UKRI
Trustworthy
Autonomous
Systems Hub

University of Southampton

Robust and Safe Artificial Intelligence

David Bossens

January 6, 2023;
CFAR Rising Star Lecture Series

Overview

- Introduction
- Meta-Reinforcement learning
  - Self-improvement
  - Policy reuse
- Resilient robot teams
  - Offline evolution with online adaptation
- Safe reinforcement learning
  - Robust constrained MDPs
  - Off-policy Evaluation
  - Safe RL Workshop
- Concluding remarks

Introduction

What do we want? Safe and Robust AI

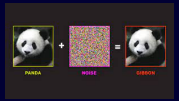
- pattern recognition and decision-making algorithms
- that are robust: do their task as expected even in the face of disturbances
- and safe: do not have harmful consequences in any part of their operation

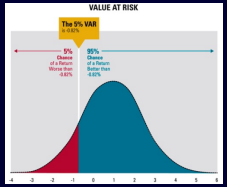
Counter Example 1: Unsafe exploration

- exploration: decisions made to obtain more information on the environment
- often random as this allows covering the space of outcomes
- unfortunately this may have negative consequences ...

Counter Example 2: worst-case outcomes

- often the agent cannot foresee some of the worst-case outcomes
- these may come from nature (e.g. weather conditions), adversarial attacks, or model errors
- in general, they are due to uncertainty in the environment





Reinforcement learning with MDPs

Markov decision process (MDP)

- Agent observes environment state $s \in \mathcal{S}$ and then performs action $a \in \mathcal{A}$.
- Environment returns the reward $r(s, a) \in \mathbb{R}$ and then samples the next state $s' \sim P^*_{s,a}$.
- Decision problem:
  - Find policy $\pi : \mathcal{S} \to \mathcal{A}$
  - (stochastic policy also often-used)
  - Which maximises $V_\pi(s) = \mathbb{E}_{\pi, P^*}\left[\sum_{k=0}^{\infty} \gamma^t r_t | s_0 = s\right]$
- MDP is defined by tuple $\langle \mathcal{S}, \mathcal{A}, P^*, r, \gamma \rangle$.
- Markov property: transition probability depends only on current state-action pair $(P^*_{s,a})$

Reinforcement learning with MDPs: Practice

- In RL, the dynamics and reward distribution of the MDP are unknown.
- Value-based: learn the value for each state-action pair.
  For example, Q-learning [10] maintains a Q-table:
  $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r(s,a) + \gamma \max_{a'} Q(s',a')\right)$
  - Approximation is needed for scalability
  - For example, Deep Q-Network (DQN) [6] minimises the loss
    $L(\theta) = \mathbb{E}_{(s,a,r,s')\sim U(\mathcal{B})}\left[\left(r + \gamma \max_{a'} Q(s',a';\hat{\theta}) - Q(s,a;\theta)\right)^2\right]$
    via repeated samples from a replay buffer $\mathcal{B}$.
  - DRQN [4]: modifies DQN for partially observable MDPs using LSTM network.
- Policy-based:
  Loss function is directly dependent on the policy. E.g. PPO [9]

Meta-reinforcement learning

Learning how to learn

- Meta-learning involves *learning how to learn*
- Two interpretations:
  - Self-improvement:
    - adjusting learning operations and/or representation
    - robustness to: assumption violations
    - key research question: efficient improvement on long-term hard exploration tasks
  - Lifelong/continual learning:
    - robustness to: multiple tasks
    - key research question: how many tasks can a policy learn?

Self-improvement

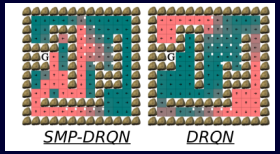
- Ambitious goal: learning a learning algorithm by modifying its basic components
- Exploring and exploiting environments of unknown class
  - Environment could be an MDP, POMDP, NMDP, ...
  - Is there some general principle for learning how to learn?
  - Can the agent autonomously figure out which approach works?
- Schmidhuber's work on **Incremental Self-improvement** [8]:
  - Encoding learning procedures as part of the learning process
  - **Policy-modifying process (PMP)**:
    - a time interval allocated to performing learning procedures as well as interaction with the environment.
    - PMP is ended when SSA is called.
    - **Success Story Algorithm (SSA):** evaluates PMPs on the *success story criterion (SSC)* $V(\pi_{t_N}, t_N) > V(\pi_{t_{N-1}}, t_{N-1}) > \cdots > V(\pi_{t_0=0}, t_0 = 0)$
      where $V(\pi_{t_k}, t_k) = (R_t - R_{t_k})/(t - t_k)$ is reward velocity since the start of the PMP.
      Pops back entries from a stack until SSC is true.

Self-improvement application: Active Adaptive perception

- Self-improvement with recurrent neural networks to provide
  - Selective application of perception module with goal-based exploration (SMP-DRQN implementation):
    - `doQuntil`(experience, maximal time, exploration rate)
    - initiate DRQN for the chosen time steps or until experience (goal) is found
    - modifiable experience set (using `setExperience`) to determine when to halt the network ;
    - when to use which exploration rate.
  - Learn how to modify the perception module based on long-term reinforcement (SMP-Fixed and SMP-Constructive implementations)
- Learns how to use instructions for long-term reward acceleration, for example:
  - using or modifying neural network
  - working memory instructions
  - self-modification
- Does not get stuck in long-term environment with no time-outs and no terminal states

	Instruction/Parameter					
ProgramCell/IP	0	1	2	3	4	5
6						
7						
8						
9	.05	.01	**.88**	.02	.02	.02
10	.05	.03	.34	.28	.10	**.20**
11	.22	.04	.02	.24	.28	**.15**
12	.05	.30	**.55**	.05	.01	.04

Self-modifying policy π



Non-episodic maze

Limitations of Self-improvement

- no guarantees
- success depends on instructions (problem of prior knowledge does arise)

Active Adaptive perception paper

Available as

- Bossens, D. M., Townsend, N. C., & Sobey, A. J. (2019). Learning to learn with active adaptive perception. *Neural Networks*, 115, 30–49.

