Reinforcement Learning for Complex Sequential Decision Problems[1]

Workshop TED4LAT, 2024

Régis Sabbadin (Inrae-MIAT, Toulouse) with
Prasanna Maddila, Patrick Taillandier, Meritxell Vinyals
Eric Casellas and Patrick Chabrier (Inrae-MIAT, Toulouse)
Orlane Rossini (IMAG, Montpellier and Inrae-MIAT, Toulouse)
Alice Cleynen and Benoîte de Saporta (IMAG, Montpellier)

[1]Work supported by PRC project Project-ANR-21-CE40-0005 HSMM-INCA and by PRCI ANR-DFG Project-ANR-22-CE92-0011 CHIP-GT

Principles of Reinforcement Learning

What is reinforcement Learning?

Definition (Wikipedia)

Reinforcement learning (RL) is an interdisciplinary area of machine learning and optimal control concerned with how an intelligent agent ought to take actions in a dynamic environment in order to maximize the cumulative reward.
Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

RL is based on the Markov Decision Process (MDP) framework, which allows to represent sequential decision problems under uncertainty.

Markov Decision Processes

A MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{D}, P, r/c \rangle$, where:

- $\mathcal{S}$ is a (generally finite) set of states of the controlled system.
- $\mathcal{D}$ is a (generally finite) set of decisions.
- P is a transition function. $P(s'|s,d)$ is the probability that the system, initially in state $s \in \mathcal{S}$ transitions to state $s' \in \mathcal{S}$ when decision $d \in \mathcal{D}$ is applied.
- r (resp. c) is a reward (resp. cost) function associated to transitions. $r(s,d,s')$ is the reward obtained when a transition (s,d,s') occurs.

Solving a MDP amounts to finding a policy $\pi : \mathcal{S} \times \{0, \ldots, H-1\} \to \mathcal{D}$, optimizing the expected sum of rewards/costs obtained during a finite number of steps[2], H.

$$V^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{H-1} r(S_t, \pi^t(S_t), S_{t+1}) | S_0 = s, \pi\right]$$

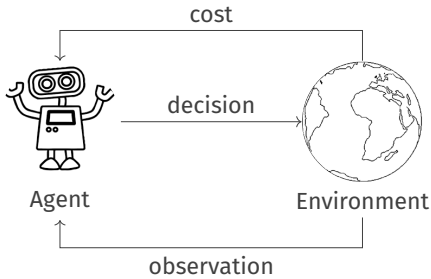
[2]Classical other objective exist for infinite horizon: Average cost, discounted cost...

Solving Markov Decision Processes

In the finite state/decision spaces case, there exists an optimal policy π^* (such that $V^{\pi^*}(s) \geq V^{\pi}(s), \forall \pi, \forall s \in \mathcal{S}$).
Furthermore,

- π^* can be computed efficiently (i.e. in time polynomial in $H, |\mathcal{S}|, |\mathcal{D}|$), using, e.g. dynamic programming, or linear programming.
- However, when $\mathcal{S}, \mathcal{D}$ are multidimensionnal or continuous, or when the problem is partially observed, or when the model is unknown and accessible only through simulation...
- **Dynamic Programming can no more be efficiently applied!**
- **(Deep) Reinforcement Learning approaches may be the solution...**

Reinforcement Learning in MDP (ex: infinite H. Q-learning)



$$\underbrace{Q^{\pi}(s,d)}_{\text{Policy Q-function}} = \underbrace{\mathbb{E}^{\pi}\left[\sum_{t=0}^{+\infty}\gamma^{t}c(S_{t-1},D_t,S_t)|S_0=s,d\right]}_{\text{Value of applying } d \text{ in } s \text{ and then following } \pi}$$

The optimal policy is progressively computed from observed transitions $\langle s,d,s',c\rangle$

$$\underbrace{Q^{\star}(s,d)}_{\text{Q function}} = \min_{\pi\in\Pi} Q^{\pi}(s,d)$$

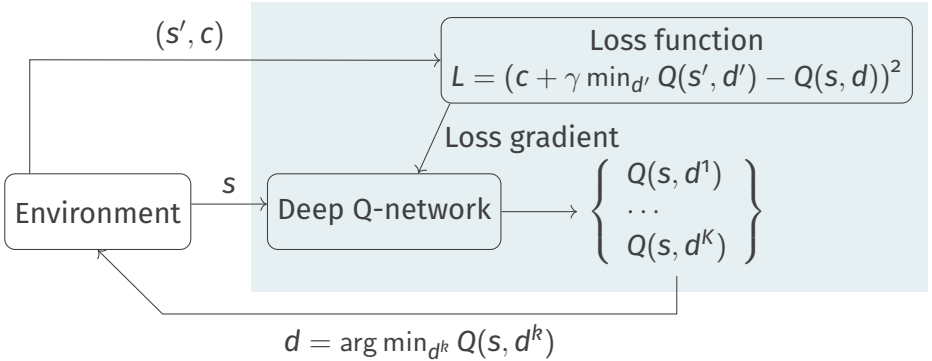
$$\pi^{\star}(s) = \arg\min_{d\in\mathcal{D}} Q^{\star}(s,d)$$

$$\langle s,d,s',c\rangle \Rightarrow Q^{n}(s,d) \leftarrow (1-\alpha)Q^{n-1}(s,d) + \alpha\left(c+\gamma\min_{d'}Q^{n-1}(s',d')\right)$$

