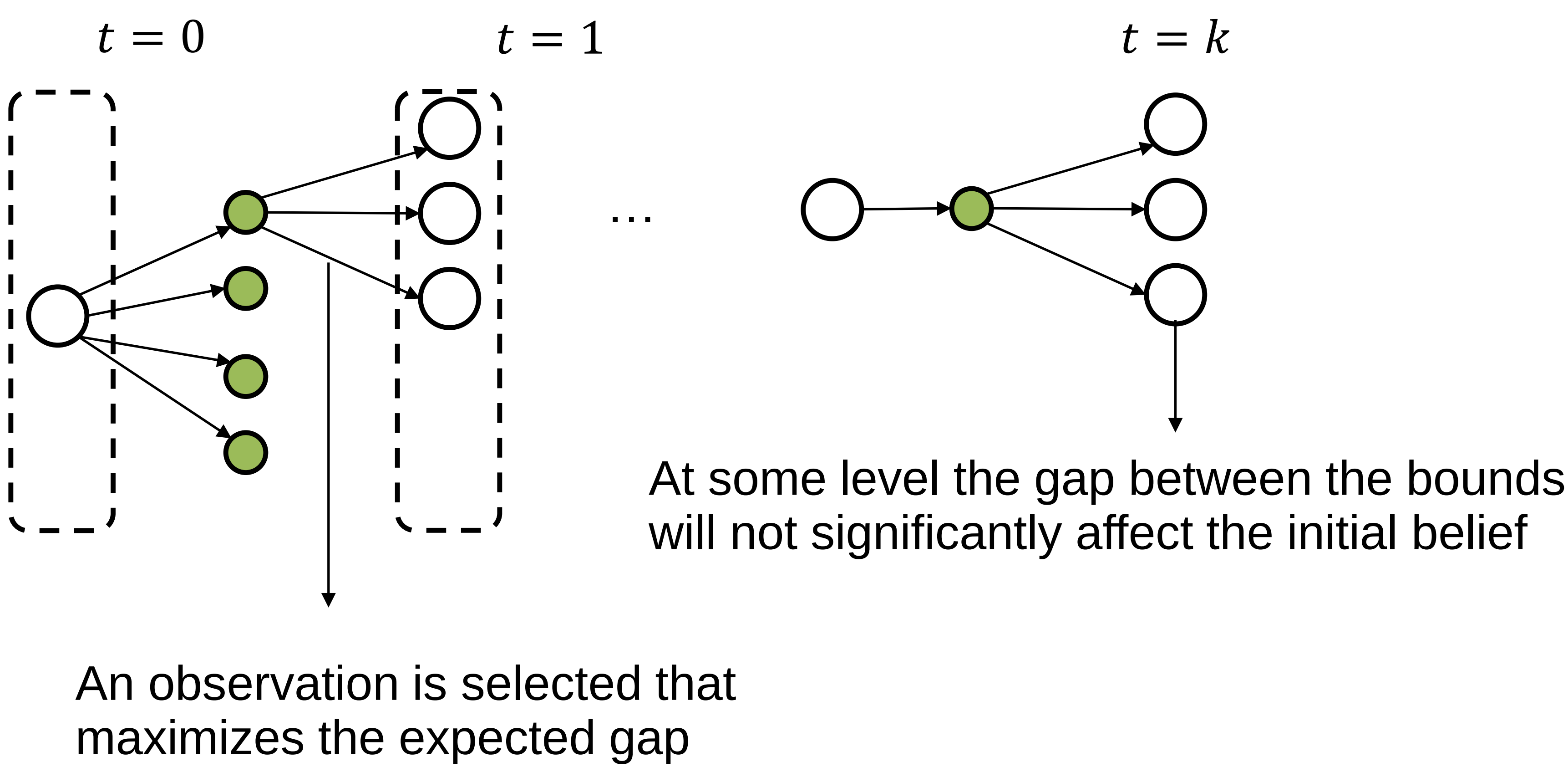
$$excess(b,t) = \left(\overline{V}(b) - \underline{V}(b)\right) - \frac{\epsilon}{\gamma^t}$$

Heuristic Search Value Iteration (HSVI)

