EXPLOITING STRUCTURE IN DECISION MAKING UNDER THE LENS OF RECENT ADVANCES IN STARAI

MARCEL GEHRKE[1], NAZLI NUR KARABULUT, FLORIAN MARWITZ, AND TANYA BRAUN[2]

[1]Institute of Humanities-Centered Artificial Intelligence, University of Hamburg
[2]Computer Science Department, University of Münster

AGENDA

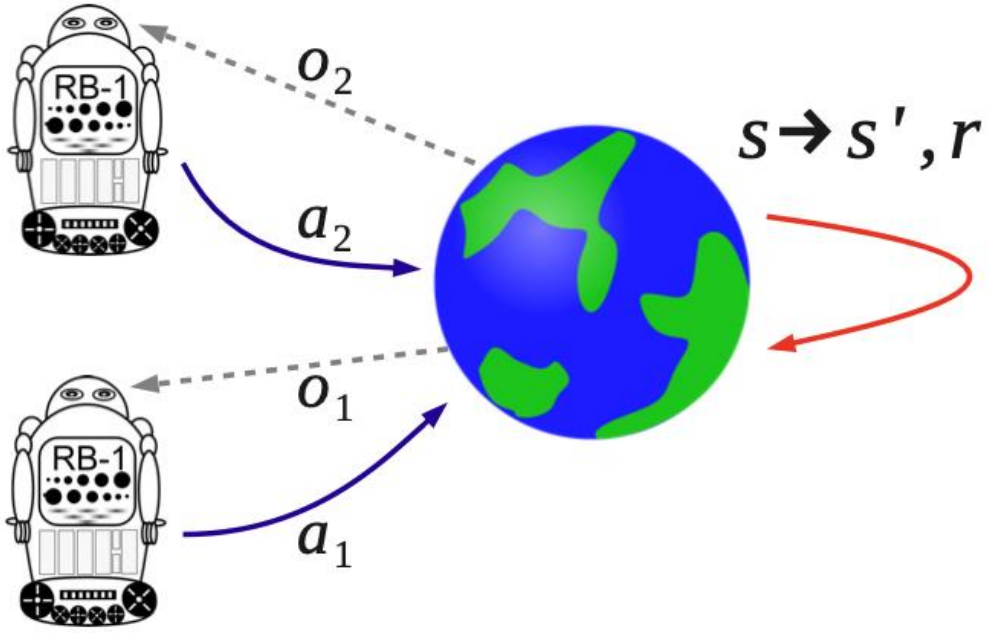
1. Introduction to Relational Models and Online Decision Making [Marcel]
2. Lifting Offline Decision Making [Flo]
3. Lifting Multi-Agent Decision Making [Nazlı Nur]
   - Lifting Decentralized Partially Observable Markov Decision Processes
   - Lifting Partially Observable Stochastic Games
4. Summary [Marcel]

LIFTING MULTI-AGENT DECISION MAKING

PART 3

MULTI-AGENT DECISION MAKING

Multi-agent decision-making refers to the process of making decisions in a setting where multiple autonomous agents interact and collaborate to achieve their individual or collective objectives.



Schematic representation of a Dec-POMDP

MULTI-AGENT DECISION MAKING: KEY CONCEPTS

- Agents: Autonomous entities capable of making decisions based on their perceptions and goals. These can be robots, software programs, or even humans.
- Environment: The space in which agents operate, which can be physical (like a factory floor) or virtual (like a simulated market).
- States and Actions: The state represents the current situation of the environment and agents, while actions are the possible moves or decisions agents can make.
- Observations: Due to partial observability, agents may have limited or noisy information about the environment or the states of other agents.
- Rewards and Objectives: Agents typically seek to maximize their individual rewards, which can be defined in terms of short-term gains or long-term objectives.

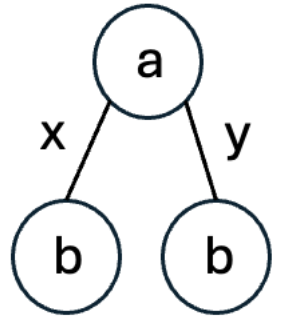
DEC-POMDPS: POLICIES

In a Dec-POMDPs, a policy is like a "roadmap" that guides each agent's decisions. It tells them what actions to take at each step, considering their observations and what the team has done so far.

- The policy helps agents work together and adapt to uncertain situations, making the best choices towards achieving their shared goals.

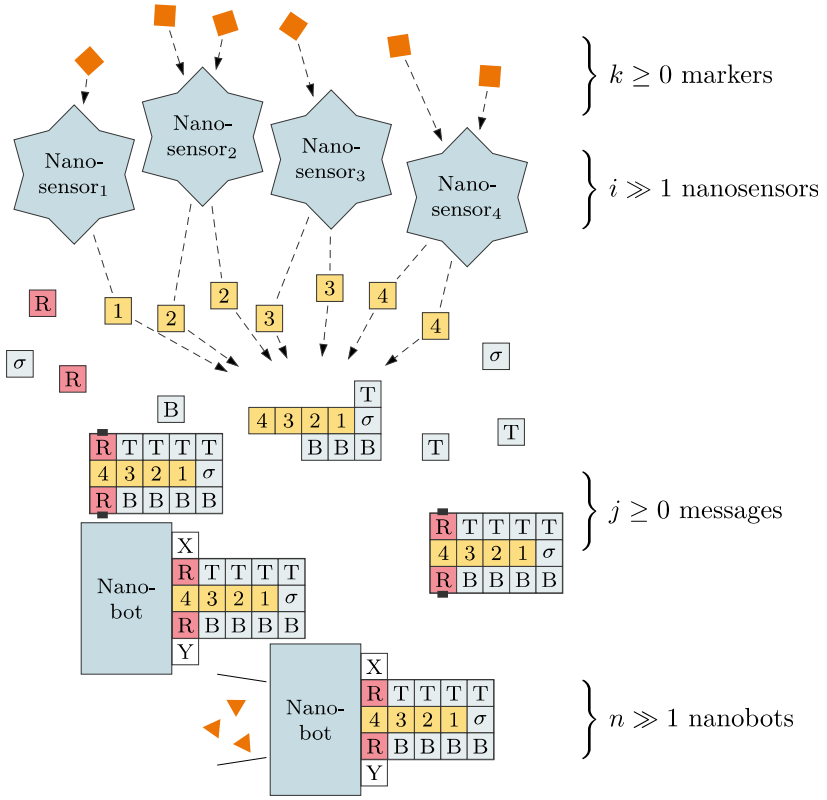
Action space = {a, b}
Observation space = {x, y}