(Q^n) converges to Q^* when $n \to +\infty$!

Deep Q-Network algorithm (infinite horizon)

- Uses a neural network to "learn" the Q-function from observed trajectories
- **Handles multidimensional continuous state spaces (finite decision space)**
- **Data Consuming !**



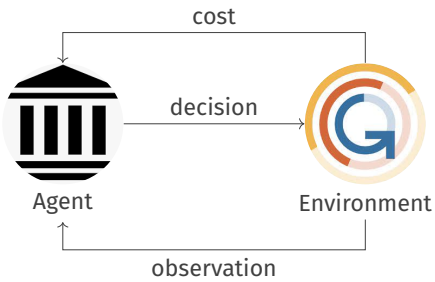
Some challenges in RL raised by applications

- **Partial observability**: Innovation adoption in agriculture
- **Non-Markovian process**: Cancer treatment follow-up
- **Multiple Learning Agents**: Anti-poaching

Challenge 1: partial observability
Innovation Adoption in agriculture

Innovation adoption example[3]

Objective: Design of public policies aiming at increase the adoption rate of an innovation (communicating water meters in irrigated agriculture)

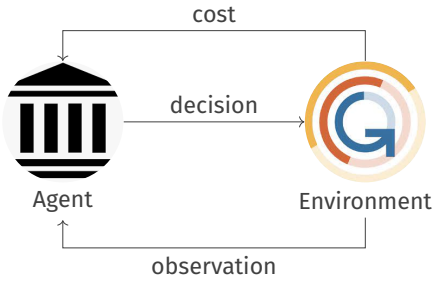


- Agent: Public Authority
- State: mental state of farmers and interaction network (complex state, unobserved)
- Reward: Increase of #adopters
- Decision: parameters of 3 levers. Environmental protection awareness, training, subsidies
- Observation: #adopters, available budget, remaining time steps

[3][Vinyals et al., 2023] Towards AI-designed innovation diffusion policies using agent based simulations and reinforcement learning: the case of digital tool adoption in agriculture

Innovation adoption example[3]

Objective: Design of public policies aiming at increase the adoption rate of an innovation (communicating water meters in irrigated agriculture)



- The state is perceived only through partial observations
- Complex transitions, simulated by the Gama simulator
- The mental reconstruction of the state depends on histories of past actions/observations
- States and actions are continuous / high-dimensional
- Fits the POMDP framework!

[3] [Vinyals et al., 2023] Towards AI-designed innovation diffusion policies using agent based simulations and reinforcement learning: the case of digital tool adoption in agriculture

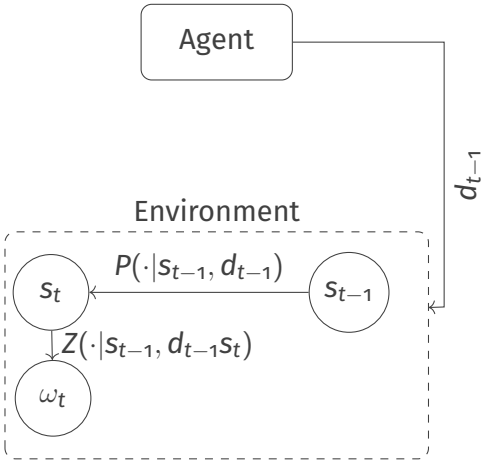
Partially Observable Markov Decision Process

POMDP Definition

A POMDP is defined by a tuple $(\mathcal{S}, \mathcal{D}, P, \Omega, Z, c)$.

- State of the process: $s \in \mathcal{S}$;
- Decision: $d \in \mathcal{D}$;
- Transition probability: $P(s'|s, d)$;
- Observation: $\omega \in \Omega$;
- Observation function: $Z(\omega|s, d, s')$;
- Cost function: $c(s, d, s')$.