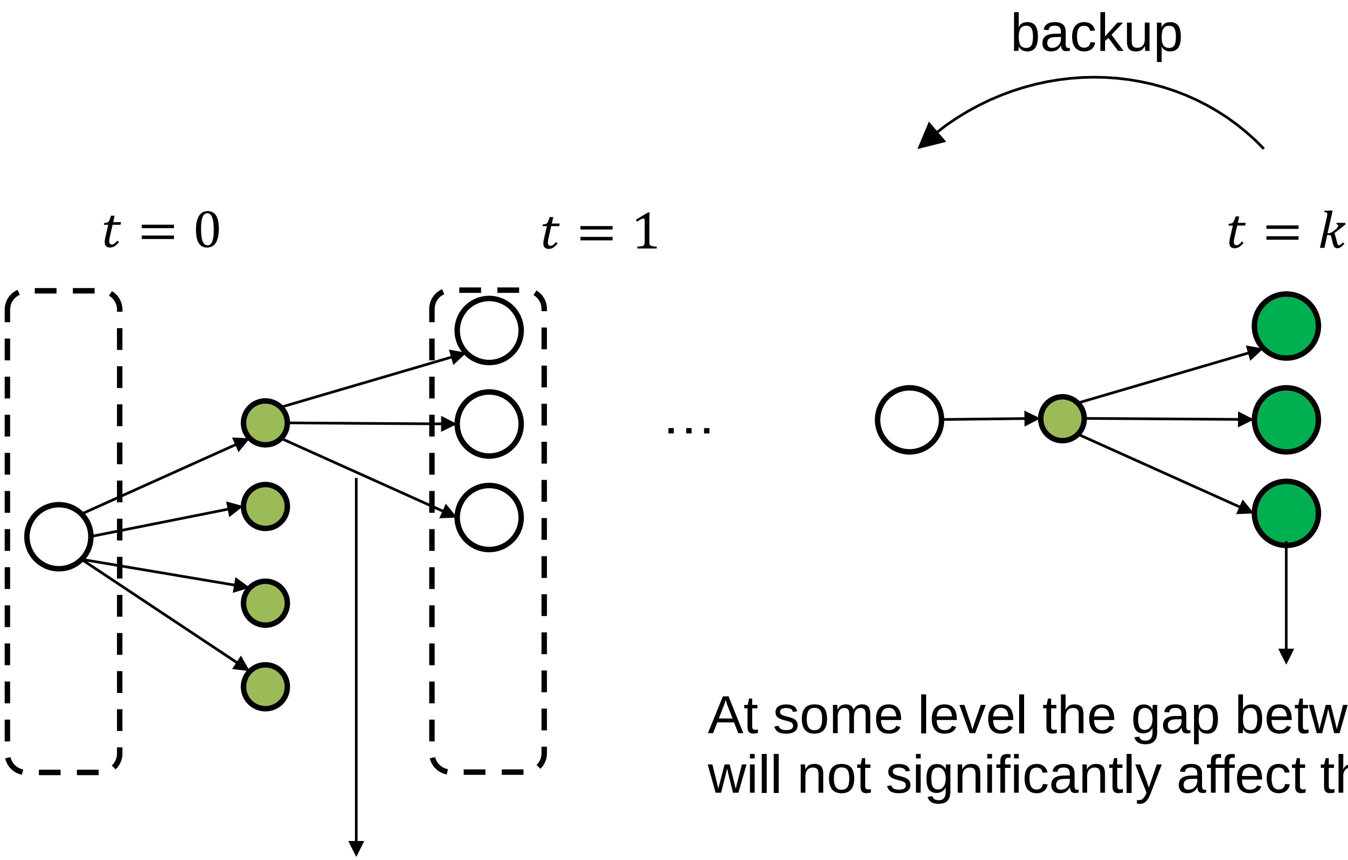


An action with maximal value based on $\overline{V}$ is selected

Heuristic Search Value Iteration (HSVI)