Example of policy tree



EXAMPLE: MEDICAL NANOSCALE SYSTEMS

- Nanoscale systems regularly consist of $> 10{,}000$ nanoagents
  - Different types of agents: nanosensors, nanobots
- Application: DNA-based medical system
  - E.g., for diagnosis (modelled as an AND gate)
    - Nanosensors receptive to individual markers for a specific disease
      - Release individual tiles in presence of their individual markers
    - Tiles assemble themselves to form messages
    - Nanobots receptive to completely formed messages
      - Release markers of their own that signify presense of the disease
- Formal model necessary to argue about
  - Success rates
  - Sizes of agent sets

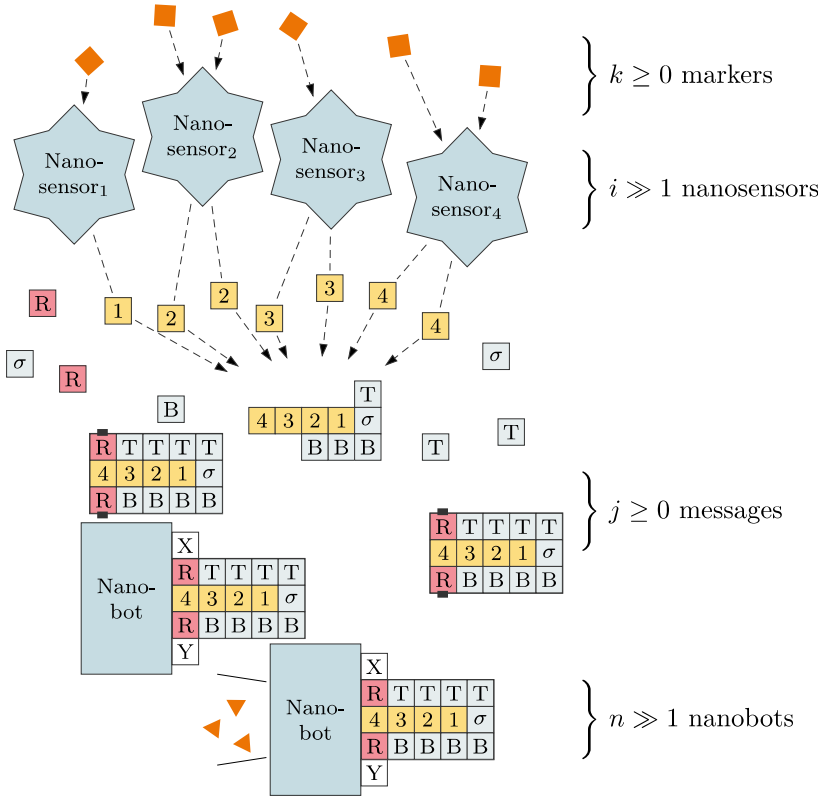


DECENTRALISED PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES (DEC-POMDPS)

- Dec-POMDP: tuple $(I, S, \{A_i\}_{i \in I}, \{O_i\}_{i \in I}, T, R, \Omega)$
  - I = a finite set of agents indexed $1, \ldots, N$
  - $\mathrm{dom}(S)$ = a finite set of states
  - A_i = a finite set of actions available to agent $i \in I$
    - $\vec{A} = \otimes_{i \in I} A_i$ set of joint actions
  - O_i = a finite set of observations available to agent $i \in I$
    - $\vec{O} = \otimes_{i \in I} O_i$ set of joint observations
  - Transition function $T(s', s, \vec{a}) = P(s' | s, \vec{a})$
  - Reward function $R(s)$ or $R(\vec{a}, s)$
  - Sensor model (observation function) $\Omega(\vec{o}, \vec{a}, s) = P(\vec{o} | \vec{a}, s)$
- Co-operative, decision-theoretic setting: Joint reward function R, joint state space S

EXAMPLE: MEDICAL NANOSCALE SYSTEMS AS A DEC-POMDP

- Set of agents I consisting of nanosensors, nanobots
- Observations O_i: markers / messages present (or not)
  - Noisy process → probabilistic behaviour
- Actions A_i: release of tiles / markers (or not)
  - Noisy process → probabilistic behaviour
- Environment → probabilistic behaviour
  - Presence in general of agents, markers, tiles, messages, or position more specifically → movement over time
- Reward: Qualitative measure
  - Positive diagnosis only in presence of disease