Lifelong reinforcement learning (LRL)

LRL aims to learn when presented sequence of tasks (typically MDPs or POMDPs)

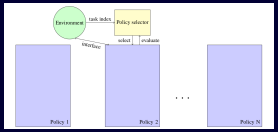
- Wide variety of approaches
- Commonality: combining task-specific representations with global representations where representation is e.g. parameters, NN layers, or policies
- Key question: *how many tasks can a representation handle?*

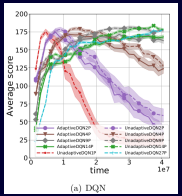
Lifetime policy reuse and the importance of task capacity

- Fixed set policies $\mathcal{P} = \{\pi_1, \ldots, \pi_{N_\pi}\}$, where $N_\pi \leq N_\tau$
- Two policy selection strategies
  - Adaptive: select based on past task-success
  - Unadaptive: random choice at start of the run
- Policies are continually refined

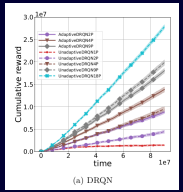
Lifetime policy reuse and the importance of task capacity

- Successful application on:
  - 27-task Cartpole MDP sequence: move cart to keep pole up under 27 different configurations of the cart and pole
  - 18-task POcman POMDP sequence: partially observable mazes with different maze topologies and object of interest (positive/negative, static/random dynamic/strategic)
- Effect of task-similarity: fewer policies needed
- Significant memory reduction possible when compared to task-specific policies; also avoids temporary policies





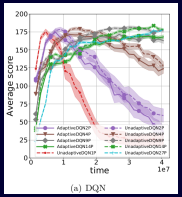
Cartpole



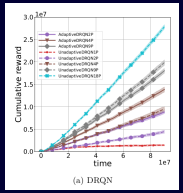
POcman

Lifetime policy reuse and **the importance of task capacity**

- **theoretical task capacity**:
  - number of policies to ϵ-optimally represent all tasks
  - task cluster analysis
- **empirical task capacity**:
  - the smallest number of policies to ϵ-optimally represent all its tasks, *compared to observed performance of task-specific policy*
  - empirically computed by manipulating $N_{\text{pol}} \rightarrow$ for a subset of tasks reasonably covering the full task space
- importance of using task capacity for pre-selecting number of policies
  - lowered memory consumption for the same performance
  - scalability to 125 Cartpole domain: worst-case of both metrics is 3 on 27-task domain $\rightarrow$ pre-selection of the number of policies $N_\pi = 9 \rightarrow$ with $N_\pi = 9$, LPR is near-optimal on the 125-task domain.

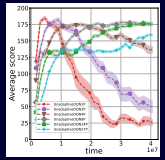


(a) DQN



(a) DRQN

27-task Cartpole 18-task POcman



125-task Cartpole

Lifetime policy reuse paper

Available as

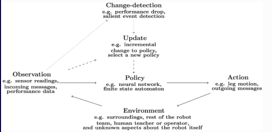
- Bossens, D. M., & Sobey, A. J. (2021). Lifetime policy reuse and the importance of task capacity. ArXiv Preprint ArXiv:2106.01741, 1–27. http://arxiv.org/abs/2106.01741

Resilient robot teams

Resilient robot teams

- Resilience: ability to rapidly recover performance from sudden, impactful disturbances
- Integrated decision framework:
  - template for decision-making strategies in resilient robot teams
  - integrates decentralised control, change-detection, and learning
  - helps identify 4 resilience strategy prototypes in the literature
- Applications examples:
  - active and distributed sensing
  - identifying and tracking dynamic anomalies
  - collaborative mapping dynamic environments
  - search-and-rescue
  - traditional multi-agent tasks (foraging, coverage, flocking)
- Review paper with framework available as:
  Bossens, D. M., Ramchurn, S., & Tarapore, D. (2022). Resilient robot teams: a review integrating decentralised control, change-detection, and learning. *Current Robotics Reports*.

From: Resilient Robot Teams: a Review Integrating Decentralised Control, Change-Detection, and Learning



Integrated decision process framework. The framework integrates decentralised control, change-detection, and learning into a single loop describing the decision problem faced by each robot in a robot team. Solid lines indicate the control cycle, dashed lines indicate the learning cycle performing policy updates periodically, and dotted lines indicate the optional change-detection cycle

Table 1 Adaptation prototypes within the integrated decision framework

From: Resilient Robot Teams: a Review Integrating Decentralised Control, Change-Detection, and Learning

Prototype	Policy space	Change-detection	Learning	Approaches
Diagnose-and-solve	Domain-specific	Diagnosis of a priori known faults	No learning or limited to generalising solutions to new faults	Fault-detection and fault-diagnosis
Trial-and-error	Generic	Generic change-detection	Learning by trial-and-error	Multi-agent reinforcement learning, embodied evolution, and offline evolution with online adaptation
Teacher-student	Generic	New data provided by a teacher, often a human operator	Imitation learning on provided data set	Learning perception-action-communication loops
Robust-by-design	Domain-specific	Implicit or explicit communication	Frequent updates on desired task allocation	Explicit task allocation, stigmergy in swarm robotics

