T10. A Concise Introduction to Cooperative Multi-Agent Reinforcement Learning

AAMAS'2025 Tutorial

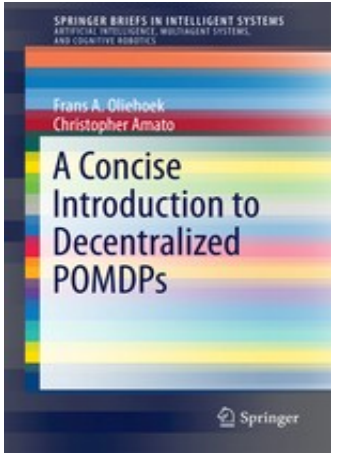
Part 1

Christopher Amato & Frans A. Oliehoek

Outline

Part 1:

- Multiagent decision problems
- Toward multiagent RL
- Some fundaments from multiagent planning
- Multiagent RL through the lens of Dec-POMDPs



available from my website

Part 1a: Multiagent decision problems

Why Multiagent Systems (MASs)?

why do intelligent agents (humans) interact the way they do?

I have a bunch of robots… how do I get them to do something useful?

subjective perspective:
- focus on protagonist agent

Why MASs?
- will need to interact with other agents!

Economist

Engineer

objective perspective:
- focus on team

Why MASs?
- flexible, robust, cost-effective.
- reduces complexity: agent can take care of some aspects itself.

(John Nash – foto by Elke Wetzig)

Why Multiagent Reinforcement Learning?

protagonist needs to learn about other agents

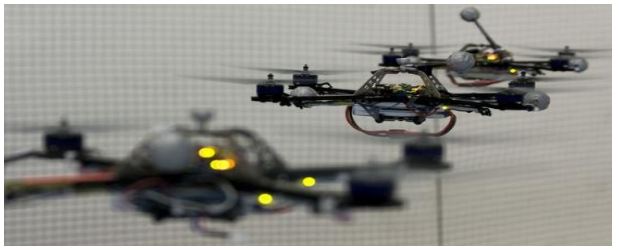
design tool...!
(MAS are hard to program!)

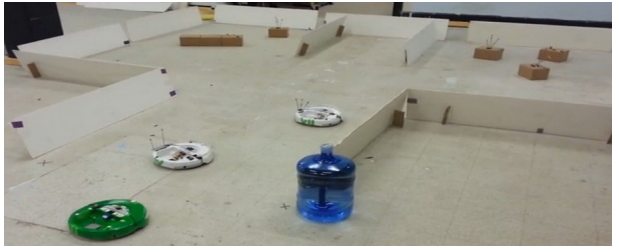
Economist

Engineer

(John Nash – foto by Elke Wetzig)

MASs under Uncertainty



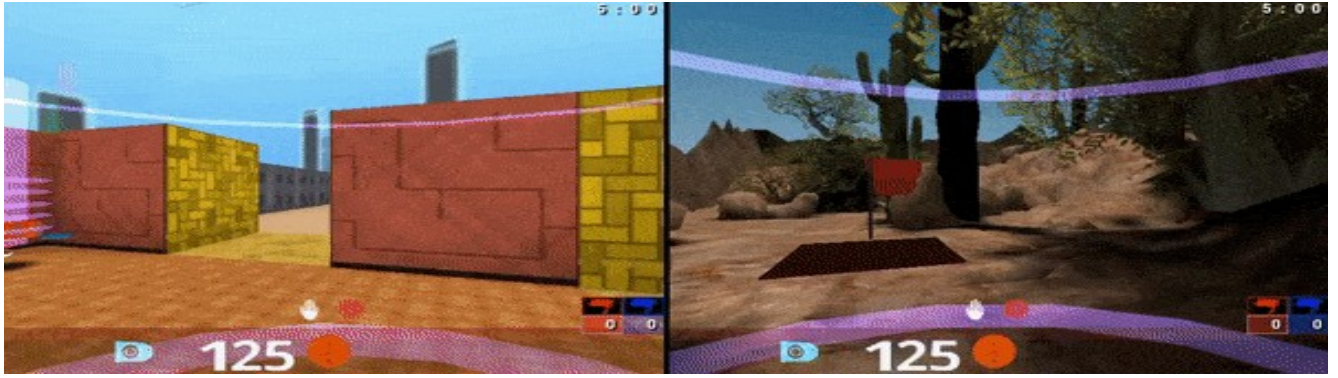


Uncertainty galore:

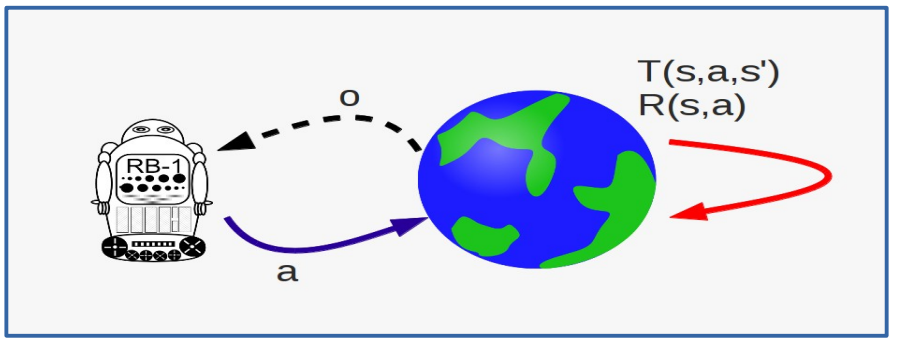
- ▶Outcome Uncertainty
- ▶Partial Observability
- ▶about others

And will this work…?

- Well… **MASs and MARL is complex…!**
- but some encouraging developments…
  - Alpha Go
  - Deepstack
  - Capture the flag:

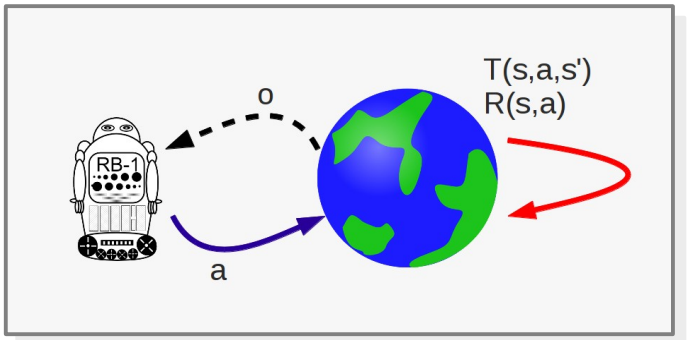


Single agent problems



Single-Agent Decision Making

- An MDP
  - S – set of states
  - A – set of actions
  - P_T – transition function: $P(s'|s,a)$
  - R – reward function: $R(s,a)$ or $R(s,a,s')$
  - T – horizon (finite)



- A POMDP
  - O – set of observations
  - P_O – observation function: $P(o|a,s')$

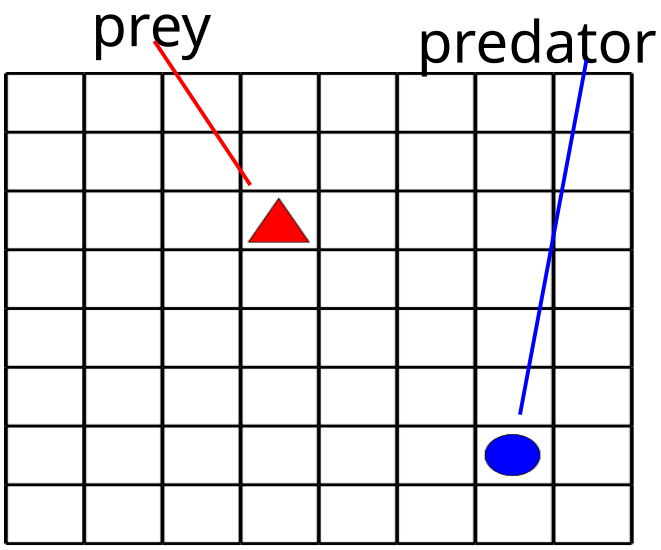
- Initial state distribution: b_0

▶time is sub- or superscript
▶not completely consistent in this tutorial…

Example: Predator-Prey Domain

- Predator-Prey domain
  - 1 agent: predator
  - prey: part of environment
  - on a torus

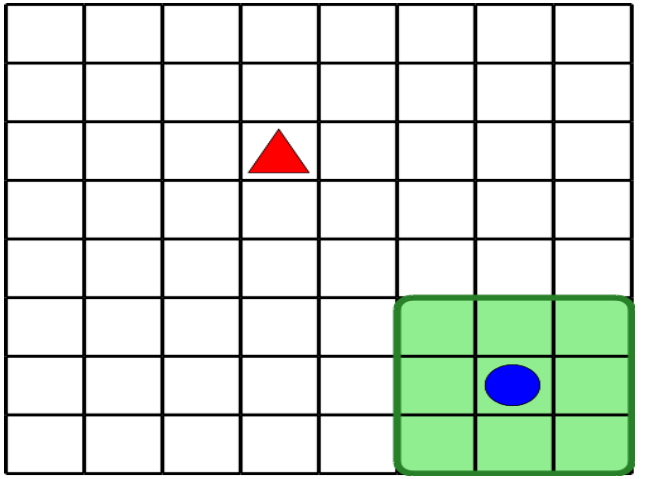
- Formalization:
  - states (-3,4)
  - actions N,W,S,E
  - transitions failing to move, prey moves
  - rewards reward for capturing



Partial Observability

- Now: partial observability
  - E.g., limited range of sight

- MDP + observations
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)

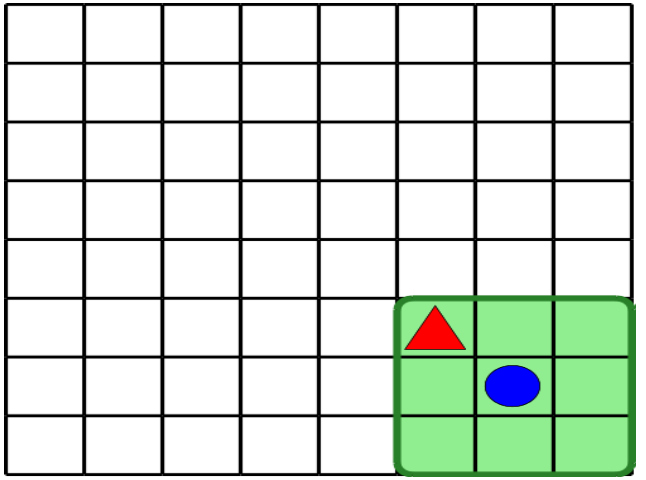


$o = \text{'nothing'}$

Partial Observability

- Now: partial observability
  - E.g., limited range of sight

- MDP + observations
  - explicit observations
  - observation probabilities
    - noisy observations (detection probability)



$o = (-1, 1)$

“Solving” single agent problems

Fully observable

Partially observable

Model / simulator available

F.O. planning

- generalized policy iteration
- online planning: e.g. MCTS

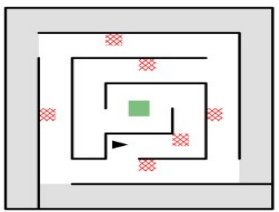
P.O. planning

- belief MDP, POMDP VI
- online planning e.g. POMCP

“Solving” single agent problems

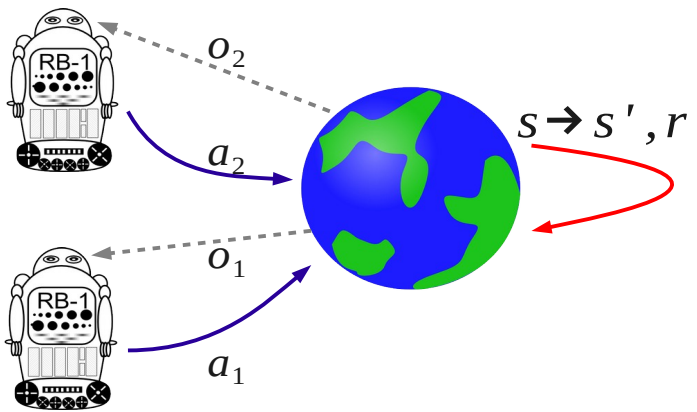
Fully observable	Partially observable	
F.O. planning • generalized policy iteration • online planning: e.g. MCTS	P.O. planning • belief MDP, POMDP VI • online planning e.g. POMCP	Model / simulator available
FORL • (deep) Q-learning, SARSA, etc. • (deep) model-based RL	PORL • policy gradient with FSC • deep RL with recurrent layers	No model / simulator available

In practice: RL methods is (almost only) used when a simulator is available

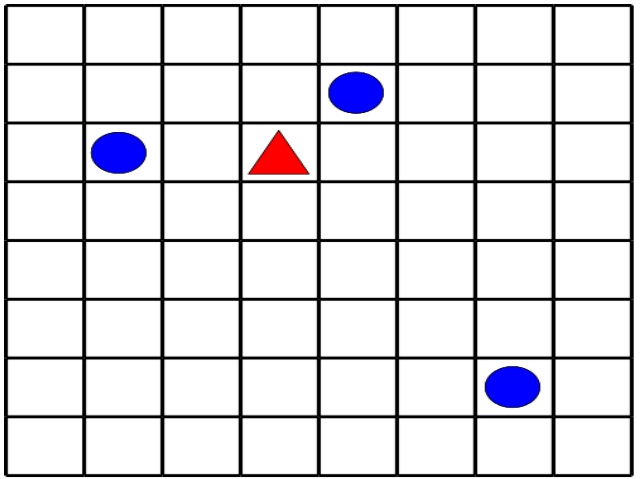


[Corneil et al. 2018 ICML]

Multiple Agents



Multiple Agents (Fully observable)

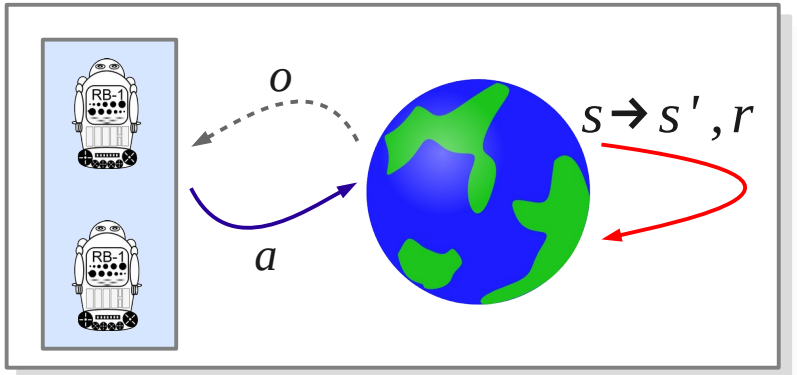


- Now: multiple agents
  - fully observable

- Formalization:
  - states ((3,-4), (1,1), (-2,0))
  - actions {N,W,S,E}
  - **joint** actions {(N,N,N), (N,N,W),…,(E,E,E)}
  - transitions probability of failing to move, prey moves
  - rewards reward for capturing jointly

Multiple Agents (Fully observable)

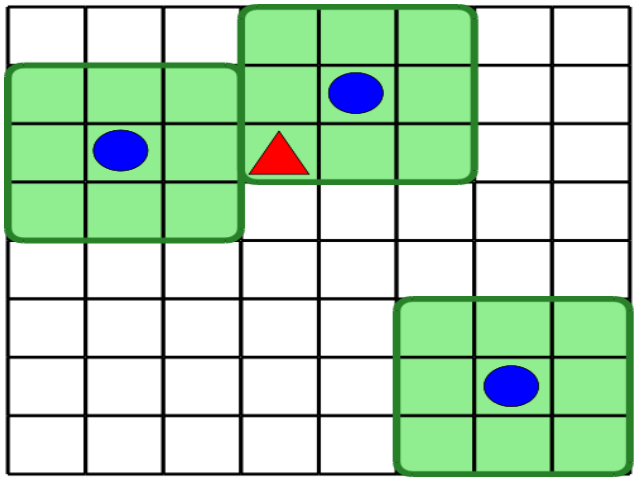
- **Multiagent MDP** [Boutilier 1996]
  - n agents
  - joint actions $\boldsymbol{a} = \langle a_1, \ldots a_n \rangle$
  - transitions and rewards depend on these joint actions

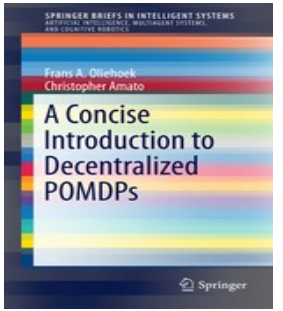


- Solution:
  - Treat as normal MDP with 1 'puppeteer agent' → optimal policy π^* → specifies $\boldsymbol{a}$ for each s
  - Every agent simple executes its part: a_i

Multiple Agents & Partial Observability

- Now both…
  - partial observability
  - multiple agents
- “Decentralized POMDP” (Dec-POMDP) framework*
- both
  - joint actions and
  - joint observations

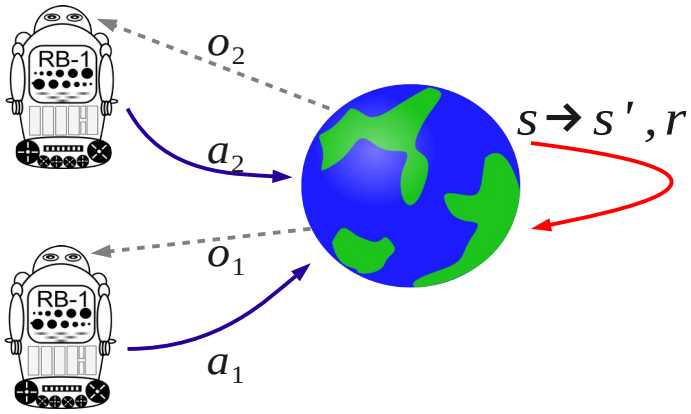




*See, e.g., “A Concise Introduction to Decentralized POMDPs”
http://www.fransoliehoek.net/publications/htmlfiles/b2hd-OliehoekAmato16book.html

The Formal Dec-POMDP Model

- A Dec-POMDP:
  - n agents
  - S – set of states
  - A – set of **joint** actions $\boldsymbol{a} = \langle a_1,\ldots,a_n \rangle$
  - P_T – transition function: $P(s' \mid s,\boldsymbol{a})$
  - O – set of **joint** observations $\boldsymbol{o} = \langle o_1,\ldots,o_n \rangle$
  - P_O – observation function: $P(\boldsymbol{o} \mid \boldsymbol{a},s')$
  - R – reward function: $R(s,\boldsymbol{a})$
  - T – horizon (finite)



►agents indices are generally subscript

Goal:

- Find the optimal **joint** policy $\pi^* = \langle \pi_1, \pi_2 \rangle$

- What is the optimal one?
  - Define **value** as the expected (discounted) sum of rewards:

$$V(\pi) = \boldsymbol{E}\left[\sum_{t=0}^{T-1} R(s_t, a_t) \mid \pi, b^0\right]$$

  - an optimal joint policy is one with maximal value

Acting in Dec-POMDPs…

- No clear reductions to single agent case…

- Idea: use **joint belief**, *b(s)* [Pynadath and Tambe 2002]
  - compute *b(s)* using joint actions and observations
  - Problem: agents do not know those during execution***

- So… now what?
  - How do we plan, when we have the model?
  - How do we learn, otherwise?

***** "puppeteer reduction" requires broadcasting observations!**
Under instantaneous, cost-free, noise-free communication this is optimal [Pynadath and Tambe 2002]

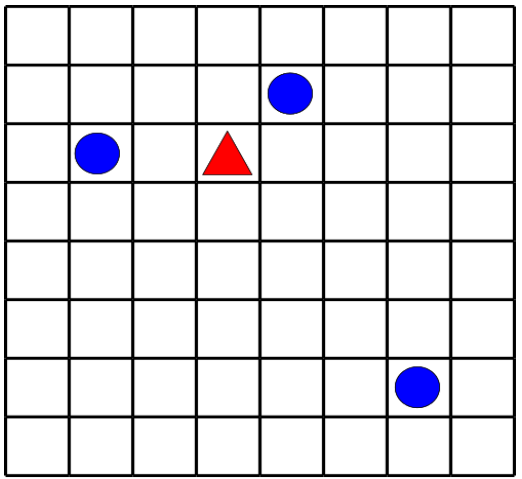
“Solving” multiagent problems

	Fully observable	Partially observable
Model / simulator available	F.O. planning • “puppeteer” reduction→ solve MMDP • online planning / exploit factorization for scalability	P.O. planning • … ?
No model / simulator available***	FO MARL • …?	PO MARL • …?

This is what we are interested in this tutorial…
→ let’s try to go there directly

*** Again MARL methods are (almost only) used when a simulator is available

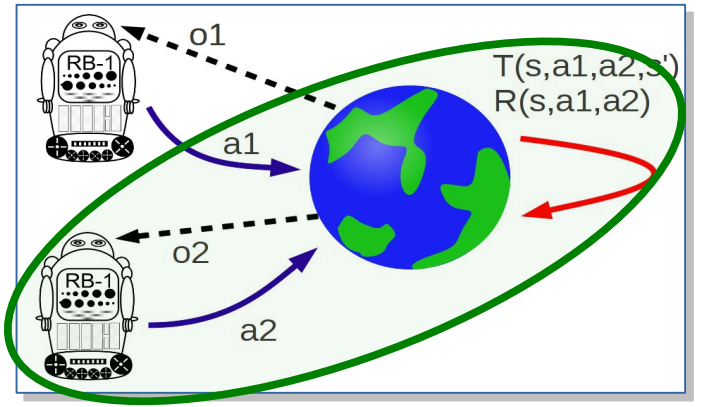
Part 1b: Towards multiagent RL (Two first ideas)



Idea 1: Individual learning

- E.g. just use Q-learning per agent:

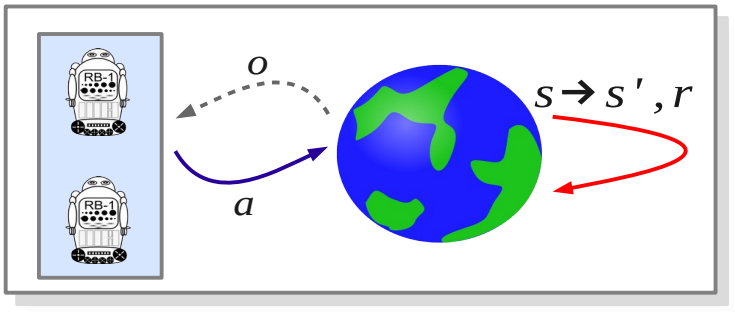
$$Q(s,a_i) := (1-\alpha)\, Q(s,a_i) + \alpha\, [r+\gamma\, V(s')\,]$$



- Can work well…**but the environment is changing**
  - "non-stationary learning problem"

- Convergence?
  - State observable → converges to a **local** optimum
  - Otherwise → no guarantees

Idea 2: Coupled Learners



- Multiagent MDP:
  MDP with joint actions…
- E.g. “joint Q-learning”: $Q(s,\boldsymbol{a}) = (1-\alpha)\, Q(s,\boldsymbol{a}) + \alpha\,[\, r + \gamma \max_{\boldsymbol{a}'} Q(s',\boldsymbol{a}')\,]$
  - A.k.a.: “Joint action learners”
  - Note: can be implemented decentrally in FORL (each agent executes its component)
- Guarantees:
  - Same guarantees as normal Q-learning
  - will converge in the limit
- Scalability: **exponential** in number of agent…

Limitations so far….

- OK. These were reasonable ideas… but:
  - individual learners → no guarantees (non-stationarity)
  - coupled learners → not scalable

- Plus… limited applicability
  - agents need to observe full state
  - in most MASs this is not possible

We need formal frameworks that:
- ►can deal with partial observability
- ►and represents knowledge: who learned what?

“Solving” multiagent problems

fundaments of single agent RL based on single agent planning!

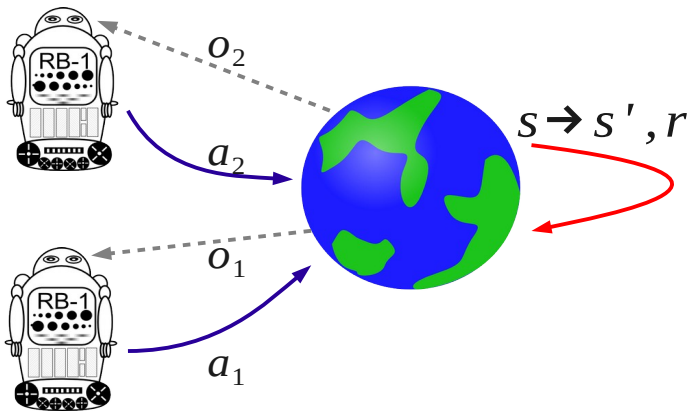
Fully observable	Partially observable
F.O. planning • “puppeteer” reduction→ solve MMDP • online planning / exploit factorization for scalability	P.O. planning • … ?
FO MARL • Individual Q learners: $Q(s,a_i)$ • Joint Q-learners $Q(s,\boldsymbol{a})$	PO MARL • Individual Q learners: $Q(h_i,a_i)$

Model / simulator available

No model / simulator available***

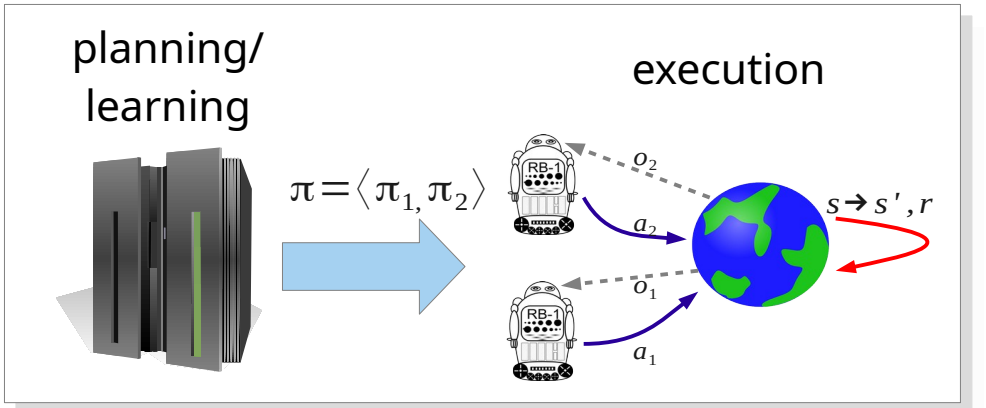
*** Again MARL methods are (almost only) used when a simulator is available

Part 1c: Some fundaments from multiagent planning



Dec-POMDPs: Off-line / On-line phases

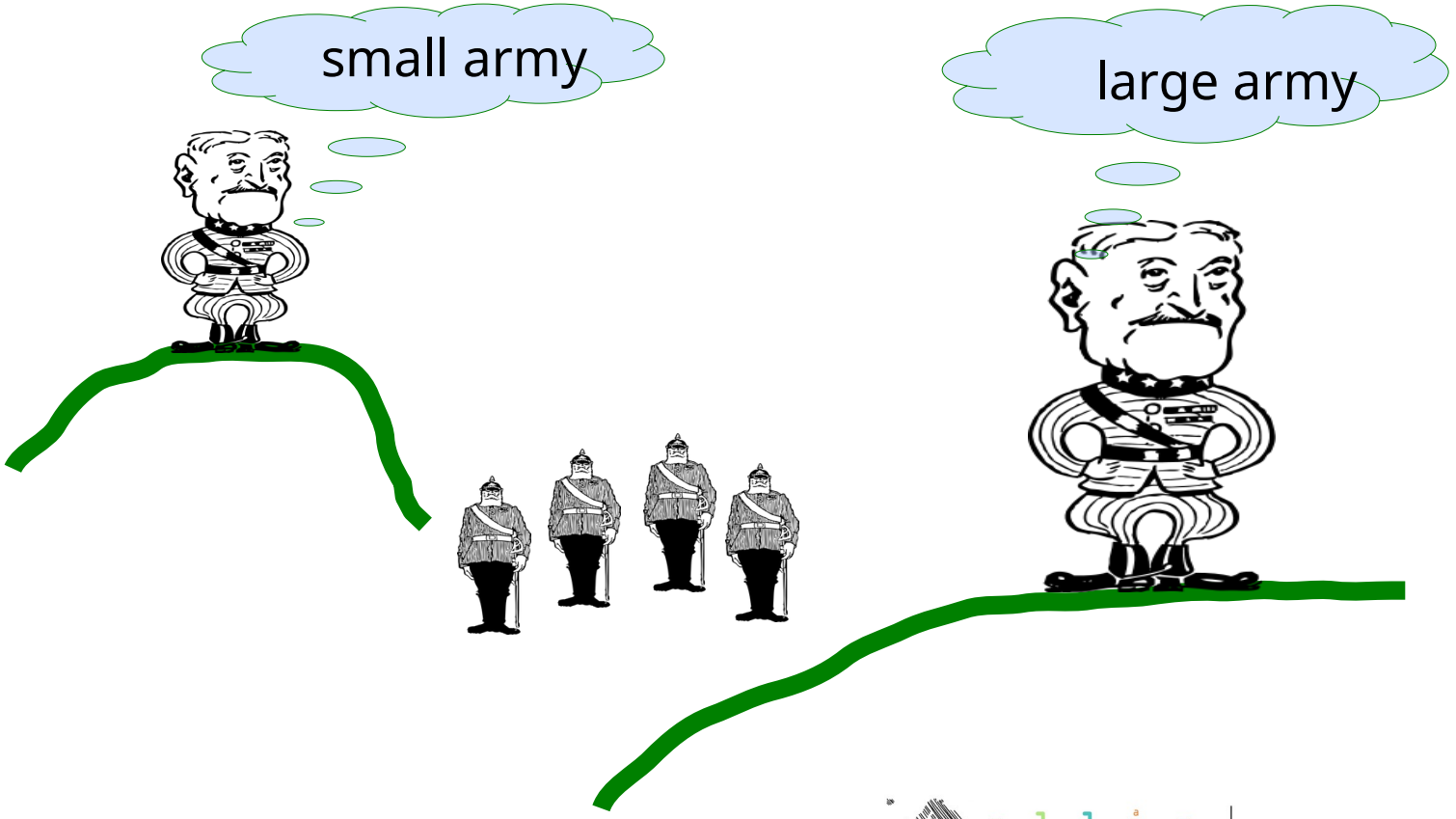
When is planning and/or learning taking place?



- True online learning
- Planning / Simulation based planning
  - plan offline given model
  - do 'learning' offline using simulator
  - limits applicability… but common assumption in MARL (and in fact in nearly all RL)

Running Example

- 2 generals problem



Running Example

- 2 generals problem

suppose T=3,
what do you think is optimal in this problem?

S – { s_L, s_S }
A_i – { (O)bserve, (A)ttack }
O_i – { (L)arge, (S)mall }

Transitions
- ►Both Observe → no state change
- ►At least 1 Attack → reset (50% probability s_L, s_S)

Observations
- ►Probability of correct observation: 0.85
- ►E.g., P(<L, L> | s_L) = 0.85 * 0.85 = 0.7225
- ►(reset is not observed!)