Partially Observable Markov Decision Process

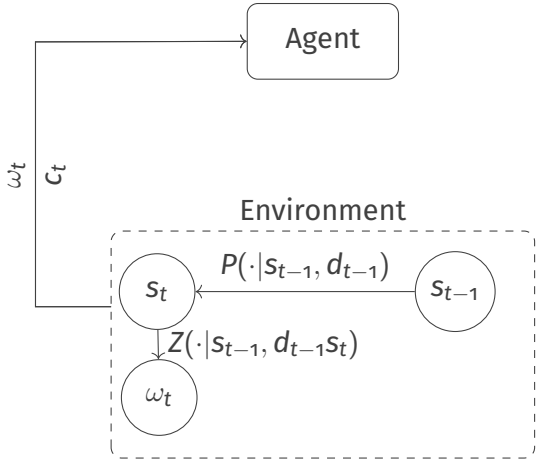


POMDP Definition

A POMDP is defined by a tuple $(\mathcal{S}, \mathcal{D}, P, \Omega, Z, c)$.

- State of the process: $s \in \mathcal{S}$;
- Decision: $d \in \mathcal{D}$;
- Transition probability: $P(s'|s, d)$;
- Observation: $\omega \in \Omega$;
- Observation function: $Z(\omega|s, d, s')$;
- Cost function: $c(s, d, s')$.

Partially Observable Markov Decision Process



POMDP Definition

A POMDP is defined by a tuple $(\mathcal{S}, \mathcal{D}, P, \Omega, Z, c)$.

- State of the process: $s \in \mathcal{S}$;
- Decision: $d \in \mathcal{D}$;
- Transition probability: $P(s'|s, d)$;
- Observation: $\omega \in \Omega$;
- Observation function: $Z(\omega|s, d, s')$;
- Cost function: $c(s, d, s')$.

Solving a Partially Observable Markov Decision Process

A POMDP can be seen as a MDP where ”states” are replaced with histories
or, equivalently, by belief states, computed from trajectories

Solving a Partially Observable Markov Decision Process

A POMDP can be seen as a MDP where "states" are replaced with histories or, equivalently, by belief states, computed from trajectories

- In a POMDP, we do not observe s_t, but noisy observations ω_t
- A t-step history $h_t = (\omega_0, d_0, \omega_1, d_1, \ldots, \omega_{t-1}) \in \mathcal{H}_t$ summarizes our probabilistic belief b_t about the state s_t
- In a finite-horizon POMDP, optimal policies are history dependent

Solving a Partially Observable Markov Decision Process

- In a POMDP, we do not observe s_t, but noisy observations ω_t
- A t-step history $h_t = (\omega_0, d_0, \omega_1, d_1, \ldots, \omega_{t-1}) \in \mathcal{H}_t$ summarizes our probabilistic belief b_t about the state s_t
- In a finite-horizon POMDP, optimal policies are history dependent

$$\pi = \{\pi_t,\}_{t=0,..,H-1}, \text{ where } \pi_t : \mathcal{H}_t \to D_t$$

$$\underbrace{V(\pi, s)}_{\text{Criterion to optimize}} = \underbrace{\mathbb{E}\left[\sum_{t=0}^{H-1} c(H_t, D_t, S_t) \mid S_0 = s, \pi, D_t \sim \pi_t(H_t)\right]}_{\text{Expected long-term cost following policy } \pi}$$

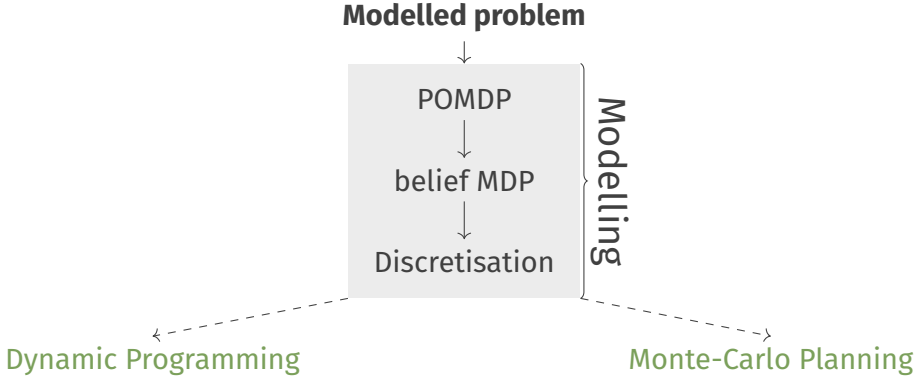
$$\underbrace{V^\star(s)}_{\text{Value function}} = \underbrace{\min_{\pi \in \Pi} V(\pi, s)}_{\text{Minimization over the set of policies } \Pi.}$$