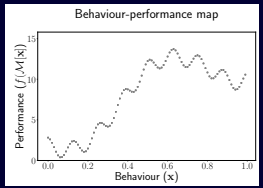
Columns indicate the prototypes and their various properties: **Policy space** indicates the space of possible policies for decentralised control; **Change-detection** indicates how change-detection is typically implemented; **Learning** indicates how the policy is adapted upon a detected environmental change. **Approaches** lists the different classes of algorithms within the adaptation prototype

Overview of approaches

- Perception-Action-Communication Loops
- Multi-Agent Reinforcement Learning
- Embodied Evolution
- **Offline Evolution with Online Adaptation**: key benefit of *rapid* adaptation to high-impact changes
- Explicit Task Allocation
- Stigmergy in Swarm Robotics

Offline Evolution with Online Adaptation

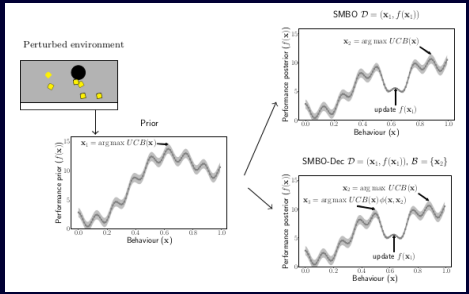
- Inspired by the "Intelligent Trial and Error" algorithm [2]:
  - offline evolution: evolve large, diverse repertoire of policies (thousands of policies in a behaviour-performance **map** $\mathcal{M}$) in simulation using quality-diversity algorithm; EA with mutation and/or crossover operators but
    - map bins a population based on diversity over behavioural features.
    - selection operator also applied across the map
  - online adaptation: use map-based Bayesian optimisation to optimise the policy after a sudden change requiring resilience
- Traditionally, applied in single-robot contexts and pre-defined behavioural diversity
- Research questions: a) how does the approach apply to robot teams? b) more useful and automated notions of diversity?



Behaviour-performance map

Swarm Map-based Bayesian Optimisation (SMBO)

- Map-based Bayesian optimisation for robot teams
- Inter-robot communication of the selected controller and its estimated performance
- Two algorithms, SMBO and SMBO-Dec



SMBO vs SMBO-Dec

SMBO

- 1 GP model
- Traditional UCB:

$$\mathbf{x}_t = \underset{\mathbf{x}\in\mathcal{M}\setminus\{\mathbf{x}_1,\ldots,\mathbf{x}_{t-1}\}}{\arg\max} \mu_{t-1}(\mathbf{x}) + \alpha\sigma_{t-1}(\mathbf{x})\,, \tag{1}$$

- One central decision-maker

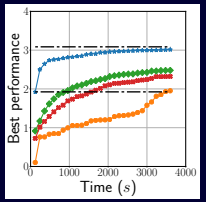
SMBO-Dec

- 1 GP model for each group (e.g. fault condition)
- Batch-based Bayesian optimisation (busy samples)
- Acquisition based on UCB with hard local penalisation [1]:

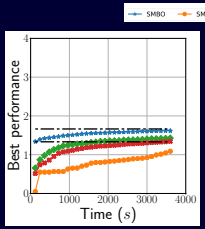
$$\mathbf{x}_t = \underset{\mathbf{x}\in\mathcal{M}\setminus\{\mathbf{x}_1,\ldots,\mathbf{x}_{t-1}\}}{\arg\max} [\mu_{t-1}(\mathbf{x}) + \alpha\sigma_{t-1}(\mathbf{x})] \prod_{i=1}^{N_b} \phi(\mathbf{x}, \tilde{\mathbf{x}}_i)\,, \tag{2}$$

$$\phi(\mathbf{x}_t, \tilde{\mathbf{x}}_i) = \min\left(1, \frac{||\mathbf{x}_t - \tilde{\mathbf{x}}_i||}{\mathbb{E}[r_i] + \gamma\frac{\sigma_{t-1}(\tilde{\mathbf{x}}_i)}{L_i}}\right). \tag{3}$$

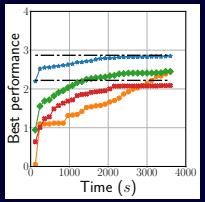
Centralised adaptation experiments (SMBO)



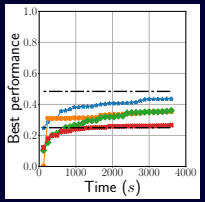
Proximity sensor



Actuator



Software-Food

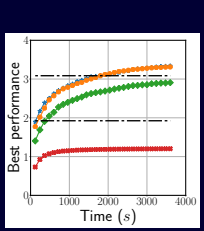


Food-scarcity

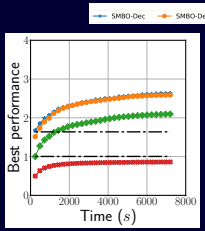
Key findings:

- SMBO rapidly close to maximal performance in behaviour-performance map (upper dotted line);
- SMBO-Uniform initially low performance $\rightarrow$ importance of prior;
- Random search and Gradient ascent over behaviour performance-map: low performance.

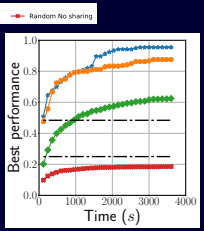
Decentralised adaptation experiments (SMBO-Dec)



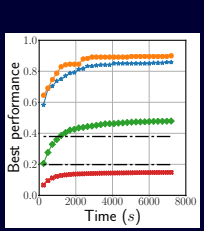
Proximity sensor



Proximity sensor 2x



Food-scarcity



Food-scarcity 2x

Key findings:

- SMBO-Dec outperforms other algorithms and even the homogeneous swarm maximum:
  - Selective sharing is beneficial (heterogeneous swarm)
  - Improved adaptation speed by using multiple workers
  - Local penalisation is beneficial
- 2x-tasks (larger arena and more robots): lowered performance but higher percentage improvement

Swarm Map-based Bayesian Optimisation paper

Available as

- Bossens, D. M., & Tarapore, D. (2021). Rapidly adapting robot swarms with Swarm Map-based Bayesian Optimisation. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2021), 9848–9854.