THE PROBLEM WITH DEC-POMDPS

- Space complexity
  - Transition model: $\mathcal{O}(s \cdot s \cdot a^N)$
  - Sensor model: $\mathcal{O}(s \cdot o^N)$ or $\mathcal{O}(s \cdot o^N \cdot a^N)$
  - Reward function: $\mathcal{O}(s)$ or $\mathcal{O}(s \cdot a^N)$
- Runtime complexity of brute-force search
  - Evaluation cost of a joint policy: $\mathcal{O}(s \cdot o^{Nh})$
  - Policy space: $\mathcal{O}\left(a^{\frac{N(o^h-1)}{o-1}}\right)$
- Exponential dependence on number of agents

Notations

$s = |S|$
State space size

$a = \max_{i \in I} |A_i|$
Largest individual action space size

$o = \max_{i \in I} |O_i|$
Largest individual observation space size

h
Horizon

N
the number of agents

SOLUTION METHODS FOR DEC-POMDPS

In a Dec-POMDPs,

- Dynamic Programming (DP)
  - DP for Dec-POMDPs incrementally builds policies for each stage but faces exponential complexity due to the rapidly increasing number of policy combinations (Frans A. Oliehoek and Christopher Amato. 2015).
- Multi-Agent A* (MAA*) (Szer, D., Charpillet, F., & Zilberstein, S. 2012)
  - MAA* is a method that uses the temporal structure in policies to search through the space of past joint policies φ_t.
- Bounded Dynamic Programming (BDP) (Amato, C., Carlin, A., & Zilberstein, S. 2007)
  - BDP manages memory constraints by using ε-pruning to approximate solutions, allowing for controlled reductions in solution quality.
- Memory-Bounded Dynamic Programming (MBDP) (Seuken, S.,& Zilberstein, S. 2007)
  - MBDP effectively manages large state spaces by limiting the number of sub-tree policies and applying heuristics to optimize performance.
- Collaborative Bayesian Game (CBG)
  - CBG has agents using their individual histories to maximize identical payoffs (Frans A. Oliehoek and Christopher Amato. 2015).

LIFTING DEC-POMDPS

COUNTING AND ISOMORPHIC DEC-POMDPS

AGENT TYPES & PARTITIONED DEC-POMDPS

- Types: Agents with the same sets of actions and observations
  - E.g., two nanosensors $1,2$ receptive to the same marker and releasing the same tile
    - $A_1 = A_2 = \{0,1\}$; 0: do nothing, 1: release tile
    - $O_1 = O_2 = \{0,1\}$; 0: marker not present, 1: marker present

→ Partitions the set of agents regarding actions, observations

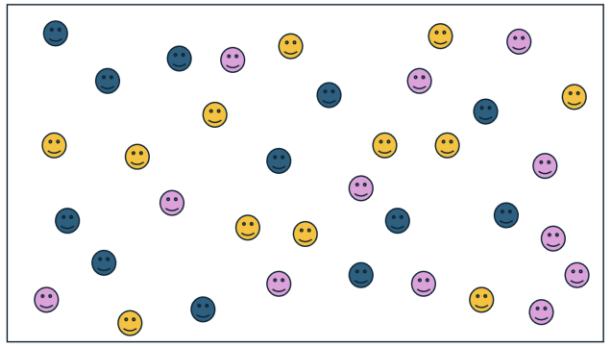
  - Agent set $I = \{I_1, \dots, I_K\}$ with $I_1, \dots, I_K$ a partitioning of I ($I = \cup_k I_k$, $I_k \cap I_{k'} = \emptyset, I_k \neq \emptyset$)
  - For each partition I_k: one set of actions A_k, one set of observations O_k for all agents in I_k
  - Expectation that $K \ll N$

- Additional constraints / assumptions on same behaviour in T, R, Ω

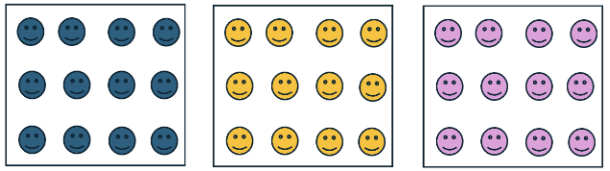
→ Partitions the set of agents completely, enabling more compact encodings

- How?

Set of agents



Partitions



COUNTING DEC-POMDPS

- Simplifies complexity by grouping agents based on state and action counts.
- Reduces state and action space, making it more tractable to compute optimal policies in scenarios with a large number of agents.
- The reduction in the number of distinct states and actions allows for more efficient policy computation.

Braun, T. et al 22

COUNTING DEC-POMDPS

- Counting constraint / assumption in T, R, Ω
  - Formal: All permutations $\sigma(\vec{a}_k)$ of a partition action $\vec{a}_k$ map to the same probability
  - Enables counting how many agents do something and not which in particular did
    - Encode in a histogram $[\#(a_1), \dots, \#(a_l)]$ how many agents did actions $A_k = \{a_1, \dots, a_l\}$
    - Number of histograms $\binom{|I_k|+l-1}{l-1} \leq |I_k|^l$

S	S'	$A_1^{\#}$	$\bar{T}(s, s', a_1') = P(s'\|s, a_1')$
0	0	[0,2]	0.01
0	0	[1,1]	0.02
0	0	[2,0]	0.03
0	1	[0,2]	0.015
0	1	[1,1]	0.012
0	1	[2,0]	0.01
1	0	[0,2]	0.01
⋮			

S	S'	A_1	A_2	$T(s, s', a_1, a_2) = P(s'\|s, a_1, a_2)$
0	0	0	0	0.01
0	0	0	1	0.02
0	0	1	0	0.02
0	0	1	1	0.03
0	1	0	0	0.015
0	1	0	1	0.012
0	1	1	0	0.012
0	1	1	1	0.01
1	0	0	0	0.01
⋮				