Solving a Partially Observable Markov Decision Process

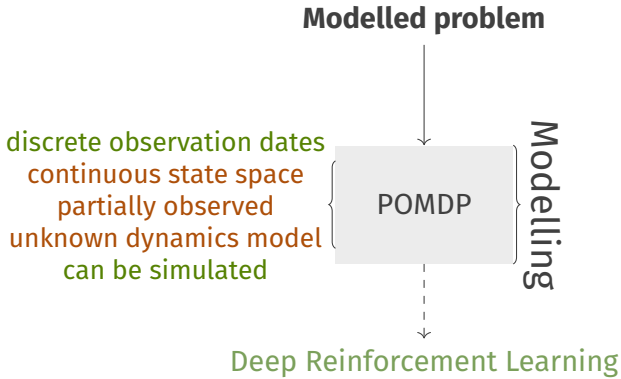
- In a POMDP, we do not observe s_t, but noisy observations ω_t
- A t-step history $h_t = (\omega_0, d_0, \omega_1, d_1, \ldots, \omega_{t-1}) \in \mathcal{H}_t$ summarizes our probabilistic belief b_t about the state s_t
- In a finite-horizon POMDP, optimal policies are history dependent

POMDP are far more difficult to solve than usual MDP!

Solution approaches



Solution approaches



Challenge 2: Non-Markovian process
Cancer treatment follow-up
(related to Orlane Rossini's PhD thesis)

Medical Context

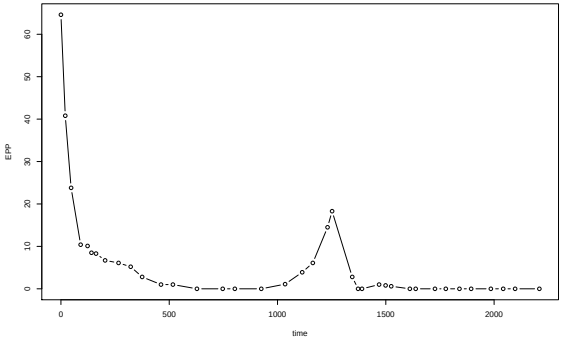


Figure: Example of a patient's data[a]

- Patients who have had a cancer benefit from regular follow-up;
- The concentration of clonal immunoglobulin is measured over time;
- The doctor must make new decisions at each visit.

[a] IUCT Oncopole and CRCT, Toulouse, France

Medical Context

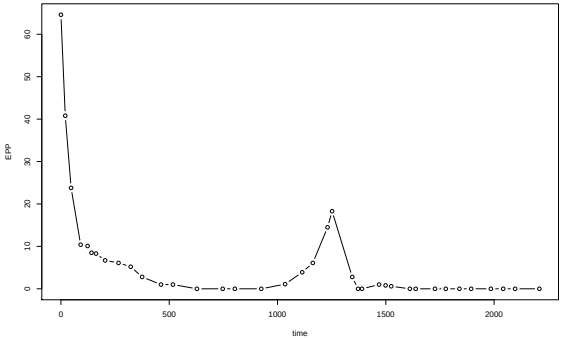


Figure: Example of a patient's data[a]

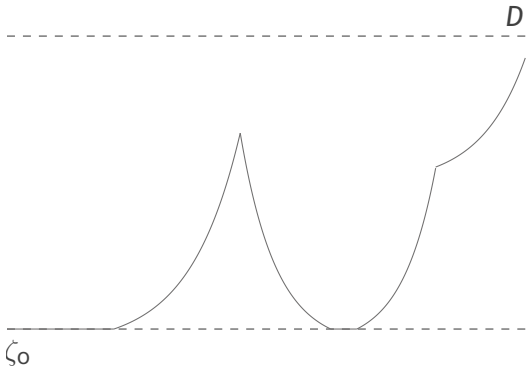
- Patients who have had a cancer benefit from regular follow-up;
- The concentration of clonal immunoglobulin is measured over time;
- The doctor must make new decisions at each visit.

$\Longrightarrow$ **Optimize decision-making to ensure the patient's quality of life**