Quality Environment Diversity (QED)

- The environment in which solutions are evolved/learned is crucial for generalisation, robustness, and complex skills (see curriculum learning, works on reducing simulation-reality gap, adversarial settings)
- How to define useful behavioural diversity? Does it depend on *environmental diversity*, i.e. the environment in which solutions are evolved?
- Quality Environment Diversity:
  - define an *environment*-performance map
  - bin solutions based on the environment which it solves (e.g. cluttered arena with larger agent density)
  - based on a user-defined environment space
  - selective pressure in different bins in evolution leads to *implicitly defined behaviour space*

Normal environment

Cluttered agent-dense environment

Attribute	Description	Value	Perturbations injected
A_1	Robots' maximal linear speed	$10\,\mathrm{cm/s}$	$\mathbf{P}_1 = \{5, 10, 15, 20\}\mathrm{cm/s}$
A_2	Number of robots in the swarm	10	$\mathbf{P}_2 = \{5, 10, 15, 20\}$
A_3	Arena size	$16\,\mathrm{m}^2$	$\mathbf{P}_3 = \{4, 9, 16, 25\}\mathrm{m}^2$
A_4	Number of obstacles	0	$\mathbf{P}_4 = \{0, 2, 4, 6\}$
A_5	Robots' range-and-bearing sensor range	$1\,\mathrm{m}$	$\mathbf{P}_5 = \{25, 50, 100, 200\}\mathrm{cm}$
A_6	Robots' proximity sensor range	$11\,\mathrm{cm}$	$\mathbf{P}_6 = \{5.5, 11, 22, 44\}\mathrm{cm}$

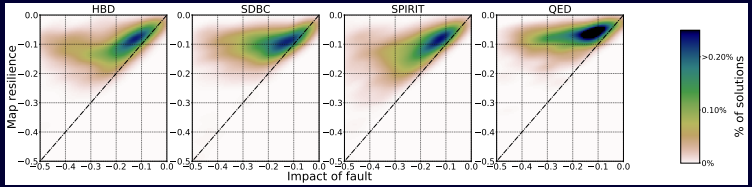
User-defined environment space

QED Results: performance

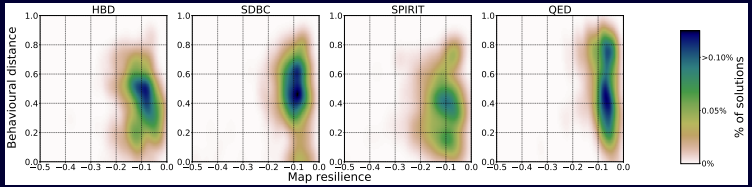
- Demonstrated across 250 fault injections (to proximity sensor, wheel, range-and-bearing sensor)
- Compared to various *behaviour* descriptors (Systematic Behaviour Characterisation, SPIRIT, Hand-coded Behavioural Descriptor)
- Trade-off: reduced performance in normal environment but much higher resilience (closer to zero)

$$\mathscr{R}(\mathcal{M}, \mathcal{E}_{\mathscr{F}} | \mathcal{E}) = \frac{\max_{c \in \mathcal{M}} f(\mathcal{E}_{\mathscr{F}}, c) - \max_{c' \in \mathcal{M}} f(\mathcal{E}, c')}{\max_{c' \in \mathcal{M}} f(\mathcal{E}, c')}, \tag{4}$$

QED Results: Qualitative analysis



Resilience across fault impact spectrum (and smaller fault impact in general)



Recovery solutions across the behaviour space

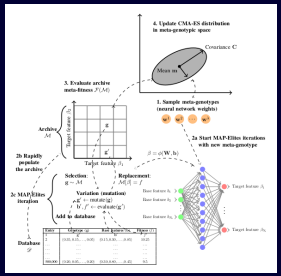
QED Paper

Available as

- Bossens, D. M., & Tarapore, D. (2021). QED: Using Quality-Environment-Diversity to Evolve Resilient Robot Swarms. *IEEE Transactions on Evolutionary Computation*, 25(2), 346–357.

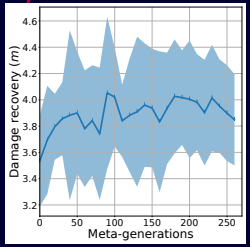
Quality-Diversity Meta-evolution (QDME)

- We don't know what will be the best behaviour space for an adaptation problem. Why not define behaviour descriptor automatically?
- Quality-diversity meta-evolution:
  - Define a *meta-genotype* W, which parametrises the behaviour space as $\beta = \phi(W\mathbf{b})$, where ϕ is a feature-map, $\mathbf{b}$ is a large set of base-features
  - Evolve the meta-genotype to optimise *meta-fitness*, which represents the objective of the behaviour-performance map on the adaptation problem
  - This meta-evolutionary step is implemented using CMA-ES, which allows relatively quick convergence with a limited population size
  - Behaviour-performance maps are formed from meta-genotypes using a large database (continuously updated with new genotypes and their base-features)

