Integration of behavior planning and reinforcement learning in robotics

Aleksandr Panov

Center for Cognitive Modeling MIPT
Artificial Intelligence Research Institute (AIRI)

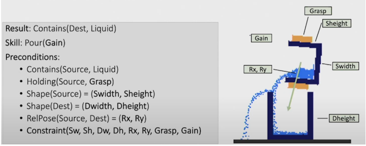
26 November 2021 – Vladivostok AI Week 2021

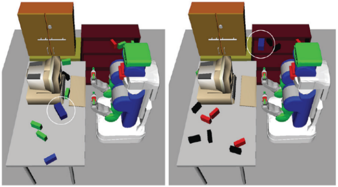
Content

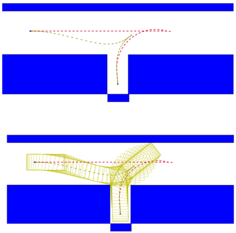
Intro: behavior planning

Behavior planning in robotics

- The agent prepare the sequence of high-level actions
- The agent do not interact with environment during planning process
- The set of actions is predefined
- The agent uses a certain knowledge representation technique







Kim, B., Wang, Z., Kaelbling, L.P., Lozano-Pérez, T.: Learning to guide task and motion planning using score-space representation. International Journal of Robotics Research. 38, 793–812 (2019).

Behavior planning in STRIPS

- State – a certain set of facts, closed atomic formulas of the predicate calculus language of the first order, represents a model of the environment in which an intelligent agent acts, an example:

$$s = \{ATR(a), AT(B, b), AT(C, c), \forall u \forall x \forall y((AT(u, x) \wedge (x \neq y)) \rightarrow \neg AT(u, y))\}$$

  - $ATR(a)$ – robot at the room a,
  - $AT(B, b)$ – box B at the room b

- The agent's actions are described using rules: both in the sets of added and deleted facts, only atomic formulas without functional symbols are used, for example:
  - Rule name: $Push(x, y, z)$
  - Condition: $C(R) = \{ATR(y), AT(x, y)\}$
  - Added facts: $A(R) = \{ATR(z), AT(x, z)\}$
  - Deleted facts: $D(R) = \{ATR(y), AT(x, y)\}$

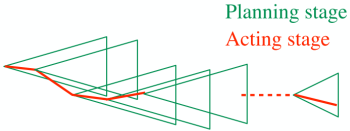
- The agent's execution of an action is reduced to the application of a rule ot modification of the state $s \xRightarrow{R,\theta} s'$, where θ – substitution of domain elements instead of variables:

$$s' = (s \setminus (D(R)\theta)) \cup (A(R)\theta))$$

Domain and planning task

- Planning domain $P = \langle s_0, \Sigma R \rangle$, where s_0 – initial state, ΣR – finite set of rules
- If G – agent's target fact, or goal, than planning task $T = \langle P, G \rangle$
- The solution of the planning task T is to find a plan that achieves the goal G
- *Plan* – is a sequence of states $s_0, \ldots, s_n$, sequence of rules $R_1, \ldots, R_n$ and sequence of substitutions $\theta_1, \ldots, \theta_n$, such that G feasible in s_n, length of the plan is equal n:

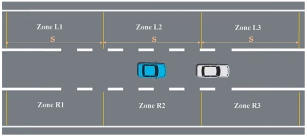
$$Plan : s_0 \xRightarrow{R_1, \theta_1} s_1 \xRightarrow{R_2, \theta_2} s_2 \ldots \xRightarrow{R_n, \theta_n} s_n$$

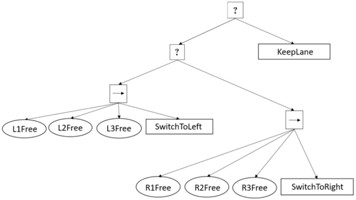


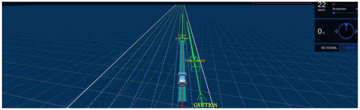
Ghallab, M., Nau, D., Traverso, P.: Automated Planning and Acting. Automated Planning and Acting. (2016).

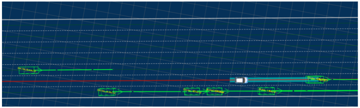
Overtaking scenario in Apollo

- Maneuver planning in autonomous driving – predefined scenarios
- Behavior tree – search space representation
- We can decode BT as string – /(&(cegY)&(ikmZ)X)
- Idea - apply genetic programming to force the search over BT variants





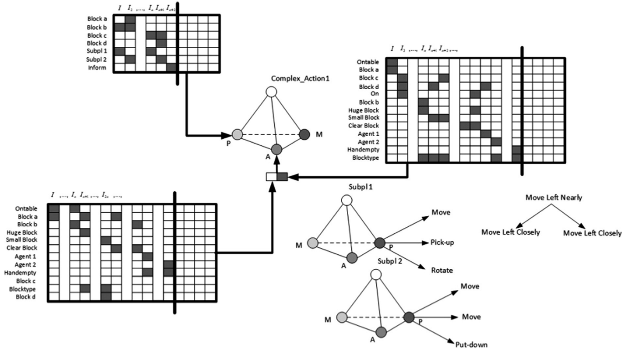




Jamal, M., Panov, A.I.: Adaptive maneuver planning for autonomous vehicles using behavior tree on Apollo Platform. In: Bramer, M. and Ellis, R. (eds.) Artificial Intelligence XXXVIII. SGAI 2021. Lecture Notes in Computer Science. (2021).