[a] IUCT Oncopole and CRCT, Toulouse, France

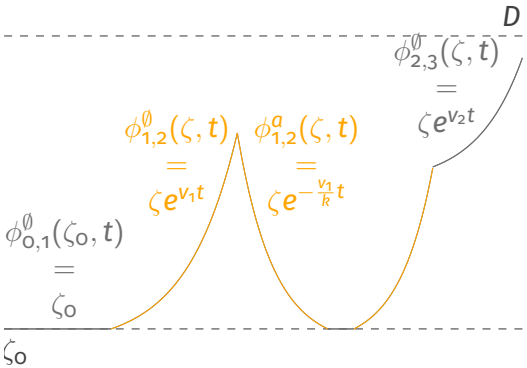
Piecewise-Deterministic Markov Process

A Piecewise-Deterministic Markov Process is defined through three local features



Piecewise-Deterministic Markov Process

A Piecewise-Deterministic Markov Process is defined through three local features



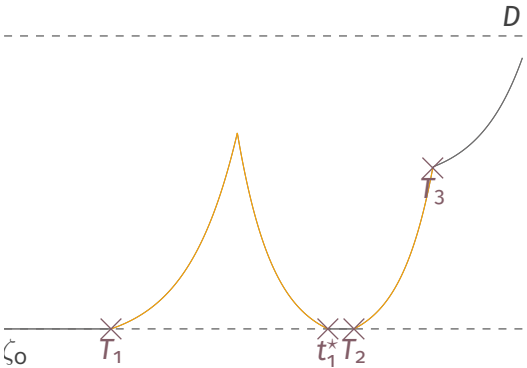
The flow

The deterministic part of the process

$$\phi^{\ell}_{m,k}(\zeta,t)$$

Piecewise-Deterministic Markov Process

A Piecewise-Deterministic Markov Process is defined through three local features



Jump intensity

Description of the jump mechanism of the process

- Boundary jump (deterministic)

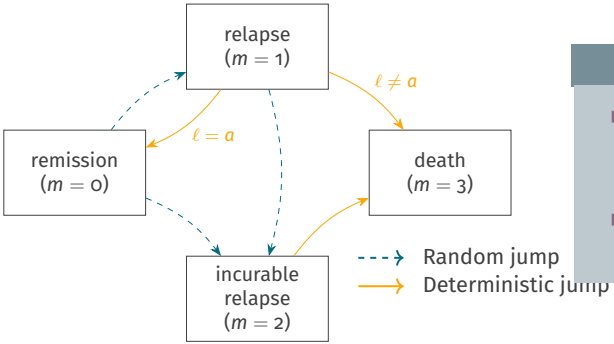
$$t_m^{\ell\star}(\zeta) = \inf\{t > 0 : \phi_{m,k}^{\ell}(\zeta, t) \in \{\zeta_0, D\}\}$$

- Random (continuous time) jumps

$$\mathbb{P}(T > t) = e^{-\int_0^t \lambda_m^{\ell}(\phi_{m,k}^{\ell}(\zeta,t))\,\mathrm{d}s}$$

Piecewise-Deterministic Markov Process

A Piecewise-Deterministic Markov Process is defined through three local features



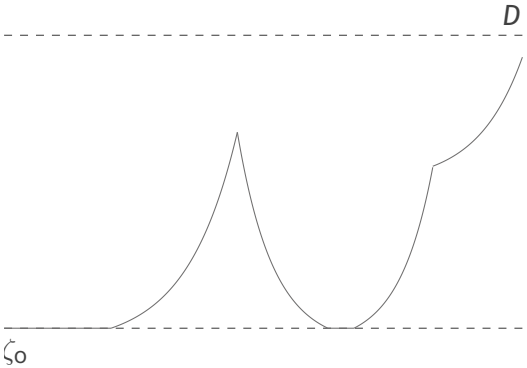
The kernel

- Discrete part of the state:
  - Mode $m \in \mathbb{N}$
  - Number of relapses $k \in \mathbb{N}$
- Law of the state of the process after each jump:

$$\mathbb{P}(m', k' | m, k, \zeta, t, \ell)$$

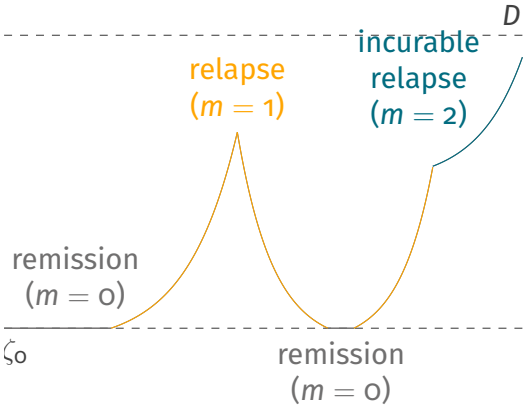
Controlled Piecewise-Deterministic Markov Process
Example of cancer treatment follow-up

Random jump from a deterministic regime to the next



Controlled Piecewise-Deterministic Markov Process
Example of cancer treatment follow-up

Random jump from a deterministic regime to the next

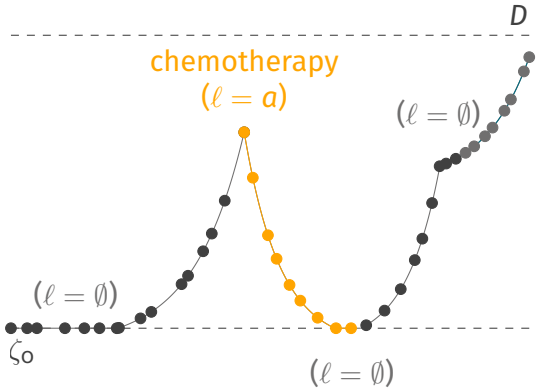


Let the patient state be $x = (m, k, \zeta, u)$:

- m disease "mode";
- k number of relapses;
- ζ biomarker value;
- u time since last jump.