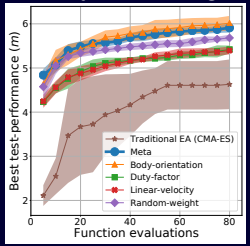


QDME flow diagram

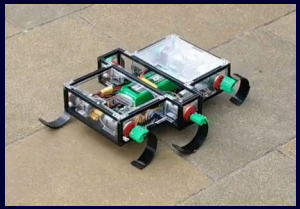
QDME experiments with linear feature-maps: hexapod robot



Improved recovery to training set of faults

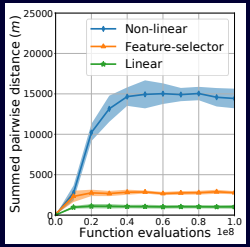


Recovery to test set of faults
on par with best base-features

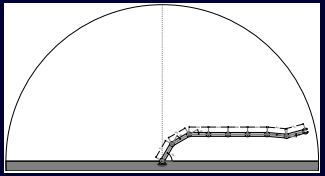


Rhex hexapod robot domain

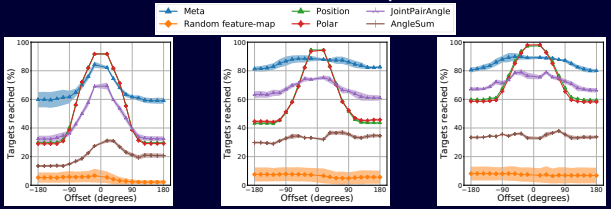
QDME experiments with non-linear feature-maps: Robot arm experiments



Offline evolution: improvement in meta-fitness over linear feature-maps



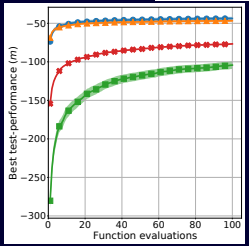
8-joint robot arm



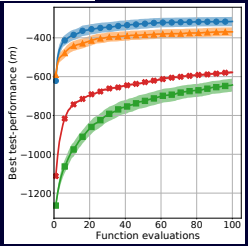
Leg 1 Leg 5 Leg 8

Online adaptation: improved recovery on high-impact

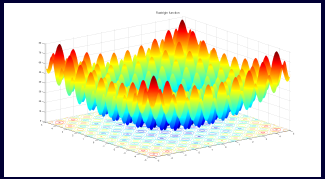
QDME experiments with non-linear feature-maps: Rastrigin function



Dimension test

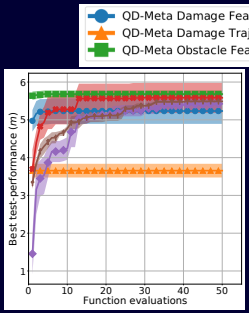


Translation test

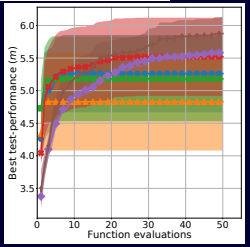


Rastrigin function

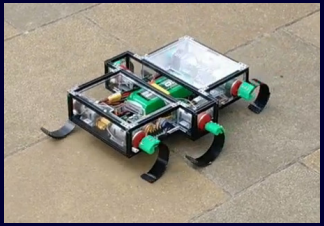
QDME experiments with non-linear feature-maps: hexapod robot revisited



Obstacle test



Damage test



Rhex hexapod robot domain

QDME papers

Available as

- Bossens, D. M., Mouret, J., & Tarapore, D. (2020). Learning behaviour-performance maps with meta-evolution. Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO 2020), 49–57.
- Bossens, D. M., & Tarapore, D. (2021). On the use of feature-maps for improved quality-diversity meta-evolution. Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion (GECCO 2021 Companion), 83–84.
- Bossens, D. M., & Tarapore, D. (2022). Quality-Diversity Meta-Evolution: customising behaviour spaces to a meta-objective. *IEEE Transactions on Evolutionary Computation*, 1–11.

Safe reinforcement learning

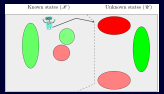
constrained MDPs (CMDPs)

- Agent observes environment state $s \in \mathcal{S}$ and then performs action $a \in \mathcal{A}$.
- Environment returns the reward $r(s,a) \in \mathbb{R}$ **as well as the constraint-cost** $c(s,a) \in \mathbb{R}$, and then samples the next state $s' \sim P^*_{s,a}$.
- Decision problem:
  - Find policy $\pi : \mathcal{S} \to \mathcal{A}$
  - Which maximises $V_\pi(s) = \mathbb{E}_{\pi,P^*}\left[\sum_{k=0}^{\infty} \gamma^t r_t | s_0 = s\right]$
    subject to $C_\pi(s) = \mathbb{E}_{\pi,P^*}\left[\sum_{k=0}^{\infty} \gamma^t c_t | s_0 = s\right] \leq d$
    where d **is the constraint-cost budget.**
- CMDP is defined by tuple $\langle \mathcal{S}, \mathcal{A}, P^*, r, c, d, \gamma \rangle$.
- Markov property: transition probability depends only on current state-action pair $(P^*_{s,a})$
- Key question: *How to ensure constraints are satisfied during exploration as well as exploitation?*

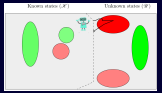
Explicit Explore, Exploit, or Escape (E^4)

- Learning near-optimal CMDP policies with safe exploration
- Introduces constraints and safe exploration to E^3 [5]
- Model-based approach:
  - Model the CMDP $\hat{P}$, $\hat{r}$, $\hat{c}$ from limited T-step trajectories.
  - Offline optimisation: use model to generate data, then estimate value of policy.
- Exploitation policy (known states):
  - CMDP model over the known states; known-state budget $d'' \leq d$.
- Exploration policy (known states):
  - CMDP model over the known states; reward for finding an unknown state; known-state budget $d'' \leq d$.
- Escape policy (unknown states):
  - before applying escape, balanced wandering to make states known
  - applying escape policy allows to return back to known state before budget d' exceeded.
  - CMDP model over worst-case transition, $P \leftarrow \arg\max_{P' \in \mathcal{P}} \sum_{s' \in \mathcal{S}} P'_{s,a}(s') \hat{C}_{\pi,P'}(s', T|\hat{M}_{\mathscr{S}})$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.