Rewards
- ►1 general attacks:
  he loses the battle
  - R(*,<A,O>) = -10
- ►Both generals Observe:
  small cost
  - R(*,<O,O>) = -1
- ►Both Attack:
  depends on state
  - R(s_L,<A,A>) = -20
  - R(s_S,<A,A>) = +5

Policy Domain

- What do policies look like?
  - In general (action-observation) histories → actions
  - in MDP/POMDP: compact representations: states/beliefs

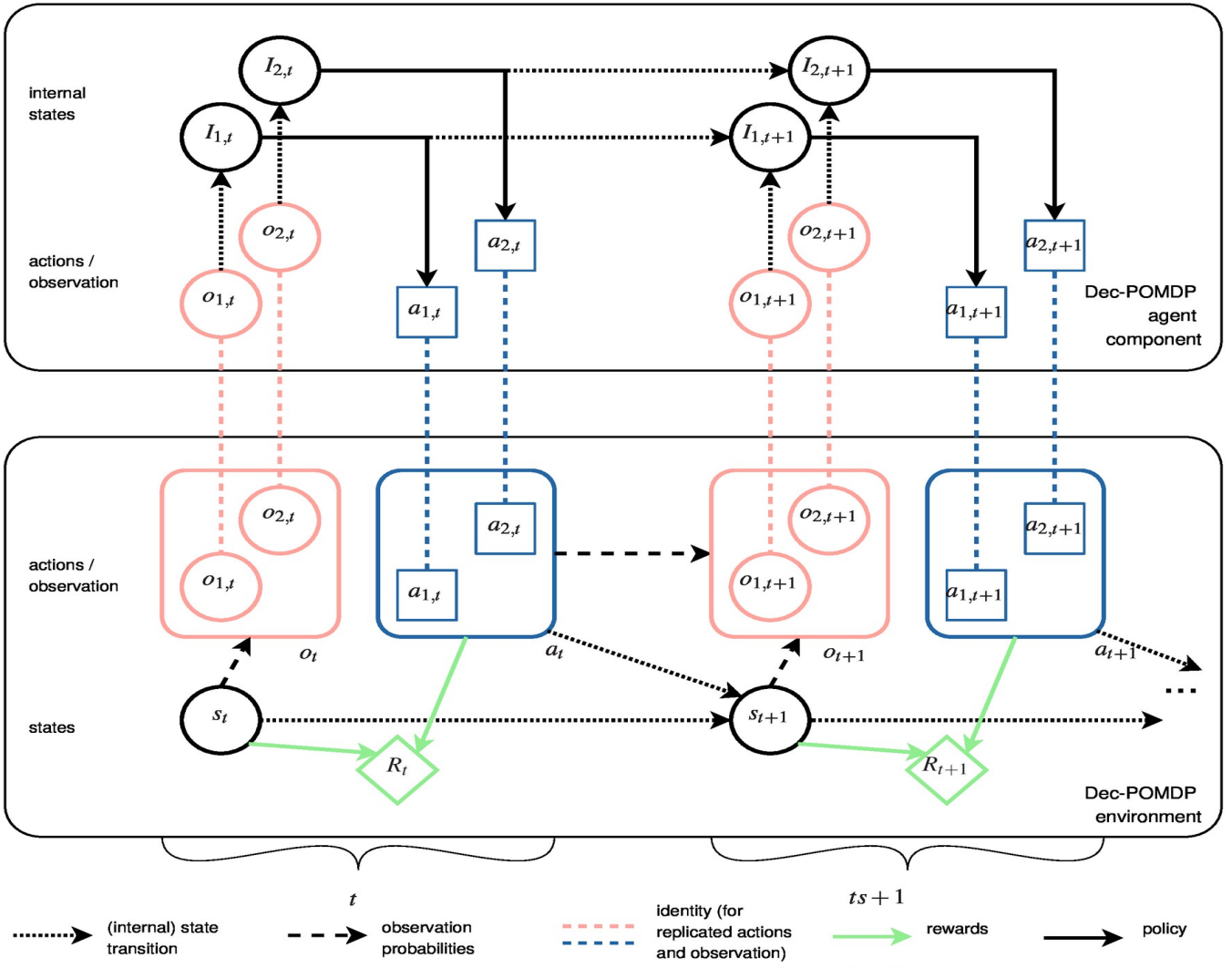
- For Dec-POMDPs: no such representation known!

  → If we want optimal policies: we are **stuck with histories**

- Of course, can try and compress…
  - → approximate internal states I_i
  - cf. RNNs

} more general picture

The general picture

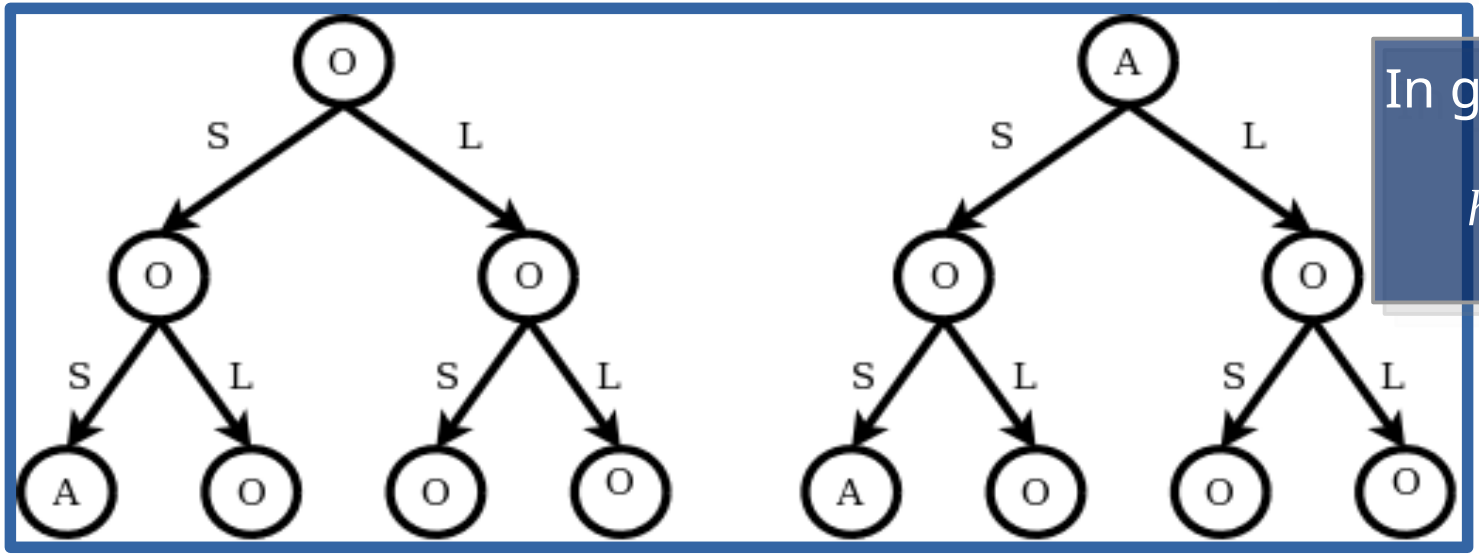


Observation histories and policy trees

- What type of histories?
  - observation histories

In cooperative case **deterministic** policies only need OHs:

$$\vec{o}_i = (o_i^1, \dots, o_i^t)$$



In general, AOHs:

$$h_i = (a_i^0, o_i^1, a_i^1, \dots, a_i^{t-1}, o_i^t)$$

Policies for Two Generals...

Optimal policy for 2 generals, h=3
value=-2.86743

General 1:
() --> observe
(o_small) --> observe
(o_large) --> observe
(o_small,o_small) --> attack
(o_small,o_large) --> attack
(o_large,o_small) --> attack
(o_large,o_large) --> observe

General 2:
() --> observe
(o_small) --> observe
(o_large) --> observe
(o_small,o_small) --> attack
(o_small,o_large) --> attack
(o_large,o_small) --> attack
(o_large,o_large) --> observe

Policies for Two Generals...

Optimal policy for 2 generals, h=3
value=-2.86743

General 1:
() --> observe
(o_small) --> observe
(o_large) --> observe
(o_small,o_small) --> attack
(o_small,o_large) --> attack
(o_large,o_small) --> attack
(o_large,o_large) --> observe

Anything that seems strange...?

General 2:
() --> observe
(o_small) --> observe
(o_large) --> observe
(o_small,o_small) --> attack
(o_small,o_large) --> attack
(o_large,o_small) --> attack
(o_large,o_large) --> observe

Policies for Two Generals...

Optimal policy for 2 generals, h=3
value=-2.86743

General 1:	General 2:
() --> observe	() --> observe
(o_small) --> observe	(o_small) --> observe
(o_large) --> observe	(o_large) --> observe
(o_small,o_small) --> attack	(o_small,o_small) --> attack
(o_small,o_large) --> attack	(o_small,o_large) --> attack
(o_large,o_small) --> attack	(o_large,o_small) --> attack
(o_large,o_large) --> observe	(o_large,o_large) --> observe

Anything that seems strange...?

Coordination vs. Exploitation

- Inherent trade-off:

coordination vs. **exploitation of local information**

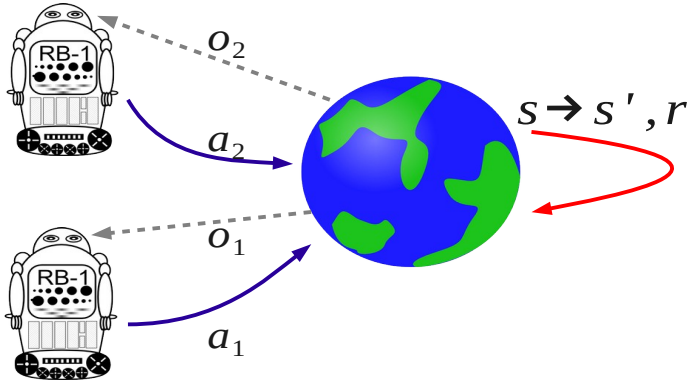
  - Ignore own observations → 'open loop plan'
    - E.g., “ATTACK on 2nd time step”

      + maximally predictable
      - low quality
  - Ignore coordination → 'individual POMDP plan'
    - E.g., try to form an 'individual belief' $b_i(s)$

      (e.g., assume other agents act random…)
      + uses local information
      - likely to result in mis-coordination

- **Optimal policy should balance between these!**

Dec-POMDP Planning Techniques & Optimal value functions



Value of a Joint Policy – history perspective

- Iterative:

$$V(\boldsymbol{\pi}) = \sum_{t=0}^{T-1} \sum_{\boldsymbol{h}_t \in \boldsymbol{\mathcal{H}}_t} \Pr(\boldsymbol{h}_t|\boldsymbol{\pi},b_0) \sum_{\boldsymbol{a}_t \in \boldsymbol{\mathcal{A}}} R(\boldsymbol{h}_t,\boldsymbol{a}_t)\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{h}_t),$$

$$R(\boldsymbol{h}_t,\boldsymbol{a}_t) = \sum_{s_t \in \mathcal{S}} R(s_t,\boldsymbol{a}_t) \Pr(s_t|\boldsymbol{h}_t,b_0)$$

Value of a Joint Policy – history perspective

- Iterative:

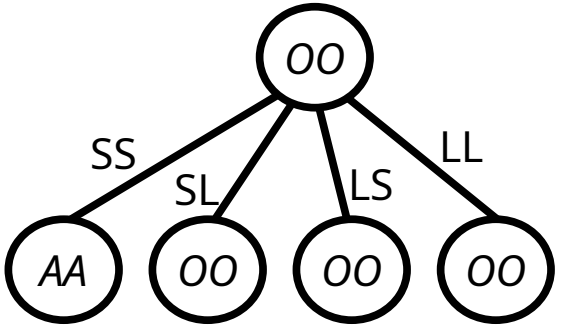
$$V(\boldsymbol{\pi}) = \sum_{t=0}^{T-1} \sum_{\boldsymbol{h}_t \in \boldsymbol{\mathcal{H}}_t} \Pr(\boldsymbol{h}_t|\boldsymbol{\pi},b_0) \sum_{\boldsymbol{a}_t \in \boldsymbol{\mathcal{A}}} R(\boldsymbol{h}_t,\boldsymbol{a}_t)\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{h}_t),$$

$$R(\boldsymbol{h}_t,\boldsymbol{a}_t) = \sum_{s_t \in \mathcal{S}} R(s_t,\boldsymbol{a}_t) \Pr(s_t|\boldsymbol{h}_t,b_0)$$

- Recursive:
  Bellman equation on joint AO-histories

$$V^{\boldsymbol{\pi}}(\boldsymbol{h}_t) = \sum_{\boldsymbol{a}_t \in \boldsymbol{\mathcal{A}}} \boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{h}_t) Q^{\boldsymbol{\pi}}(\boldsymbol{h}_t,\boldsymbol{a}_t)$$

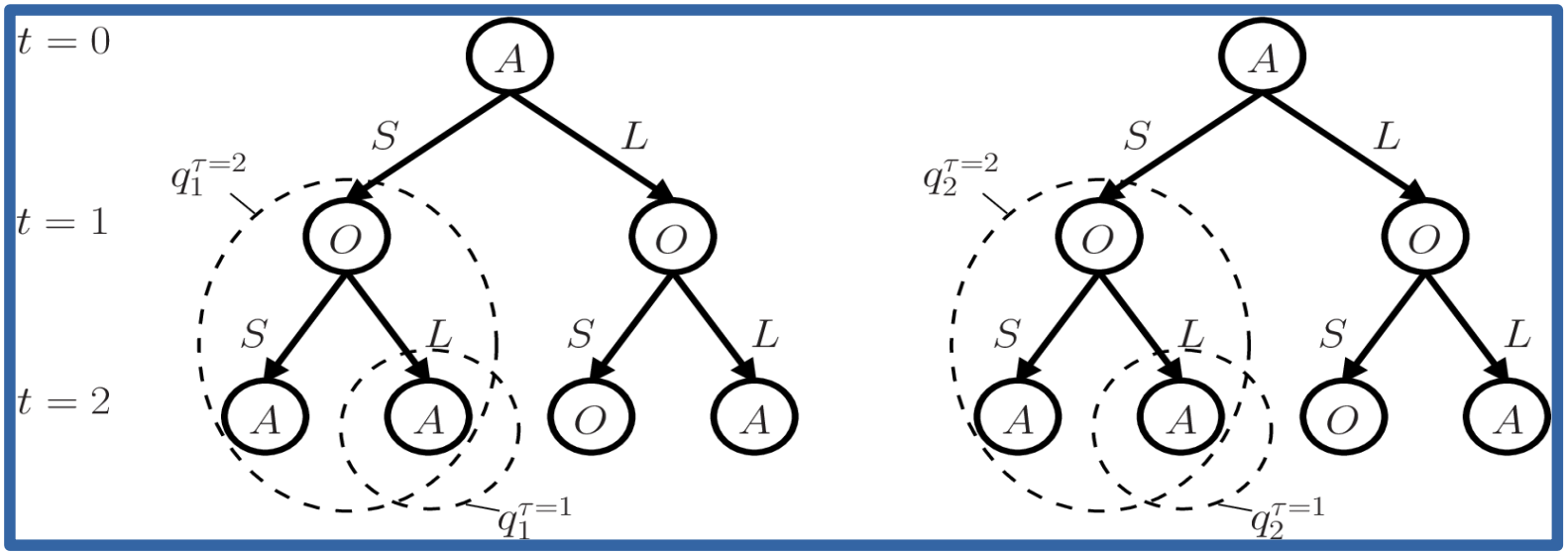
$$Q^{\boldsymbol{\pi}}(\boldsymbol{h}_t,\boldsymbol{a}_t) = R(\boldsymbol{h}_t,\boldsymbol{a}_t) + \sum_{\boldsymbol{o}_{t+1} \in \boldsymbol{\mathcal{O}}} \Pr(\boldsymbol{o}_{t+1}|\boldsymbol{h}_t,\boldsymbol{a}_t) V^{\boldsymbol{\pi}}(\boldsymbol{h}_{t+1})$$



value propagation on a tree

Value of a Joint Policy – subtree policy perspective

- Sub-tree policies:



- Different formulation of value:

$$V(s, q^{\tau=k}) = R(s,a) + \Sigma_o P(o|s,a) V(s', q^{\tau=k-1})$$

  - no need to explicitly remember AOHs… but number of q's is huge!

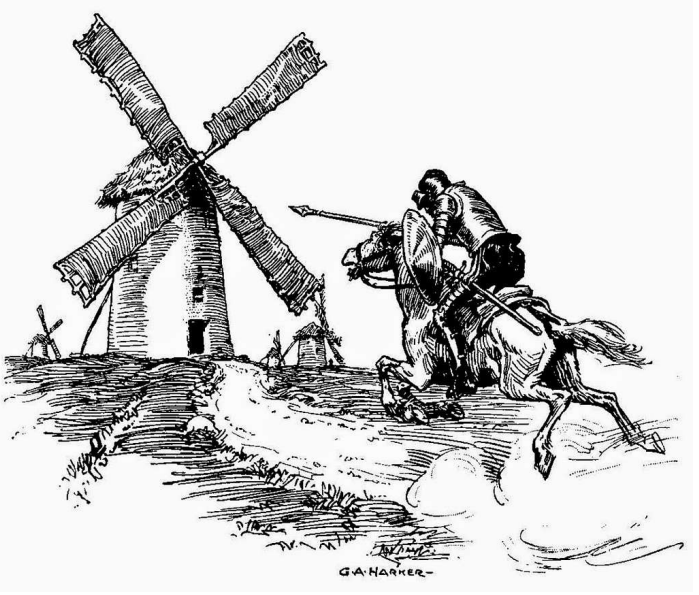
Brute Force Search

- We can compute the value of a joint policy $V(\pi)$
- So the **stupidest algorithm** is:
  - compute $V(\pi)$, for all π
  - select a π with maximum value
- Number of joint policies is huge! (doubly exponential in horizon h)
- Clearly intractable…

h	num. joint policies
1	4
2	64
3	16384
4	1.0737e+09
5	4.6117e+18
6	8.5071e+37
7	2.8948e+76
8	3.3520e+153

Why Dec-POMDP planning?

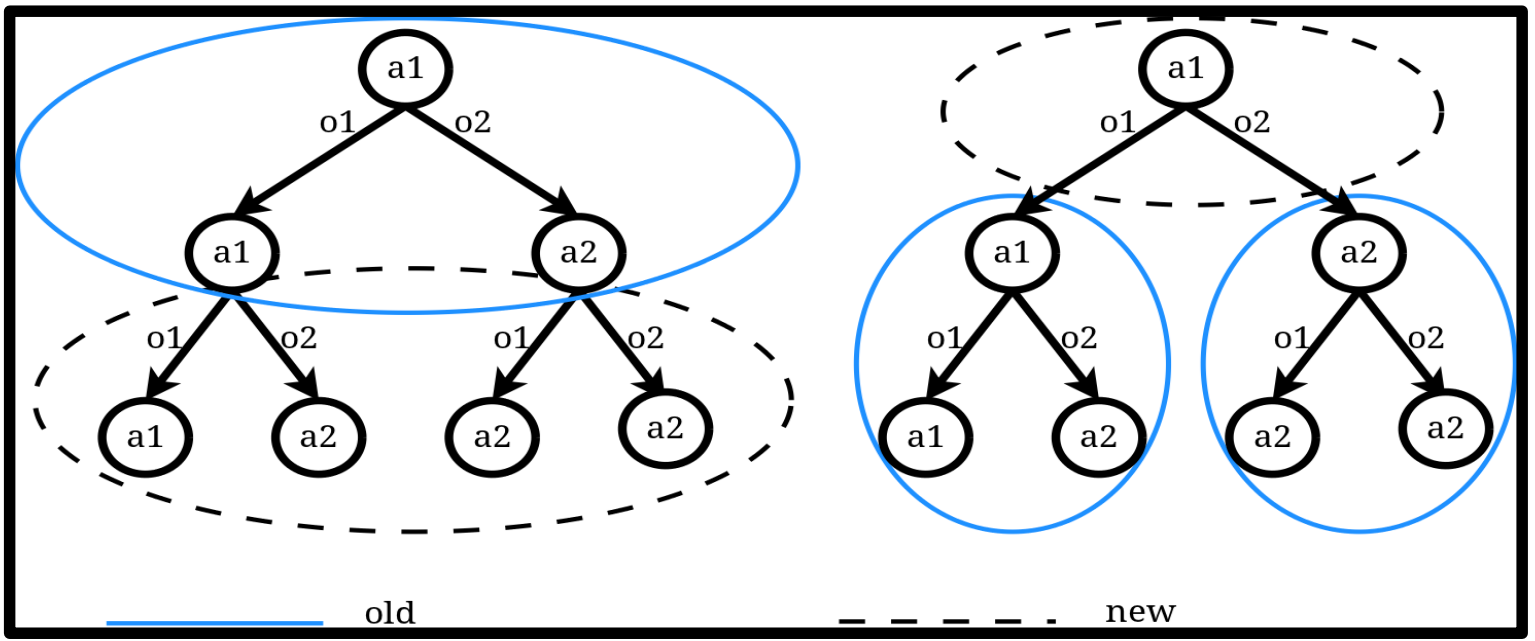
- Finding an optimal plan is (very!) intractable: **NEXP-complete**
- So… should we not just give up?
- Perhaps, but many reasons to care:
  - Interesting..!
  - Understand the problem better
  - Problems do not get easier by ignoring their complexity (and we want collaborating agents… right?)
  - Theory of MDPs (e.g., value functions) are the foundation of RL **→ for effective MARL, we need Dec-POMDP theory.**



Heuristic Search

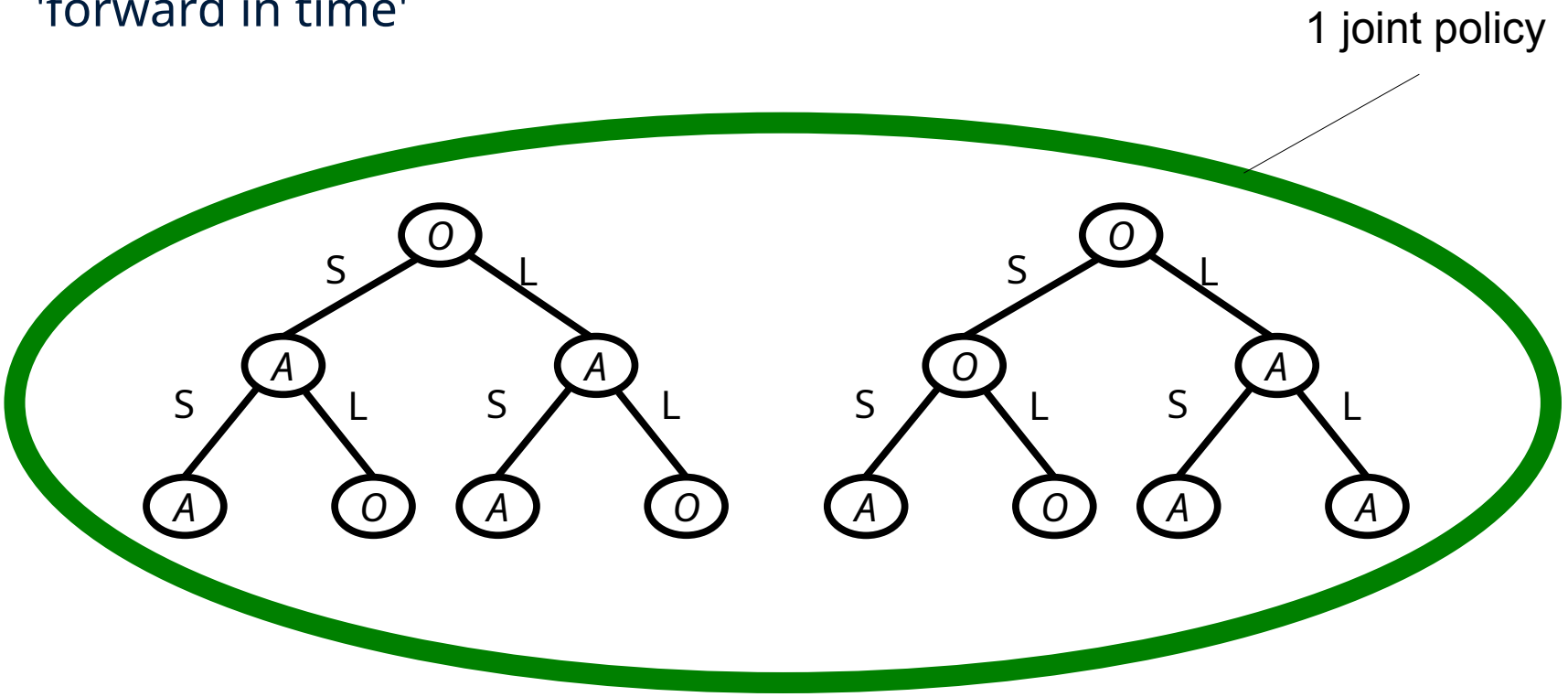
Bottom-up vs. Top-down

- DP constructs bottom-up… alternative: top-down
  → heuristic search [Szer et al. 2005, Oliehoek et al. 2008]



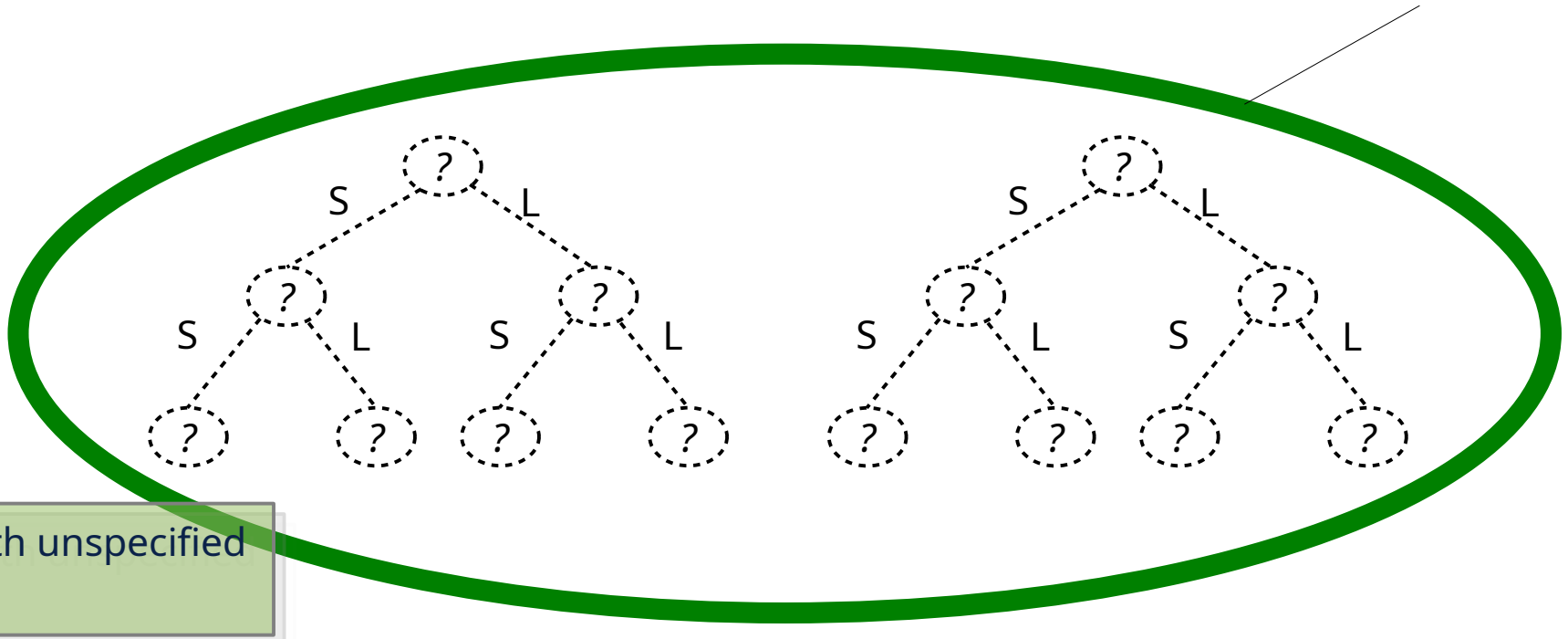
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'



Heuristic Search – 1

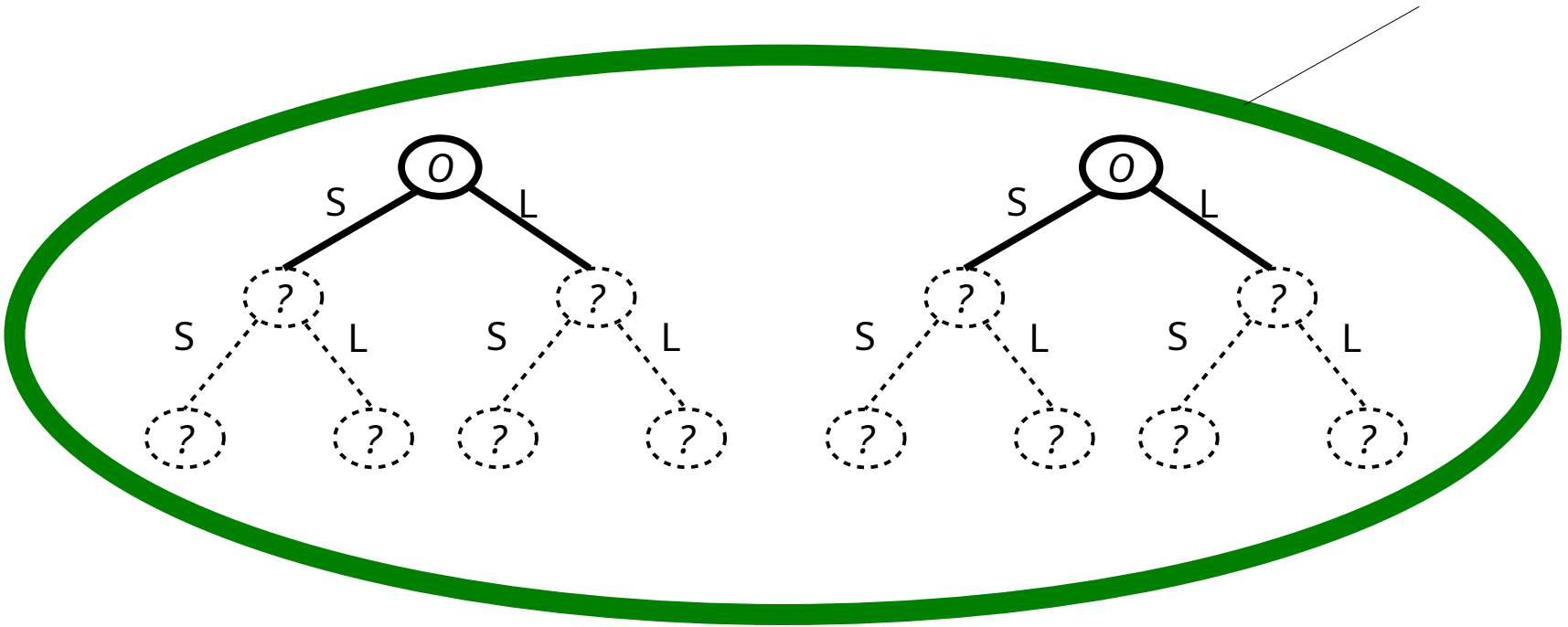
- Incrementally construct all (joint) policies
  - 'forward in time'



Start with unspecified policy

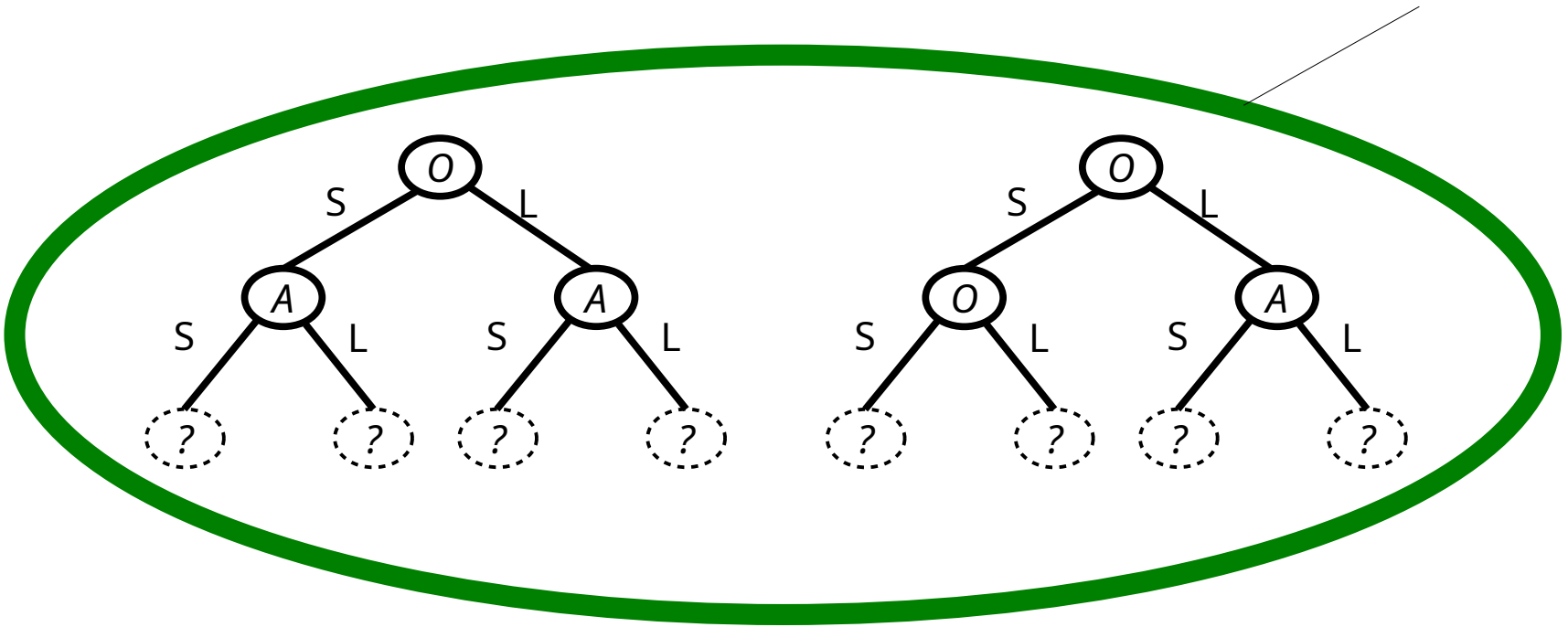
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'