Braun, T. et al 22

COUNTING DEC-POMDPS

- Complexity-wise, with $n = \max_k |I_k|$
  - Transition model: $\mathcal{O}(s \cdot s \cdot n^{Ka})$
  - Sensor model: $\mathcal{O}(s \cdot n^{Ko})$
  - Reward function: $\mathcal{O}(s \cdot n^{Ka})$
  - Evaluation cost: $\mathcal{O}(s \cdot n^{Koh})$
  - Reduction if $K \ll N$
- Unfortunately,
  - Policy space: $\mathcal{O}\left(n^{\frac{aK(n^{ho}-1)}{n^{o}-1}}\right)$
- Ongoing research how to use counting efficiently

S	S'	$A_1^{\#}$	$\bar{T}(s, s', a_1') = P(s'\|s, a_1')$
0	0	[0,2]	0.01
0	0	[1,1]	0.02
0	0	[2,0]	0.03
0	1	[0,2]	0.015
0	1	[1,1]	0.012
0	1	[2,0]	0.01
1	0	[0,2]	0.01
⋮			

S	S'	A_1	A_2	$T(s, s', a_1, a_2) = P(s'\|s, a_1, a_2)$
0	0	0	0	0.01
0	0	0	1	0.02
0	0	1	0	0.02
0	0	1	1	0.03
0	1	0	0	0.015
0	1	0	1	0.012
0	1	1	0	0.012
0	1	1	1	0.01
1	0	0	0	0.01
⋮				

Braun, T. et al 22

ISOMORPHIC DEC-POMDPS

- *Dec-POMDPs Challenge:* Complexity increases with more agents.
- *Isomorphism Approach:* Groups agents with indistinguishable agents to reduce complexity.
- *Efficiency Gains:* Leverages structural similarities and assumes conditional independence within groups, simplifying decision-making.

Braun, T. et al 22

ISOMORPHIC DEC-POMDPS

- Isomorphic constraint / assumption in T, R, Ω:
  Conditional independence between agents of a partition given joint state
  → Enables factorisation of T, R, Ω
  - E.g., $T(s, s', a_1, a_2) = T_1(s, s', a_1) \cdot T_2(s, s', a_2) = \prod_{i \in I_k} T'(s, s', a_i)$

$$T_1 = T_2 = T'$$

S	S'	A_i	$T'(s, s', a_i) = P(s' \mid s, a_i)$
0	0	0	0.01
0	0	1	0.03
0	1	0	0.015
0	1	1	0.01
1	0	0	0.01
		⋮	

- Space complexities
  - Transition model: $\mathcal{O}(s \cdot s \cdot a^K)$
  - Sensor model: $\mathcal{O}(s \cdot o^K)$
  - Reward function: $\mathcal{O}(s \cdot a^K)$
- Ongoing research how to solve isomorphic DecPOMDPs efficiently

Braun, T. et al 22

INTERIM SUMMARY: STRUCTURE BY GROUPS IN THE AGENT SET

- Types of agents with identical action and observation space
- Partitioned DecPOMDP if agent types + constraints of transition / sensor / reward function
- Counting DecPOMDP
  - Permutations of actions of agents of the same partition map to the same probability / reward
  - Count occurrences → encode in histograms
- Isomorphic DecPOMDP
  - Further independences between agents of a partition

Space complexity polynomial at worst but using compact encoding for efficient decision making not yet solved

PARTIALLY OBSERVABLE STOCHASTIC GAME (POSG)

DIFFERENCES BETWEEN DEC-POMDP AND POSG

In Dec-POMDPs;

- *Single Reward Function:* All agents share a common reward function. This means the agents are working cooperatively towards a common goal.
- *Objective:* The primary objective is to maximize the cumulative reward shared by all agents.

In POSGs;

- *Individual Reward Function:* Each agent has its own reward function, reflecting its individuality.
- *Objective:* Each agent aims to maximize its own reward.

PARTIALLY OBSERVABLE STOCHASTIC GAME (POSG)

- Partially observable stochastic game (POSG)
  - Dec-POMDP $(I, S, \{A_i\}_{i \in I}, \{O_i\}_{i \in I}, T, \cancel{R}, \Omega)$ but with individual reward functions $\{R_i\}_{i \in I}$
  - Reward function R_i for each agent $i \in I$

- For self-interested or adversarial acting
  - Local optimum not guaranteed to be the global optimum
  - Dominant strategy equilibrium: best response (highest utility) given any state
    - Not guaranteed to exist
    - Prisoner's dilemma: Dominant strategy not always pareto-optimal strategy
  - Nash Equilibrium: No agent has incentive to change its strategy if no other agent changes its strategy
    - Always exists

DYNAMIC PROGRAMMING FOR POSG

The goal of applying dynamic programming to POSGs is to extend the principles used in partially observable Markov decision processes (POMDPs) to the multi-agent setting of POSGs (Hansen, Bernstein, and Zilberstein 2004).

Multi-agent dynamic programming solves a finite-horizon POSG by iteratively eliminating weakly-dominated strategies using a dynamic programming operator.

This approach allows for optimizing decision-making in environments where agents have incomplete information about the state and the actions of other agents.

- Extend dynamic programming with using partially observable Markov decision processes (POMDPs).
- Uses a multi-agent dynamic programming operator.

MULTI-AGENT DYNAMIC PROGRAMMING OPERATOR

The multi-agent dynamic programming operator is an extension of the single-agent operator used in POMDPs, adapted to handle the interactions between multiple agents.

This operator iteratively refines the strategies of the agents by considering their beliefs and the potential actions and observations of other agents.