Planning in sign-based world model

Cognitive Dynamic Systems



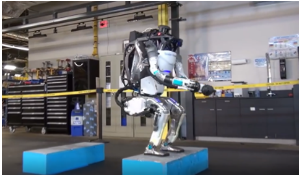
- Sign-based world model as special case of knowledge representation
- Human-robot interaction and transparent decision making
- Multi-agent planning and role distribution
- Case-based hierarchical planning

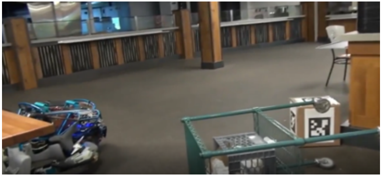
Kiselev, G., Panov, A.: Hierarchical Psychologically Inspired Planning for Human-Robot Interaction Tasks. In: Ronzhin, A., Rigoll, G., and Meshcheryakov, R. (eds.) Interactive Collaborative Robotics. ICR 2019. Lecture Notes in Computer Science. pp. 150–160. Springer (2019).

Intro: reinforcement learning

Learning and adaptation for intelligent agents

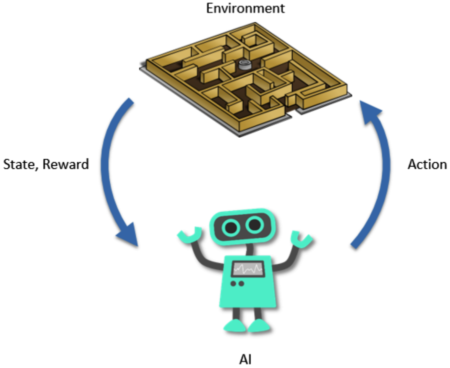
Cognitive Dynamic Systems





- Sufficient properties of the intelligent agent: adaptation and autonomy
- Robotic applications: we can replace handcrafted and analytical methods with learnable models

Reinforcement learning: preliminaries



- Split the environment and the agent – the source and acceptor of the data are explicitly present in the statement of the problem
- There is no teacher or supervisor, i.e. the error of the model is not set explicitly, but is indirectly transmitted through a reward
- Feedback from the environment may arrive with a delay
- The time parameter has a special meaning – sequential data
- The agent's actions affect the incoming data in the future

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning. An Introduction*. 2018.
Laura Graesser and Wah Loon Keng. *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. 2020.

Markov decision process

Lets $\langle S, A, T, R, G, \gamma \rangle$ – Markov decision process, where:

- S – state space,
- A – the set of actions (discrete or continuous),
- $T : S \times A \to S$ – transition function,
- $R : S \times A \to \mathbb{R}$ – reward function,
- $G : S \to \{0, 1\}$ – goal function defining termination state,
- γ – discounting factor.

The agent acts using policy function that maps S to A (stochastic or determined):

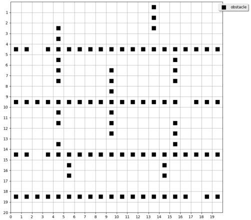
$$\pi : S \to A$$

Agent's goal – maximize expected return by policy π:

$$\mathbb{E}_\pi \sum_{t=0}^{\tau} \gamma^t R(s_t, a_t)$$

Q-function and grid example

- $E = (M, R)$ - environment, where M - grid map, $R(p_s, p_f)$ - reward generator,
- $a_t = p_t \to p_{t+1}$ - movement actions,
- $s_t \in R^{(2d)^2}$ -agent's observations (not the same as state)



Lets $Q^*(s_t, a_t) = \max_\pi \mathbb{E}[R | s_t, a_t, \pi]$ - optimal value function, then given the definition R we obtain the following Bellman equation:

$$Q^*(s, a) = \mathbb{E}_{s_t \sim E}\left[\boxed{r_t + \gamma \max_{a_t} Q^*(s_t, a_t)} \middle| s, a \right]$$

Value function approximation

To solve the Bellman equation by iterative methods, various approximations of the function are used $Q^*(s,a)$: $Q(s,a;\theta) \approx Q^*(s,a)$.