Exploitation



Exploration



Escape

Exploitation End

E^4 theory

- Constrained simulation lemma: If model approximation close enough for a state, then $|\hat{V}_\pi(s) - V_\pi(s)| < \epsilon$ and $|\hat{C}_\pi(s) - C_\pi(s)| < \epsilon$ (=*known state*)
- l-safe explore-or-exploit lemma: budget for known states should be set to some $l \leq d - 2\epsilon$ for safe exploration and exploitation from s (safe meaning $C_\pi(s) \leq d$).
- Safe balanced wandering lemma: stop balanced wandering as soon as predicted cost satisfies

$$C(p_t) + \gamma^t \max_{P \in \mathcal{P}} \max_{a_t \in \mathcal{A}} \sum_{s_{t+1} \in \mathcal{S}} P_{s_t, a_t}(s_{t+1}) \hat{C}_{P,\pi}(s_{t+1}, T | \hat{M}_{\mathscr{U}}) \geq d' - c_{\max} \tag{5}$$

- Escape budget lemma: a budget for the escape policy is safe if it satisfies the following four requirements for the trajectory of the true CMDP (combining the true CMDP in known and unknown states) to be safe:
  - diameter lower than minimal time steps to escape budget
  - unknown-state requirement: escape budget accounts for future steps for safe return in known states
  - known-state requirement: escape budget accounts for path already taken in known states and future steps for safe return in known states
  - safe return requirement: there is sufficient budget d_s left for safe return in known states, after accounting for known states and the trajectory in unknown states.

Escape policy optimisation

- Uncertainty sets:
  - Choice is very flexible.
  - Usual robust optimisation: initially broad then more data makes it tight
  - For unknown states this does not work (few visitations $\rightarrow$ too broad set $\rightarrow$ no guarantee to escape)
  - Local inference or prior knowledge can help
    - narrow Bayesian priors
    - action models: you know the effect of actions (e.g. effect of going north in xy-coordinate state space)
    - local conditions: e.g. weather, near wall of arena
- MDP has limited diameter (D):
  - $D = \max_{s \neq s'} \min_{\pi: \mathcal{S} \to \mathcal{A}} T(s' | M, \pi, s)$
  - Otherwise no escape guarantee
- Different possible optimisers: policy gradient, dynamic programming, linear programming

E^4 key features

Main results:

- Optimality: E^4 finds near-optimal constrained policy within polynomial time
- Safe exploration: satisfies the constraint-cost budget throughout the whole lifetime *despite model errors*
- Explicit separation between known and unknown states, allowing e.g. targeted and safe exploration policies
- Contrast to alternative safe RL methods:
  - traditional CMDP optimisation still violates constraints throughout exploration (see e.g. [7])
  - LP or DP methods that assume dynamics are known
  - Off-policy RL and others which do not include explicit constraint-functions
- Applicability: reachability assumptions
- Flexibility: incorporate prior knowledge and inference methods for unknown states
- Optimisation: use variants of DP, LP, or policy gradient based on worst-case dynamics

E^4 paper

Available as

- Bossens, D. M., & Bishop, N. (2022). Explicit Explore, Exploit, or Escape (E^4): near-optimal safety-constrained reinforcement learning in polynomial time. *Machine Learning*.

Off-policy Evaluation

- Can we avoid exploration?
- Off-policy evaluation: evaluate a evaluation policy π_e using trajectories from behaviour policy π_b
- Traditionally, using **importance sampling**:

$$\begin{aligned}\mathscr{G} &= \mathbb{E}_{\pi_e, P^*}\left[\sum_{t=1}^{H} r_t\right] \\ &= \mathbb{E}_{\pi_b, P^*}\left[\sum_{t=1}^{H} r_t \left(\prod_{t=1}^{H} \frac{\pi_e(a_t|s_t)}{\pi_b(a_t|s_t)}\right)\right]\end{aligned}$$

sample from π_b and correct for the probability under π_e

The curse of horizon

- The variance of the importance sampling estimator grows exponentially with the horizon of the decision problem
- The estimator is given by

$$\hat{G} = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{H} r_t^{(i)} \left(\prod_{t=1}^{H} \frac{\pi_e(a_t^{(i)}|s_t^{(i)})}{\pi_b(a_t^{(i)}|s_t^{(i)})} \right)$$

  where i indexes the trajectory of the state $s_t^{(i)}$, action $s_t^{(i)}$, and reward $r_t^{(i)}$ at time t.
- Via Popoviciu's inequality, $\mathsf{Var}(X) \leq (M-m)^2/4$ for any random variable X with range $[m, M]$.
- Therefore:

$$\begin{aligned} \mathsf{Var}(\hat{G}) &\leq \left(\sum_{t=1}^{H} r_{\mathsf{max}} \left(\prod_{t=1}^{H} \rho_{\mathsf{max}} \right) \right)^2 /4 \\ &= \left(H r_{\mathsf{max}} \rho_{\mathsf{max}}^H \right)^2 /4 \,, \end{aligned}$$

  where $\rho_{\mathsf{max}} = \max_{(s,a) \in \mathcal{S} \times \mathcal{A}} \pi_e(a|s)/\pi_b(a|s)$.