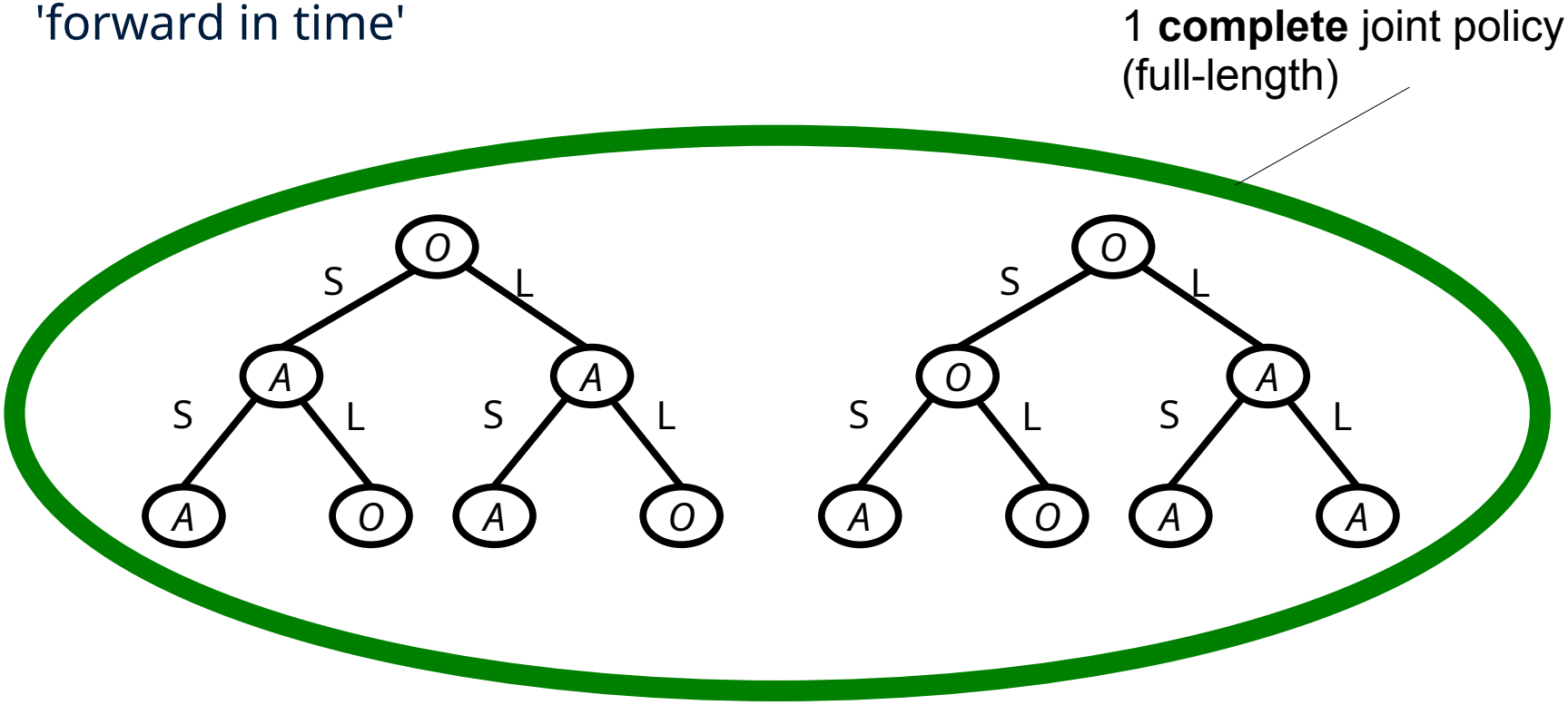
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'



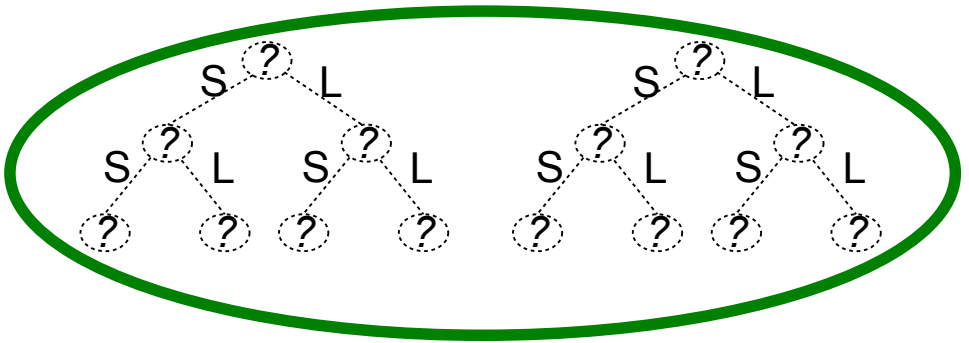
Heuristic Search – 1

- Incrementally construct all (joint) policies
  - 'forward in time'



Heuristic Search – 2

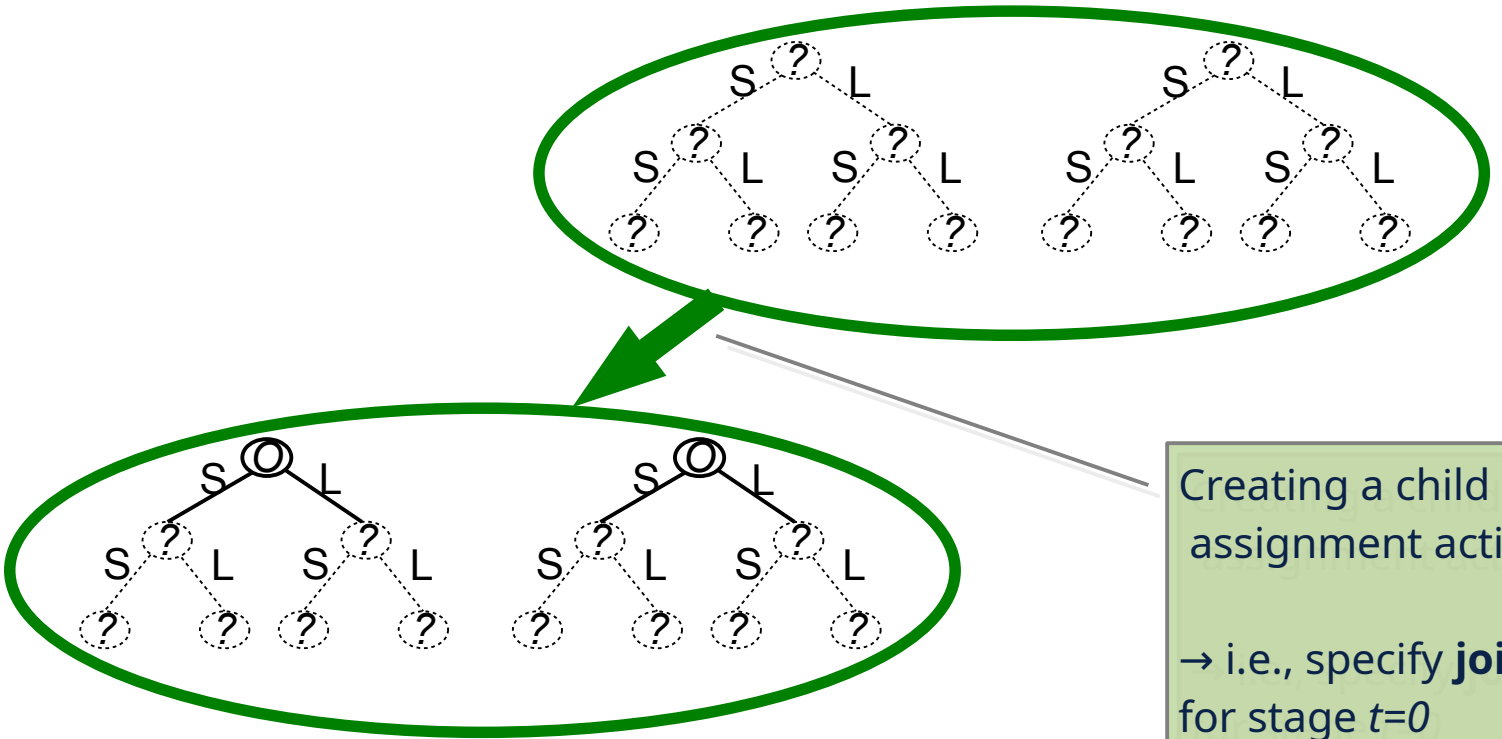
- Creating **ALL** joint policies → tree structure!



Root node:
unspecified joint policy

Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!



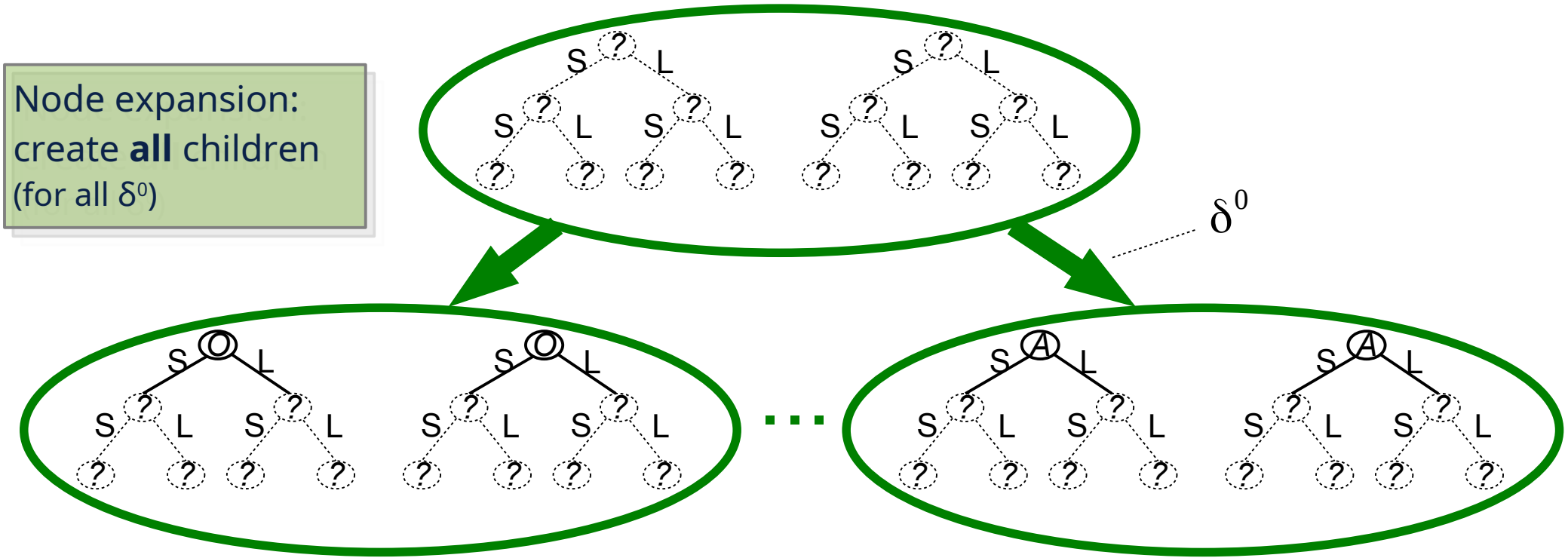
Creating a child node:
assignment actions at *t=0*

→ i.e., specify **joint decision rule** δ^0 for stage *t=0*

Heuristic Search – 2

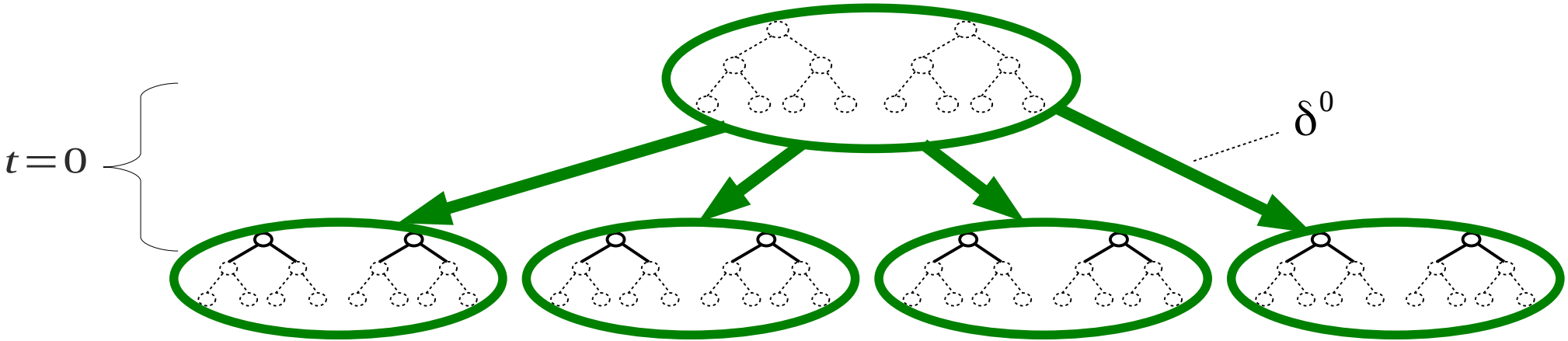
- Creating **ALL** joint policies → tree structure!

Node expansion:
create **all** children
(for all δ^0)



Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!

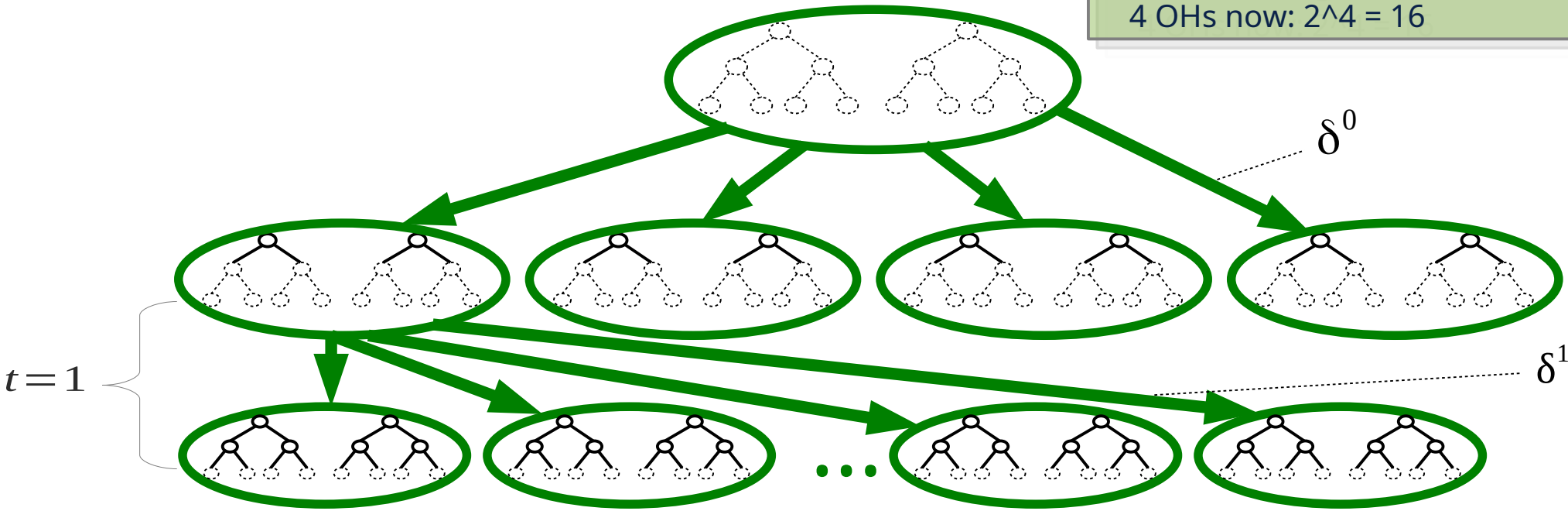


Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!

Next expansion: more children!

→ need to assign actions to 4 OHs now: 2^4 = 16

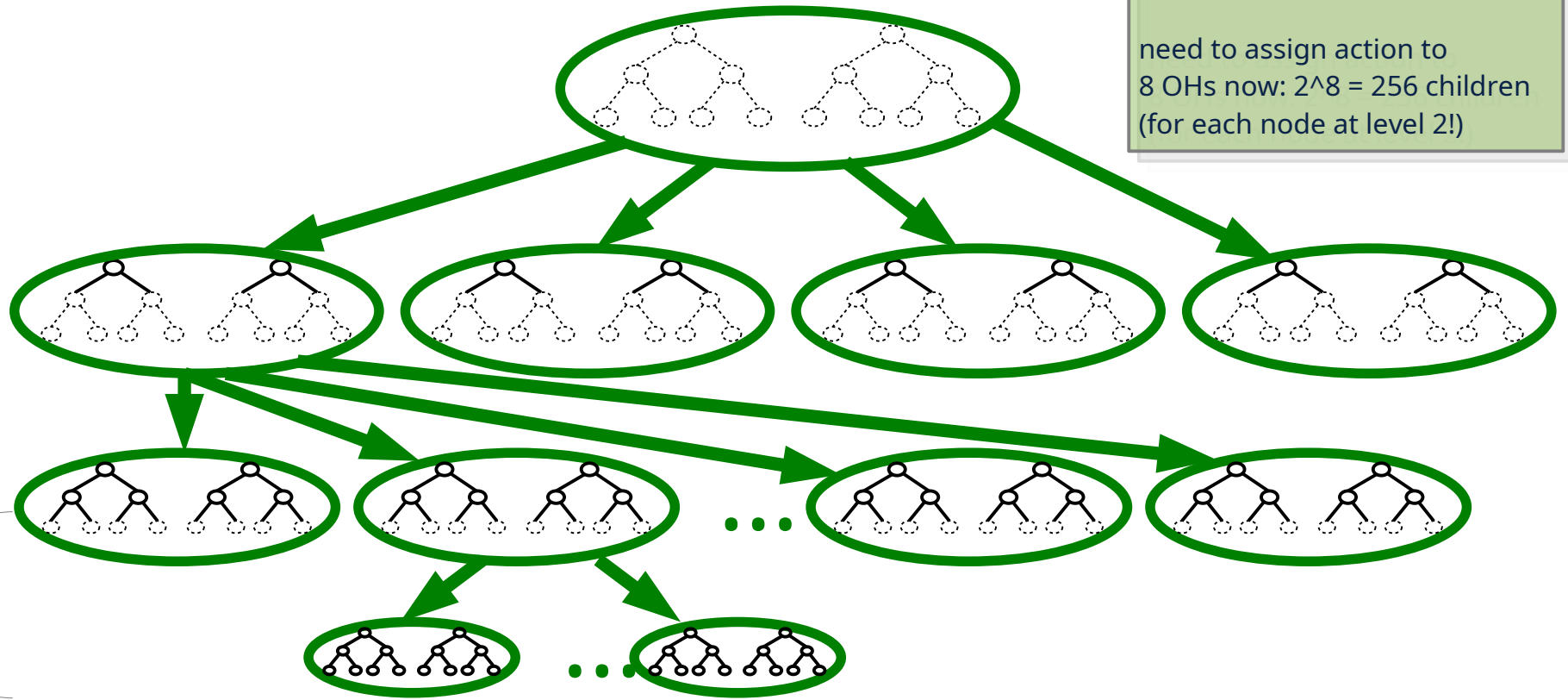


Heuristic Search – 2

- Creating **ALL** joint policies → tree structure!

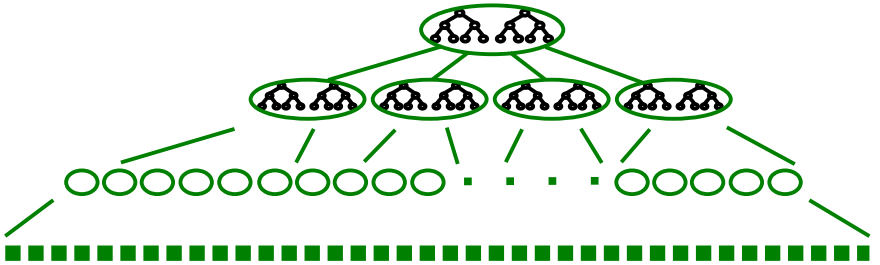
Last stage: even more!

need to assign action to
8 OHs now: 2^8 = 256 children
(for each node at level 2!)



Heuristic Search – 3

- Too big to create completely…
- Idea: use **heuristics**
  - avoid going down non-promising branches!



- Apply A* → **Multiagent A*** [Szer et al. 2005]

- Techniques for further scaling [Oliehoek et al. 2013 JAIR]:
  - lossless clustering of histories
  - Incremental expansion

How About Optimal Value Functions?

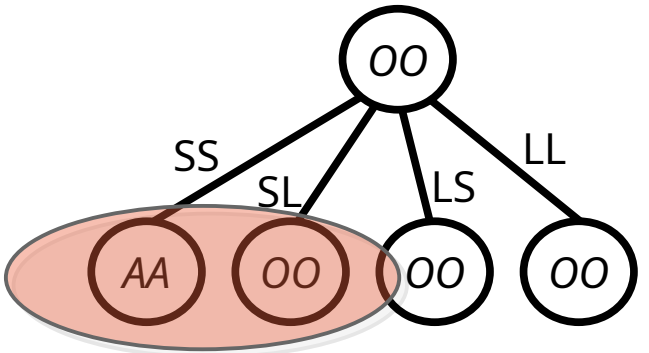
- We saw a value function **for a given joint (sub-tree) policy**
  …how about **optimal** value functions…?

- E.g., how about something like

  $Q^*(\mathbf{h},\mathbf{a}) = R(\mathbf{h},\mathbf{a}) + \Sigma_{\mathbf{o}} P(\mathbf{o}\,|\,\mathbf{h},\mathbf{a})\, V^*(\mathbf{h}'=\langle \mathbf{h},\mathbf{a},\mathbf{o}\rangle)$
  $V^*(\mathbf{h}') = \max_{\mathbf{a}'} Q^*(\mathbf{h}',\mathbf{a}')$

  ?

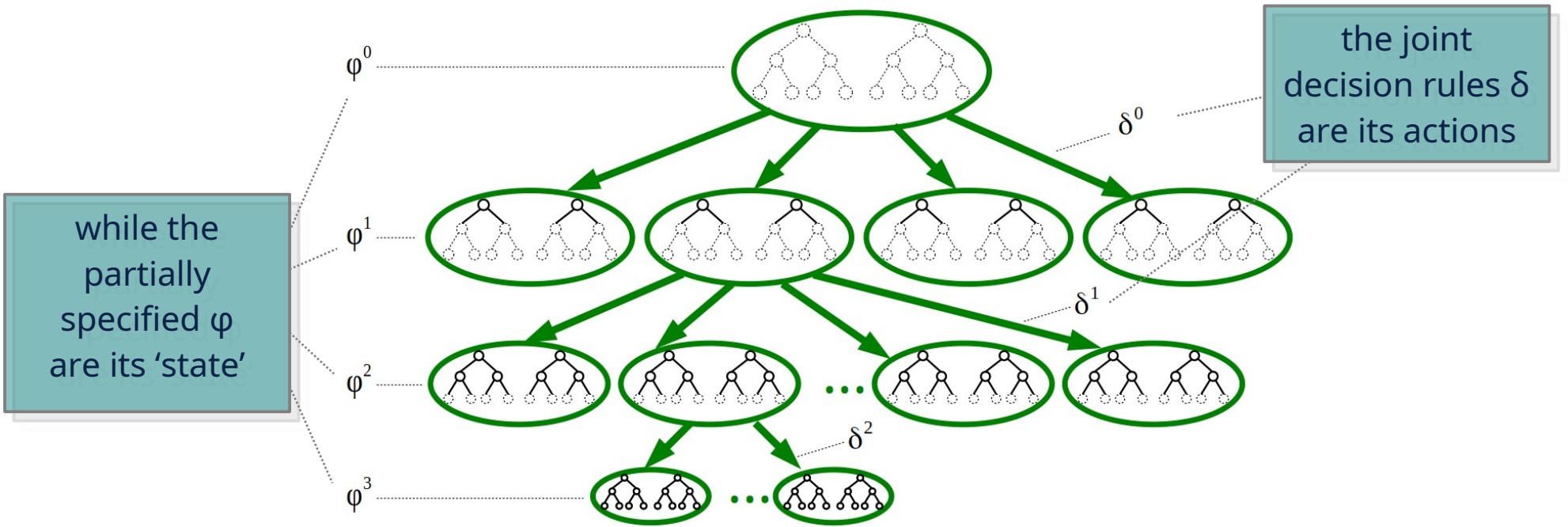
- A bit tricky…
  - would work for “multiagent POMDP”
  - but for not Dec-POMDPs:
    policies need to be decentralized!



not possible…!
requires agent 1 to select different actions, while it gets the same observation (‘S’)

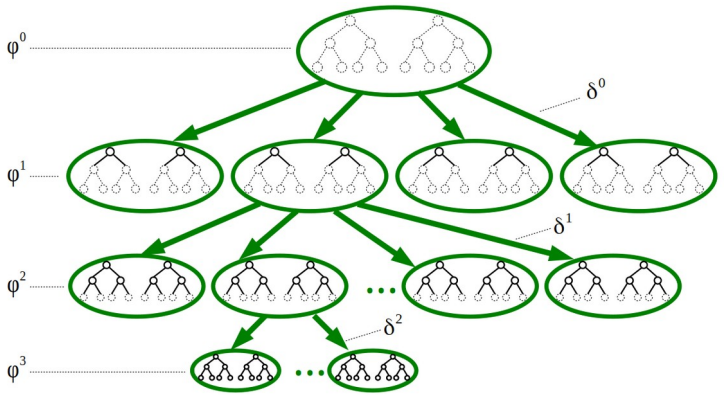
Reinterpreting the GMAA search tree

- Can view this as the decision making of the “planner”



Optimal value function of “Plan-time MDP”

- Leads to “Plan-time MDP”
  - states $\varphi_t = <\delta_0, \ldots, \delta_{t-1}>$
  - actions δ_t



- That has Bellman optimality equations:

$$Q^*(\varphi_t, \delta_t) = R(\varphi_t, \delta_t) + V^*(\varphi_{t+1} = <\varphi_t, \delta_t>)$$

$$V^*(\varphi_t) = \max_\delta Q^*(\varphi_t, \delta)$$

- With $R(\varphi_t, \delta_t)$ the expected reward at stage t:

$$R(\varphi_t, \delta_t) = \Sigma_s \Sigma_h P(s_t, h_t \mid \varphi_t) R(s_t, \delta_t(h_t))$$

(no bold to increase readability, but all entities are ‘joint’)

Of course, dependence on past joint policy φ_t is inconvenient…

→ instead: “plan-time statistics” $\sigma(s_t, h_t)$ to define value $V^*(\sigma_t)$

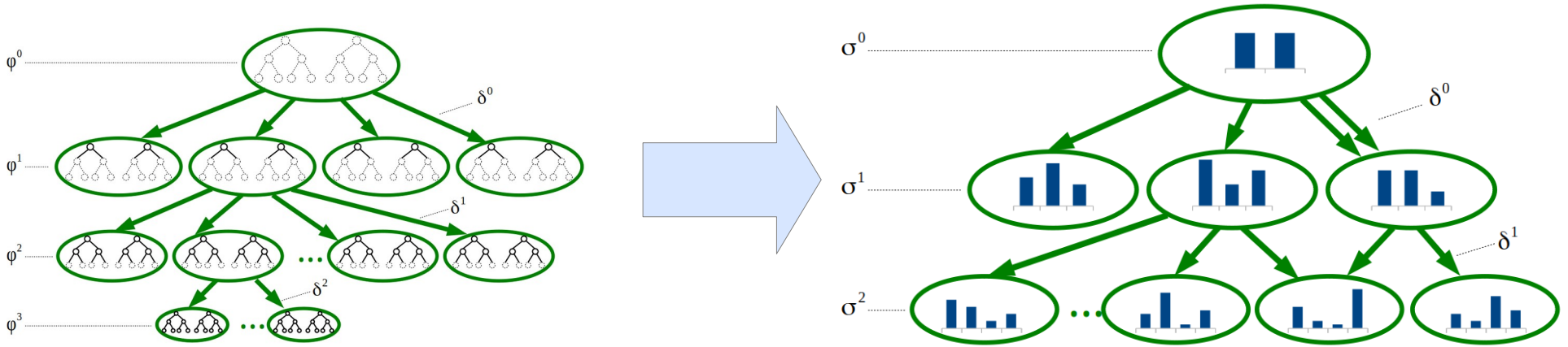
→ only need OHs: $\sigma(s_t, \overline{o}_t)$

So… what is relevant:

- ►the state
- ►and who knows what

The view with Plan-time statistics

- The σ_t allow for reuse
  [Oliehoek 2013 IJCAI]



- And it turns out that value function is PWLC on σ_t …
  [Dibangoye et al. 2013 IJCAI]

… so yes it is a (special case of) POMDP!

- A **plan-time non-observable MDP (NOMDP)**! [Oliehoek and Amato '14]
  - States: $\langle s_t, h_t \rangle$ (or $\langle s_t, \bar{o}_t \rangle$)
  - Actions: δ_t
  - Observations: NULL
  - In this NOMDP, the planner's belief is the "plan-time statistics" $\sigma(s_t, h_t)$

Terminology in decentralized control:

plan-time … =
"the designer's approach"

- Extend to deal with **common (i.e. shared) information** [Nayyar et al 2013]:
  → Plan-time POMDP
  - States: $\langle s_t, h_t \rangle$ ← h_t are joint histories of private information
  - Actions: δ_t
  - Observations: $\{o_{common}\}$
  - For each history of common observations h_{common} we get a different $\sigma(s_t, h_t)$
  - Select actions based on $\langle h_{common}, \sigma(s_t, h_t) \rangle$
  - E.g., used in MARL for Hanabi in "Bayesian action decoder" [Foerster et al. 2019]

"partial history sharing information structure"

Of course… histories still don’t scale

- E.g., for infinite horizon…? → do some compression on memory
- One option: finite-state controllers
  - each agent has information state I_i
  - and updates in some way: $I_i' = \iota(I_i, a_i, o)$
- Can incorporate in definition of Dec-POMDP problem…:

Definition 7 (ISA-Dec-POMDP). A *Dec-POMDP with information state abstraction (ISA-Dec-POMDP)* is a Dec-POMDP framework together with the specification of the sets $\{\mathcal{I}_i\}$ of information states (ISs).

For an ISA-Dec-POMDP, using the notation of the agent components defined above, there are two optimizations that need to be performed jointly:

1. the optimization of the joint action selection policy $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_n \rangle$, and
2. the optimization of the joint information state function $\iota = \langle \iota_1, \ldots, \iota_n \rangle$.

Oliehoek, Frans A., and Christopher Amato. "Dec-POMDPs as non-observable MDPs." (2014).

Plan-time sufficient statistics & NOMDP formulation

- Can extend these concepts…

Definition 8 (Plan-time ISA sufficient statistic)**.** The plan-time sufficient statistic for an ISA-Dec-POMDP is $\sigma_t(s,I) \triangleq \Pr(s,I|\boldsymbol{\delta}_0,\ldots,\boldsymbol{\delta}_{t-1})$.