During the training process, the θ parameters are adjusted as a result of minimizing the loss function $L(\theta)$:

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_i - Q(s,a;\theta_i)\right)^2\right],$$

$$y_i = \mathbb{E}_{s_t\sim E}\left[r_t + \gamma \max_{a_t} Q(s_t,a_t;\theta_{i-1})|s,a\right]$$

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s_t\sim E}\left[\left(r_t + \gamma \max_{a_t} Q(s_t,a_t;\theta_{i-1}) - \right.\right.$$
$$\left.\left. - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right].$$

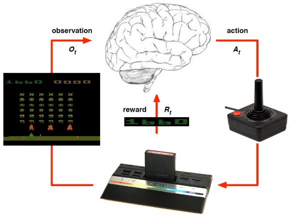
Current state-of-the-art in RL

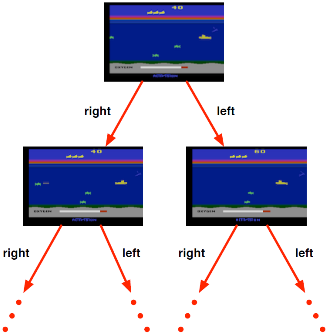
- Model-free success cases: Atari, OpenAI Five, AlphaStar etc.
- Problems with sample-efficiency:
  - DQN , A3C and Rainbow DQN have been applied to ATARI 2600 games and require from 44 to over 200 million frames (200 to over 900 hours) to achieve human-level performance
  - OpenAI Five utilizes 11,000+ years of Dota 2 gameplay
  - AlphaZero uses 4.9 million games of self-play in Go
  - AlphaStar uses 200 years of Starcraft II gameplay
- Low robustness to task-irrelevant perturbations
- Promising approaches: hierarchical methods, demonstration and imitation learning, and more effective world models

Silver D. et al. *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*. Science. 2018.
Berner C. et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019.
Vinyals O. et al. *Alphastar: Mastering the real-time strategy game Starcraft II*. DeepMind blog. 2019.

Planning and learning

Planning and learning in Atari

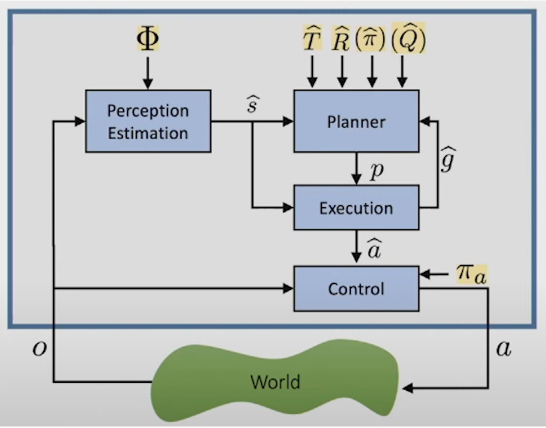




- Learning:
  - the agent does not know game rules,
  - interactive learning in the environment,
  - discrete actions and raw pixels as a state
- Planning:
  - the agent knows game rules – ideal world model,
  - the agent can star simulator,
  - planning without interaction with the environment – tree search

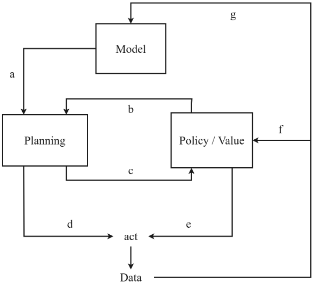
Some realization of integrated models

Cognitive Dynamic Systems



Common scheme for planning and learning

Cognitive Dynamic Systems



a – plan over a learned model,

b – use information from a policy/value network to improve the planning procedure,

c – use the result from planning as training targets for a policy/value,

d – act in the real world based on the planning outcome,

e – act in the real world based on a policy/value function,

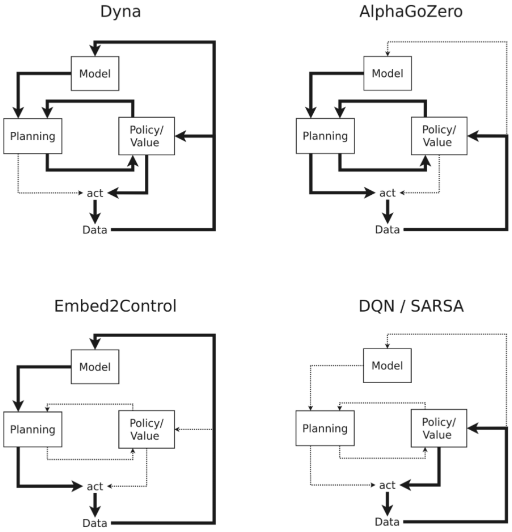
f – generate training targets for the policy/value based on real world data,

g – generate training targets for the model based on real world data

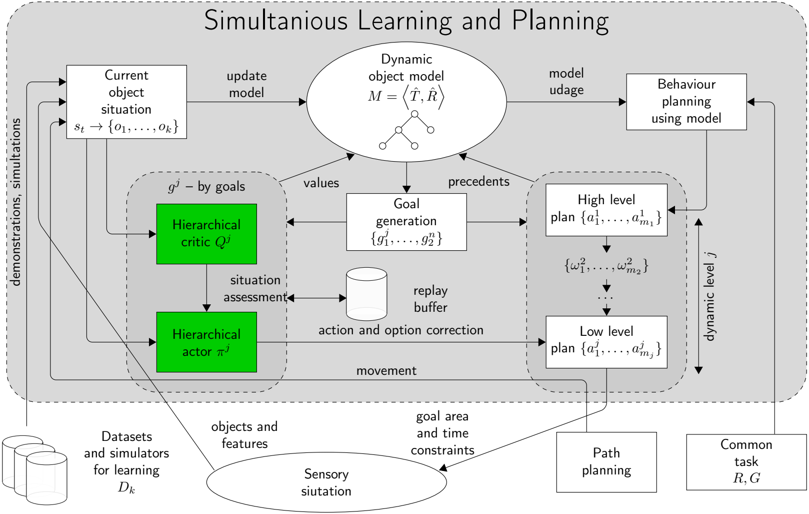
Yi, Fengji, Wenlong Fu, and Huan Liang. *Model-based reinforcement learning: A survey*. 2020.