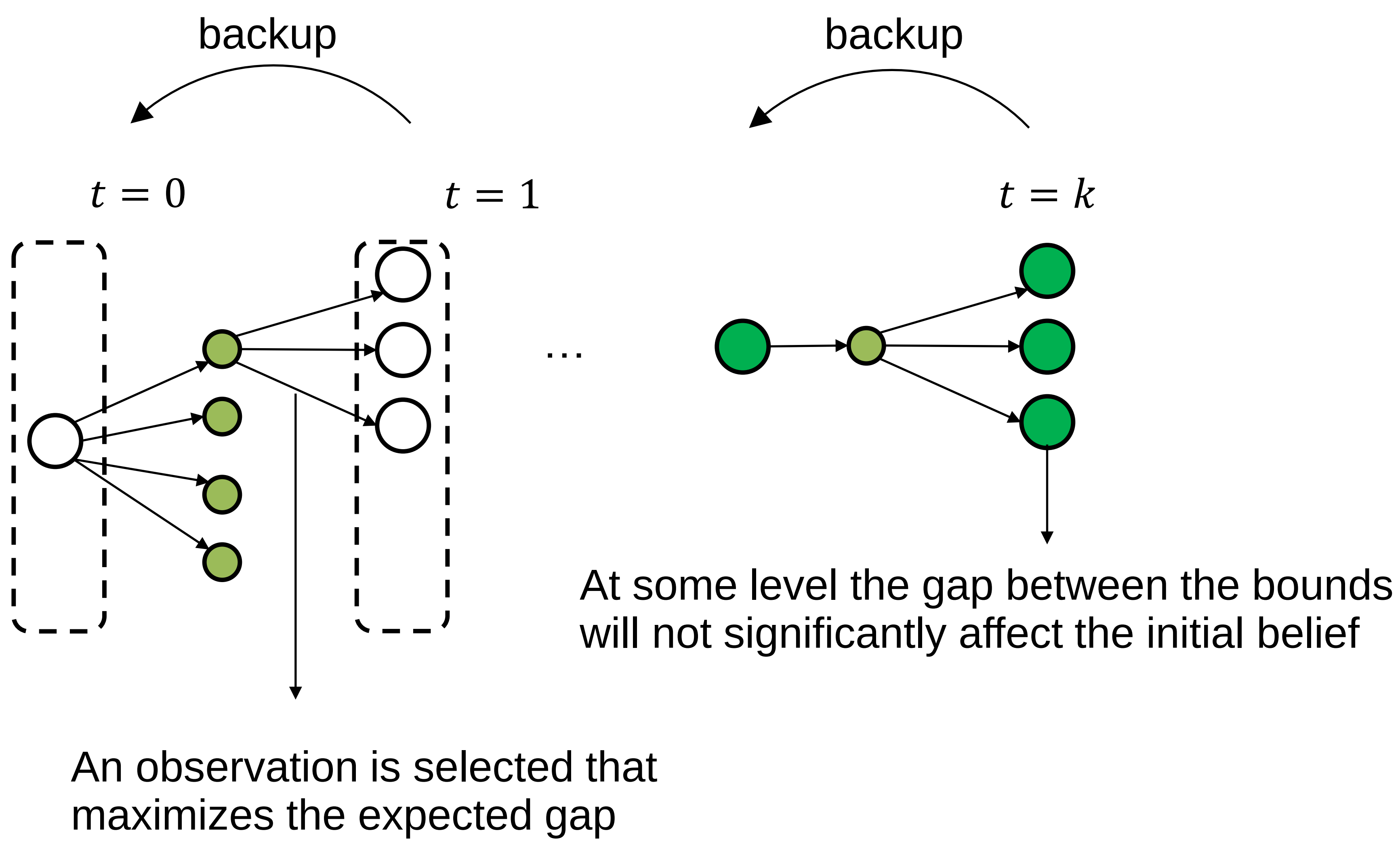
Heuristic Search Value Iteration (HSVI)



At some level the gap between the bounds will not significantly affect the initial belief

An observation is selected that maximizes the expected gap

Heuristic Search Value Iteration (HSVI)



Heuristic Search Value Iteration (HSVI)

Algorithm 5 HSVI

Function HSVI

Initialize $\underline{V}$ and $\bar{V}$
while $\bar{V}(b_0) - \underline{V}(b_0) > \epsilon$ **do**
  BoundUncertaintyExplore$(b_0, 0)$

Function BoundUncertaintyExplore$(b,\ t)$

if $\bar{V}(b) - \underline{V}(b) > \epsilon\gamma^{-t}$ **then**
  // Choose the action according to the upper bound value function
  $a^* \leftarrow \text{argmax}_a\, Q_{\bar{V}}(b, a')$
  // Choose an observation that maximizes the gap between bounds
  $o^* \leftarrow \text{argmax}_o(\Pr(o|b, a^*)(\bar{V}(b^{a,o}) - \underline{V}(b^{a,o}) - \epsilon\gamma^{-(t+1)}))$
  BoundUncertaintyExplore$(b^{a^*,o^*}, t+1)$
  // After the recursion, update both bounds
  $\underline{V} = \underline{V} \cup backup(b, \underline{V}))$
  $\bar{V}(b) \leftarrow J\bar{V}(b)$

Heuristic Search Value Iteration (HSVI)