Controlled Piecewise-Deterministic Markov Process
Example of cancer treatment follow-up

Random jump from a deterministic regime to the next



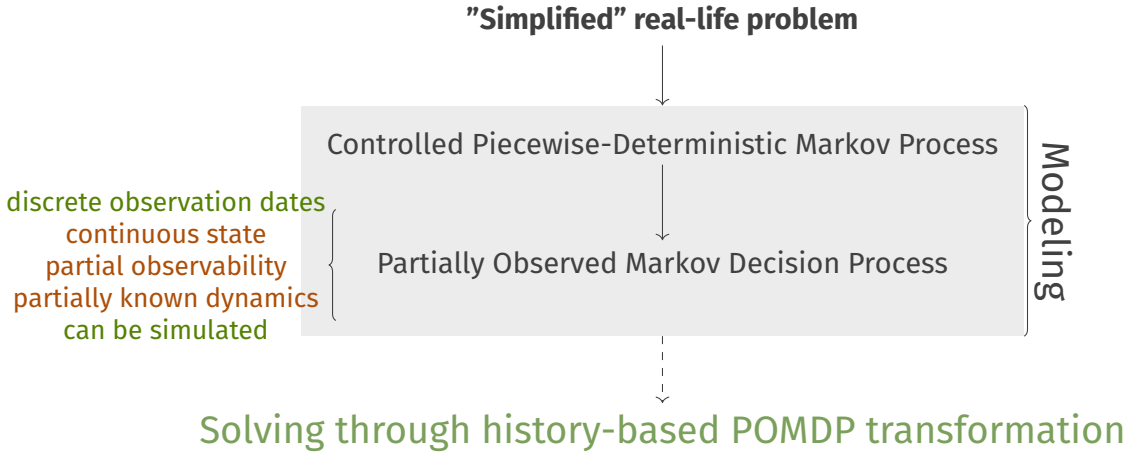
Let the patient state be $x = (m, k, \zeta, u)$:

- m disease "mode";
- k number of relapses;
- ζ biomarker value;
- u time since last jump.

Let d be the decision: $d = (\ell, r)$:

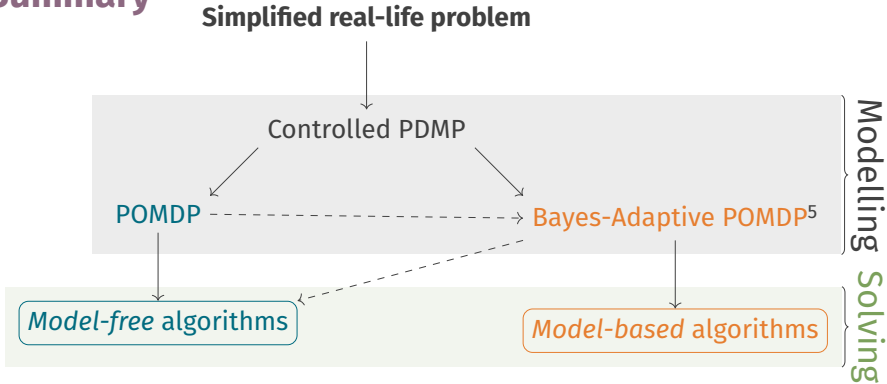
- ℓ treatment;
- r time till next visit.

Monte-Carlo planning solution approach[4]



[4]de Saporta et al., Medical follow-up optimization: A Monte-Carlo planning strategy, 2024, https://hal.science/hal-04382747v1

Summary



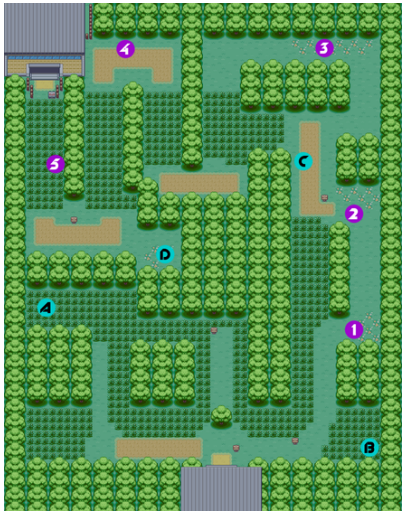
- Model-free algorithms require lots of data or a simulator
- Model-based algorithms require less data (but prior knowledge of model structure)[6]

[5] Orlane Rossini's PhD thesis

[6] Codes available at https://forgemia.inra.fr/orlane.le-quellennec/controlled_pdmp_po

Challenge 3: Multiple learning agents
Anti-poaching
(Prasanna Maddila's PhD thesis)

Context



- We want to solve a subclass of stochastic games, where
  - Some agents are cooperative, . . .
  - and the others independently compete with the team.
- We can formally model Anti-Poaching in this sub-class
  - where rangers cooperate against independent poachers.