Some realization of integrated models

Cognitive Dynamic Systems



Simultaneous learning and planning architecture



Panov A.I. *Simultaneous Learning and Planning in Hierarchical Control System for the Cognitive Agent*. 2021.

Integration options

Cognitive Dynamic Systems

Sequentional: planning one part of task and use learning policy for another part – not in the focus of current talk

Hierarchical: **planning over skills and learn skill policy**

Hybrid: **on-line switching between planner and learning policy**

On-model: **planning on learnable model** – "MCTS-based"

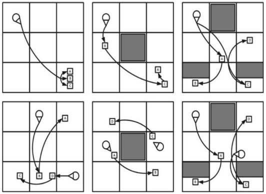
Dreamy: **learn the policy on planned trajectories**– "Dreamer-based"

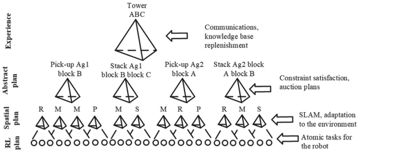
Aitygulov, E., Kiselev, G., Panov, A.I.: Task and Spatial Planning by the Cognitive Agent with Human-like Knowledge Representation. In: Ronzhin, A., Rigoll, G., and Meshcheryakov, R. (eds.) Interactive Collaborative Robotics. ICR 2018. Lecture Notes in Computer Science. pp. 1–12. Springer (2018).

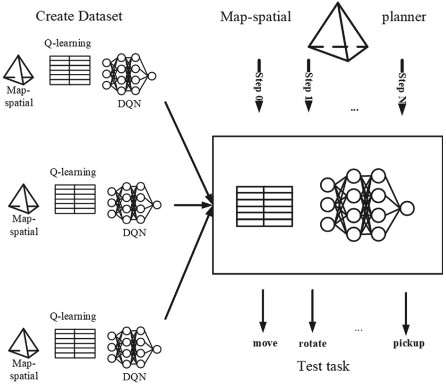
Hierarchical integration

Learning of spatial actions

- Hierarchical planning – MAP planner
- Blocks World environment
- Actions: spatial movements and object manipulations
- Using Q-learning to find a low-level policy

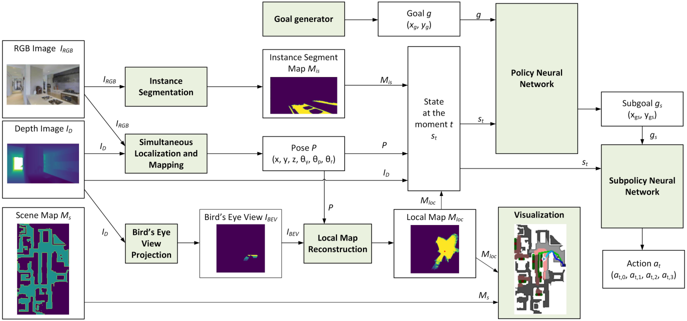




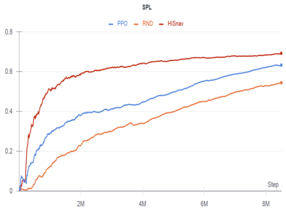


Kiselev, G., Panov, A.I.: Q-learning of Spatial Actions for Hierarchical Planner of Cognitive Agents. In: Ronzhin, A., Rigoll, G., and Meshcheryakov, R. (eds.) Interactive Collaborative Robotics. ICR 2020. Lecture Notes in Computer Science. pp. 160–169. Springer International Publishing (2020).

Reinforcement learning in robot navigation task

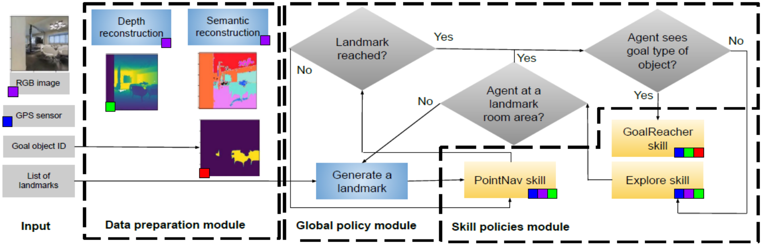


- Object navigation task: set of object classes
- Semantic scene representation: object segmentation
- End-to-end reinforcement learning approach

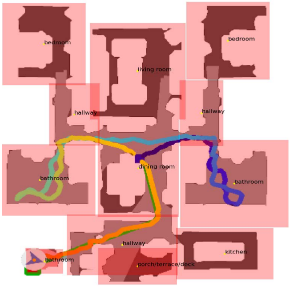


Staroverov A. et al. *Real-Time Object Navigation with Deep Neural Networks and Hierarchical Reinforcement Learning* // IEEE Access. 2020.