Definition 9 (Plan-time ISA-NOMDP)**.** Given the internal state update functions, an ISA-Dec-POMDP can be converted to a *plan-time ISA-NOMDP* $\mathcal{M}_{PT-ISA-NOMDP}$ for a Dec-POMDP $\mathcal{M}$ is a tuple $\mathcal{M}_{PT-ISA-NOMDP}(\mathcal{M}) = \langle \check{\mathcal{S}},\check{\mathcal{A}},\check{T},\check{R},\check{\mathcal{O}},\check{O},\check{h},\check{b}_0 \rangle$, where:

- $\check{\mathcal{S}}$ is the set of augmented states, each $\check{s} = \langle s,I \rangle$.
- $\check{\mathcal{A}}$ is the set of actions, each $\check{a}$ corresponds to a joint decision rule $\boldsymbol{\delta}$ (which is a joint (stationary) action selection policy) in the ISA-Dec-POMDP.
- $\check{T}$ is the transition function:

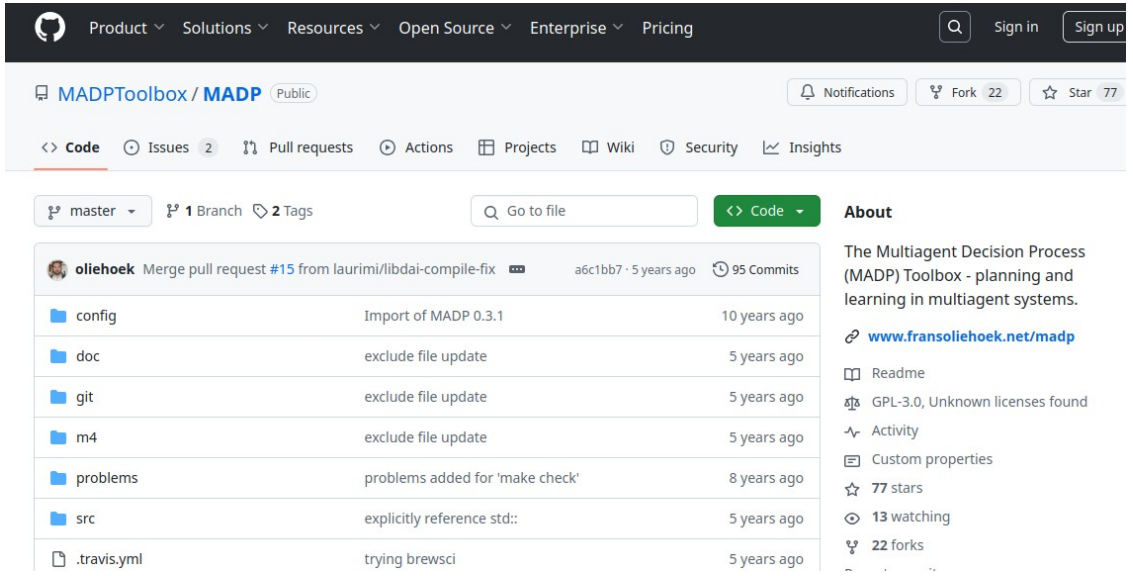
$$\begin{aligned}\check{T}(\langle s,I' \rangle \,|\, \langle s,I \rangle ,\boldsymbol{\delta}) &= \Pr(s',I'|s,\boldsymbol{a}=\boldsymbol{\delta}(I)) \\ &= \Pr(s'|s,\boldsymbol{a}) \sum_{\boldsymbol{o}} \iota(I'|I,\boldsymbol{a},\boldsymbol{o}) \Pr(\boldsymbol{o}|\boldsymbol{a},s')\end{aligned}$$

- $\check{R}$ is the reward function: $\check{R}(\langle s,I \rangle ,\boldsymbol{\delta}) = R(s,\boldsymbol{\delta}(I))$.
- $\check{\mathcal{O}} = \{NULL\}$ is the observation set which only contains the $NULL$ observation.
- $\check{O}$ is the observation function that specifies that $NULL$ is received with probability 1 (irrespective of the state and action).

For more info:
Oliehoek & Amato. "Dec-POMDPs as non-observable MDPs." (2014)

Try out some Dec-POMDPs?

- Try it out!




```
#include "ProblemDecTiger.h"
#include "JESPExhaustivePlanner.h"
int main()
{
    ProblemDecTiger dectiger;
    JESPExhaustivePlanner jesp(3,&dectiger);
    jesp.Plan();
    cout << jesp.GetExpectedReward() << endl;
    cout << jesp.GetJointPolicy()->SoftPrint() << endl;
    return(0);
}
```

https://github.com/MADPToolbox/MADP

Part 1d: Multiagent RL through the lens of Dec-POMDPs

Non-stationarity revisited…

- So why are we talking about Dec-POMDPs…?
  → **gives us the tools to formalize ‘non-stationarity’**

  - before… “other agent changes”
  - but could not specify ***how***…
    - the other agent changes due to individual observations
    - but individual learners do not represent those…!

Non-statio

- So why are v
  → **gives us t**
  - before... "o

But my other agents are Q-learners...?

- they can be seen as policies $\pi_i(a_i|h_i)$!

In particular, let's think of π_i as an individual Q-learning agent... it receives its own observations and remembers its own actions in $\vec{h}_i^t$, and uses this for updates and action selection:

$$Q_i(o_i^{t-1},a_i^{t-1}) \leftarrow (1-\alpha)Q_i(o_i^{t-1},a_i^{t-1}) + \alpha(r + \gamma \max_{a_i} Q(o_i^t,a_i))$$

$$\pi_i(a_i|\vec{h}_i^t) = \text{epsilon-greedy}(Q(o_i^t,\cdot),a_i,\epsilon)$$

So, even though it does not need to remember $\vec{h}_i^t$, each $\vec{h}_i^t$ deterministically induces an updated Q_i and thus action selection probabilities.

Example: predicting rewards

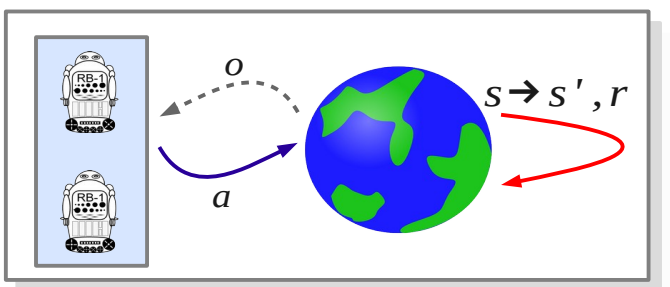
Let's predict the reward for agent *i* in a Dec-POMDP, given π_{-i}

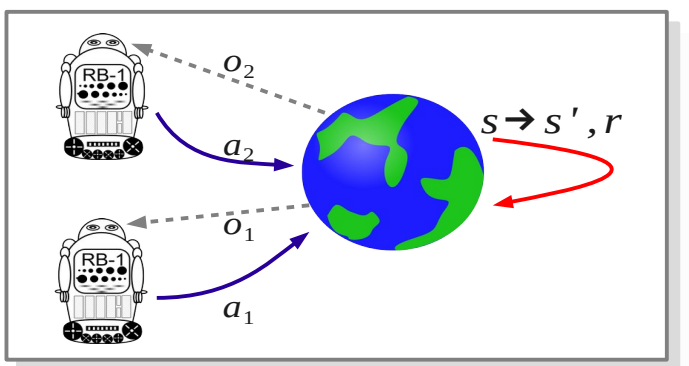
$$R_i(\vec{h}_i^t,a_i) = \sum_{s^t,\vec{\boldsymbol{h}}_{-i}^t} \Pr(s^t,\vec{\boldsymbol{h}}_{-i}^t|\vec{h}_i^t) \sum_{\boldsymbol{a}_{-i}} \boldsymbol{\pi}_{-i}(\boldsymbol{a}_{-i}|\vec{\boldsymbol{h}}_{-i}^t) R_i(s^t,a_i,\boldsymbol{a}_{-i})$$

with

- $\boldsymbol{\pi}_{-i}(\boldsymbol{a}_{-i}|\vec{\boldsymbol{h}}_{-i}^t) = \pi_1(a_1^t|\vec{h}_1^t) \times \cdots \times \pi_n(a_n^t|\vec{h}_n^t)$ the application of the policies of the other agents
- $\Pr(s^t,\vec{\boldsymbol{h}}_{-i}^t|\vec{h}_i^t)$ the belief over both states and AOHs of other agents, given our own AOH $\vec{h}_i^t$. This can be computed, given $\boldsymbol{\pi}_{-i}$, in a similar way as the normal belief update in a POMDP.

MARL: an objective perspective



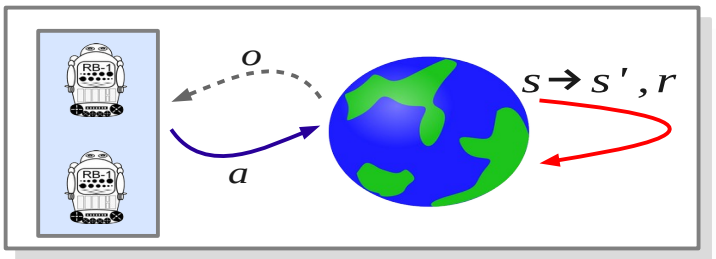


Objective perspective

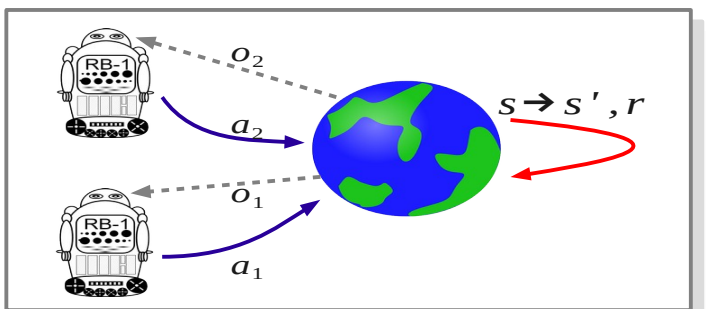
- Objective perspective:
  reason for the entire team

- Two approaches to decision making:

  - centralized: “puppeteer”



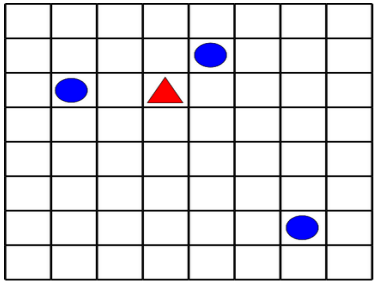
  - decentralized



Centralized Objective Approach

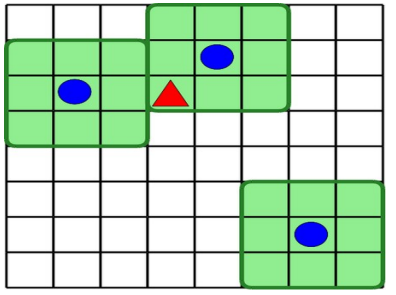
- Centralize all information (communication or fully observable)

centralized problem is...



fully observable:
multiagent MDP

(standard) RL methods



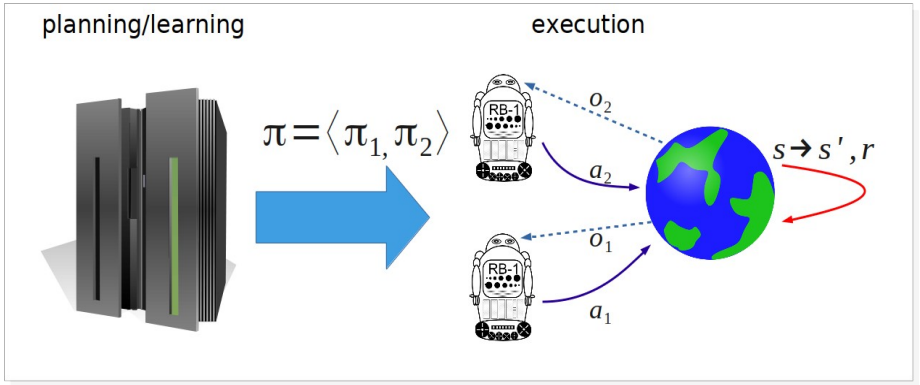
partially observable:
multiagent POMDP

PORL methods

- Example: joint Q-learning!
- Overcoming the scalability hurdle is topic of much research.

Decentralized Objective Approach
("Dec-POMDP RL")

- Learn for all agents in the team
- Truly decentralized **execution**
- **optionally**: offline training phase
  - CTDE (with centralized components)



- Based on all kinds of RL techniques:
  - value-based (e.g., Q-learning)
  - policy search
  - actor-critic

Policy search methods:
- ▶no dependence on Markov property
- ▶policy gradient
  - can be decentralized [Peshkin et al. 2000]
  - has convergence guarantees
  - many recent deep versions

Policy gradient for partially observable RL

- Let's look at policy gradient ("REINFORCE") in P.O. settings (e.g., POMDPs)

- history: $h_t = (o_0, a_0, \ldots, o_{t-1}, a_{t-1}, o_t)$
- value:

$$V(\pi_\theta) = \sum_{t=0}^{T-1} \sum_{h_t \in \mathcal{H}_t} \Pr(h_t|\pi_\theta) \sum_{a_t} \pi_\theta(a_t|h_t) R(h_t, a_t)$$

Cf. the value of a joint policy we saw before:

$$V(\boldsymbol{\pi}) = \sum_{t=0}^{T-1} \sum_{\boldsymbol{h}_t \in \boldsymbol{\mathcal{H}}_t} \Pr(\boldsymbol{h}_t|\boldsymbol{\pi}, b_0) \sum_{\boldsymbol{a}_t \in \boldsymbol{\mathcal{A}}} R(\boldsymbol{h}_t, \boldsymbol{a}_t) \boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{h}_t),$$

Derive gradient...

- Gradient:

$$\nabla_\theta V(\pi_\theta) = \nabla_\theta \left[\sum_{t=0}^{T-1} \sum_{h_t \in \mathcal{H}_t} \Pr(h_t|\pi_\theta) \sum_{a_t} \pi_\theta(a_t|h_t) R(h_t,a_t) \right]$$

$$\{\text{next slide}\} = \sum_{t=0}^{T-1} \sum_{h_t,a_t} \Pr(h_t,a_t|\pi_\theta) R(h_t,a_t) \nabla_\theta \log \Pr(h_t,a_t|\pi_\theta)$$

Derive gradient...

- Gradient:

$$\nabla_\theta V(\pi_\theta) = \nabla_\theta \left[\sum_{t=0}^{T-1} \sum_{h_t \in \mathcal{H}_t} \Pr(h_t|\pi_\theta) \sum_{a_t} \pi_\theta(a_t|h_t) R(h_t,a_t) \right]$$

$$\{\text{next slide}\} = \sum_{t=0}^{T-1} \sum_{h_t,a_t} \Pr(h_t,a_t|\pi_\theta) R(h_t,a_t) \nabla_\theta \log \Pr(h_t,a_t|\pi_\theta)$$

- with $\nabla_\theta \log \Pr(h_t,a_t|\pi_\theta)$

$$= \nabla_\theta \log \left[\Pr(a_t|h_t;\pi_\theta) \left[\prod_{k=0}^{t-1} \Pr(a_k|h_k;\pi_\theta) \Pr(o_{k+1}|h_k,a_k) \right] P(o_0) \right]$$

$$= \nabla_\theta \left[\log P(o_0) + \sum_{k=0}^{t} \log \Pr(a_k|h_k;\pi_\theta) + \sum_{k=0}^{t-1} \log \Pr(o_{k+1}|h_k,a_k) \right]$$

$$\{\text{only middle term depends on } \theta\} = \sum_{k=0}^{t} \nabla_\theta \log \Pr(a_k|h_k;\pi_\theta)$$

(Full deriv.)

- for completeness:

$$\nabla_\theta V(\pi_\theta) = \nabla_\theta \left[\sum_{t=0}^{T-1} \sum_{h_t \in \mathcal{H}_t} \Pr(h_t|\pi_\theta) \sum_{a_t} \pi_\theta(a_t|h_t) R(h_t,a_t) \right]$$

$$= \sum_{t=0}^{T-1} \nabla_\theta \underbrace{\sum_{h_t \in \mathcal{H}_t} \Pr(h_t|\pi_\theta) \sum_{a_t} \pi_\theta(a_t|h_t) R(h_t,a_t)}_{R_t(\theta)}$$

$$= \sum_{t=0}^{T-1} \sum_{h_t \in \mathcal{H}_t} \sum_{a_t} \nabla_\theta \left[\Pr(h_t|\pi_\theta) \pi_\theta(a_t|h_t) R(h_t,a_t) \right]$$

$$\{\text{prod. rule:}\} = \sum_{t=0}^{T-1} \sum_{h_t \in \mathcal{H}_t} \sum_{a_t} \left[\nabla_\theta \left(\Pr(h_t|\pi_\theta) \pi_\theta(a_t|h_t) \right) R(h_t,a_t) + \Pr(h_t|\pi_\theta) \pi_\theta(a_t|h_t) \nabla_\theta R(h_t,a_t) \right]$$

$$= \sum_{t=0}^{T-1} \sum_{h_t,a_t} \left[\left(\nabla_\theta \Pr(h_t,a_t|\pi_\theta) \right) R(h_t,a_t) + \Pr(h_t|\pi_\theta) \pi_\theta(a_t|h_t) \cdot 0 \right]$$

$$= \sum_{t=0}^{T-1} \sum_{h_t,a_t} \left(\nabla_\theta \Pr(h_t,a_t|\pi_\theta) \right) R(h_t,a_t)$$

$$= \sum_{t=0}^{T-1} \sum_{h_t,a_t} \Pr(h_t,a_t|\pi_\theta) \left(\frac{\nabla_\theta \Pr(h_t,a_t|\pi_\theta)}{\Pr(h_t,a_t|\pi_\theta)} \right) R(h_t,a_t)$$

$$\{\text{log trick:}\} = \sum_{t=0}^{T-1} \sum_{h_t,a_t} \Pr(h_t,a_t|\pi_\theta) R(h_t,a_t) \nabla_\theta \log \Pr(h_t,a_t|\pi_\theta)$$

Multiagent PG

- Single agent:

$$\nabla_\theta V(\pi_\theta) = \sum_{t=0}^{T-1} \sum_{h_t, a_t} \Pr(h_t, a_t | \pi_\theta) R(h_t, a_t) \sum_{k=0}^{t} \nabla_\theta \log \Pr(a_k | h_k; \pi_\theta)$$

- Multiagent:

$$\nabla_{\boldsymbol{\theta}} V(\boldsymbol{\pi_\theta}) = \sum_{t=0}^{T-1} \sum_{\boldsymbol{h}_t, \boldsymbol{a}_t} \Pr(\boldsymbol{h}_t, \boldsymbol{a}_t | \boldsymbol{\pi_\theta}) R(\boldsymbol{h}_t, \boldsymbol{a}_t) \sum_{k=0}^{t} \nabla_{\boldsymbol{\theta}} \log \Pr(\boldsymbol{a}_k | \boldsymbol{h}_k; \boldsymbol{\theta})$$

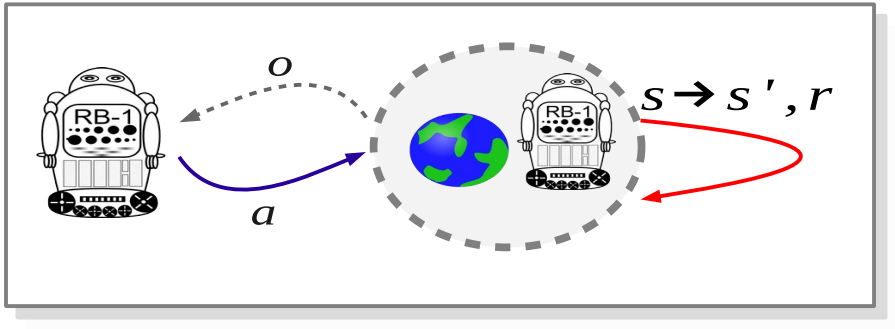
$$\{\text{policy is decentralized}\} = \sum_{t=0}^{T-1} \sum_{\boldsymbol{h}_t, \boldsymbol{a}_t} \Pr(\boldsymbol{h}_t, \boldsymbol{a}_t | \boldsymbol{\pi_\theta}) R(\boldsymbol{h}_t, \boldsymbol{a}_t) \sum_{k=0}^{t} \nabla_{\boldsymbol{\theta}} \log \prod_{i=1}^{n} \Pr(a_{i,k} | h_{i,k}; \theta_i)$$

$$\{\text{gradient is per agent}\} = \sum_{t=0}^{T-1} \sum_{\boldsymbol{h}_t, \boldsymbol{a}_t} \Pr(\boldsymbol{h}_t, \boldsymbol{a}_t | \boldsymbol{\pi_\theta}) R(\boldsymbol{h}_t, \boldsymbol{a}_t) \sum_{i=1}^{n} \sum_{k=0}^{t} \nabla_{\theta_i} \log \Pr(a_{i,k} | h_{i,k}; \theta_i)$$

Peshkin L, Kim KE, Meuleau N, Kaelbling LP. Learning to cooperate via policy search. In Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence 2000 Jun 30 (pp. 489-496). **IFAAMAS Influential paper award 2024**

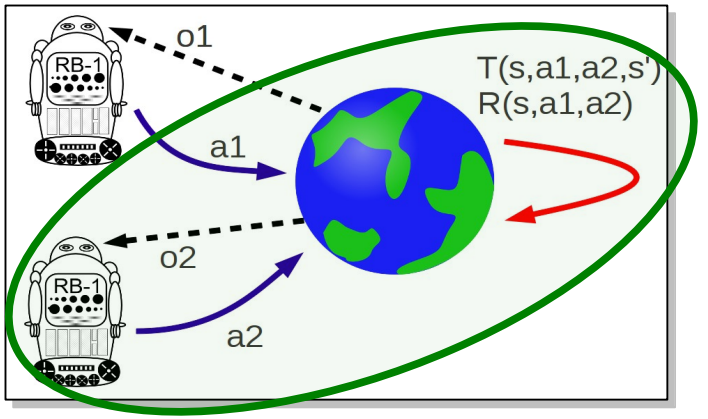
Upshot:
convergent coop. MARL that can be decentralized!
(only return needs to be observed)

MARL: a subjective perspective



Subjective Perspective

- So now let's further formalize…
  - Other agents: part of the environment
  - **Best response model** = multiagent environment + models of other agents

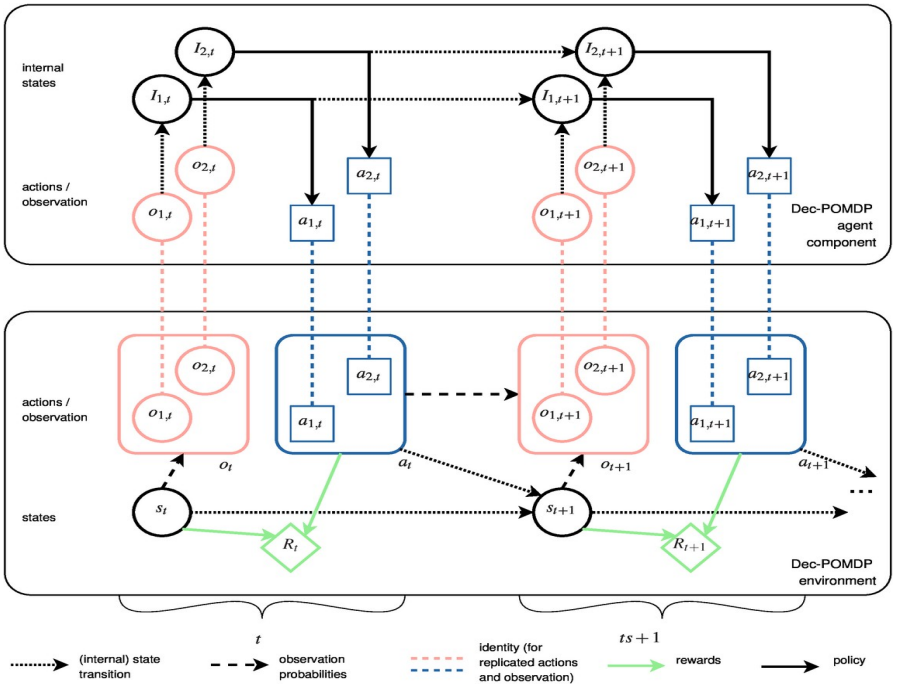


Frans A. Oliehoek and Christopher Amato. Best Response Bayesian Reinforcement Learning for Multiagent Systems with State Uncertainty. In Proceedings of the Ninth AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM), 2014.

Models & Environment

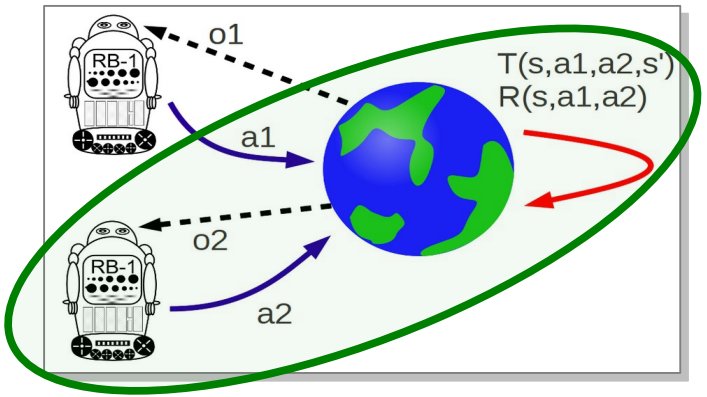
- Multiagent **environment**
  - $MEA_i = \langle S, \{A_j\}, \{O_j\}, T, O, R_i \rangle$

- **Models of other agents**
  - think: finite state controllers
  - $m_j = \langle A_j, O_j, IS_j, \pi_j, \beta_j, I_j \rangle$
  - but… very general!
    - no real restrictions on internal states or functions
    - i.e., policy and belief update can be computational procedures
    - so includes MDPs, POMDPs, etc.



Best-response model (BRM)

- Formalized as a (non-standard) POMDP



- States: $\bar{s}_i = \langle s, I_{-i} \rangle$
- Actions: a_i
- Observations: o_i
- Dynamics function:

$$\bar{D}_i(\bar{s}_i', o_i | \bar{s}_i, a_i) = \sum_{a_{-i}} \sum_{o_{-i}} T(s'|s,a) O(o|a,s') \beta_{-i}(I_{-i}'|I_{-i}, a_{-i}, o_{-i}) \pi_{-i}(a_{-i}|I_{-i})$$

- Rewards:

$$\bar{R}_i(\bar{s}_i, a_i) = \sum_{a_{-i}} R_i(s,a) \pi_{-i}(a_{-i}|I_{-i})$$

Planning / Learning for BRMs

- Since a BRM is a POMDP…
  - If you have all the models → use POMDP solver
  - otherwise → POMDP RL
    - yes… difficult… (but all methods apply)

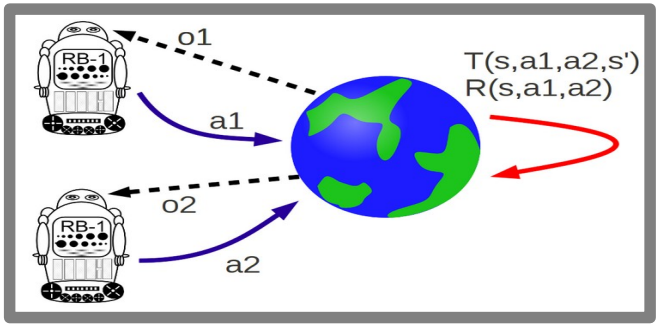
- Open question: **is RL for a BRM easier (or more difficult) than other POMDP RL?**

  - easier to have priors of ‘sane’ behavior for agents (rather than other environmental aspects) ?
  - but if other agents are learning…
    …continual extrapolation problem?

Our final agent solving the *VizDoom: Take Cover* environment. [33, 34]

https://worldmodels.github.io/

Summary Part 1



- MARL is complex…! (and interesting!)
- Two initial ideas…
  - fully centralized → issues with scalability (action spaces, communication)
  - fully decentralized → issues with convergence
- Dec-POMDP: framework that **allows reasoning over private observations**
- Fundaments of multiagent planning
  - 2 generals: need to also factor in ‘predictability’
  - Heuristic search: past joint policy matters
    → because it determines the **distribution over states and knowledge**
  - Enables formulation of **“plan-time models”** or **“designer’s approach”**
- Dec-POMDP perspective to MARL:
  - helps to understand ‘non-stationarity’
  - objective approach: policy gradient still works (since value of a given joint policy is analogue to a POMDP)
  - subjective approach: formalize a “best-response model” → RL in difficult POMDPs

References

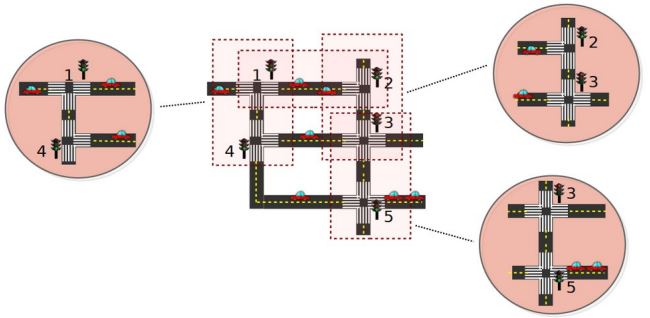
- Most references are in:
  - Frans A. Oliehoek and Christopher Amato. **A Concise Introduction to Decentralized POMDPs**, SpringerBriefs in Intelligent Systems, Springer, May 2016.

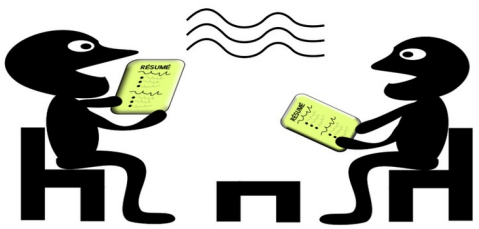
- Some more details:
  - Frans A. Oliehoek. **Decentralized POMDPs**. In Wiering, Marco and van Otterlo, Martijn, editors, *Reinforcement Learning: State of the Art*, Adaptation, Learning, and Optimization, pp. 471–503, Springer Berlin Heidelberg, Berlin, Germany, 2012.
  - Frans A. Oliehoek and Christopher Amato. **Dec-POMDPs as Non-Observable MDPs**. IAS technical report IAS-UVA-14-01, Intelligent Systems Lab, University of Amsterdam, 2014.