Partially Observable Stochastic Games

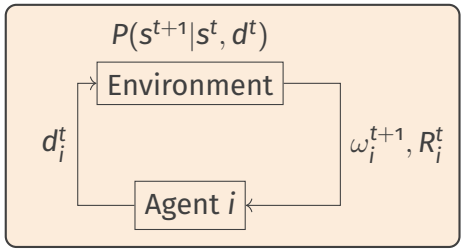


Figure: Simulation of a time-step in a POSG

- Each agent has a local history $h_i^t = (d_i^0, \omega_i^1, d_i^1, \ldots, d_i^{t-1}, \omega_i^t)$.
- Local (mixed) policy $\pi_i(h_i^t) \in \Delta(\mathcal{D}_i)$
- Transitions and rewards depend on joint decisions
- value of a joint policy:

$$v_{\pi,i}(h_i^t) = \mathbb{E}_\pi \left[\sum_{\tau=t}^{H} \gamma^{\tau-t} R_i^\tau \middle| h_i^t \right] \quad (1)$$

- A Nash Equilibrium is a joint policy $(\pi_i^*)_{i \in \mathcal{I}}$ from which no agent has interest to deviate: $\forall i \in \mathcal{I}, \forall \pi_i$

$$v_{\pi_i^*}(h) \geq v_{(\pi_i, \pi_{-i}^*)}(h) \quad (2)$$

Solving Partially Observable Stochastic Games

- POSG are an extension of both POMDP and normal-form games
- How can we compute joint equilibrium policies ?

- **Exact solution method:**
  - Dynamic programming[7]
  - . . . but only for *small* games

- **Reinforcement learning approaches?**
  - There exists Reinforcement Learning approaches[8]
  - ... which are often used for 2-player games
  - ... or for for purely competitive/cooperative games

[7] [Hansen et al., 2004]
[8] [Yang and Wang, 2021]

The Anti-Poaching Game

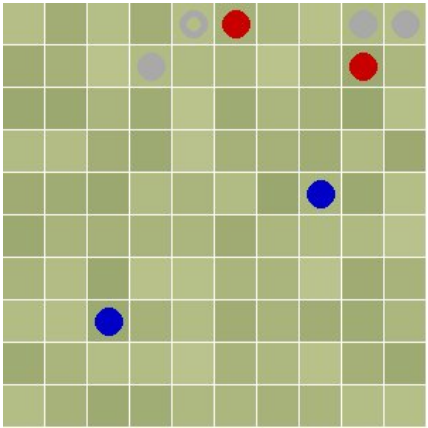


Figure: Visualisation of the Anti-Poaching Game

- The game has two types of agents: Rangers ($i \in \mathcal{I}$) and Poachers ($j \in \mathcal{P}$)
  - playing in a grid-world of fixed size
  - during a finite horizon ($0 \leq t \leq H$).
- Rangers can move or skip their turn $\forall i \in \mathcal{R}$,

$$\mathcal{D}_i = \{\emptyset, \uparrow, \downarrow, \leftarrow, \rightarrow\}$$

- Poachers can also place traps. $\forall j \in \mathcal{P}$,

$$\mathcal{D}_j = \{\emptyset, \uparrow, \downarrow, \leftarrow, \rightarrow, \text{place-trap}\}$$

The Anti-Poaching Game

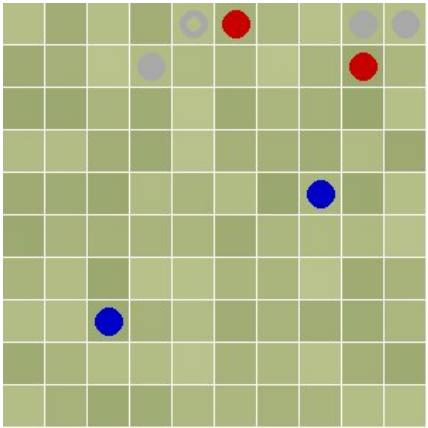


Figure: Visualisation of the Anti-Poaching Game

- A state $s \in \mathcal{S}$, is described by the state of all *"active"* objects in the grid.
- Each agent receives *noisy* observations of their cell at each timestep.
- A poacher is
  - rewarded for a prey recovered from a trap,
  - and penalised if he loses a trap, or gets captured.