Hierarchical policy optimization with landmarks



- Task formulation with landmarks (brief information about rooms)
- Dividing policy into a set of skills and hierarchical structure
- Smooth policy transfer to new real-world scenes

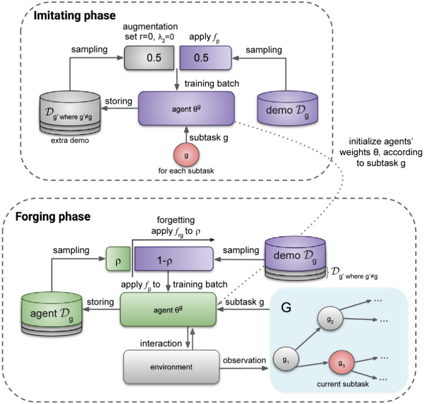


Staroverov A., Panov A. I. *Landmark Policy Optimization for Object Navigation Task*. ArXiv:2109.09512.

Robotic realization of HLPO

Staroverov A., Panov A. I. *Landmark Policy Optimization for Object Navigation Task*. ArXiv:2109.09512.

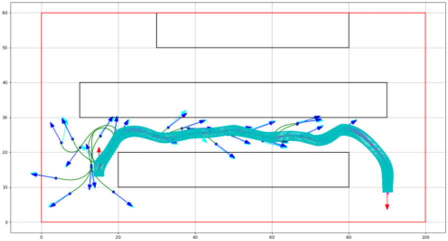
Forger as object-oriented skill formation from demonstration



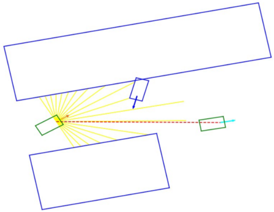
- Forgetting mechanism for learning from demonstrations
- High-level expert plan extraction from demonstrations

Skrynnik A. et al. *Forgetful experience replay in hierarchical reinforcement learning from expert demonstrations*. Knowledge-Based Systems. 2021.

Policy Optimization to Learn Adaptive Motion Primitives



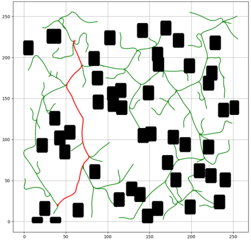
- Path planning with kinodynamic constraints
- Learnable steering function for sample-based planner
- Curriculum PPO on specially collected dataset in simulator

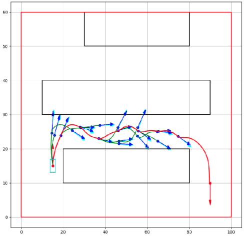


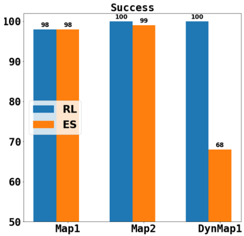
Angulo, B., Yakovlev, K., Panov, A.I.: Policy Optimization to Learn Adaptive Motion Primitives in Path Planning with Dynamic Obstacles (2021).

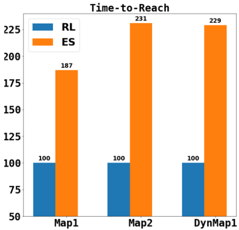
POLAMP preliminary results

Cognitive
Dynamic
Systems



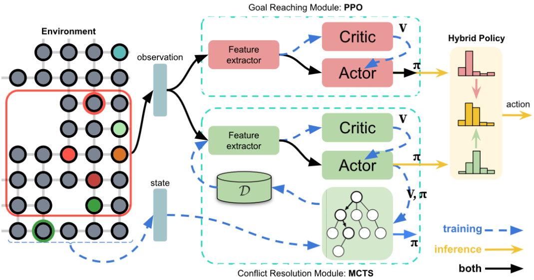






Angulo, B., Yakovlev, K., Panov, A.I.: Policy Optimization to Learn Adaptive Motion Primitives in Path Planning with Dynamic Obstacles (2021).

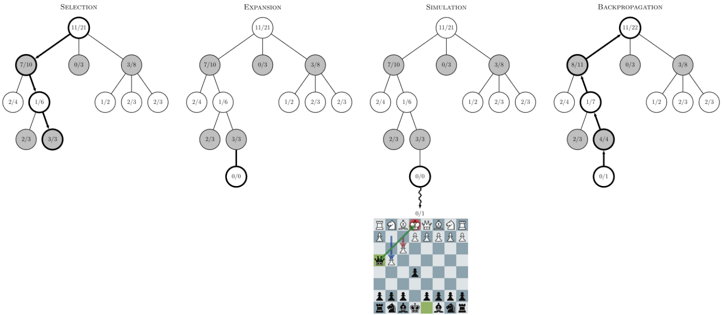
Hybrid policy optimization: decomposition



- Multi-agent pathfinding setting
- Partial observations of the grid world
- Two subproblems: goal reaching and conflict resolution

Skrynnik, A., Yakovleva, A., Davydov, V., Yakovlev, K., Panov, A.I.: Hybrid Policy Learning for Multi-Agent Pathfinding. IEEE Access. 9, 126034–126047 (2021).