Steps:

1. Initialize: start with initial policy trees and value vectors.
2. Exhaustive Backup: generate depth t+1 policy trees from depth t trees.
3. Pruning: eliminate dominated policy trees using linear programming tests.

MULTI-AGENT DYNAMIC PROGRAMMING OPERATOR

Exhaustive Backup: Build policy trees

Algorithm 1 Multi-agent Dynamic Programming Operator

function MA-DP(set of policies Π_i^{t-1} for each agent $i \in \boldsymbol{I}$ with value vectors $\boldsymbol{V}_i^{t-1}$)
  $\Pi_i^t \leftarrow$ Perform exhaustive backup using Π_i^{t-1} for each agent $i \in \boldsymbol{I}$
  $\boldsymbol{V}_i^t \leftarrow$ Calculate new value vectors for each agent $i \in \boldsymbol{I}$
  while $\exists \pi_{i,j}^t \in \Pi_i^t$: Eq. (1) holds **do**
    $\Pi_i^t \leftarrow \Pi_i^t \setminus \{\pi_{i,j}^t\}$, $\boldsymbol{V}_i^t \leftarrow \boldsymbol{V}_i^t \setminus \{v_{i,j}^t\}$
  return $\{(\Pi_i^t, \boldsymbol{V}_i^t)\}_{i \in \boldsymbol{I}}$

MULTI-AGENT DYNAMIC PROGRAMMING OPERATOR

Compute the value of each policy

Algorithm 1 Multi-agent Dynamic Programming Operator

function MA-DP(set of policies Π_i^{t-1} for each agent $i \in \boldsymbol{I}$ with value vectors $\boldsymbol{V}_i^{t-1}$)
  $\Pi_i^t \leftarrow$ Perform exhaustive backup using Π_i^{t-1} for each agent $i \in \boldsymbol{I}$
  $\boldsymbol{V}_i^t \leftarrow$ Calculate new value vectors for each agent $i \in \boldsymbol{I}$
  while $\exists \pi_{i,j}^t \in \Pi_i^t$: Eq. (1) holds **do**
    $\Pi_i^t \leftarrow \Pi_i^t \setminus \{\pi_{i,j}^t\}$, $\boldsymbol{V}_i^t \leftarrow \boldsymbol{V}_i^t \setminus \{v_{i,j}^t\}$
  return $\{(\Pi_i^t, \boldsymbol{V}_i^t)\}_{i \in \boldsymbol{I}}$

MULTI-AGENT DYNAMIC PROGRAMMING OPERATOR

Pruning: eliminate dominated policy trees using linear programming tests.

Algorithm 1 Multi-agent Dynamic Programming Operator

function MA-DP(set of policies Π_i^{t-1} for each agent $i \in \boldsymbol{I}$ with value vectors $\boldsymbol{V}_i^{t-1}$)
  $\Pi_i^t \leftarrow$ Perform exhaustive backup using Π_i^{t-1} for each agent $i \in \boldsymbol{I}$
  $\boldsymbol{V}_i^t \leftarrow$ Calculate new value vectors for each agent $i \in \boldsymbol{I}$
  while $\exists \pi_{i,j}^t \in \Pi_i^t$: Eq. (1) holds **do**
    $\Pi_i^t \leftarrow \Pi_i^t \setminus \{\pi_{i,j}^t\}$, $\boldsymbol{V}_i^t \leftarrow \boldsymbol{V}_i^t \setminus \{v_{i,j}^t\}$
  return $\{(\Pi_i^t, \boldsymbol{V}_i^t)\}_{i \in \boldsymbol{I}}$

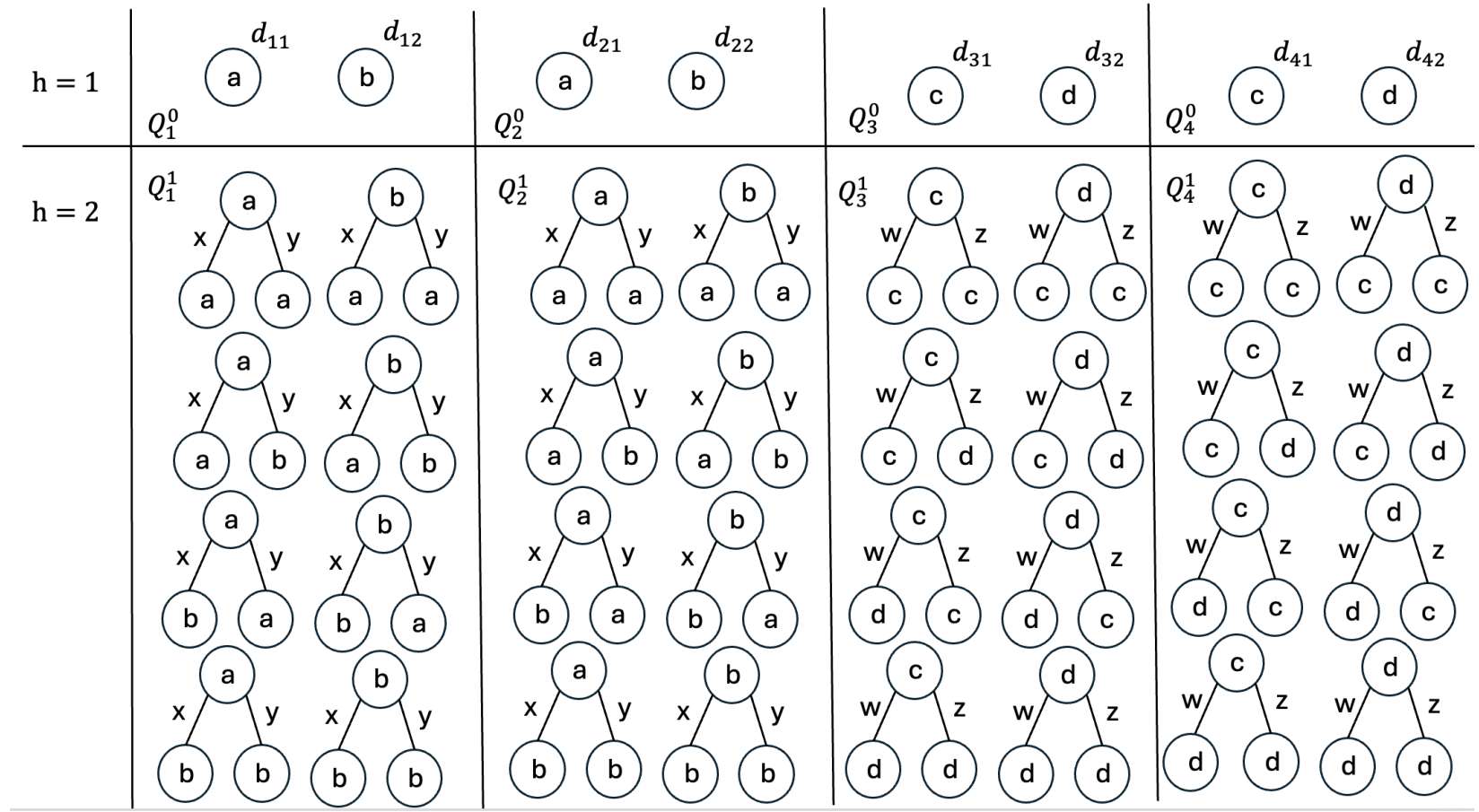
Equation 1:

$$\forall s \in ran(S), \boldsymbol{\pi}_{-i}^t \in \boldsymbol{\Pi}_{-i}^t \exists \pi_{i,j'}^t \in \Pi_i^t : V_i(s, \pi_{i,j}^t, \boldsymbol{\pi}_{-i}^t) \leq V_i(s, \pi_{i,j'}^t, \boldsymbol{\pi}_{-i}^t), \quad (1)$$

While loop: "idea of equation"

DYNAMIC PROGRAMMING EXHAUSTIVE BACKUP

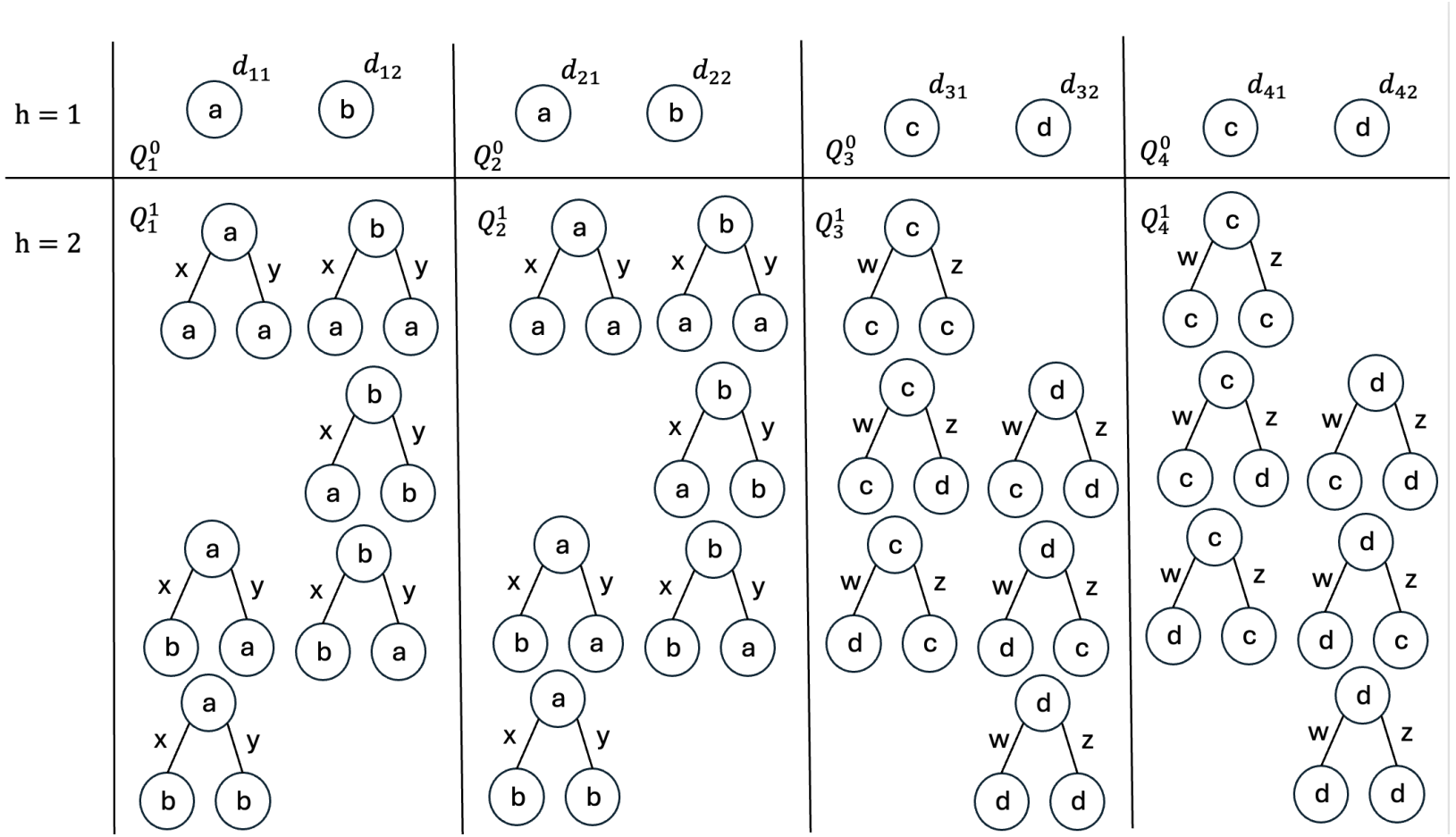
This is the exhaustive backup where you build all possible strategies