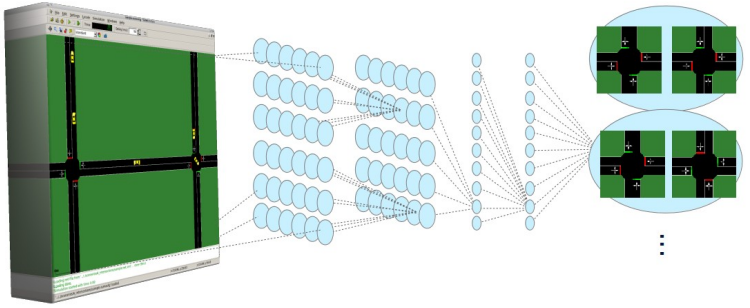
Bonus: Further MARL Topics

Further Topics in MARL

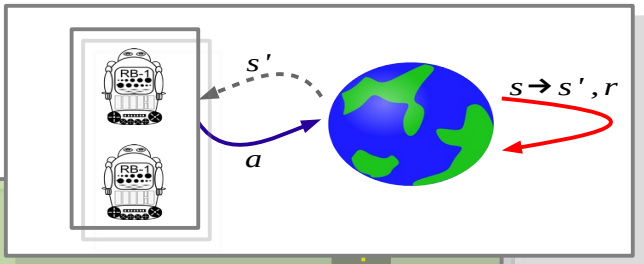
- Multiagent (reinforcement) learning is active topic of research
  - scaling up
  - deep learning
  - learning to communicate
  - multiagent approaches to ML (e.g., GANs)
  - ad-hoc teamwork: coordination without training

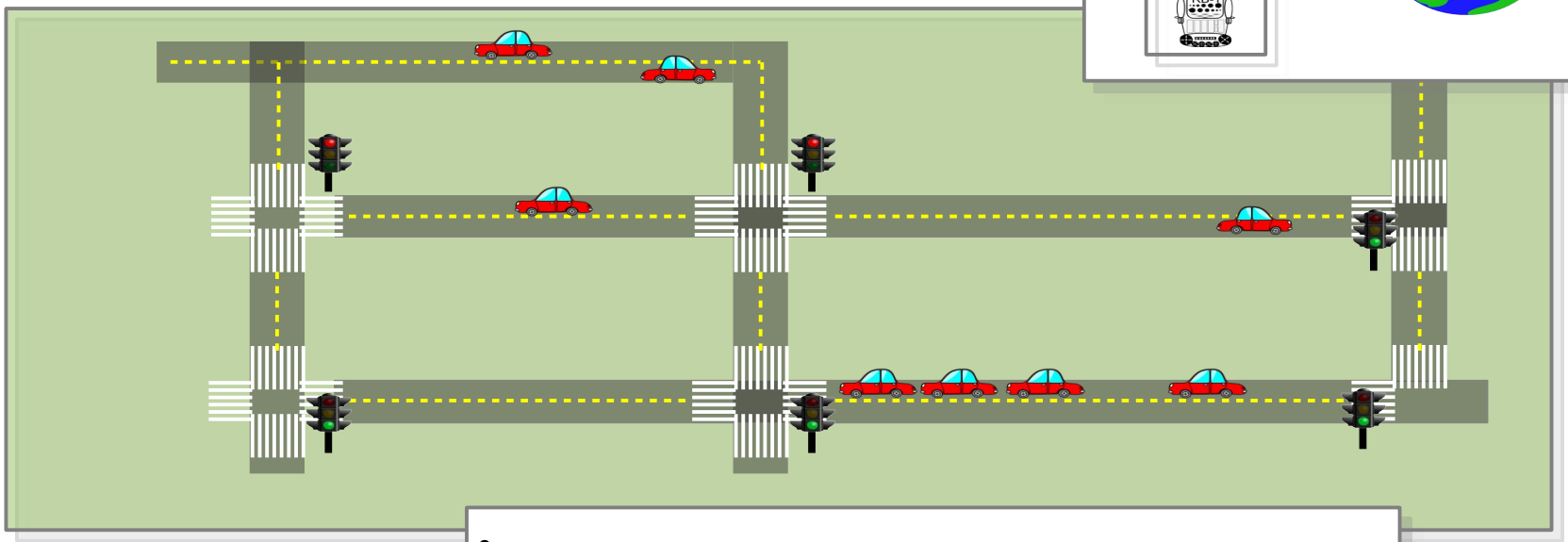






Scaling multiagent (PO)MDPs

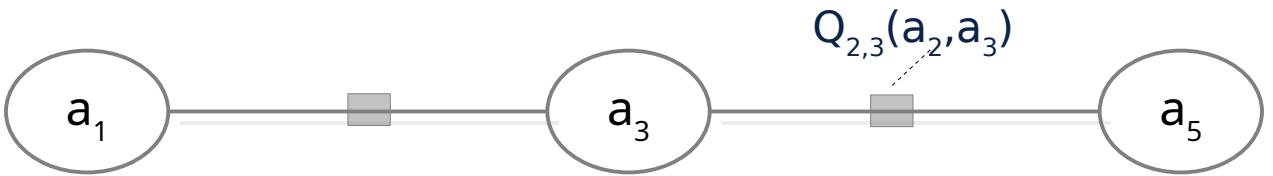




-
  - Treat as single 'puppeteer' agent
  - Main problem:
- **exponentially many joint actions**
-

Scaling multiagent (PO)MDPs

- E.g., even in stateless setting: Q(**a**) too large…
- **Coordination graphs** [Guestrin et al. NIPS 2001]
  - address by factorizing…
  - E.g., $Q(\mathbf{a}) \approx Q_{1,2}(a_1,a_2) + Q_{2,3}(a_2,a_3)$



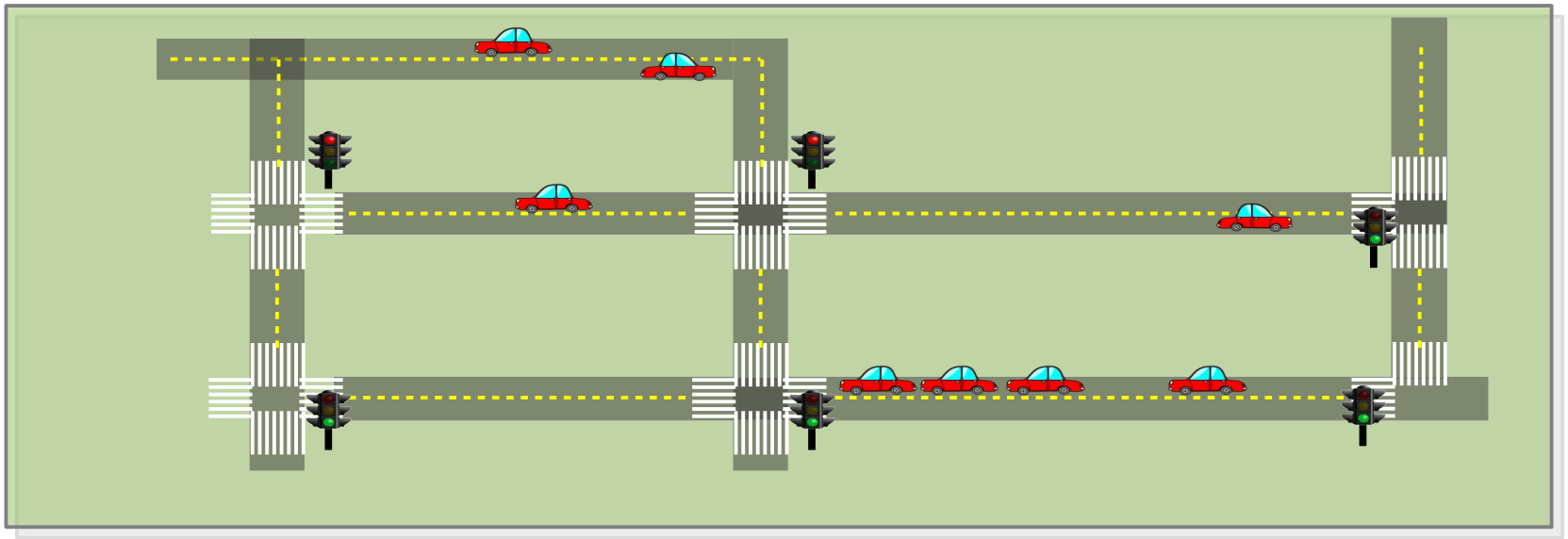
- Benefits:
  - compact representation
  - can optimize:

$$\max_{\mathbf{a}} [Q_{1,2}(a_1,a_2) + Q_{2,3}(a_2,a_3)]$$

    - with inference or COP techniques (variable elimination, max-plus, etc.)

Scaling multiagent (PO)MDPs

- back to sequential setting…

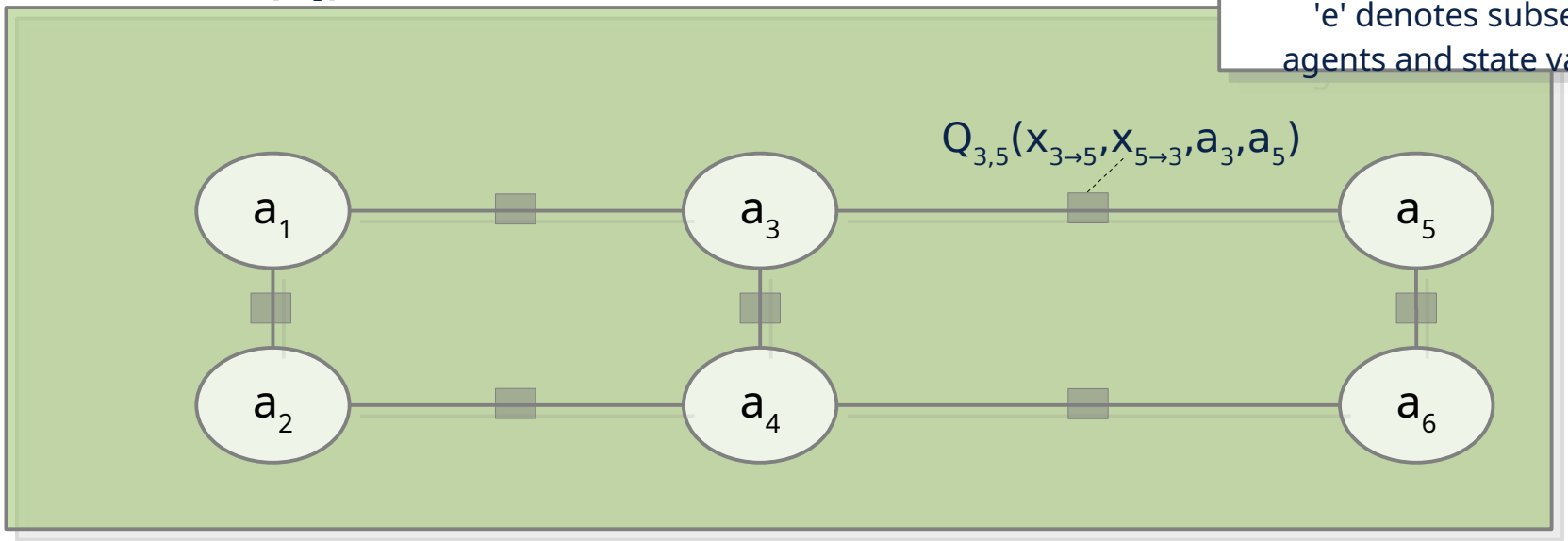


Factored Value Functions [Guestrin NIPS'01]

- Approximate with factored (Q)-value function

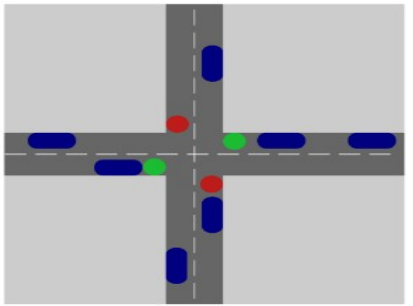
$$\hat{Q}(s,a) = \sum_e Q_e(x_e, a_e)$$

'e' denotes subsets of agents and state variables

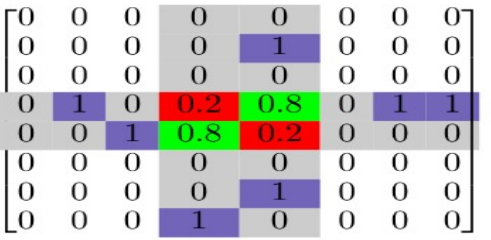


Large state spaces: deep MARL

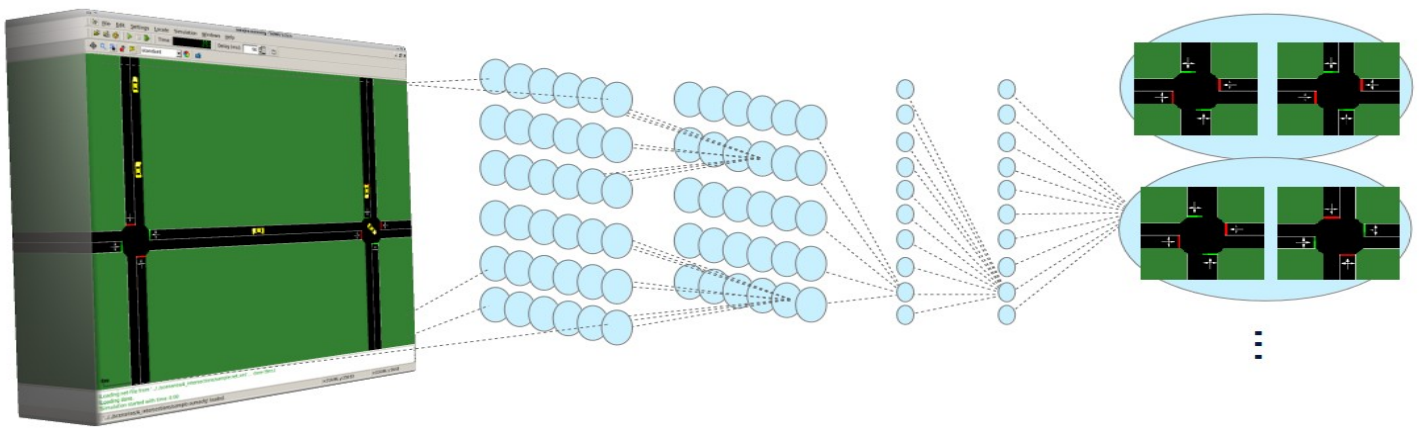
- Huge state spaces
  - even 2 intersection source problems are intractable
  - (even 1 intersection is…)
- Deep Q-learning
  - encode state as matrix



(b) Traffic situation



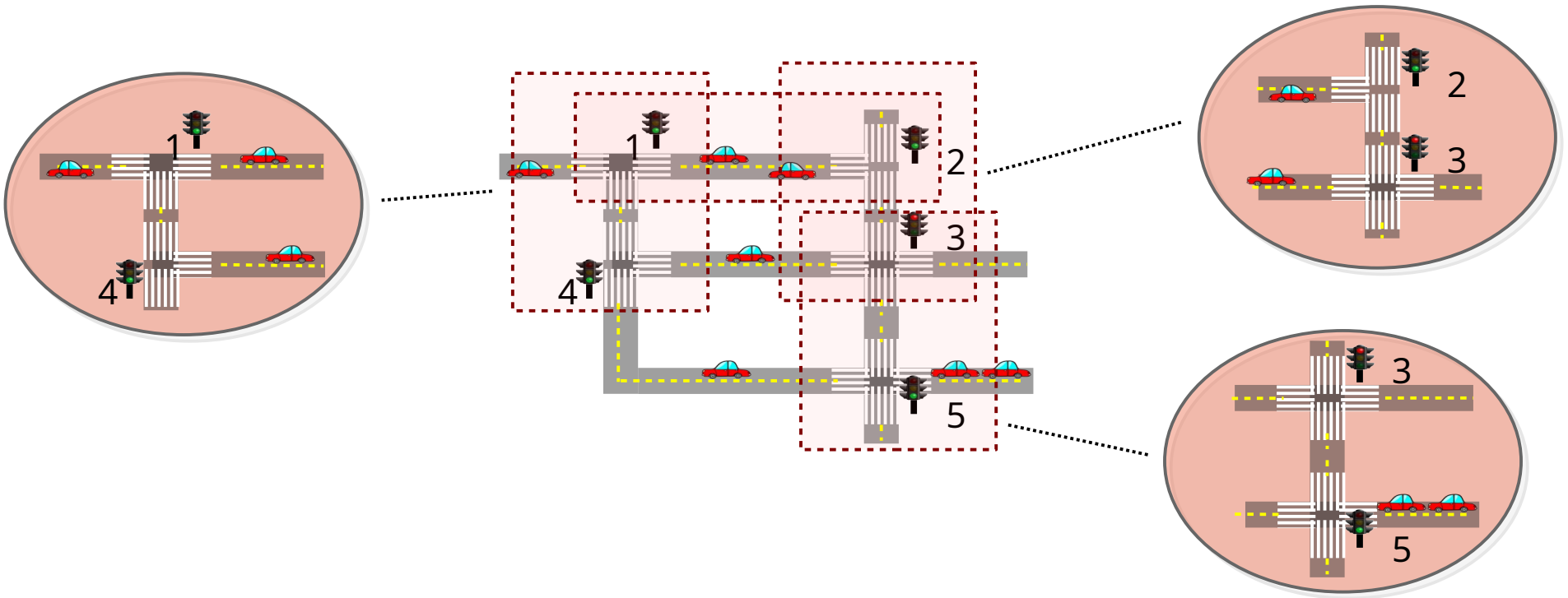
(c) Simplified example of state representation in 8×8 matrix.



Elise Van der Pol and Frans A. Oliehoek. Coordinated Deep Reinforcement Learners for Traffic Light Control. In NIPS'16 Workshop on Learning, Inference and Control of Multi-Agent Systems, 2016.

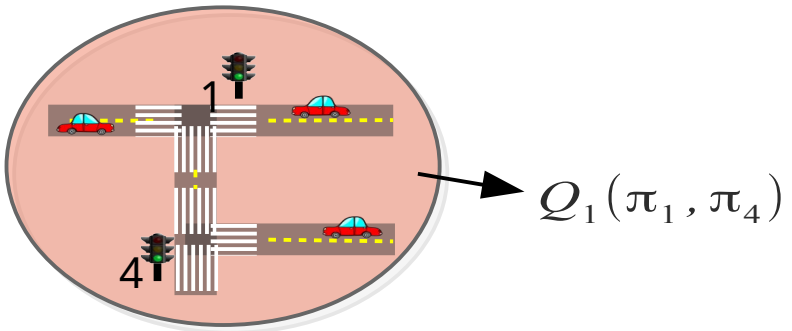
Scaling via “Transfer Planning”

- Define **source problem** for each Q-component

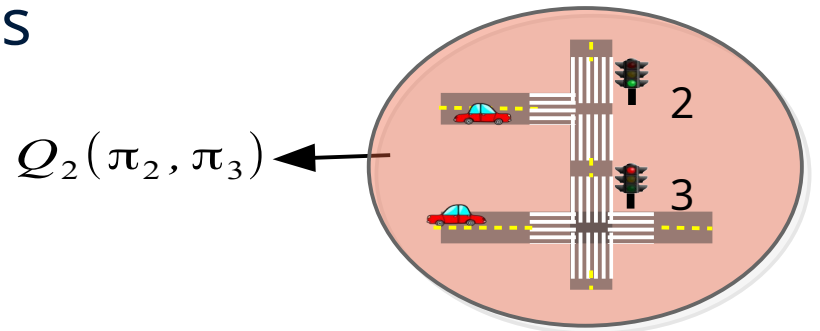


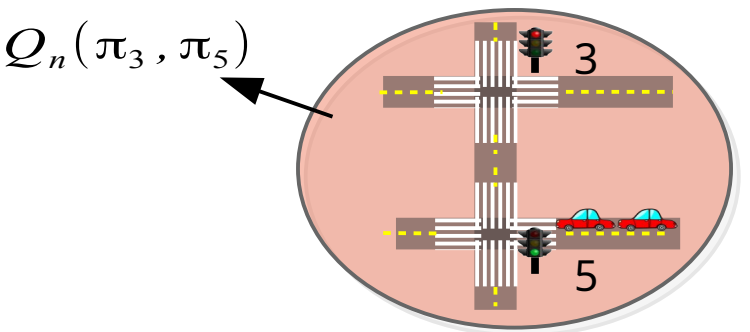
Scaling via “Transfer Planning”

- 'Solve' source problems independently
  - use Q-function as components



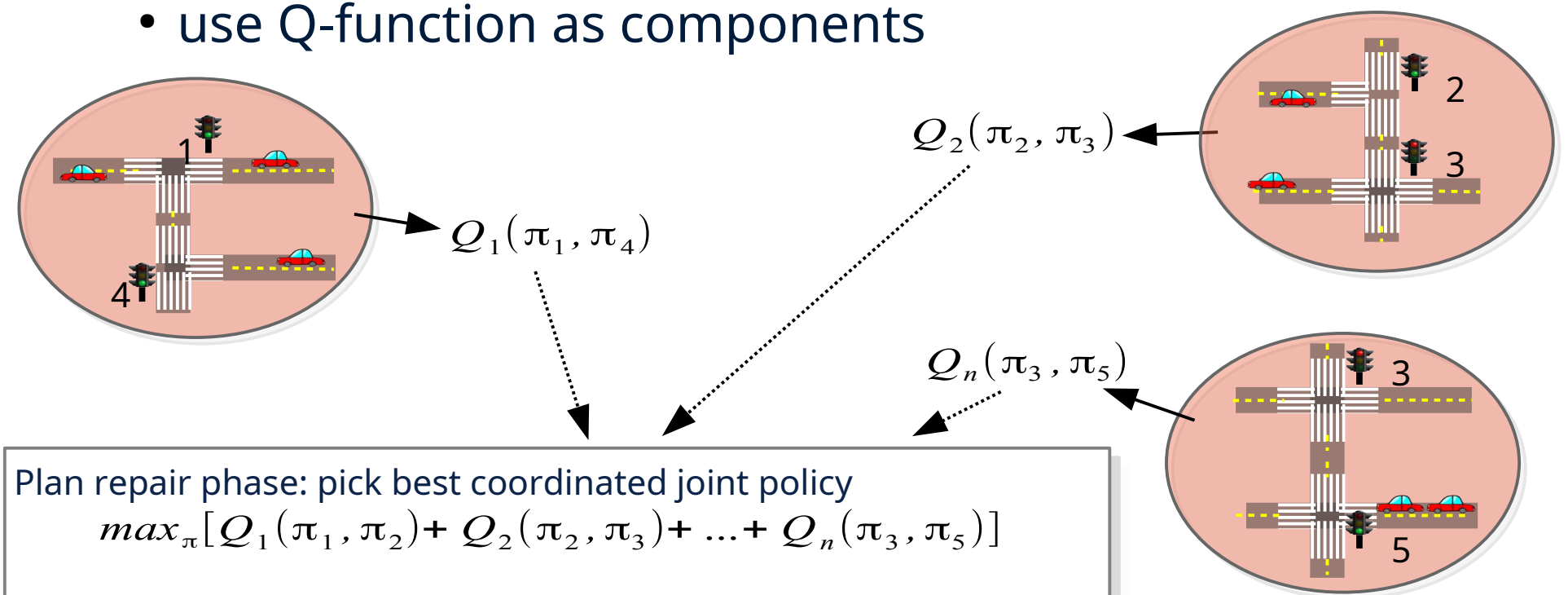
planning or learning





Scaling via “Transfer Planning”

- 'Solve' source problems independently
  - use Q-function as components



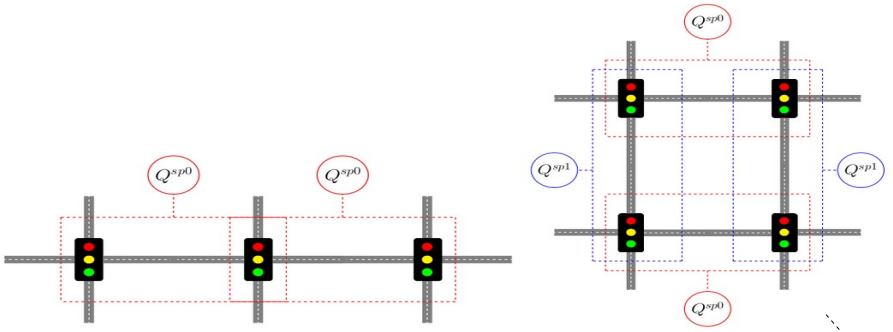
Plan repair phase: pick best coordinated joint policy

$$max_\pi[Q_1(\pi_1, \pi_2) + Q_2(\pi_2, \pi_3) + \ldots + Q_n(\pi_3, \pi_5)]$$

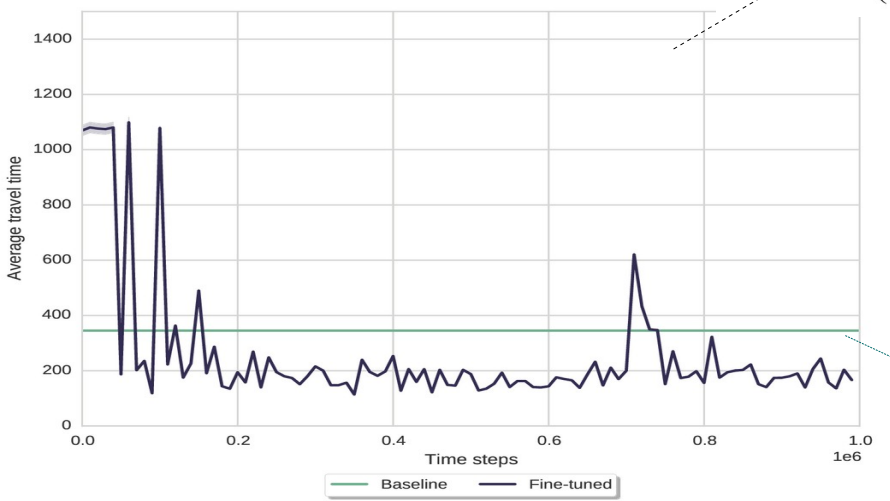
- optimization of a graphical model: inference
  - can use (adapt) variable elimination

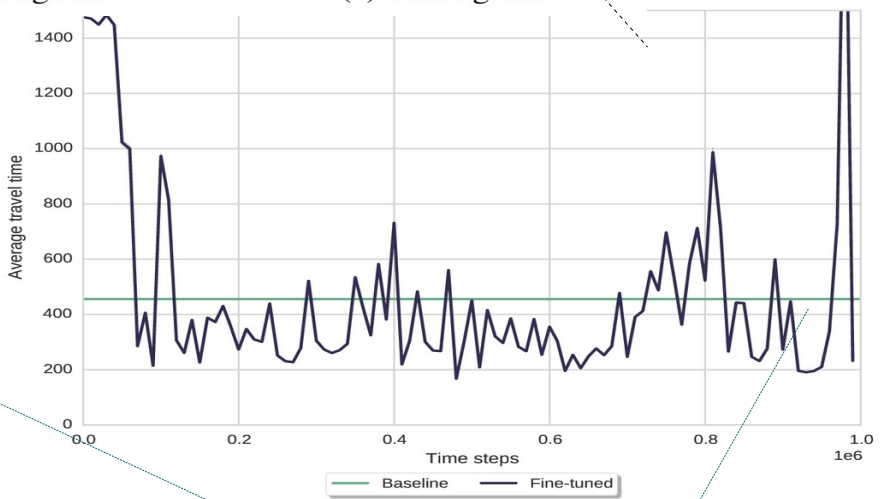
Empirical Results

- Results on SUMO traffic simulator
  [Van der Pol '16 NIPSWS]



(b) Three agents (c) Four agents





- Also: http://www.fransoliehoek.net/trafficvideo

Baseline: Kuyer et al. 08 ECML

Communication

- instantaneous, cost-free, and noise-free:
  - Dec-MDP → multiagent MDP (MMDP)
  - Dec-POMDP → multiagent POMDP (MPOMDP)

- but in practice:
  - probability of failure
  - delays
  - costs

- Also: implicit communication!
  (via observations and actions)

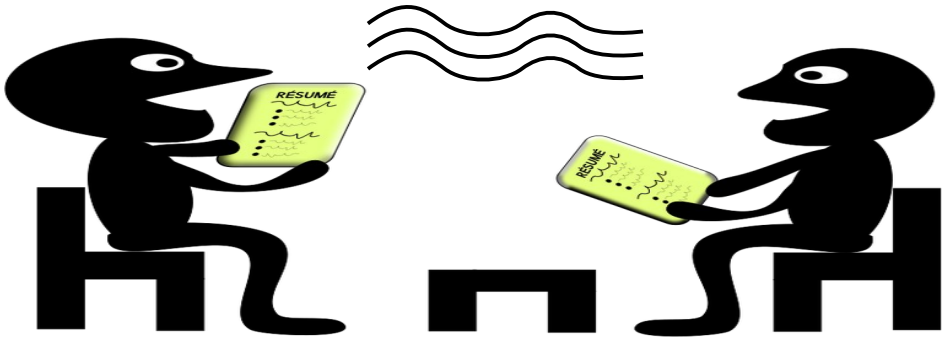
Explicit Communication

- perform a particular information update (e.g., sync) as in the MPOMDP:
  - each agent broadcasts its information, and
  - each agent uses that to perform joint belief update

- Other approaches:
  - Communication cost [Becker et al. 2005]
  - Delayed communication [Hsu et al. 1982, Spaan et al. 2008, Oliehoek & Spaan 2012]
  - Communicate every k stages [Goldman & Zilberstein 2008]

Implicit Communication

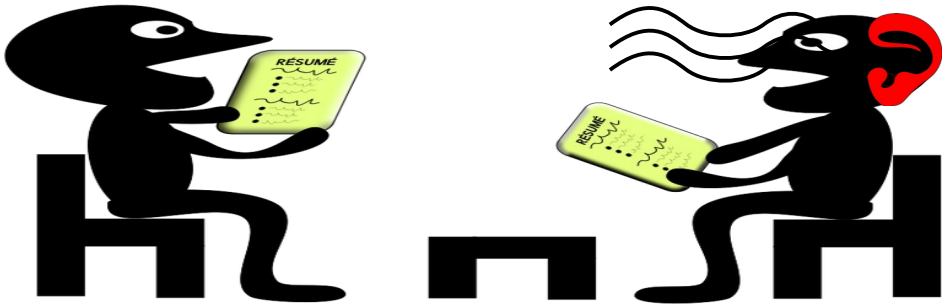
- Encode communications by actions and observations



- Embed the **optimal meaning** of messages by finding the optimal plan [Goldman and Zilberstein 2003, Spaan et al. 2006]

Implicit Communication

- Encode communications by actions and observations



- Embed the **optimal meaning** of messages by finding the optimal plan [Goldman and Zilberstein 2003, Spaan et al. 2006]

Implicit Communication

- Encode communications by actions and observations

- Embed the **optimal meaning** c
  plan [Goldman and Zilberstein 20

Me in 2014:

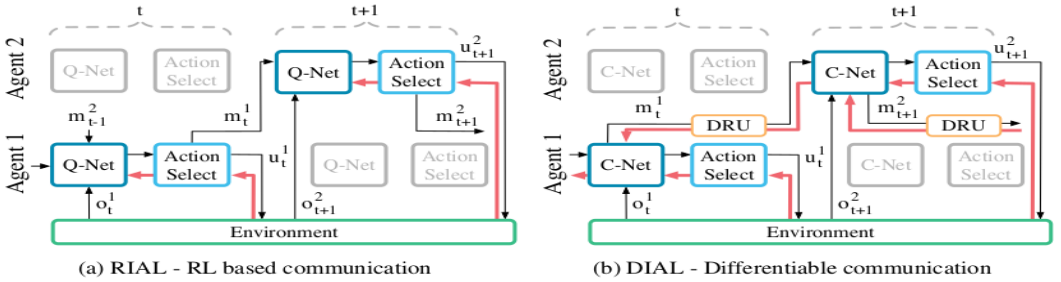
"E.g. communication bit doubles the #actions and observations!

Clearly, useful… but intractable for general settings (perhaps for analysis of very small communication systems)"

but then…

Deep learning of communication

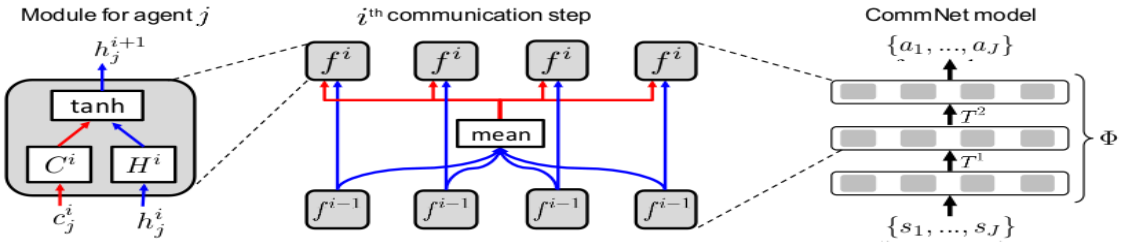
- …these scalability issues can be overcome (at least to some extend) by deep learning…



(a) RIAL - RL based communication

(b) DIAL - Differentiable communication

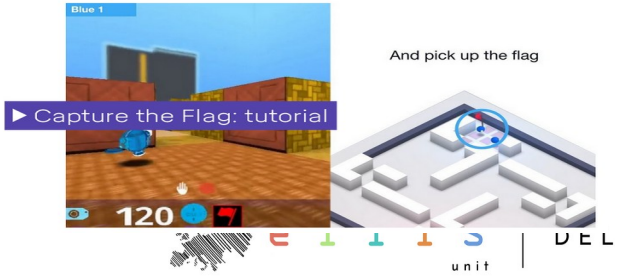
[Foerster et al. 2016]



CommNet [Sukhbaatar et al. 2016]

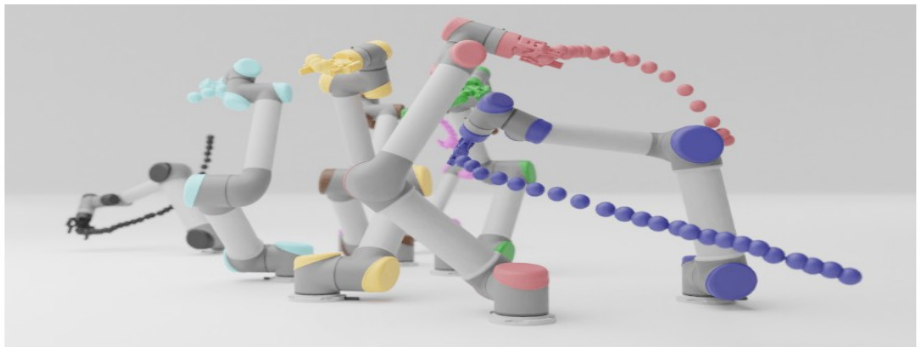
Deep MARL: where is the field…?