Hybrid policy optimization: MCTS and results



Individual Success Rate (std)

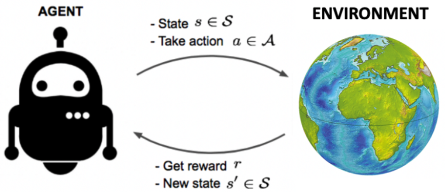
	Conflict resolution			Goal reaching	Hybrid Policy	
config	QMIX	MCTS	QMIX+MCTS	cPPO	QMIX+cPPO	MCTS+cPPO
id1-coop08x08-02	0.397 (0.015)	0.377 (0.027)	0.517 (0.006)	0.532 (0.021)	**0.607 (0.012)**	**0.607 (0.01)**
id2-coop08x08-04	0.428 (0.019)	0.307 (0.004)	0.527 (0.005)	0.497 (0.022)	**0.578 (0.008)**	0.559 (0.014)
id3-coop16x16-08	0.330 (0.003)	0.248 (0.004)	0.358 (0.004)	0.421 (0.004)	**0.450 (0.008)**	0.443 (0.005)
id4-coop32x32-16	0.220 (0.002)	0.173 (0.002)	0.271 (0.001)	0.364 (0.004)	0.367 (0.001)	**0.381 (0.005)**
id5-rnd08x08-02	0.780 (0.018)	0.715 (0.011)	0.855 (0.007)	**0.893 (0.008)**	0.883 (0.002)	0.880 (0.004)
id6-rnd08x08-04	0.768 (0.011)	0.655 (0.011)	0.837 (0.007)	0.852 (0.007)	**0.885 (0.005)**	0.852 (0.001)
id7-rnd16x16-08	0.565 (0.011)	0.486 (0.006)	0.640 (0.001)	0.718 (0.004)	**0.735 (0.007)**	0.730 (0.003)
id8-rnd32x32-16	0.329 (0.002)	0.246 (0.005)	0.416 (0.003)	0.565 (0.004)	0.562 (0.001)	**0.586 (0.008)**

Skrynnik, A., Yakovleva, A., Davydov, V., Yakovlev, K., Panov, A.I.: Hybrid Policy Learning for Multi-Agent Pathfinding. IEEE Access. 9, 126034–126047 (2021).

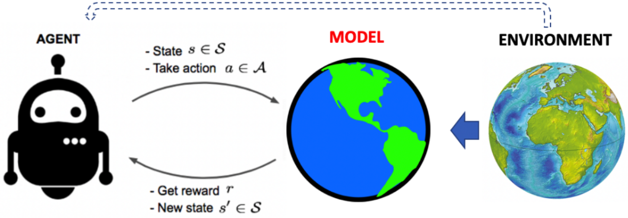
Model-based RL

Model-based RL setting

Model-free setting



Model-based setting



Formal setting

Lets $\langle S, A, T, R, G, \gamma \rangle$ – Markov decision process, where:

- S – state space,
- A – the set of actions (discrete or continuous),
- $T : S \times A \to S$ – transition function,
- $R : S \times A \to \mathbb{R}$ – reward function,
- $G : S \to \{0, 1\}$ – goal function defining termination state,
- γ – discounting factor.

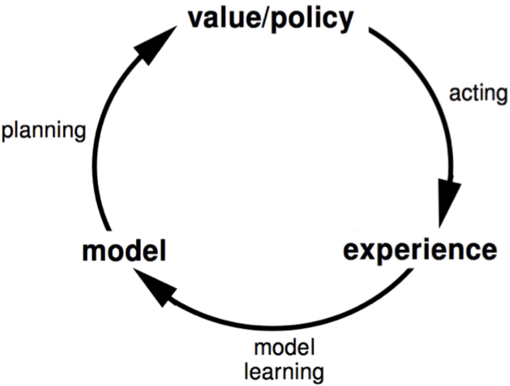
The agent has access to an updatable model $M =< \hat{T}, \hat{R} >$ and can build a plan to achieve the goal $G(s_{n+1}) = 1$ by modeling transitions:

$$Plan =< s_i, r_i, a_i, \hat{s}_{i+1}, \hat{r}_{i+1}, a_{i+1}, \dots, a_{i+n}, \hat{s}_{i+n+1} >$$

Agent's goal – maximize expected return by policy π:

$$\mathbb{E}_\pi \sum_{t=0}^{\tau} \gamma^t R(s_t, a_t)$$

Model-based RL: simple realization



Model representation

- The model M – MDP representation $\langle S, A, T, R \rangle$ parameterized by η
- Suppose that the set of states S and the set of actions A are known
- For this case the model $\mathcal{M} = \langle \hat{T}_\eta, \hat{R}_\eta \rangle$ represents a function $\hat{T}_\eta \approx T$ and a reward function $\hat{R}_\eta \approx R$:

$$s_{t+1} \sim \hat{T}_\eta(s_{t+1}|s_t, a_t),$$

$$r_{t+1} = \hat{R}_\eta(r_{t+1}|s_t, a_t)$$

- It is usually assumed that the functions of transitions and rewards are conditionally independent:

$$\mathbb{P}[s_{t+1}, r_{t+1}|s_t, a_t] = \mathbb{P}[s_{t+1}|s_t, a_t]\mathbb{P}[r_{t+1}|s_t, a_t]$$