State-based importance sampling (SIS)

- Let $\mathcal{S}^A \subset \mathcal{S}$ and $\mathcal{S}^B = \mathcal{S} \setminus \mathcal{S}^A$ its complement.
- Then

$$A = \begin{cases} \prod_{t \in [H]: s_t \in \mathcal{S}^A} \frac{\pi_e(a_t|s_t)}{\pi_b(a_t|s_t)}, & \text{if } \exists s \in \mathcal{S}^A \cap \tau \\ 1, & \text{otherwise} \end{cases}$$

and

$$B = \begin{cases} \prod_{t \in [H]: s_t \in \mathcal{S}^B} \frac{\pi_e(a_t|s_t)}{\pi_b(a_t|s_t)}, & \text{if } \exists s \in \mathcal{S}^B \cap \tau \\ 1, & \text{otherwise,} \end{cases}$$

where $[H] = \{1, 2, \ldots, H\}$.

- Following covariance testing [3], the expected return can be rewritten as

$$\begin{aligned} \mathscr{G} &= \mathbb{E}_{\pi_b, P^*}\left[G \left(\prod_{t=1}^{H} \frac{\pi_e(a_t|s_t)}{\pi_b(a_t|s_t)} \right) \right] \\ &= \mathbb{E}_{\pi_b, P^*}\left[GAB \right] \\ &= \mathbb{E}_{\pi_b, P^*}\left[A \right] \mathbb{E}_{\pi_b, P^*}\left[BG \right] + \mathrm{Cov}(A, BG) \quad \text{because } \mathrm{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]\,. \end{aligned}$$

State-based importance sampling (SIS)

- Following covariance testing [3], the expected return can be rewritten as

$$\mathscr{G} = \mathbb{E}_{\pi_b, P^*}[A]\,\mathbb{E}_{\pi_b, P^*}[BG] + \mathrm{Cov}(A, BG)$$

- We can use accurately estimate $\mathscr{G}$ using the random variable BG based on off-policy samples from π_b if:
  - Condition 1: $\mathbb{E}_{\pi_b}[A] = 1$. Verify empirically with search algorithm.
  - Condition 2: $\mathrm{Cov}(A, BG) = 0$. Verify empirically with search algorithm.
  - Condition 3: states $s \in \mathcal{S}^B$ have low maximal action probability ratio. e.g. for uniform random π_b, $\rho_{\max} = 1/\epsilon^H$.
- Denoting the subtrajectory $\tau_B := \{t \in [H] : s_t^{(i)} \in \mathcal{S}^B\}$ and $M_B := \max_{\tau \in \tau_B} |\tau|$, we have

$$\mathrm{Var}(\hat{G}_{SIS}(\mathcal{S}^A)) \leq \left(H r_{\max} \rho_{\max}^{M_B}\right)^2 / 4\,. \tag{6}$$

Search algorithm for SIS

- Goal: find the ϵ-negligible state-set $\mathcal{S}^A$ that minimises the MSE when applying SIS on $\mathcal{S}^A$.
- Start with $\texttt{best} \leftarrow \emptyset$ and user-defined $\epsilon > 0$.
- Iterate over subsets $\mathcal{S}^A \subset \mathcal{S}$.
  - Check if sample average satisfies $|\bar{A} - 1| < \epsilon$ and the estimated covariance satisfies $\widehat{\mathsf{Cov}}(A, BG) < \epsilon$. If so, this implies ϵ-negligiblity, i.e. Condition 1 and 2 are satisfied approximately, and continue to next step.
  - Compute $\widehat{\mathsf{MSE}}(\hat{G}_{SIS}(\mathcal{S}^A)) = \widehat{\mathsf{Var}}(\hat{G}_{SIS}(\mathcal{S}^A)) + \widehat{\mathsf{Cov}}(A, BG)^2$.
  - Set $\texttt{best} \leftarrow \mathcal{S}^A$ whenever either $\widehat{\mathsf{MSE}}(\mathcal{S}^A) < \widehat{\mathsf{MSE}}(\texttt{best})$ or $\widehat{\mathsf{MSE}}(\mathcal{S}^A) < \widehat{\mathsf{MSE}}(\texttt{best}) * (1 + \epsilon) \wedge |\mathcal{S}^A| > |\texttt{best}|$.
- Return best.

SIS experiments on lift domains

- **Lift states.** A lift state $s \in \mathcal{S}$ is a state for which $P^*(s,a) = P^*(s,a')$ and $r(s,a) = r(s,a')$ for all pairs of actions $a, a' \in \mathcal{A}$. The set of lift states for a domain is denoted by $\mathcal{S}^L$. Lift states intuitively captures the covariance condition (see Condition 2) since the action probabilities, and therefore their ratios, do not affect the expected return.
- Two selected domains with lift states:
  - Deterministic:
    - π_e always takes optimal action
    - transitions always according to direction (lift arrow or else agent action)
    - implies every lift state is traversed equally on average (Condition 1 OK for $\mathcal{S}^A = \mathcal{S}^L$)
  - Stochastic
    - π_e takes optimal action with 99% probability
    - transitions with 99% probability according to direction (lift arrow or else agent action)
    - more variable visitation frequencies (Condition 1 not OK for $\mathcal{S}^A = \mathcal{S}^L$ so search algorithm needs to check)

SIS Results

- SIS on lift-states works best for deterministic – this is where setting $\mathcal{S}^A = \mathcal{S}^L$ yields Condition 1 (approximately)
- SIS with search works best on stochastic – here one can selectively optimise negligible state-set (not necessarily all lift-states)

Table 1: Mean squared error (MSE) for the different estimators of the expected return in the deterministic lift domain based on 25 independent replicates for each domain size. The best estimator is highlighted in bold for each domain size.