- MARL for traffic:
  - http://www.fransoliehoek.net/trafficvideo
- Learning to communicate:
  - https://youtu.be/KhtdEvJ1F6Q?t=41
  - https://arxiv.org/abs/1810.11187
- Starcraft II via ‘value factorization’
  - https://arxiv.org/pdf/1803.11485.pdf
  - https://www.youtube.com/watch?v=4WBi8xI_8YA
- Capture the flag
  - https://deepmind.com/blog/capture-the-flag-science/



Deep MARL: where is the field…?

- "From Motor Control to Team Play in Simulated Humanoid Football"
  - https://youtu.be/KHMwq9pv7mg?t=249
- Learning a Decentralized Multi-arm Motion Planner
  - https://multiarm.cs.columbia.edu/
- Learning to fly
  - https://github.com/utiasDSL/gym-pybullet-drones
- Playing Stratego
  - https://www.science.org/doi/10.1126/science.add4679



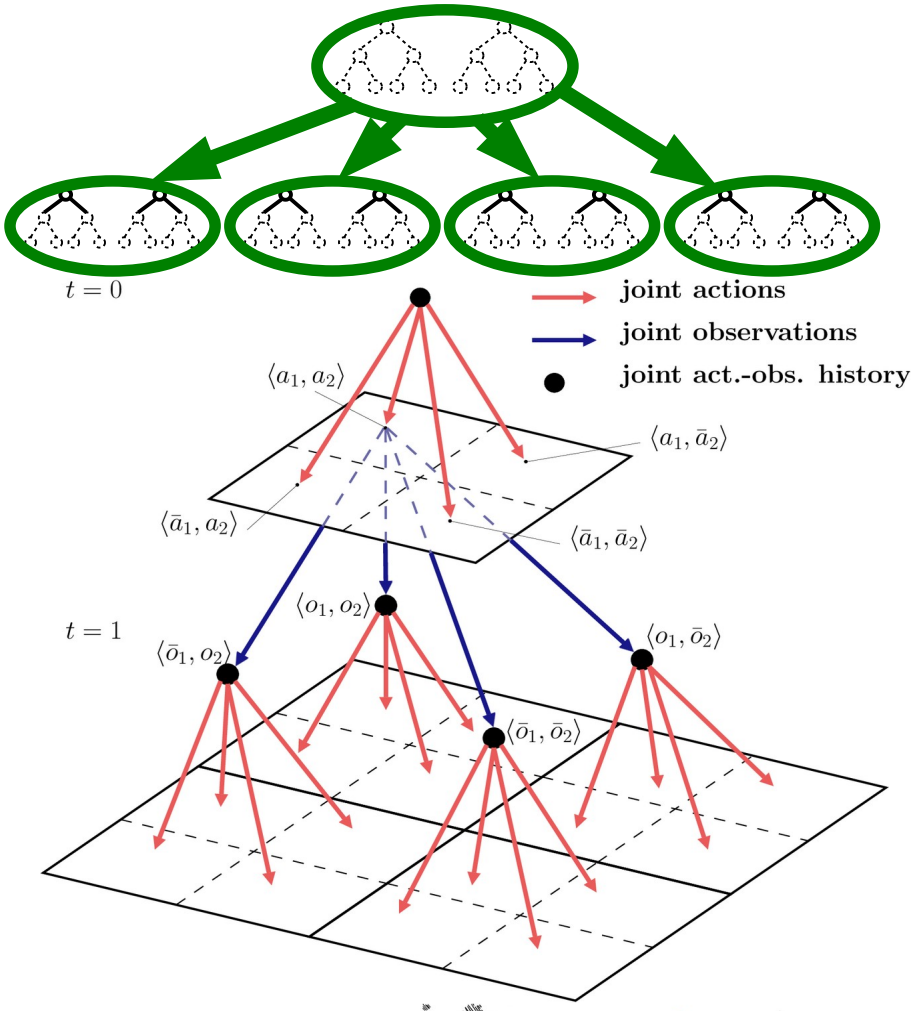
More Dec-POMDP solver improvements

MAA* via Bayesian Games

- Each node → a φ^t
- decision problem for stage t

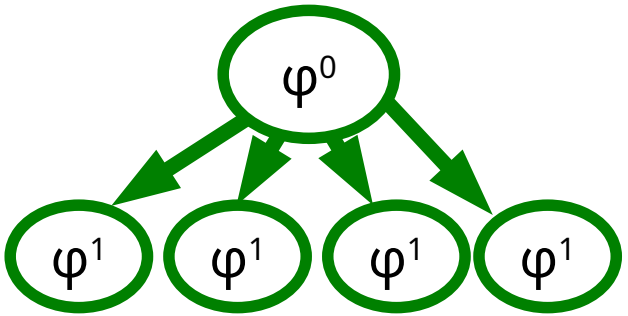
$\vec{\theta}_1^{t=0}$	$\vec{\theta}_2^{t=0}$	()	
		a_2	$\bar{a}_2$
()	a_1	+2.75	−4.1
	$\bar{a}_1$	−0.9	+0.3

$\vec{\theta}_1^{t=1}$	$\vec{\theta}_2^{t=1}$	(a_2,o_2)		$(a_2,\bar{o}_2)$		...
		a_2	$\bar{a}_2$	a_2	$\bar{a}_2$	
(a_1,o_1)	a_1	−0.3	+0.6	−0.6	+4.0	...
	$\bar{a}_1$	−0.6	+2.0	−1.3	+3.6	...
$(a_1,\bar{o}_1)$	a_1	+3.1	+4.4	−1.9	+1.0	...
	$\bar{a}_1$	+1.1	−2.9	+2.0	−0.4	
$(\bar{a}_1,o_1)$	a_1	−0.4	−0.9	−0.5	−1.0	...
	$\bar{a}_1$	−0.9	−4.5	−1.0	+3.5	...
$(\bar{a}_1,\bar{o}_1)$	...	...	...	...	...	...



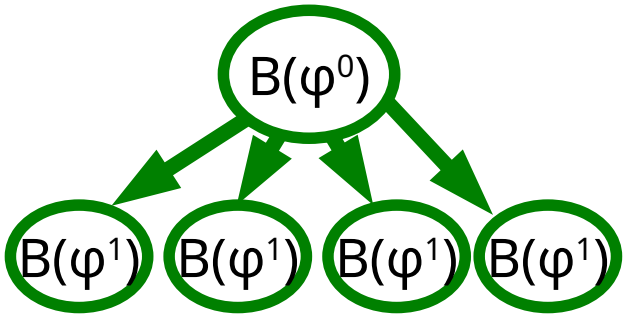
MAA* via Bayesian Games – 2

MAA* perspective



- node → φ^t
- joint decision rule δ
  maps OHs to actions
- Expansion: appending all next-stage decision rules: $\varphi^{t+1}=(\varphi^t,\delta^t)$

BG perspective

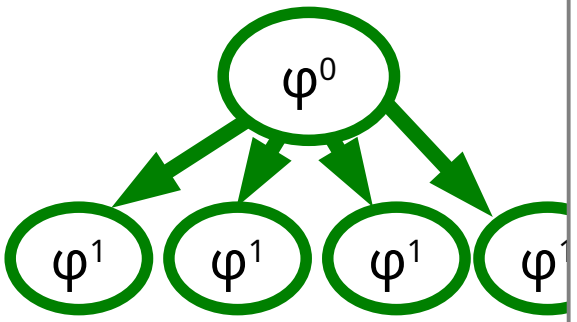


- node → a BG
- joint BG policy β
  maps 'types' to actions
- Expansion: enumeration of all joint BG policies $\varphi^{t+1}=(\varphi^t,\beta^t)$

direct correspondence: δ ↔ β

MAA* via Bayesian Games – 2

MAA* perspective



- node → φ^t
- joint decision rule δ maps OHs to actions
- Expansion: appending all next decision rules: $\varphi^{t+1}=(\varphi^t,\delta^t)$

direct co

What is the point?

- ▶ Generalized MAA* [Oliehoek & Vlassis '07]
- ▶ Unified perspective of MAA* and 'BAGA' approximation [Emery-Montemerlo et al. '04]
- ▶ No direct improvements...

However...

- ▶ BGs provide abstraction layer
- ▶ Facilitated two improvements that lead to state-of-the-art performance [Oliehoek et al. '13]
  - Clustering of histories
  - Incremental expansion

MAA* Limitations

- Number of children grows doubly exponential with nodes depth
- For a node last stage, number of children is

$$O\left(|A_*|^{n|O_*|^{h-1}}\right)$$

- Total number of joint policies $O\left(|A_*|^{(n|O_*|^h-1)/(|O_*|-1)}\right)$

→ MAA* can only solve 1 horizon longer than brute force search... [Seuken & Zilberstein '08]

- Techniques to overcome this problem [Oliehoek et al. 2013 JAIR]:
  - lossless clustering of histories
  - Incremental expansion

Lossless Clustering

- Two types (=action-observation histories) in a BG are **probabilistically equivalent** iff

$$P(\vec{\theta}_{-i}|\vec{\theta}_{i,a}) = P(\vec{\theta}_{-i}|\vec{\theta}_{i,b})$$

$$P(s|\vec{\theta}_{-i}, \vec{\theta}_{i,a}) = P(s|\vec{\theta}_{-i}, \vec{\theta}_{i,b})$$

$\vec{o}_1^2$ \ $\vec{o}_2^2$	$(o_{\text{HL}},o_{\text{HL}})$	$(o_{\text{HL}},o_{\text{HR}})$	$(o_{\text{HR}},o_{\text{HL}})$	$(o_{\text{HR}},o_{\text{HR}})$
$(o_{\text{HL}},o_{\text{HL}})$	0.261	0.047	0.047	0.016
$(o_{\text{HL}},o_{\text{HR}})$	0.047	0.016	0.016	0.047
$(o_{\text{HR}},o_{\text{HL}})$	0.047	0.016	0.016	0.047
$(o_{\text{HR}},o_{\text{HR}})$	0.016	0.047	0.047	0.261

(a) The joint type probabilities.

$\vec{o}_1^2$ \ $\vec{o}_2^2$	$(o_{\text{HL}},o_{\text{HL}})$	$(o_{\text{HL}},o_{\text{HR}})$	$(o_{\text{HR}},o_{\text{HL}})$	$(o_{\text{HR}},o_{\text{HR}})$
$(o_{\text{HL}},o_{\text{HL}})$	0.999	0.970	0.970	0.5
$(o_{\text{HL}},o_{\text{HR}})$	0.970	0.5	0.5	0.030
$(o_{\text{HR}},o_{\text{HL}})$	0.970	0.5	0.5	0.030
$(o_{\text{HR}},o_{\text{HR}})$	0.5	0.030	0.030	0.001

(b) The induced joint beliefs. Listed is the probability $\Pr(s_l|\vec{\boldsymbol{\theta}}^2,\boldsymbol{b}^0)$ of the tiger being behind the left door.

Lossless Clustering

- Two types (=action-observation histories) in a BG are **probabilistically equivalent** iff

$$P(\vec{\theta}_{-i}|\vec{\theta}_{i,a}) = P(\vec{\theta}_{-i}|\vec{\theta}_{i,b})$$

$$P(s|\vec{\theta}_{-i}, \vec{\theta}_{i,a}) = P(s|\vec{\theta}_{-i}, \vec{\theta}_{i,b})$$

$\vec{o}_1^2$ \ $\vec{o}_2^2$	$(o_{\text{HL}},o_{\text{HL}})$	$(o_{\text{HL}},o_{\text{HR}})$	$(o_{\text{HR}},o_{\text{HL}})$	$(o_{\text{HR}},o_{\text{HR}})$
$(o_{\text{HL}},o_{\text{HL}})$	0.261	0.047	0.047	0.016
$(o_{\text{HL}},o_{\text{HR}})$	0.047	0.016	0.016	0.047
$(o_{\text{HR}},o_{\text{HL}})$	0.047	0.016	0.016	0.047
$(o_{\text{HR}},o_{\text{HR}})$	0.016	0.047	0.047	0.261

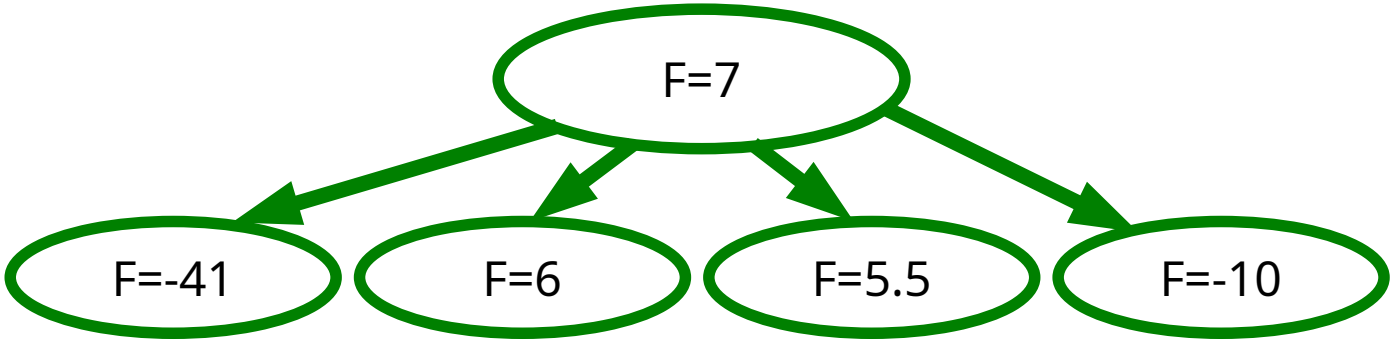
(a) The joint type probabilities.

$\vec{o}_1^2$ \ $\vec{o}_2^2$	$(o_{\text{HL}},o_{\text{HL}})$	$(o_{\text{HL}},o_{\text{HR}})$	$(o_{\text{HR}},o_{\text{HL}})$	$(o_{\text{HR}},o_{\text{HR}})$
$(o_{\text{HL}},o_{\text{HL}})$	0.999	0.970	0.970	0.5
$(o_{\text{HL}},o_{\text{HR}})$	0.970	0.5	0.5	0.030
$(o_{\text{HR}},o_{\text{HL}})$	0.970	0.5	0.5	0.030
$(o_{\text{HR}},o_{\text{HR}})$	0.5	0.030	0.030	0.001

(b) The induced joint beliefs. Listed is the probability $\Pr(s_l|\vec{\theta}^2,b^0)$ of the tiger being behind the left door.

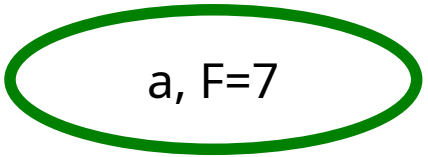
Incremental Expansion

- Key idea: even though nodes can have many children, only few are useful.
  - i.e., only few will be selected for further expansion
  - others will have too low heuristic value



- if we can generate the nodes in increasing heuristic order
  → can avoid expansion of redundant nodes

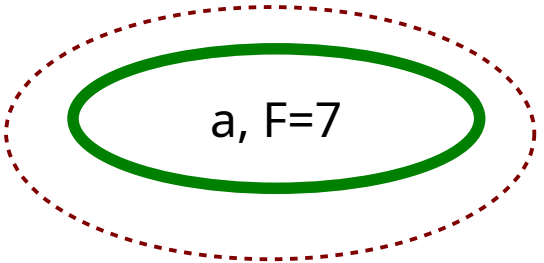
Incremental Expansion



Open list
a – 7

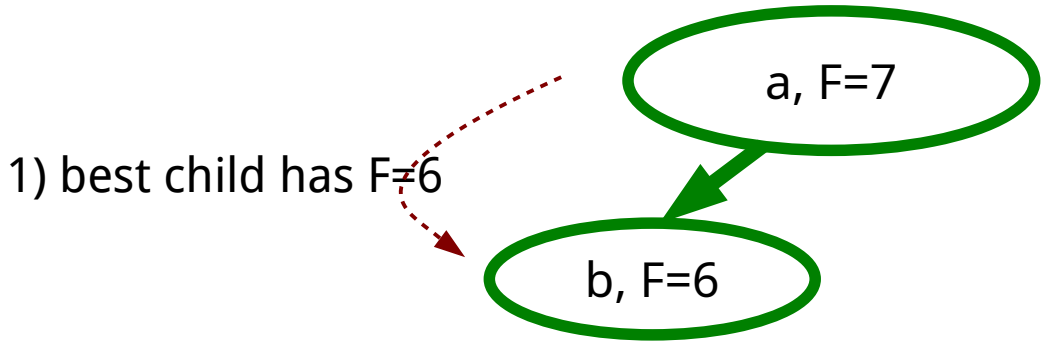
Incremental Expansion

Select for expansion →



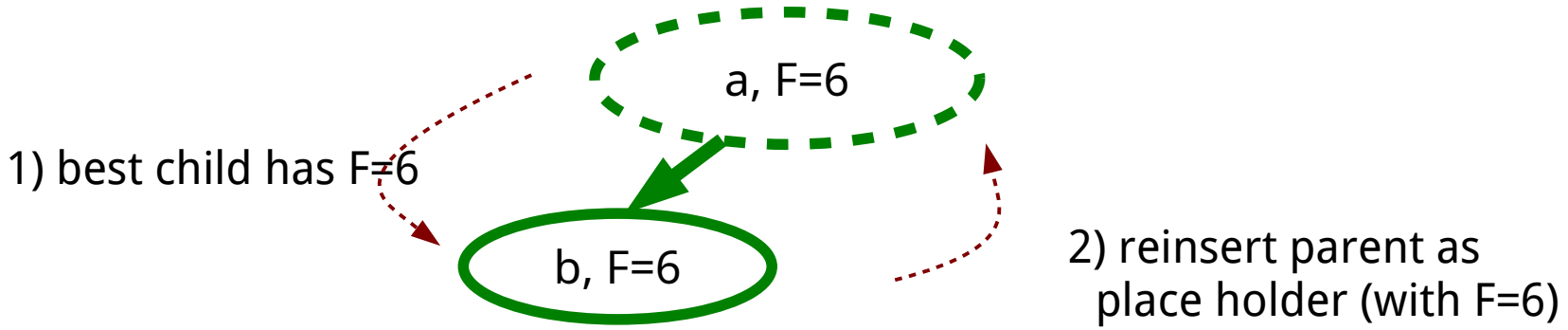
Open list
a – 7

Incremental Expansion



Open list
b – 6

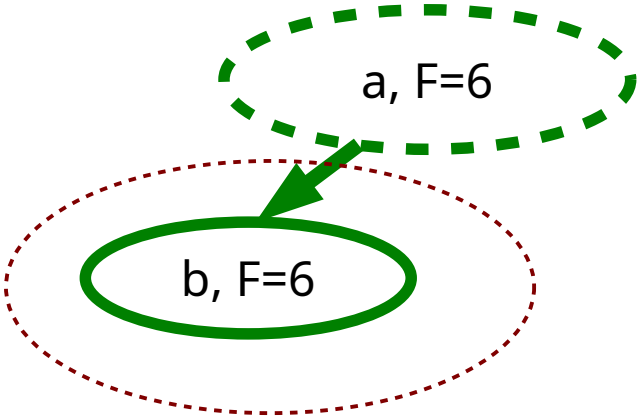
Incremental Expansion



Open list
b – 6
a – 6

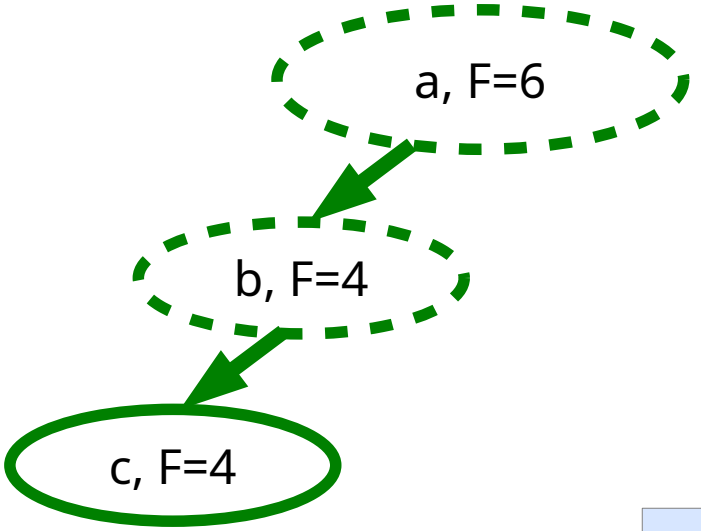
Incremental Expansion

Select for expansion →



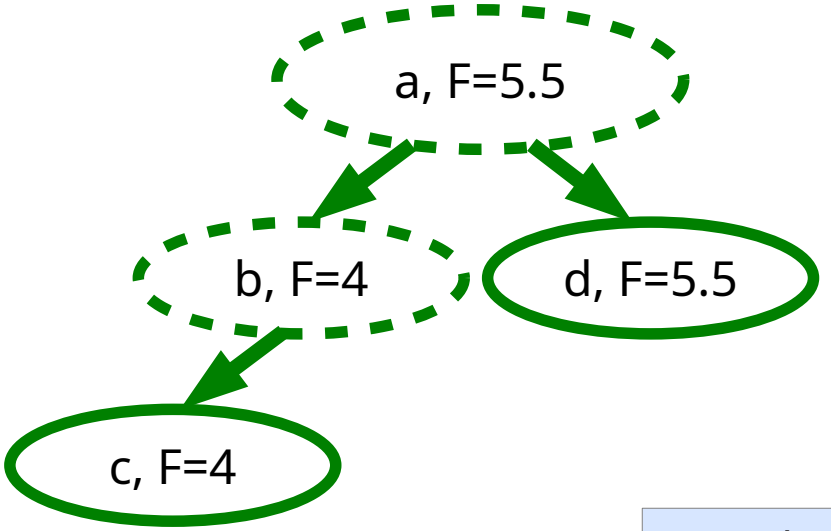
Open list
b – 6
a – 6

Incremental Expansion



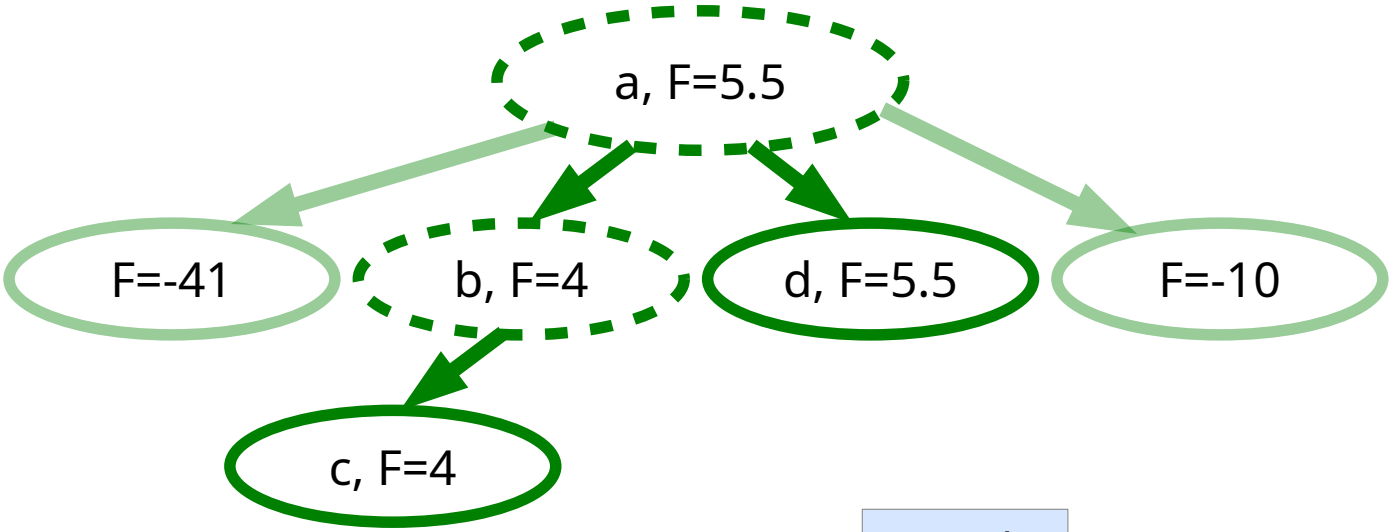
Open list
a – 6
c – 4
b – 4

Incremental Expansion



Open list
d – 5.5
a – 5.5
c – 4
b – 4

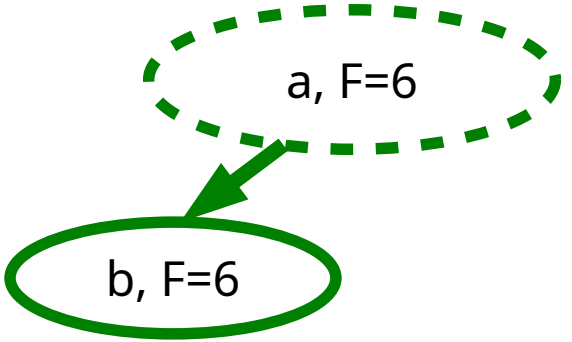
Incremental Expansion



Open list
d – 5.5
a – 5.5
c – 4
b – 4

Incremental Expansion: How?

- How do we generate the next-best child?



- Node ↔ BG, so...
  - find the solutions of the BG (in decreasing order of value)
  - i.e., 'incremental BG solver'
  - Modification of BaGaBaB [Oliehoek et al. 2010]
    - stop searching when next solution found
    - save search tree for next time visited.

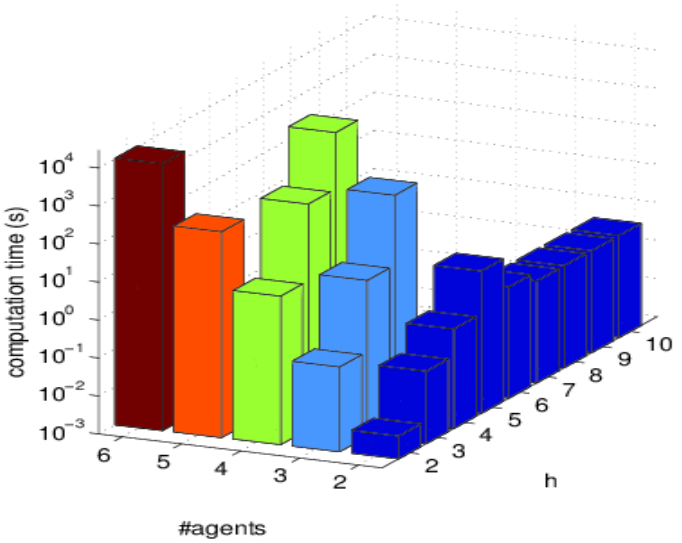
Some Results

	problem primitives			
	n	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}_i\rvert$	$\lvert\mathcal{O}_i\rvert$
DEC-TIGER	2	2	3	2
BROADCASTCHANNEL	2	4	2	2
GRIDSMALL	2	16	5	2
COOPERATIVE BOX PUSHING	2	100	4	5
RECYCLING ROBOTS	2	4	3	2
HOTEL 1	2	16	3	4
FIREFIGHTING	2	432	3	2

h	MILP	DP-LPC	DP-IPG	GMAA — Q_{BG}		
				IC	ICE	heur
BROADCASTCHANNEL, ICE solvable to $h = 900$						
2	0.38	≤ 0.01	0.09	≤ 0.01	≤ 0.01	≤ 0.01
3	1.83	0.50	56.66	≤ 0.01	≤ 0.01	≤ 0.01
4	34.06	*	*	≤ 0.01	≤ 0.01	≤ 0.01
5	48.94			≤ 0.01	≤ 0.01	≤ 0.01
DEC-TIGER, ICE solvable to $h = 6$						
2	0.69	0.05	0.32	≤ 0.01	≤ 0.01	≤ 0.01
3	23.99	60.73	55.46	≤ 0.01	≤ 0.01	≤ 0.01
4	*	–	2286.38	0.27	≤ 0.01	0.03
5			–	21.03	0.02	0.09
FIREFIGHTING (2 agents, 3 houses, 3 firelevels), ICE solvable to $h \gg 1000$						
2	4.45	8.13	10.34	≤ 0.01	≤ 0.01	≤ 0.01
3	–	–	569.27	0.11	0.10	0.07
4			–	950.51	1.00	0.65
GRIDSMALL, ICE solvable to $h = 6$						
2	6.64	11.58	0.18	0.01	≤ 0.01	≤ 0.01
3	*	–	4.09	0.10	≤ 0.01	0.42
4			77.44	1.77	≤ 0.01	67.39
RECYCLING ROBOTS, ICE solvable to $h = 70$						
2	1.18	0.05	0.30	≤ 0.01	≤ 0.01	≤ 0.01
3	*	2.79	1.07	≤ 0.01	≤ 0.01	≤ 0.01
4		2136.16	42.02	≤ 0.01	≤ 0.01	0.02
5		–	1812.15	≤ 0.01	≤ 0.01	0.02
HOTEL 1, ICE solvable to $h = 9$						
2	1.92	6.14	0.22	≤ 0.01	≤ 0.01	0.03
3	315.16	2913.42	0.54	≤ 0.01	≤ 0.01	1.51
4	–	–	0.73	≤ 0.01	≤ 0.01	3.74
5			1.11	≤ 0.01	≤ 0.01	4.54
9			8.43	0.02	≤ 0.01	20.26
10			17.40	#	#	
15			283.76			
COOPERATIVE BOX PUSHING (Q_{POMDP}), ICE solvable to $h = 4$						
2	3.56	15.51	1.07	≤ 0.01	≤ 0.01	≤ 0.01
3	2534.08	–	6.43	0.91	0.02	0.15
4	–		1138.61	*	328.97	0.63

‘–’ memory limit violations
‘*’ time limit overruns
‘#’ heuristic bottleneck

Some Results



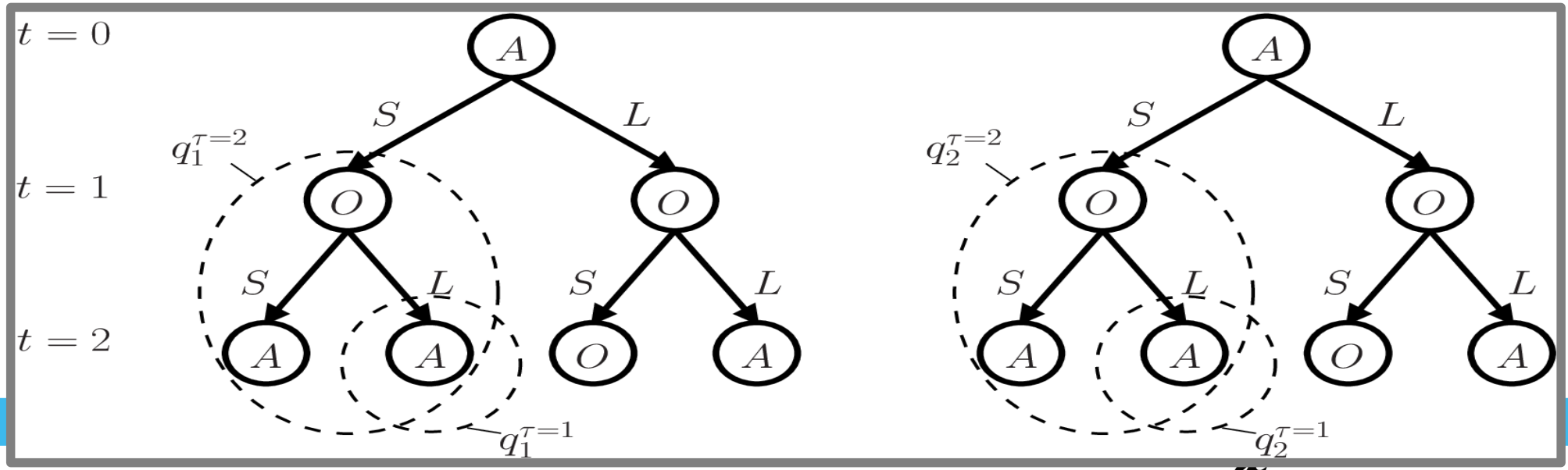
Scalability w.r.t. #agents

h	V^*	T_{GMAA*}(s)	T_{IC}(s)	T_{ICE}(s)
		RECYCLING ROBOTS		
3	10.660125	≤ 0.01	≤ 0.01	≤ 0.01
4	13.380000	713.41	≤ 0.01	≤ 0.01
5	16.486000	–	≤ 0.01	≤ 0.01
6	**19.554200**		≤ 0.01	≤ 0.01
10	**31.863889**		≤ 0.01	≤ 0.01
15	**47.248521**		≤ 0.01	≤ 0.01
20	**62.633136**		≤ 0.01	≤ 0.01
30	**93.402367**		0.08	0.05
40	**124.171598**		0.42	0.25
50	**154.940828**		2.02	1.27
70	**216.479290**		–	28.66
80			–	–
		BROADCASTCHANNEL		
4	3.890000	≤ 0.01	≤ 0.01	≤ 0.01
5	4.790000	1.27	≤ 0.01	≤ 0.01
6	**5.690000**	–	≤ 0.01	≤ 0.01
7	**6.590000**		≤ 0.01	≤ 0.01
10	**9.290000**		≤ 0.01	≤ 0.01
25	**22.881523**		≤ 0.01	≤ 0.01
50	**45.501604**		≤ 0.01	≤ 0.01
100	**90.760423**		≤ 0.01	≤ 0.01
250	**226.500545**		0.06	0.07
500	**452.738119**		0.81	0.94
700	**633.724279**		0.52	0.63
800			–	–
900	**814.709393**		9.57	11.11
1000			–	–

Cases that compress well
* excluding heuristic

Dynamic Programming – 1

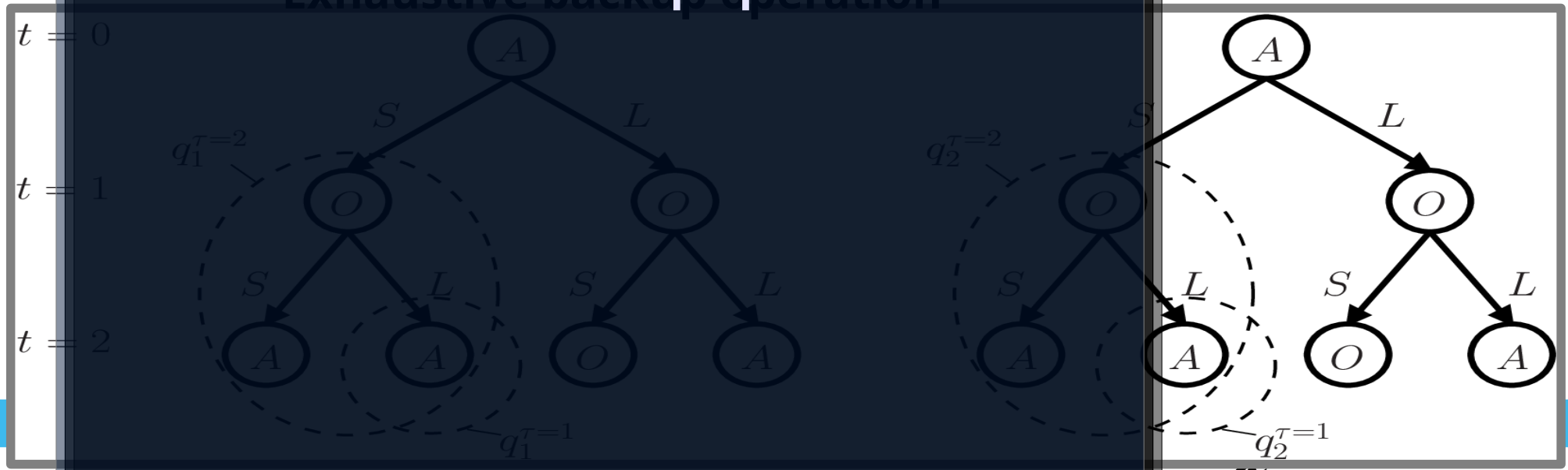
- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - construct all 2-stages-to-go policies $Q^{\tau=2}$, etc.



Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - construct all 2-stages-to-go policies $Q^{\tau=2}$, etc.

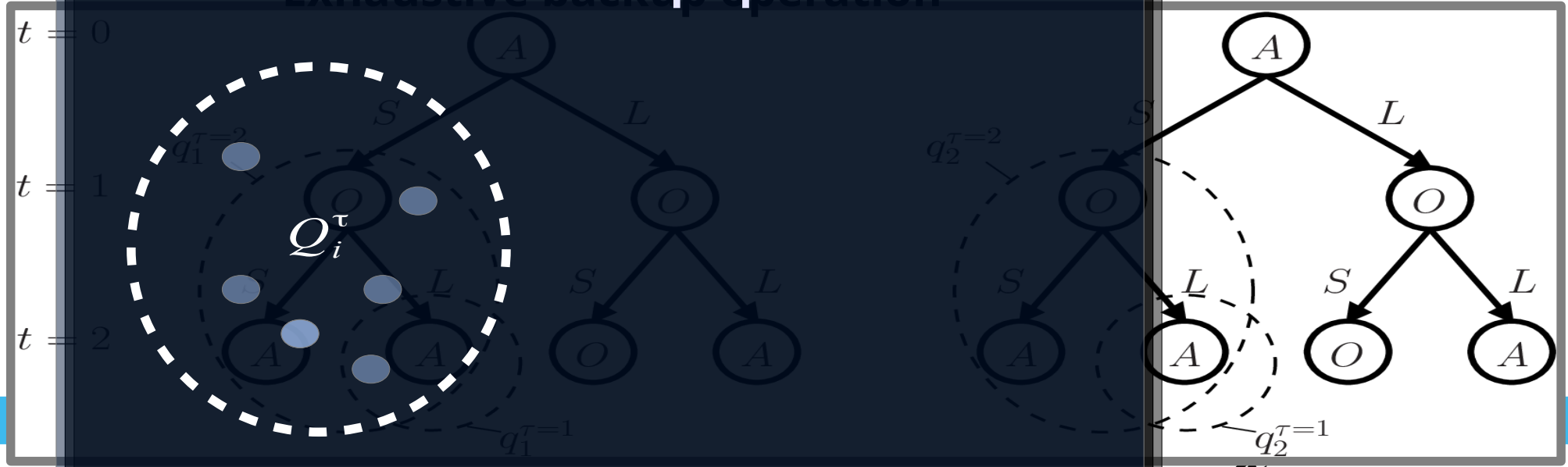
Exhaustive backup operation



Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - construct all 2-stages-to-go policies $Q^{\tau=2}$, etc.

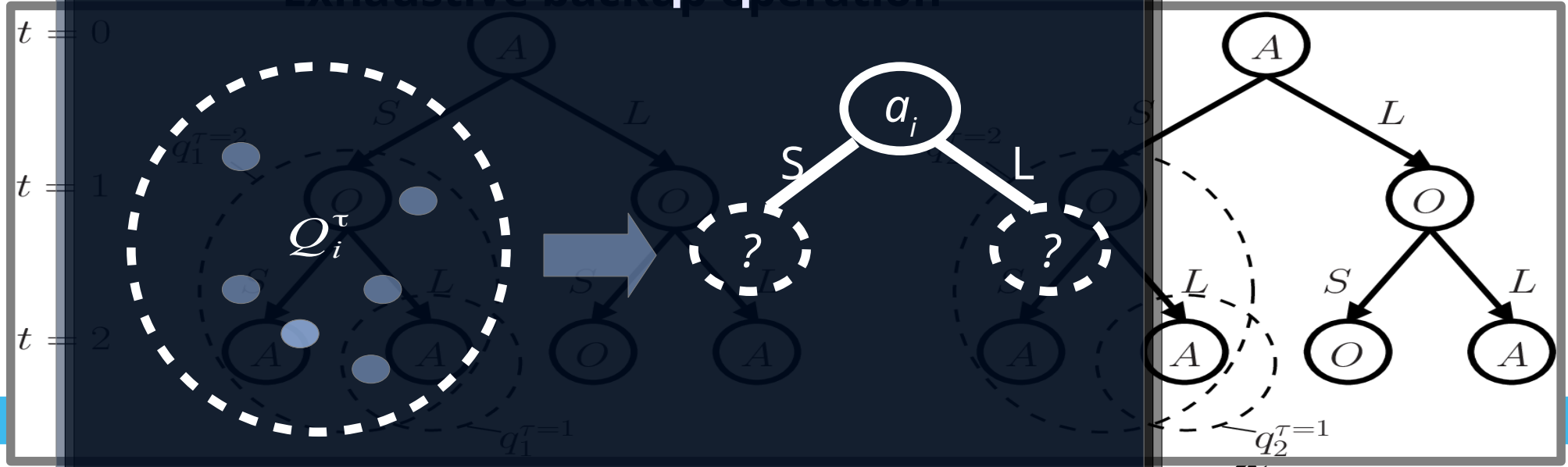
Exhaustive backup operation



Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - construct all 2-stages-to-go policies $Q^{\tau=2}$, etc.

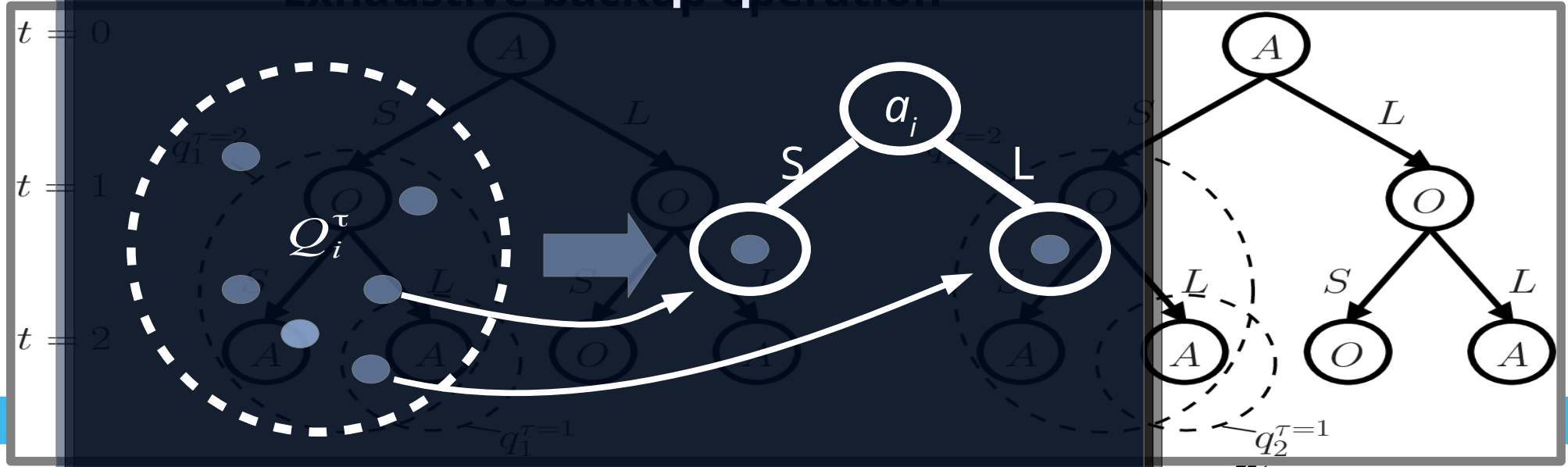
Exhaustive backup operation



Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - construct all 2-stages-to-go policies $Q^{\tau=2}$, etc.

Exhaustive backup operation

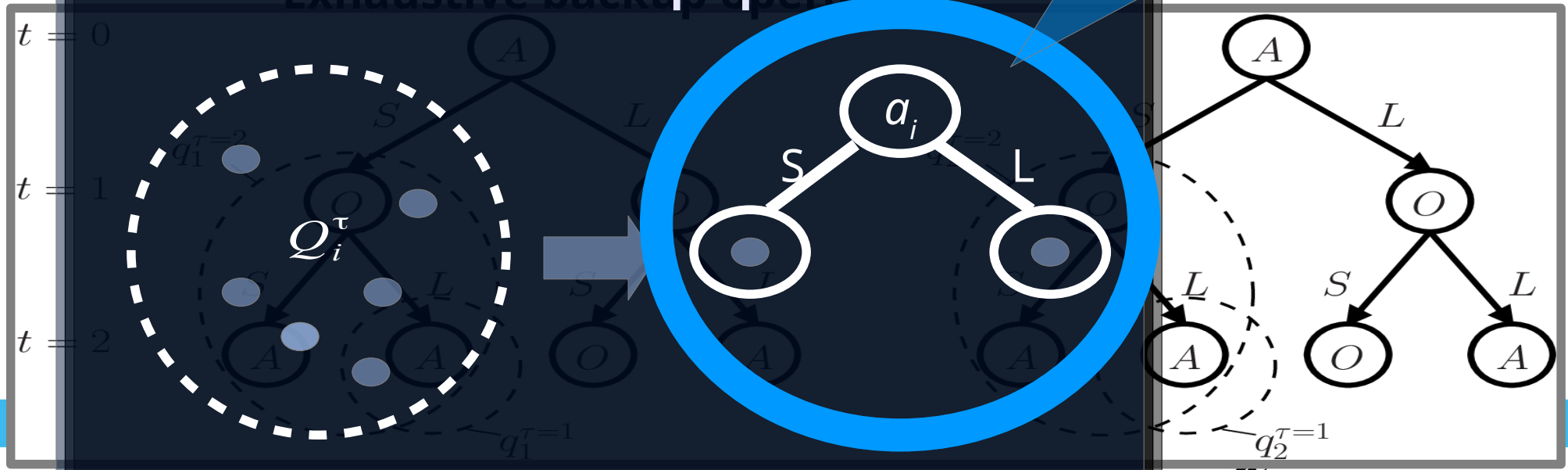


Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - construct all 2-stages-to-go policies $Q^{\tau=2}$ etc.

Exhaustive backup operation

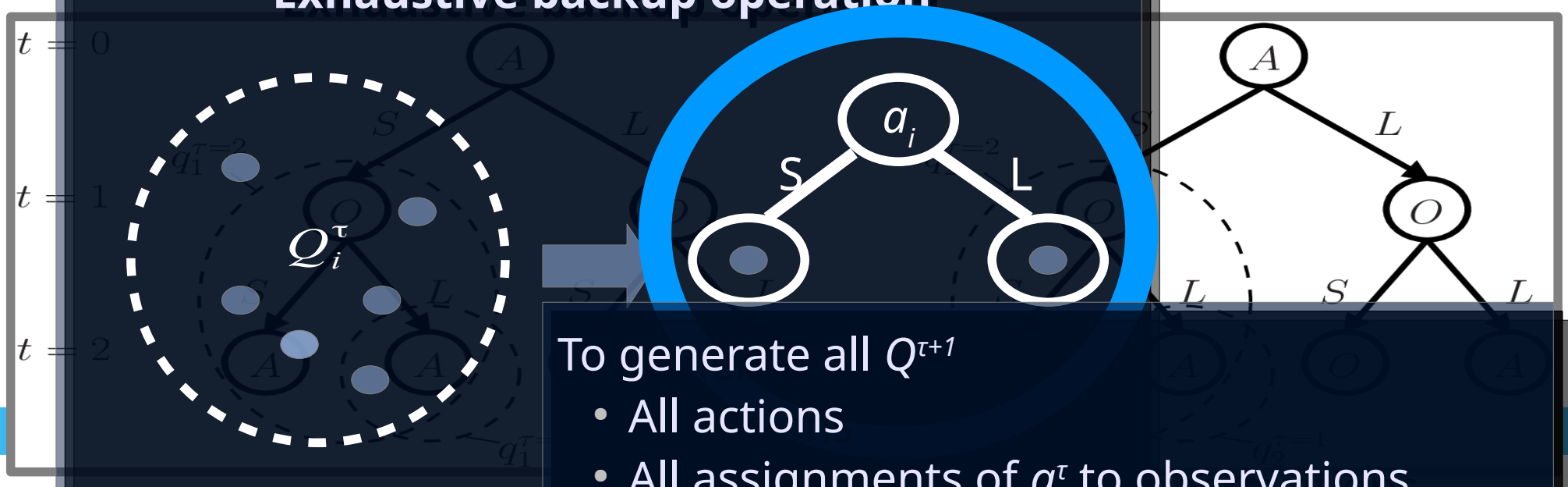
a new $q^{\tau+1}$



Dynamic Programming – 1

- Generate all policies in a special way:
  - from 1 stage-to-go policies $Q^{\tau=1}$
  - construct all 2 stages-to-go policies $Q^{\tau=2}$, etc.

Exhaustive backup operation

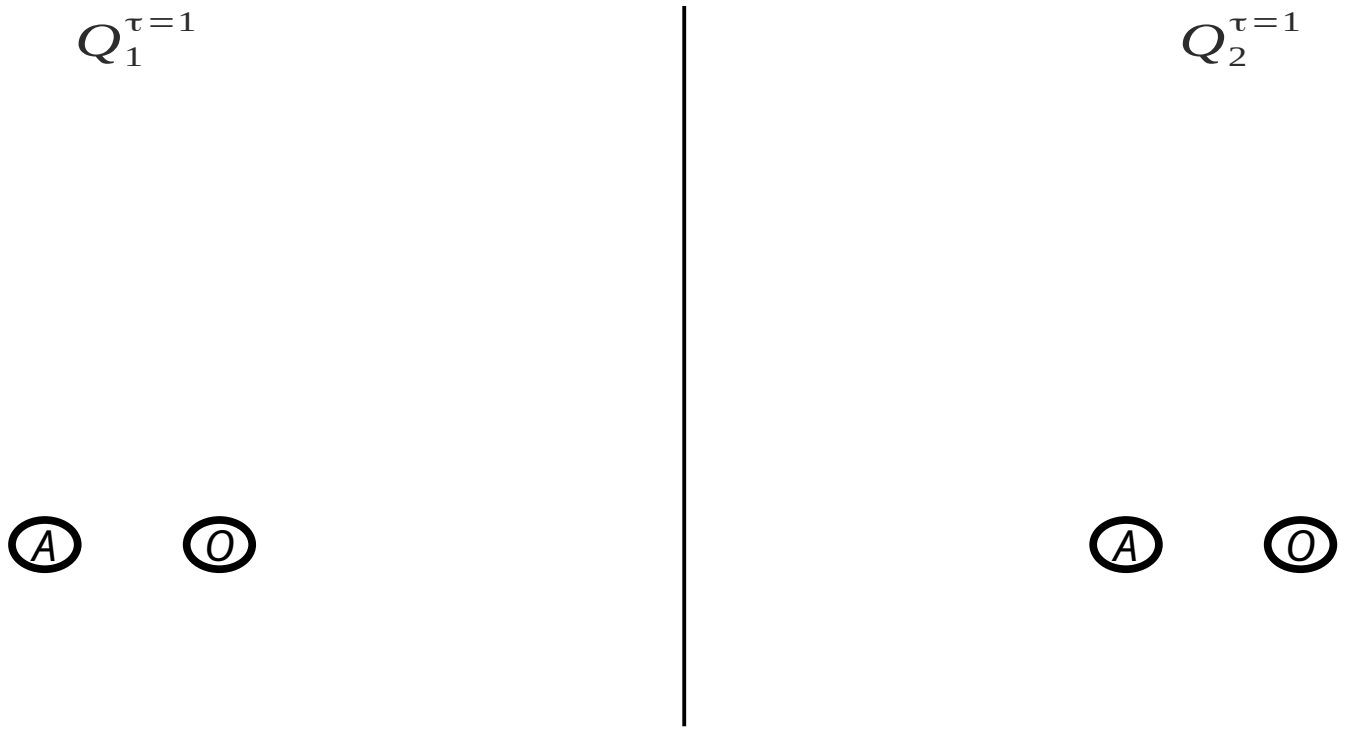


To generate all $Q^{\tau+1}$

- All actions
- All assignments of q^τ to observations

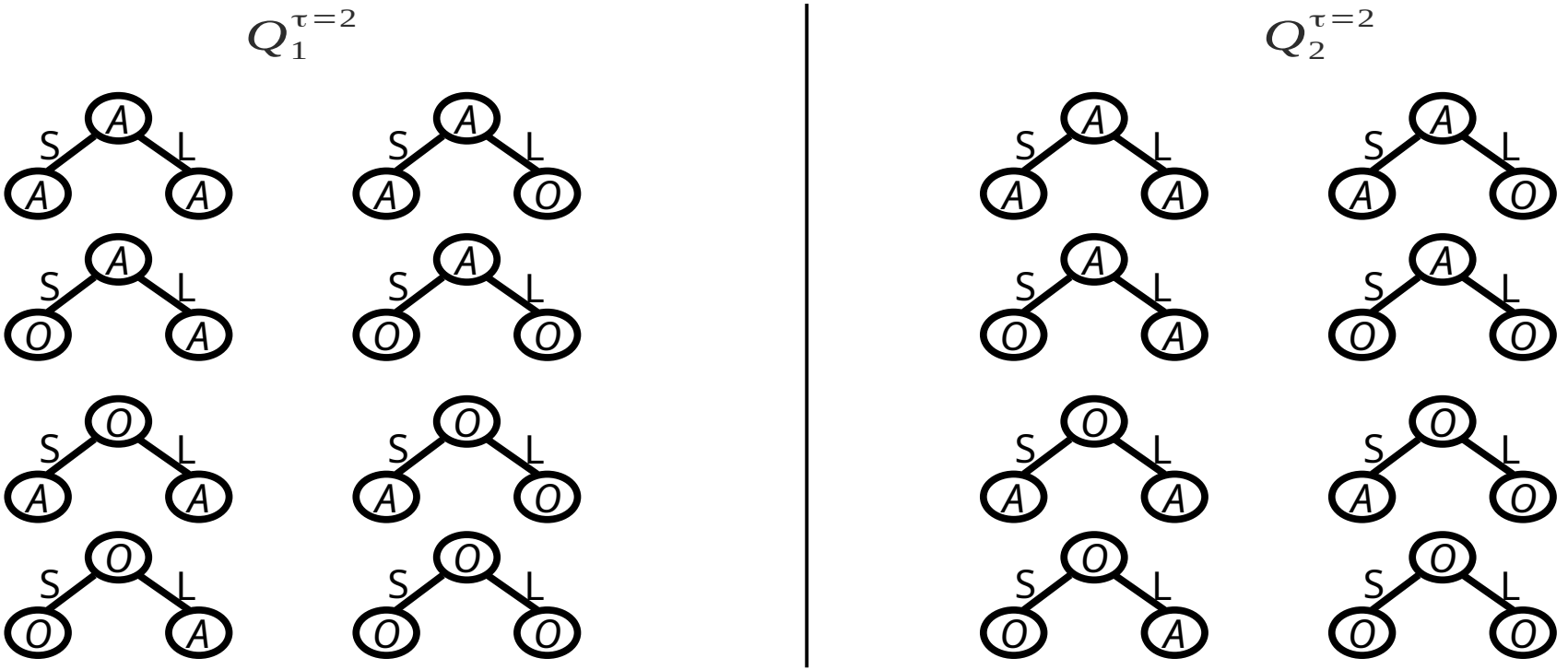
Dynamic Programming – 2

- (obviously) this scales very poorly…



Dynamic Programming – 2

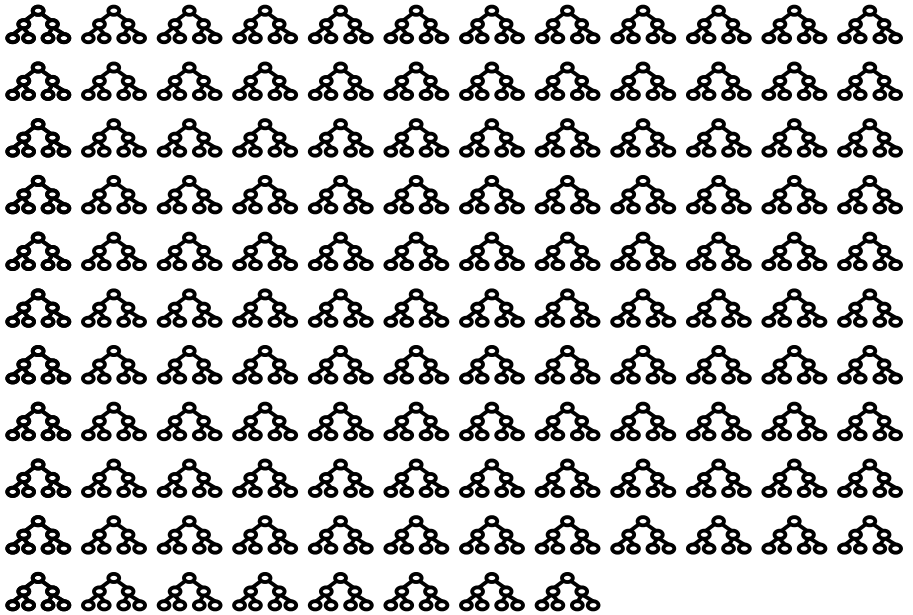
- (obviously) this scales very poorly…



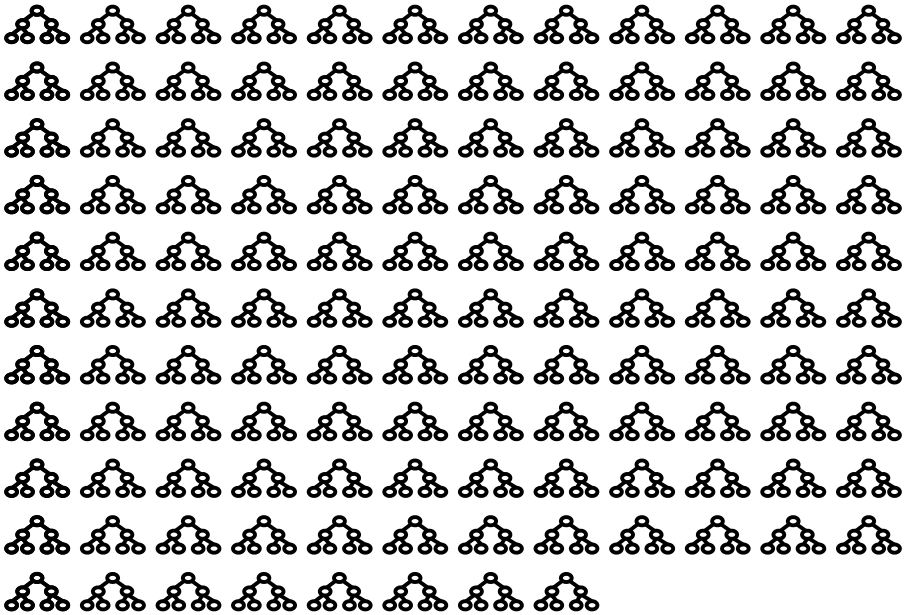
Dynamic Programming – 2

- (obviously) this scales very poorly…

$Q_1^{\tau=3}$

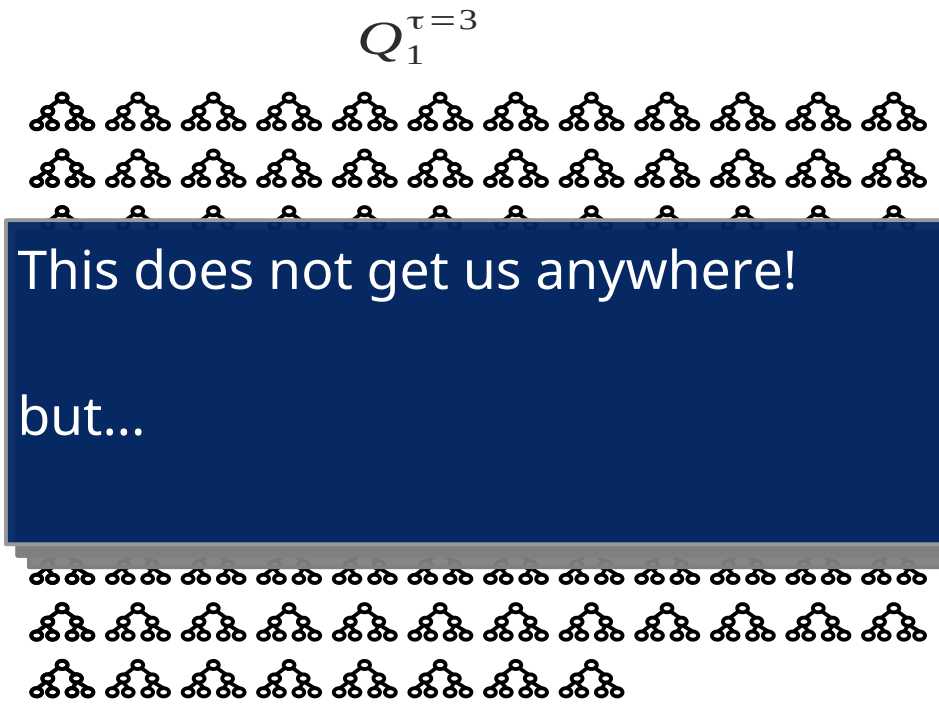


$Q_2^{\tau=3}$



Dynamic Programming – 2

- (obviously) this scales very poorly…



h	num. indiv. policies
1	2
2	8
3	128
4	32768
5	2.1475e+09
6	9.2234e+18
7	1.7014e+38
8	5.7896e+76

This does not get us anywhere!

but…

Dynamic Programming – 3

- Perhaps not all those Q_i^{τ} are useful!
  - Perform **pruning** of 'dominated policies'!

- Algorithm [Hansen et al. 2004]

$Q_i^{\tau=1} = A_i$

```
Initialize Q1(1), Q2(1)
for tau=2 to h
  Q1(tau) = ExhaustiveBackup(Q1(tau-1))
  Q2(tau) = ExhaustiveBackup(
  Prune(Q1,Q2,tau)
end
```

Note: cannot prune independently!

- ▶usefulness of a q_1 depends on Q_2
- ▶and vice versa
  → **Iterated elimination** of policies
- ▶how? linear programming.