The reward functions have a specific structure

Rewards with a Specific Structure ...

- Rangers have identical reward functions: $\forall i_1, i_2 \in \mathcal{R}$, $\forall s, a, b$[a]

$$R_{i_1}(s, a, b) = R_{i_2}(s, a, b)$$

- Poachers' rewards are independent:

$$R_j(s, a, b) = R_j(s, a, (b_j, b'_{-j}))$$

- The game is zero sum i.e. $\forall s, a, b$:

$$\sum_{i \in \mathcal{R}} R_i(s, a, b) + \sum_{j \in \mathcal{P}} R_j(s, a, b) = 0$$

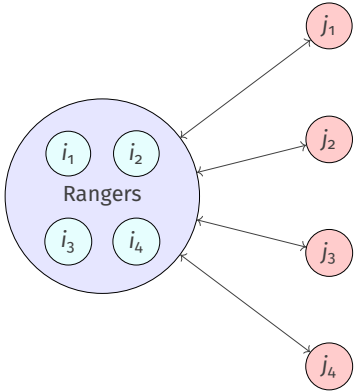


Figure: Interaction graph between agents

[a] $a = (a_i)_{i \in \mathcal{R}}$, joint team action and $b = (b_j)_{j \in \mathcal{P}}$ joint adversary action

Simulation Model for Transitions

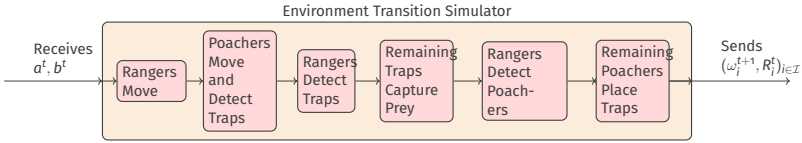


Figure: Transition Simulation Model for the Anti-Poaching game

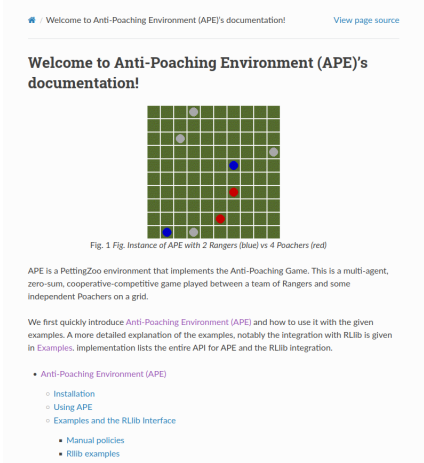
Transition probabilities $s^t \xrightarrow{d^t} s^{t+1}$ are $P(s^{t+1} \mid s^t, d^t)$

- They are difficult to calculate and store!
  - due to the size of the state and action spaces.
- So, they are simulated, which allows to
  - implement a simulator
  - use Reinforcement Learning

Benchmark

To facilitate the development of new algorithms for this game,

- The **Anti-Poaching Environment (APE)** in Python using the PettingZoo API [Terry et al., 2021] is provided :
  - https://forgemia.inra.fr/chip-gt/antipoaching
- . . . which makes it easy to use with existing RL libraries.
  - notably RLlib[Liang et al., 2018], which proposes multi-agent RL algorithms.



Concluding Remarks

POMDP and RL for complex sequential decision problems

- POMDP form a "classical" powerful representation framework for studying sequential decision problems under uncertainty
- POMDP can be conveniently be modelled, using e.g. the Gymnasium API[9]
- POMDP are far harder to solve than MDP. they can be solved using off-the-shelf RL librairies[10] (including deep-RL)
- POMDPs can also model non- (semi-) Markovian decision processes
- Finally, game-theoretic extensions of POMDPs allow to model multi-agent decision problems. Dedicated API+solution algorithms exist[11]!

[9] https://gymnasium.farama.org/
[10] For example, RLlib https://docs.ray.io/en/latest/rllib/
[11] PettigZoo:https://pettingzoo.farama.org/

Merci !

Any questions ?

References I

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004).
Dynamic programming for partially observable stochastic games.
In AAAI, volume 4, pages 709–715.

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J., Jordan, M. I., and Stoica, I. (2018).
Rllib: Abstractions for distributed reinforcement learning.
In Dy, J. G. and Krause, A., editors, Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 3059–3068. PMLR.

References II

Terry, J., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sullivan, R., Santos, L. S., Dieffendahl, C., Horsch, C., Perez-Vicente, R., et al. (2021).
Pettingzoo: Gym for multi-agent reinforcement learning.
Advances in Neural Information Processing Systems, 34:15032–15043.

Yang, Y. and Wang, J. (2021).
An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective.
arXiv:2011.00583 [cs].