DYNAMIC PROGRAMMING EXHAUSTIVE BACKUP

After exhaustive backup, we prune weakly dominated policies in each agents

Using linear programming tests



LIFTING PARTIALLY OBSERVABLE STOCHASTIC GAME

ISOMORPHIC POSG

ISOMORPHIC PARTIALLY OBSERVABLE STOCHASTIC GAME

An isomorphic POSG is a special type of POSG where agents are partitioned into groups with symmetric behaviour.

This allows for a more compact representation, which depends on the number of partitions, not the number of agents.

An isomorphic $POSG\ \bar{M}$ is a tuple $(\bar{\boldsymbol{I}}, S, \bar{\boldsymbol{A}}, \bar{\boldsymbol{O}}, \bar{\boldsymbol{T}}, \bar{\boldsymbol{R}})$, with

- $\bar{\boldsymbol{I}}$ a partitioning $\{\Im_k\}_{k=1}^K$ of agents, $n_k = |\Im_k|$ and $|\bar{\boldsymbol{I}}| = \sum_k n_k = N$,
- S a random variable with a set of states
- $\bar{\boldsymbol{A}} = \{\bar{A}_k\}_{k=1}^K$ a set of decision random variable $\bar{A}_k$, each with possible actions as range, $ran(A_i) = ran(\bar{A}_k) \forall_i \in \Im_k$, and $ran\,(\boldsymbol{A}_k) = \times_{i \in \Im_k}\ ran(A_k)$,
- $\bar{O} = \{\bar{O}_k\}_{k=1}^K$ a set of random variables $\bar{O}_k$, each with a set of observations as range, $ran(O_i) = ran(\bar{O}_k) \forall_i \in \Im_k$, and $ran\,(\boldsymbol{O}_k) = \times_{i \in \Im_k}\ ran(O_k)$,
- $\bar{\boldsymbol{T}} = \{\bar{T}_k(S, S', \bar{\boldsymbol{A}}, \bar{\boldsymbol{O}})\}_{k=1}^K$ a set of probability distributions $\bar{T}_k(S, S', \bar{\boldsymbol{A}}, \bar{\boldsymbol{O}}) = P(S', \bar{\boldsymbol{O}} | S, \bar{\boldsymbol{A}})$, and
- $\bar{\boldsymbol{R}} = \{\bar{R}_k(S, \bar{\boldsymbol{A}})\}_{k=1}^K$ a set of reward functions.

Karabulut & Braun 24

LIFTING DYNAMIC PROGRAMMING FOR ISOMORPHIC POSG

Purpose: extend multi-agent dynamic programming to handle isomorphic POSGs efficiently by leveraging the partitioned structure of agents.

Lemma: indistinguishable agents (within the same partition) share the same set of policies, beliefs, and state values.

Theorem: policies weakly dominated for one agent in a partition are weakly dominated for other agents in the same partition.

Lifted Dynamic Programming Operator:

Input: sets of policies and corresponding value vectors for each partition.

Process:

- Perform exhaustive backup using policies for each partition.
- Calculate new value vectors
- Prune weakly dominated policies per partition.

Output: Updated sets of policies and value vectors for each partition.

Karabulut & Braun 24

DYNAMIC PROGRAMMING EXHAUSTIVE BACKUP

Isomorphic case

1.Column shows the partition 1:

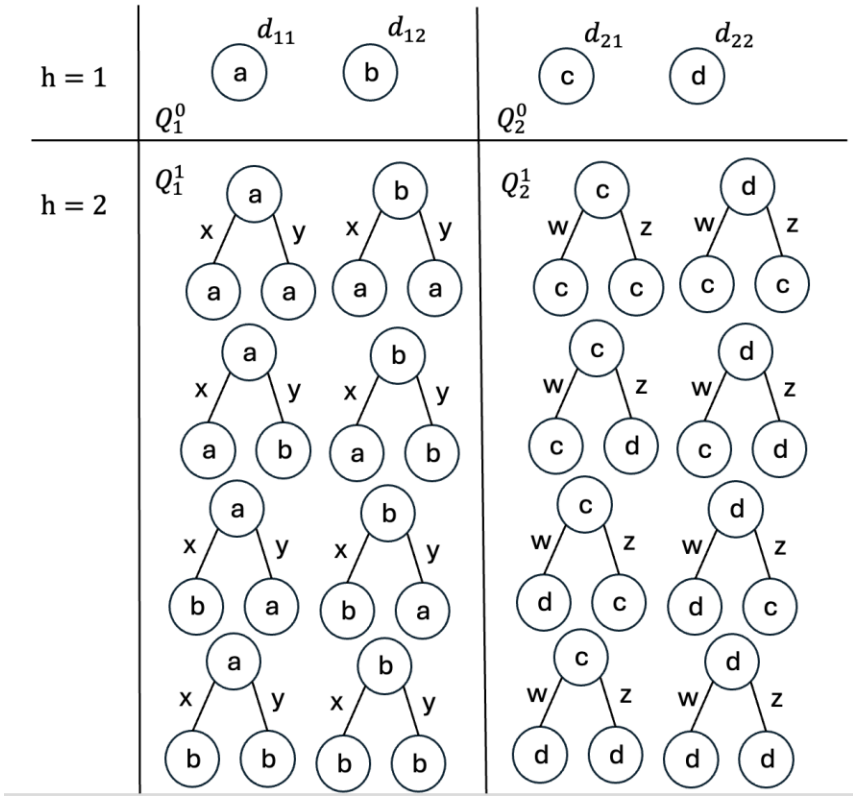
Action space = {a, b}

Observation space = {x, y}

2.Column shows the partition 2:

Action space = {c, d}

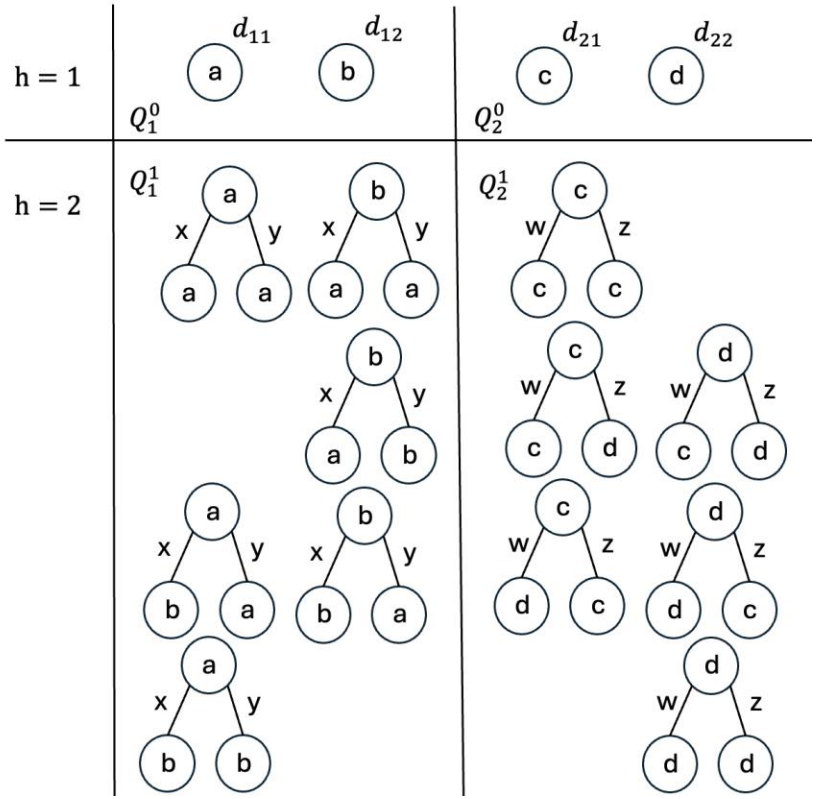
Observation space = {w, z}



Karabulut & Braun 24

DYNAMIC PROGRAMMING EXHAUSTIVE BACKUP

After exhaustive backup, we prune weakly dominated policies in each partitions.



Karabulut & Braun 24

INTERIM SUMMARY: POSG AND ISOMORPHIC POSG

- POSG
  - POSGs model multi-agent systems where each agent, with limited observations, makes decisions based on incomplete information.
  - Dynamic programming: exhaustive backup + pruning

- Isomorphic POSG
  - A special type of POSG where agents are partitioned into groups (partitions) that exhibit symmetric behavior, allowing for more efficient modeling and computation.
  - Lifting dynamic programming for isomorphic POSG: exhaustive backup for partitions + pruning

AGENDA

1. Introduction to Relational Models and Online Decision Making [Marcel]
2. Lifting Offline Decision Making [Flo]
3. Lifting Multi-Agent Decision Making [Nazlı Nur]
   - Lifting Decentralized Partially Observable Markov Decision Processes
   - Lifting Partially Observable Stochastic Games
4. Summary [Marcel]

AGENDA

1. Introduction to Relational Models and Online Decision Making [Marcel]
2. Lifting Offline Decision Making [Flo]
3. Lifting Multi-Agent Decision Making [Nazlı Nur]
4. Summary [Marcel]

BIBLIOGRAPHY

ORDERED ALPHABETICALLY

BIBLIOGRAPHY

- Amato, C., Carlin, A., & Zilberstein, S. (2007, May). Bounded dynamic programming for decentralized POMDPs. In *Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*.
- Braun, T., Gehrke, M., Lau, F., & Möller, R. (2022, August). Lifting in multi-agent systems under uncertainty. In *Uncertainty in Artificial Intelligence* (pp. 233-243). PMLR.Frans A. Oliehoek and Christopher Amato. 2015. *A Concise Introduction to Decentralised POMDPs*. Springer.
- Frans A. Oliehoek and Christopher Amato. 2015. *A Concise Introduction to Decentralised POMDPs*. Springer.
- Hansen, E.A., Bernstein, D.S., Zilberstein, S.: Dynamic programming for partially observable stochastic games. In: AAAI-04 Proc. of the 19th National Conference on Artificial Intelligence. vol. 4, pp. 709–715 (2004).
- Karabulut, N.N., and Braun, T.: Lifting Partially Observable Stochastic Games. In the International Conference on Scalable Uncertainty Management 2024.

BIBLIOGRAPHY

- Seuken, S., & Zilberstein, S. (2007, February). Memory-Bounded Dynamic Programming for DEC-POMDPs. In *IJCAI* (pp. 2009-2015).
- Szer, D., Charpillet, F., & Zilberstein, S. (2012). MAA*: A heuristic search algorithm for solving decentralized POMDPs. *arXiv preprint arXiv:1207.1359*.