Dynamic Programming – 4

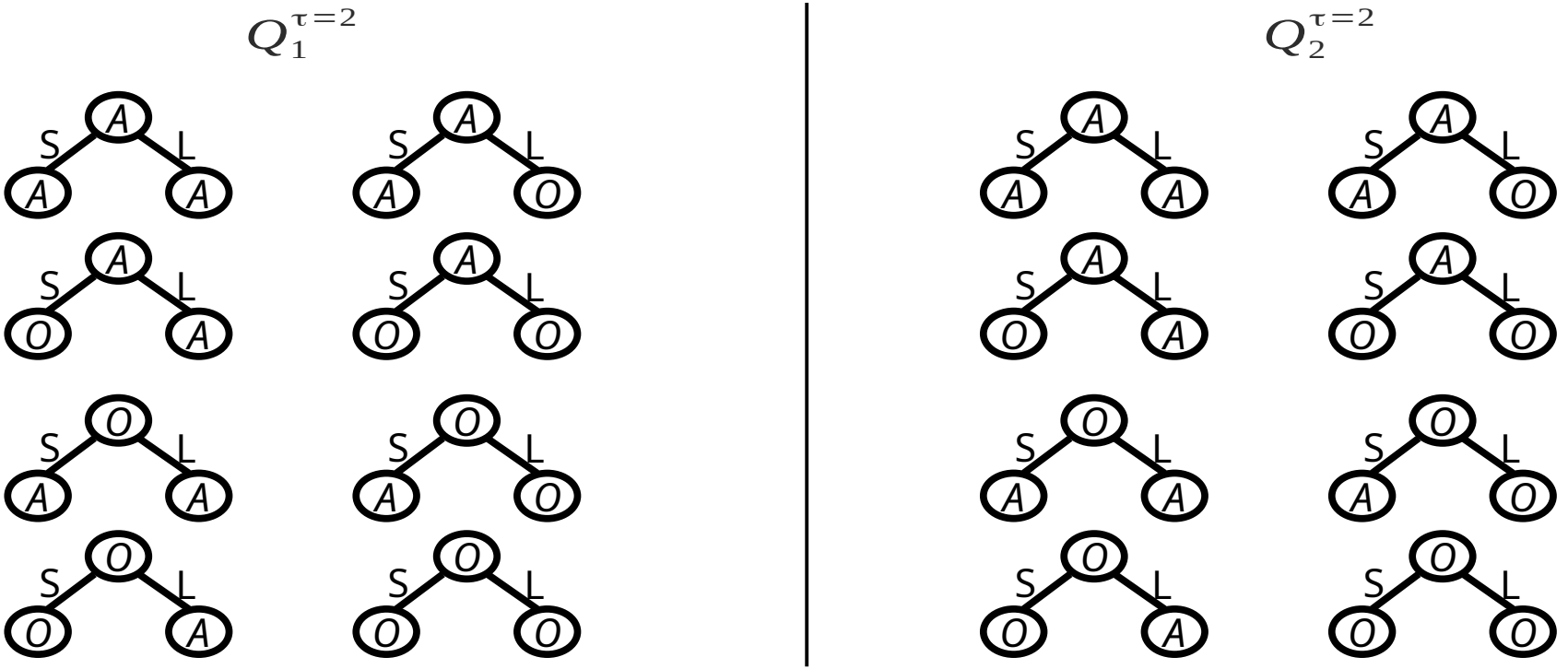
- Initialization

$Q_1^{\tau=1}$

$Q_2^{\tau=1}$

Dynamic Programming – 4

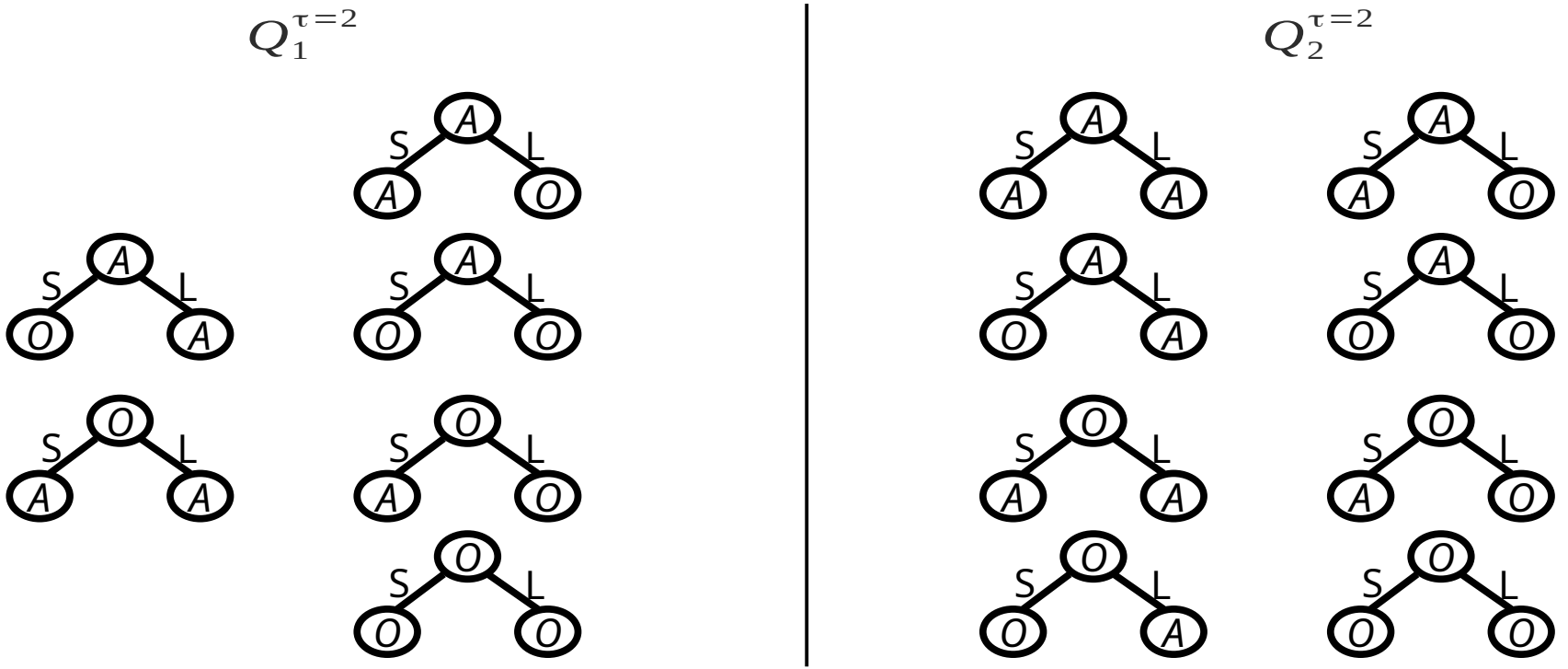
- Exhaustive Backups gives



Dynamic Programming – 4

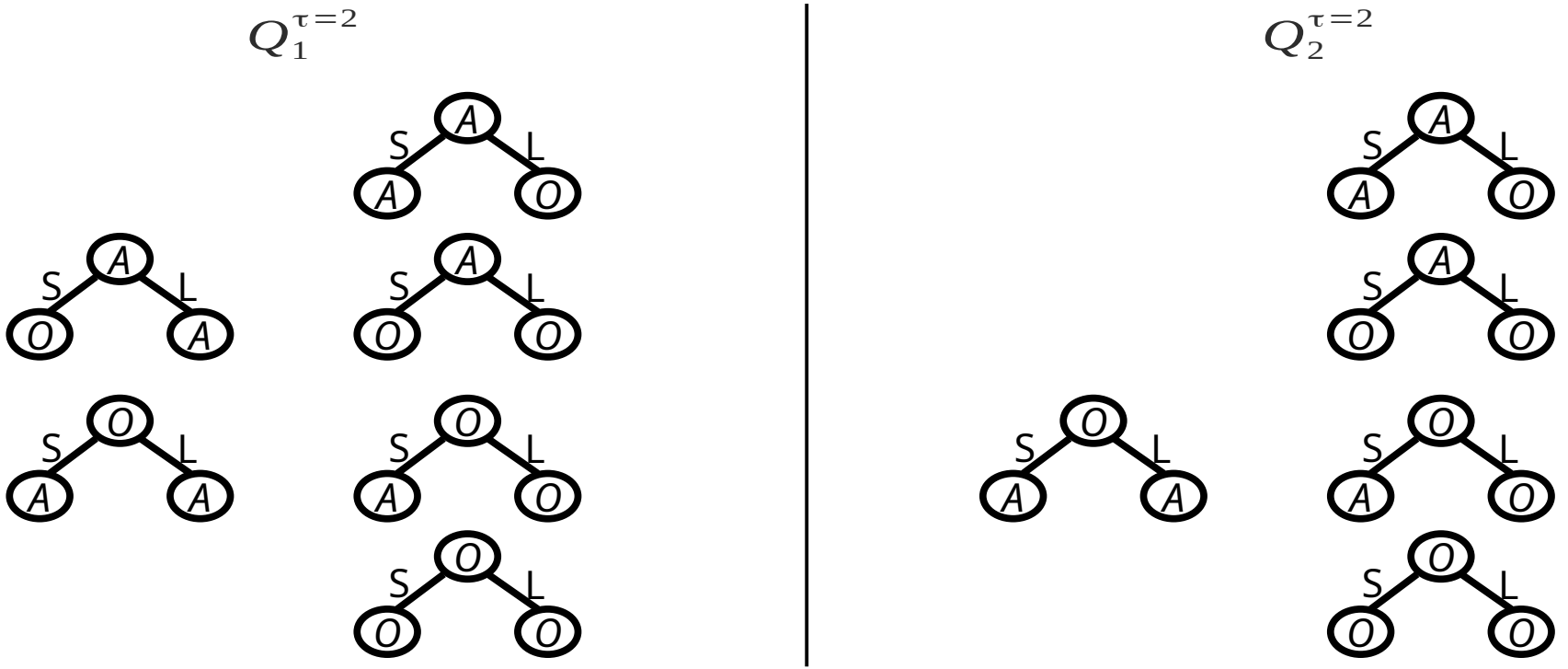
- Pruning agent 1…

Hypothetical Pruning
(not the result of actual pruning)



Dynamic Programming – 4

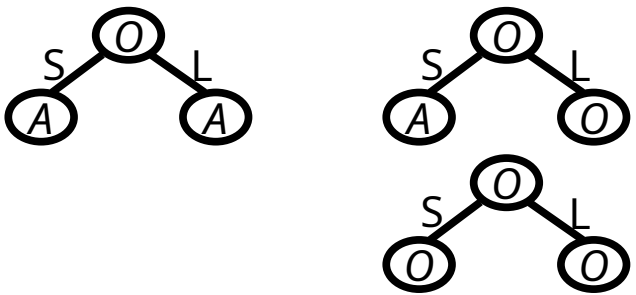
- Pruning agent 2…



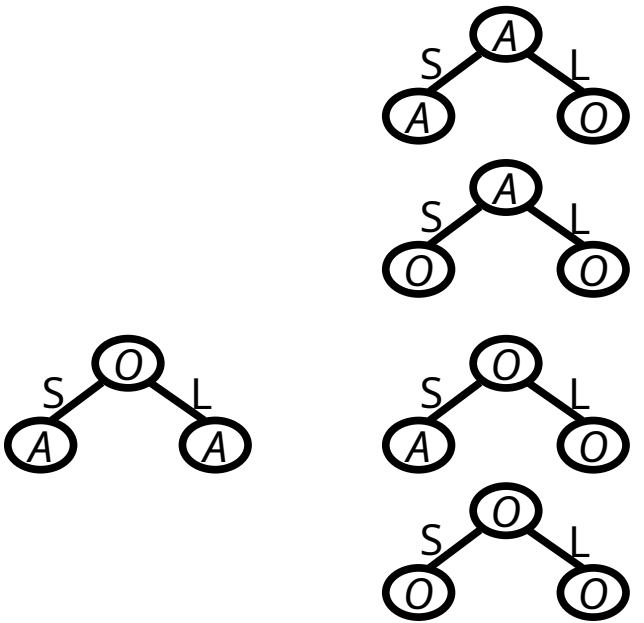
Dynamic Programming – 4

- Pruning agent 1…

$Q_1^{\tau=2}$



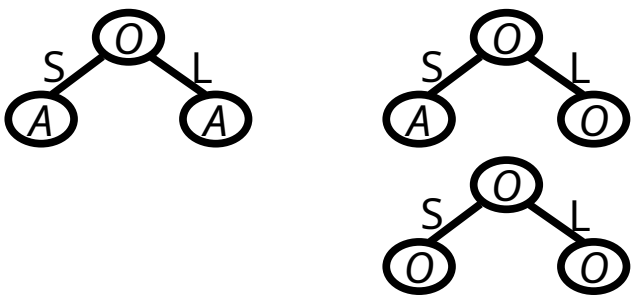
$Q_2^{\tau=2}$



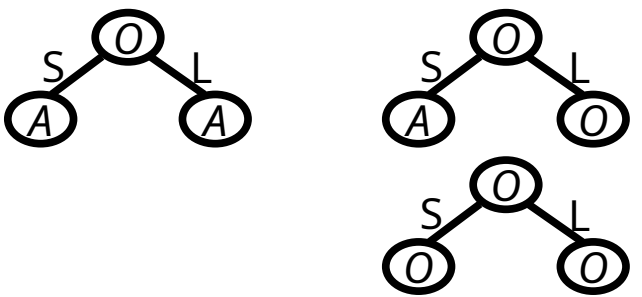
Dynamic Programming – 4

- Etc…

$Q_1^{\tau=2}$



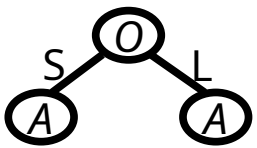
$Q_2^{\tau=2}$

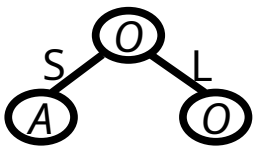


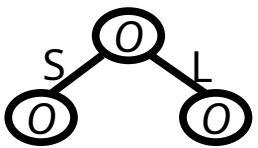
Dynamic Programming – 4

- Etc…

$Q_1^{\tau=2}$

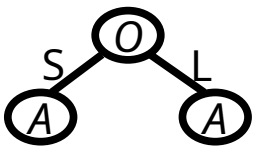


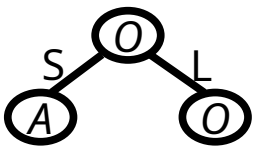


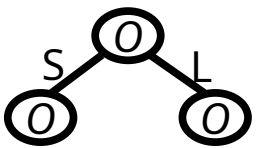


$Q_2^{\tau=2}$

In this case: symmetric
→ but need not be in general!



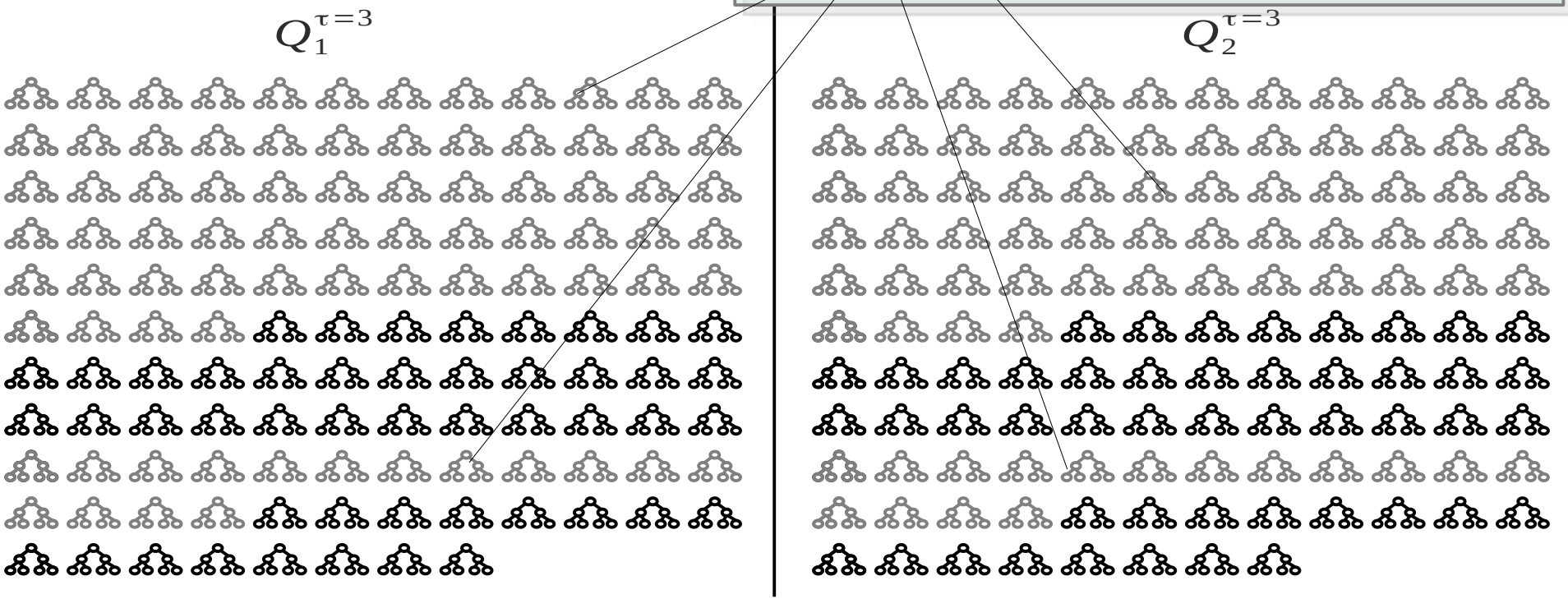




Dynamic Programming – 4

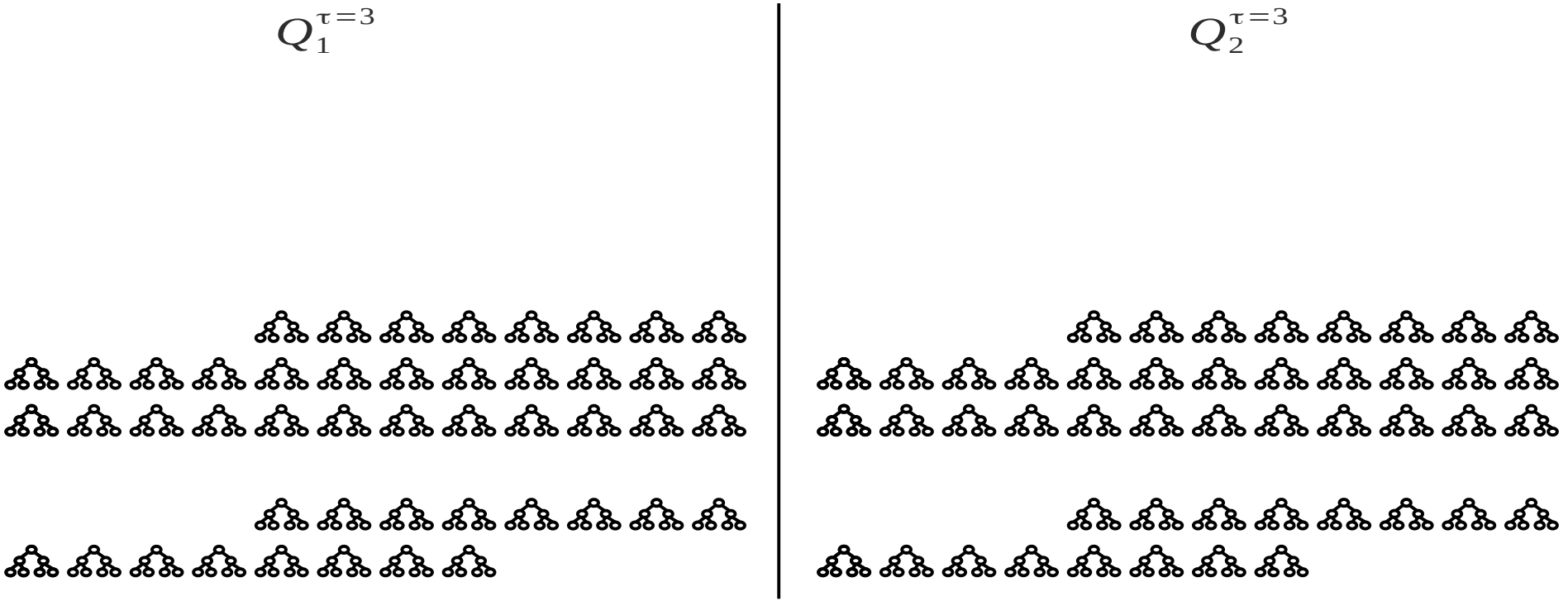
- Exhaustive backups:

We **avoid** generation of many policies!



Dynamic Programming – 4

- Exhaustive backups:



Dynamic Programming – 4

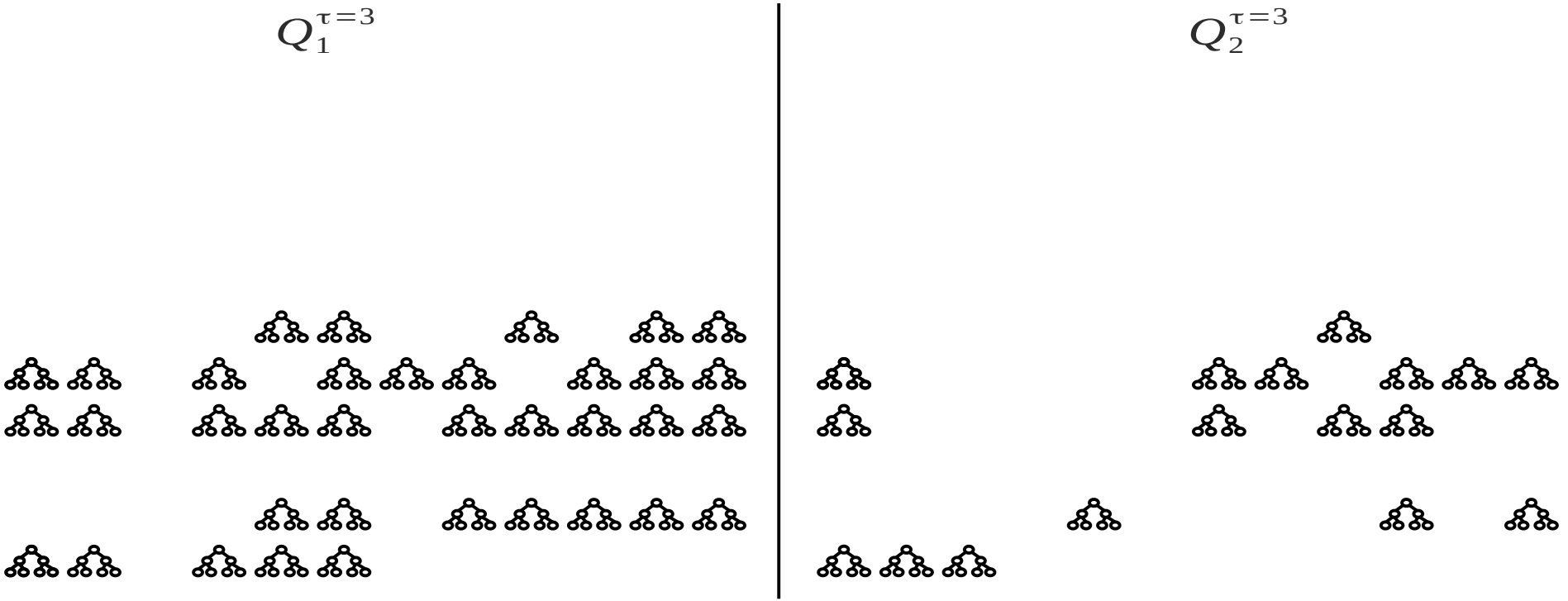
- Pruning agent 1…

$Q_1^{\tau=3}$

$Q_2^{\tau=3}$

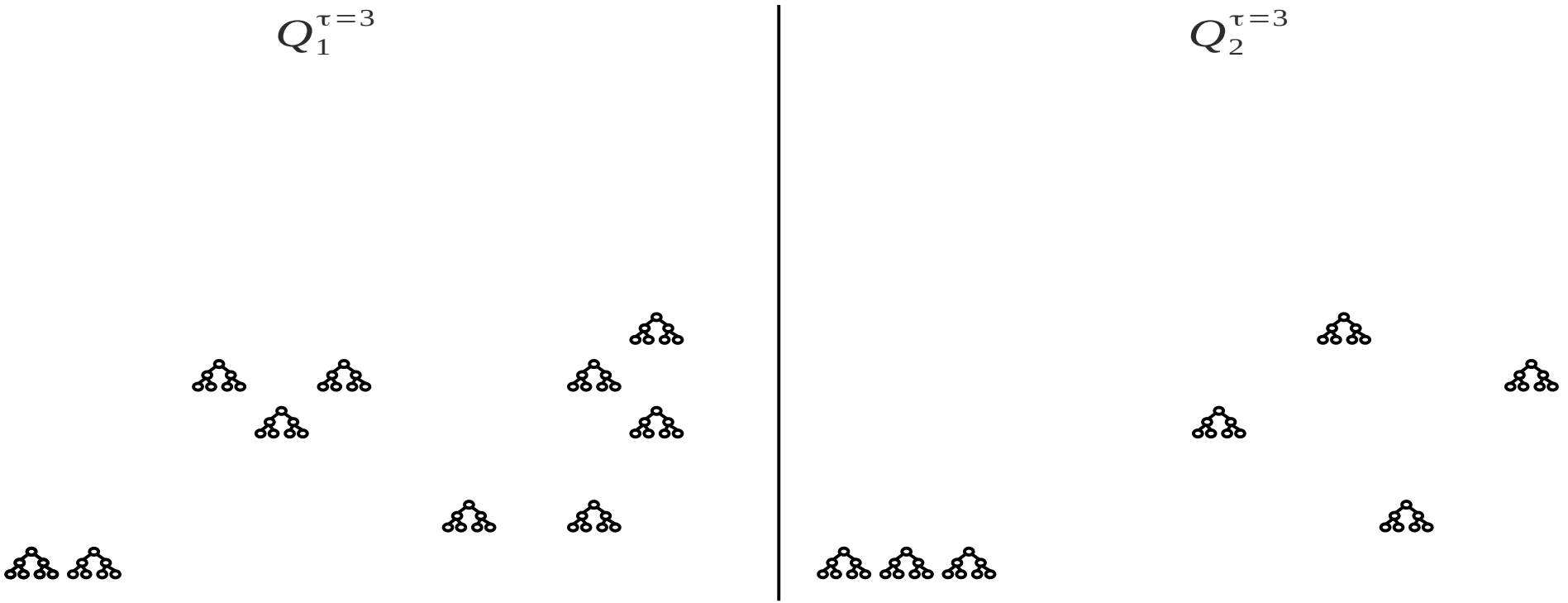
Dynamic Programming – 4

- Pruning agent 2…



Dynamic Programming – 4

- Etc…



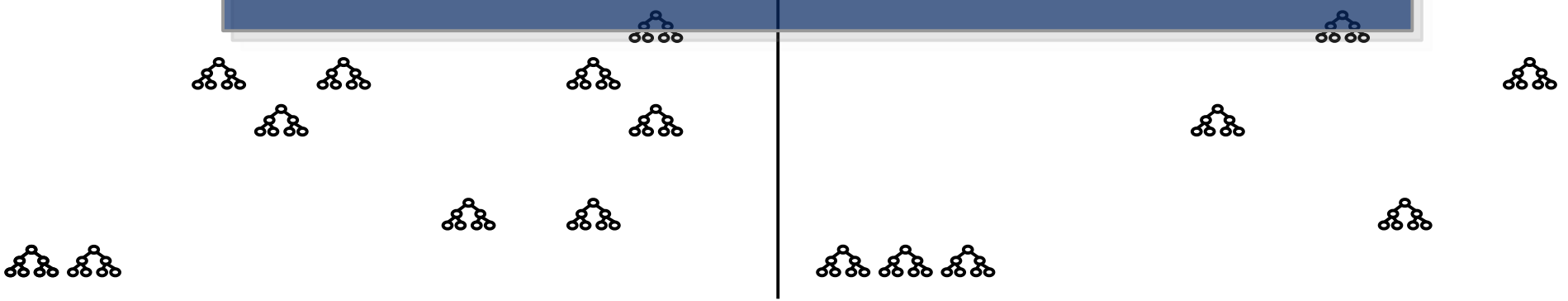
Dynamic Programming – 4

- Etc…

At the very end:

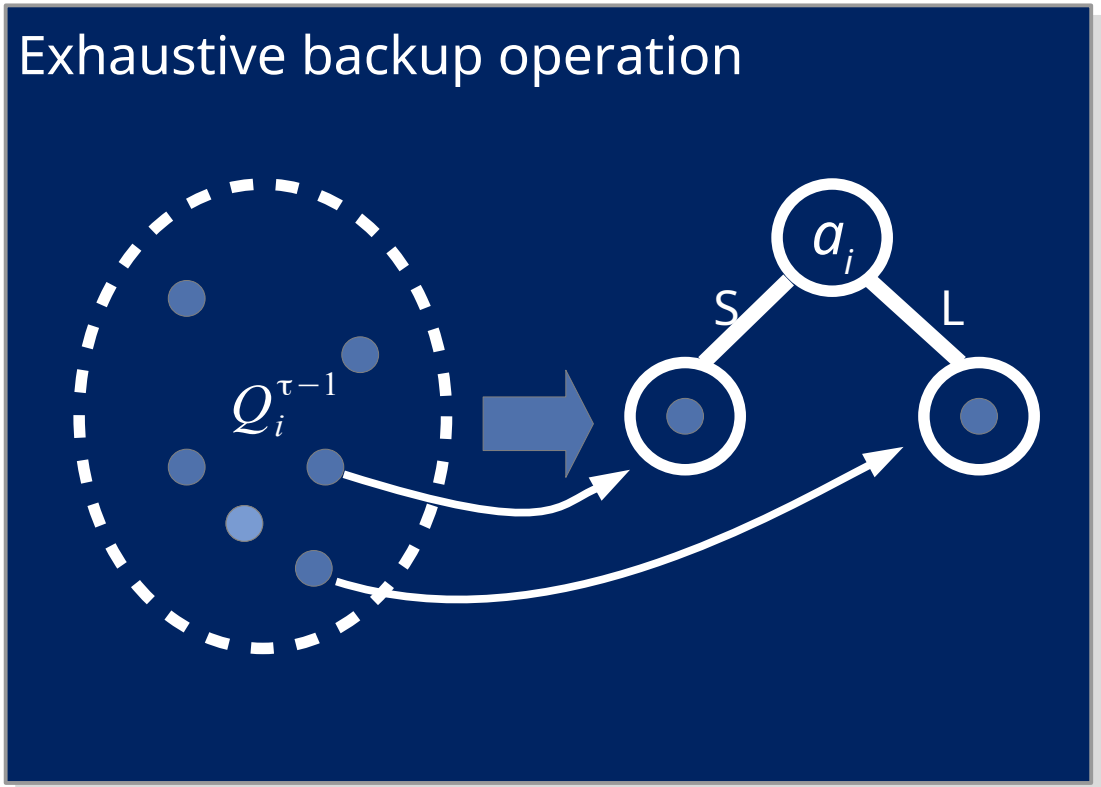
- evaluate all the remaining combinations of policies
- select the best one

$$V(q^{\tau=h}) = \sum_s b^0(s) V(s, q^{\tau=h})$$



Incremental Policy Generation – 1

- Bottleneck: exhaustive backup



$$Q_i^{\tau} = \cup_a \; Q_i^{\tau,a}$$

$$Q_i^{\tau,a} = (+)_o \; Q_i^{\tau,a,o}$$

$$Q_i^{\tau,a,o} = BackProject\left(Q_i^{\tau-1}\right)$$

Incremental Policy Generation – 1

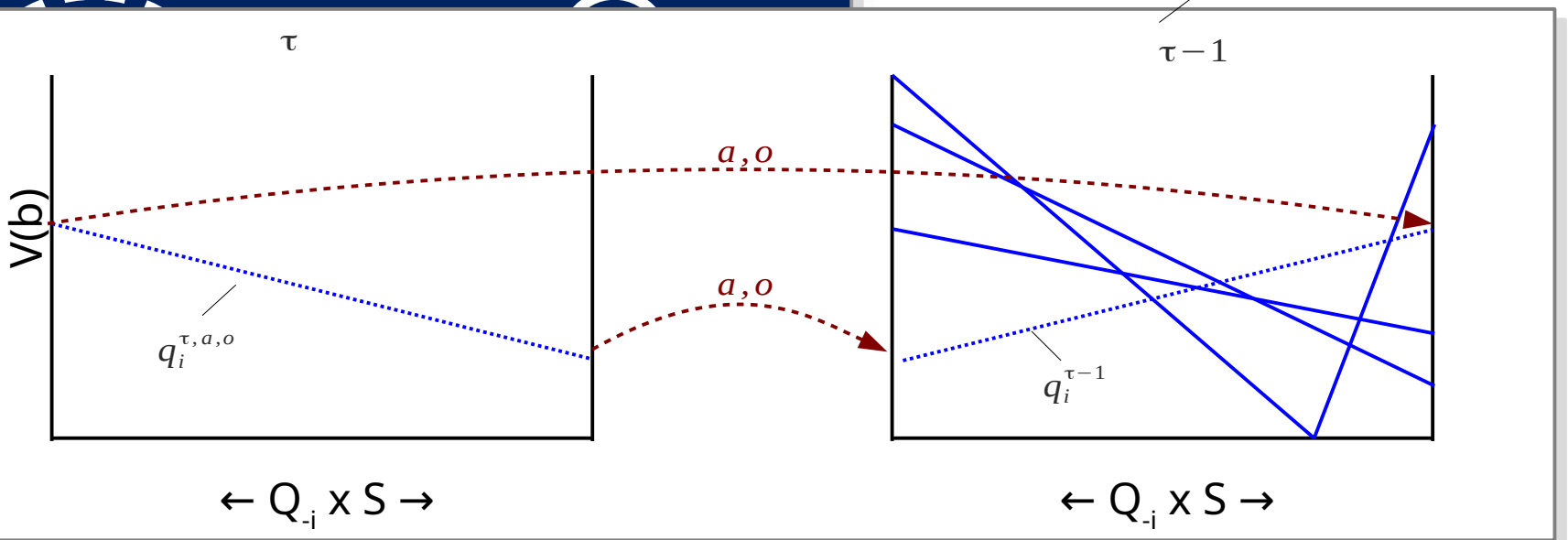
- Bottleneck: exhaustive backup

$$Q_i^{\tau} = \cup_a \; Q_i^{\tau,a}$$

$$Q_i^{\tau,a} = (+)_o \; Q_i^{\tau,a,o}$$

$$Q_i^{\tau,a,o} = BackProject\left(Q_i^{\tau-1}\right)$$

Exhaustive backup operation



Incremental Policy Generation – 2

- IPG [Amato et al 2009]:
  some states may be unreachable (for specific a,o)
  → prune only over reachable sub-space