Model learning

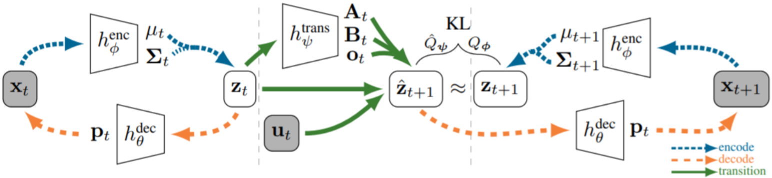
- **Goal**: evaluate a model M_η using experience $\{s_1, a_1, r_2, \ldots, s_t\}$
- Supervised learning tasks:

$$
\begin{aligned}
s_1, a_1 &\rightarrow r_2, s_2 \\
s_2, a_2 &\rightarrow r_3, s_3 \\
&\vdots \\
s_{t-1}, a_{t-1} &\rightarrow r_t, s_t
\end{aligned}
$$

- Learning a mapping $s, a \rightarrow r$ – the task of *regression*
- Learning a mapping $s, a \rightarrow s'$ – the task of *probability density estimate*
- Choose a **loss function**, for example, the root-mean-square error or KL-divergence
- Search for η parameters that minimize the empirical loss function

E2C algorithm

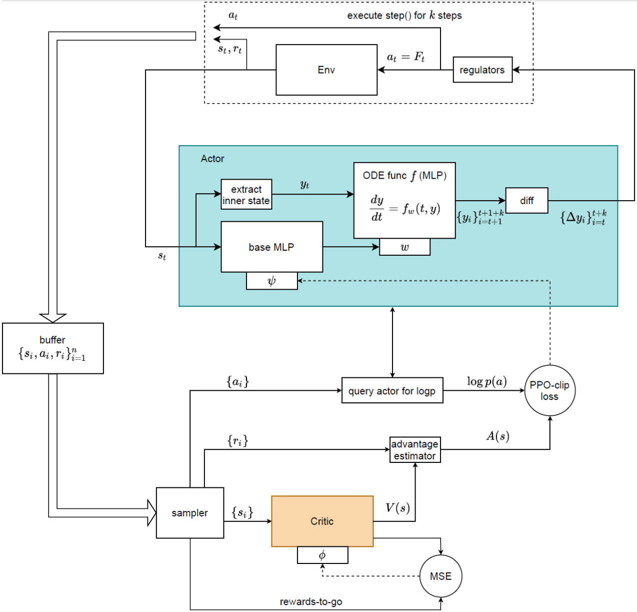
Cognitive Dynamic Systems

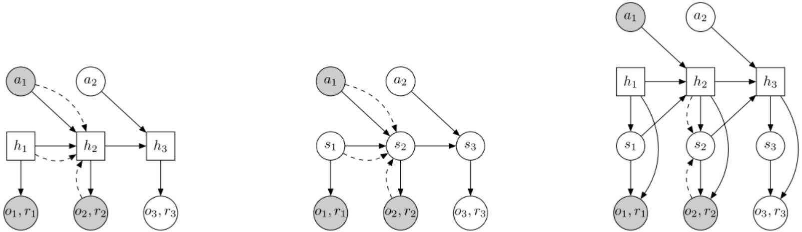


- Images of x_t as observations-states (raw pixels)
- Controlling nonlinear systems
- Variational auto-encoder $h_\phi^{enc} + h_\theta^{dec}$ as generative model
- In latent space z_t dynamic is linear:
  - using network h_ψ^{trans} to calculate matrices A_t, B_t, o_t and predict the next $\hat{z}_{t+1}$,
  - apply KL-divergence between $\hat{z}_{t+1}$ and z_{t+1} as a loss function

Watter J. et al. *Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images*. NeurIPS. 2015.

Neural ODE as a model: preliminary schema



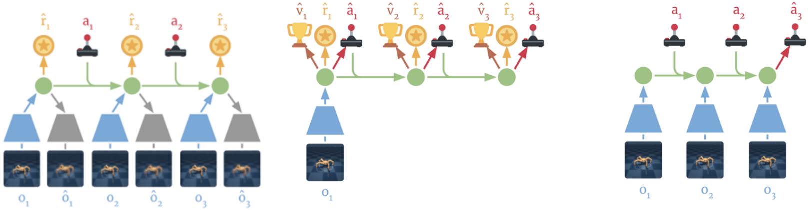


Recurrent state-space model:

- partially observable case and raw feature space,
- dynamics in latent space,
- high computational efficiency

Hafner D. et al. *Learning latent dynamics for planning from pixels*. ICML. 2019.
Hafner D. et al. *Dream to Control: Learning Behaviors by Latent Imagination*. ICLR. 2020.