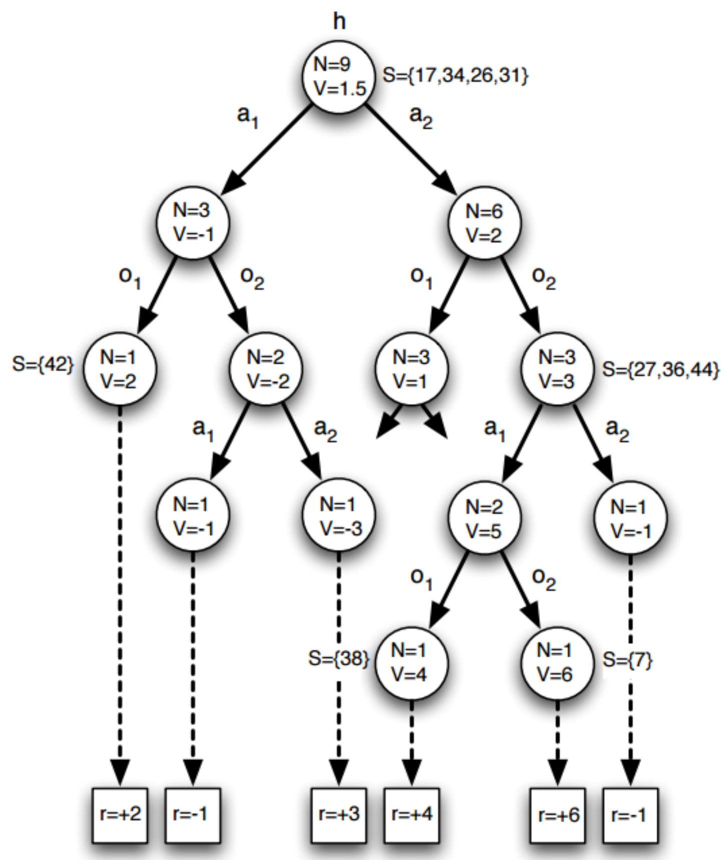
- HSVI iteratively adds points to the upper bounds and α-vectors to the lower bound sets
  - redundancy (some computation for beliefs can be done repeatedly)
  - dominance (some points / vectors can become dominated in later iterations)
  - we can periodically check and remove dominated (and irrelevant) α-vectors (and points)
- there can be other methods for the forward search
  - domain-specific heuristic
  - breadth-search variant, where the algorithm maintains the set of beliefs with positive excess and selects always the one with the maximal excess

Monte Carlo Tree Search for POMDPs

- MCTS techniques can also be used for online decision making for POMDPs
- we do not have states, the MCTS tree needs to be defined in a different way
- we can assume perfect information and aggregate statistics based on actions and observations
  - simple modification, the algorithm learns the best action for each state
  - very inaccurate if the actions have conflicting outcomes for multiple states

Monte Carlo Tree Search for POMDPs

- a node in the search tree can correspond to the history of actions and observations – POMCP (by Silver & Veness,2010)



Monte Carlo Tree Search for POMDPs

- belief updates in POMCPs
  - Bayes update
  - $b(s, hao) = \frac{\sum_{s' \in S} \Omega(a,o,s) T(s',a,s) b(s'|h)}{\sum_{s',s'' \in S} \Omega(a,o,s'') T(s',a,s'') b(s'|h)}$
  - can be too slow for large domains
- approximation using particle filtering
  - the algorithm runs K trials
  - the trails approximate belief distributions

Monte Carlo Tree Search for POMDPs

Algorithm 1 Partially Observable Monte-Carlo Planning

```
procedure SEARCH(h)
    repeat
        if h = empty then
            s ~ I
        else
            s ~ B(h)
        end if
        SIMULATE(s, h, 0)
    until TIMEOUT()
    return argmax_b V(hb)
end procedure

procedure ROLLOUT(s, h, depth)
    if γ^depth < ϵ then
        return 0
    end if
    a ~ π_rollout(h, ·)
    (s', o, r) ~ G(s, a)
    return r + γ.ROLLOUT(s', hao, depth+1)
end procedure

procedure SIMULATE(s, h, depth)
    if γ^depth < ϵ then
        return 0
    end if
    if h ∉ T then
        for all a ∈ A do
            T(ha) ← (N_init(ha), V_init(ha), ∅)
        end for
        return ROLLOUT(s, h, depth)
    end if
    a ← argmax_b V(hb) + c sqrt(log N(h) / N(hb))
    (s', o, r) ~ G(s, a)
    R ← r + γ.SIMULATE(s', hao, depth + 1)
    B(h) ← B(h) ∪ {s}
    N(h) ← N(h) + 1
    N(ha) ← N(ha) + 1
    V(ha) ← V(ha) + (R − V(ha)) / N(ha)
    return R
end procedure
```

Monte Carlo Tree Search for POMDPs

Rocksample	(7, 8)	(11, 11)	(15, 15)
States $\|\mathcal{S}\|$	12,544	247,808	7,372,800
AEMS2	21.37 ±0.22	N/A	N/A
HSVI-BFS	21.46 ±0.22	N/A	N/A
SARSOP	21.39 ±0.01	21.56 ±0.11	N/A
Rollout	9.46 ±0.27	8.70 ±0.29	7.56 ±0.25
POMCP	20.71 ±0.21	20.01 ±0.23	15.32 ±0.28

Monte Carlo Tree Search for POMDPs