(a) 100 Monte Carlo trajectories

Domain Size	$\hat{G}_{IS}$	$\hat{G}_{PDIS}$	$\hat{G}_{SIS}$ (Lift-states)	$\hat{G}_{SIS}$ (Search-based)	$\hat{G}_{INCRIS}$
7	0.0436	0.2865	**0.0183**	0.0436	0.2865
9	0.0922	1.0837	**0.0217**	0.0922	1.0837
11	0.2637	5.8209	**0.0213**	0.2980	5.8209
13	0.6969	20.2194	**0.0321**	0.7450	20.2194
15	1.0860	42.7628	**0.0213**	1.1587	42.7628
17	1.5161	80.6098	**0.0379**	1.5161	80.6098

(b) 1,000 Monte Carlo trajectories

Domain Size	$\hat{G}_{IS}$	$\hat{G}_{PDIS}$	$\hat{G}_{SIS}$ (Lift-states)	$\hat{G}_{SIS}$ (Search-based)	$\hat{G}_{INCRIS}$
7	0.0071	0.0409	**0.0020**	0.0072	0.0409
9	0.0212	0.2289	**0.0022**	0.0127	0.2289
11	0.0460	0.9075	**0.0026**	0.0380	0.9075
13	0.0832	2.1744	**0.0021**	0.0627	2.1744
15	0.1718	6.6258	**0.0040**	0.1381	6.6258
17	0.3346	16.8118	**0.0035**	0.2815	16.8118

Table 2: Mean squared error (MSE) for the different estimators of the expected return in the stochastic lift domain based on 25 independent replicates for each domain size. The best estimator is highlighted in bold for each domain size.

(a) 100 Monte Carlo trajectories

Domain Size	$\hat{G}_{IS}$	$\hat{G}_{PDIS}$	$\hat{G}_{SIS}$ (Lift-states)	$\hat{G}_{SIS}$ (Search-based)	$\hat{G}_{INCRIS}$
7	**0.2130**	1.2275	0.4170	0.2175	1.2045
9	1.3445	5.3590	41.5666	**1.2924**	3.7304
11	8.0707	20.8687	563.8311	**4.3758**	4.4781
13	1.9429	18.0928	5.3231	**1.3980**	15.1281
15	71.3497	55.9490	0.8916	**0.7503**	35.8578
17	3.4324	105.7343	3.3009	**0.8528**	42.7097

(b) 1,000 Monte Carlo trajectories

Domain Size	$\hat{G}_{IS}$	$\hat{G}_{PDIS}$	$\hat{G}_{SIS}$ (Lift-states)	$\hat{G}_{SIS}$ (Search-based)	$\hat{G}_{INCRIS}$
7	0.1904	0.7648	1.5650	0.1904	**0.1509**
9	0.1926	0.9861	0.5054	**0.1430**	0.7859
11	3.6139	5.6127	1.3685	**0.5136**	1.3429
13	2.3293	195.9315	1.5384	**0.5698**	5.6865
15	5.2129	14.1539	4.8391	**0.3287**	9.7411
17	11.1425	20.1486	1.9365	**1.6173**	20.7798

SIS paper

Available as

- Bossens, D. M., & Thomas, P. (2022). Low Variance Off-policy Evaluation with State-based Importance Sampling. ArXiv Preprint.

Looking back at Safe RL Workshop @ IJCAI 2022

- One-day event on various approaches to safe reinforcement learning
- Invited talks and contributed talks
- Contributed papers
- Overview of the workshop:
  `https://sites.google.com/view/safe-rl-2022/homepage`
- Safe RL 2023? Stay tuned ...

Concluding remarks

Concluding remarks

- Robustness and Safety are key challenges to AI
- Presented a few important results today:
  - Self-improvement: robustness to assumption violations (sparse rewards in non-episodic environment) through lifetime reward acceleration criterion
  - Policy reuse: robustness to multiple tasks improved when accounting for task capacity
  - Offline evolution with online adaptation, allows rapid adaptation to faults and environmental changes, for resilient robot teams
  - Constrained MDPs: the E^4 algorithm for safe exploration yielding near-optimal policies
  - Off-policy Evaluation: avoids exploration; state-based importance sampling technique for mitigating the curse of horizon
- Thanks for your attention! Any questions?

[1] Ahsan S. Alvi, Binxin Ru, Jan Callies, Stephen J. Roberts, and Michael A. Osborne. Asynchronous batch Bayesian optimisation with improved local penalisation. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:373–387, 2019.

[2] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, may 2015.

[3] Zhaohan Daniel Guo, Philip S. Thomas, and Emma Brunskill. Using options and covariance testing for long horizon off-policy policy evaluation. In *Advances in Neural Information Processing Systems (NeurIPS 2017)*, pages 2493–2502, 2017.

[4] Matthew Hausknecht and Peter Stone. Deep Recurrent Q-Learning for Partially Observable MDPs. In *Sequential Decision Making for Intelligent Agents. Papers from the AAAI 2015 Fall Symposium.*, pages 29–37, 2015.

[5] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[7] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. *arXiv preprint*, pages 1–25, 2019.

[8] Juergen H. Schmidhuber. A general method for incremental self-improvement and multi-agent learning. In Xin Yao, editor, *Evolutionary Computation: Theory and Applications.*, volume 1, chapter 3, pages 81–123. World Scientific, 1999.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint*, pages 1–12, 2017.

[10] Christopher J C H Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.