Dreamer algorithm

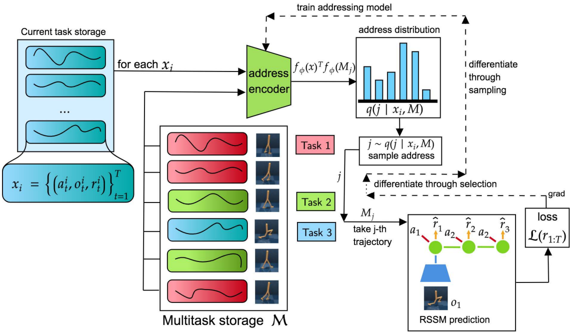


- Model components:
  - representation model $p_\theta(s_t|s_{t-1}, a_{t-1}, o_t)$,
  - observation model $q_\theta(o_t|s_t)$,
  - reward model $q_\theta(r_t|s_t)$,
  - transition model $q_\theta(s_t|s_{t-1}, a_{t-1})$
- The model is trained with a variational loss function on mutual information:

$$I(s_{1:T}; (o_{1:T}, r_{1:T})|a_{1:T}) - \beta I(s_{1:T}, i_{1:T}|a_{1:T})$$

- The agent is trained to maximize the value function along imaginary trajectories
- Experience is collected taking into account the representation model

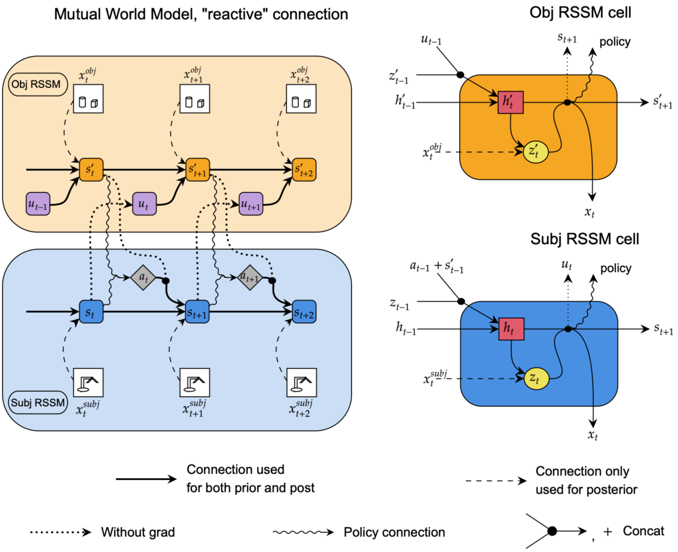
Retrospective multitask adaptation



- Adapted RSSM-like world models for a multitask case
- Original addressing mechanism, the training of which can be formalized in the form of a one-step meta-MDP
- Using model-based RL with an addressing mechanism in a photorealistic robotic simulator

Zholus A. et al. *Multitask Adaptation by Retrospective Exploration with Learned World Models*. 2021.

Dreamer with subject and object layers



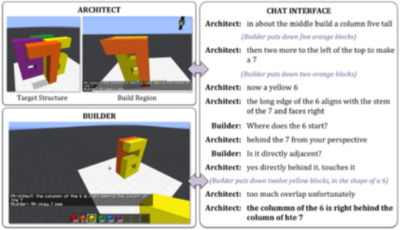
Conclusion

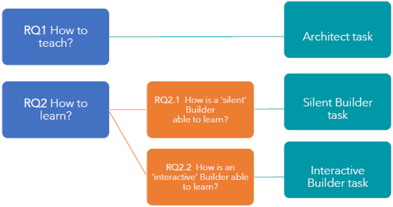
- We need to increase the level of autonomy of robots
- Behavior planning can be more adaptive when integrated with learning methods
- Learnable models can be applied for complex tasks when integrated with planning methods
- There are several options for integration: sequential, hierarchical, on-model and dreamy
- We create methods realizing all types of integration:
  - hierarchical planning and learning: MAP-RL, HLPO, HPS, ForgER, POLAMP, HPS,
  - model-based reinforcement learning: neural ODE, RAMA, dual Dreamer
- We develop the unified architecture for simultaneous learning and planning – SLAP agent

Collaborators:

Alexey Skrynnik (AIRI),Alexey Staroverov (AIRI), Artem Zholus (MIPT), Andrey Gorodetsky (MIPT), Brian Angulo (Integrant), Konstantin Yakovlev (FRC CSC RAS)

IGLU challenge





- Agent can easily extract information about objects
- Agent should learn to construct more complex objects
- We need to realize some example of object and symbol grounding

https://www.iglu-contest.net/

Special issue on Neural Symbolic Integration

Special Issue “Neural-Symbolic Cognitive Architectures” in Cognitive Systems Research (Q1 in WoS by 2021 JCR) – August 2021

- Neural-Symbolic Integration approaches
- Symbol grounding problem
- Reinforcement learning methods in cognitive systems
- Hybrid knowledge representation
- Vector-symbolic architectures
- Applied semiotics and semiotic cognitive architectures
- Cognitive and Social Robotics
- Integrated models of Learning and Reasoning
- Biologically inspired cognitive architectures
- Emotionally intelligent agents
- Simultaneous Learning and Planning
- Human-analogous active learning
- Artificial and collaborative creativity
- Explainable AI models and systems
- General theory of neural-symbolic computation

Thank you for your attention!

cogmodel.mipt.ru
airi.net
raai.org

panov.ai@mipt.ru