# Improving Sample Efficiency in Behavior Learning by Using Sub-optimal Planners for Robots

**Conference Paper** · July 2021

CITATIONS
0

READS
47

**3 authors:**

Emanuele Antonioni
Sapienza University of Rome
**7** PUBLICATIONS **5** CITATIONS
SEE PROFILE

Francesco Riccio
Sapienza University of Rome
**22** PUBLICATIONS **65** CITATIONS
SEE PROFILE

Daniele Nardi
Sapienza University of Rome
**382** PUBLICATIONS **14,553** CITATIONS
SEE PROFILE

# Improving Sample Efficiency in Behavior Learning by Using Sub-optimal Planners for Robots

Emanuele Antonioni[1], Francesco Riccio[1], and Daniele Nardi[1]

Dept. of Computer, Control, and Management Engineering
Sapienza University of Rome, Rome (Italy)
{lastname}@diag.uniroma1.it.

**Abstract.** The design and implementation of behaviors for robots operating in dynamic and complex environments are becoming mandatory in nowadays applications. Reinforcement learning is consistently showing remarkable results in learning effective action policies and in achieving super-human performance in various tasks – without exploiting prior knowledge. However, in robotics, the use of purely learning-based techniques is still subject to strong limitations. Foremost, sample efficiency. Such techniques, in fact, are known to require large training datasets, and long training sessions, in order to develop effective action policies. Hence in this paper, to alleviate such constraint, and to allow learning in such robotic scenarios, we introduce SErP (Sample Efficient robot Policies), an iterative algorithm to improve the sample-efficiency of learning algorithms. SErP exploits a sub-optimal planner (here implemented with a monitor-replanning algorithm) to lead the exploration of the learning agent through its initial iterations. Intuitively, SErP exploits the planner as an expert in order to enable focused exploration and to avoid portions of the search space that are not effective to solve the task of the robot. Finally, to confirm our insights and to show the improvements that SErP carries with, we report the results obtained in two different robotic scenarios: (1) a cartpole scenario and (2) a soccer-robots scenario within the RoboCup@Soccer SPL environment.

**Keywords:** Automated Planning, Reinforcement Learning, Decision-making

## 1 Introduction

Nowadays, robots can operate in very challenging scenarios such as healthcare, security, industry, and domestic aid [3]. These contexts are highly dynamic, not structured, and characterized by unpredictable events [22]. Consequently, the deployment of robots with predefined decision-making processes or static behavioral rules is not recommended and often not effective. Instead, they must be provided with the capability of (1) learning through continuous interaction

with their environment; (2) adapting to an unexpected situation; (3) recovering from faults; and (4) generalize their behavior to a new situation.

In the literature, proposed solutions can be coarsely categorized into two main classes: planning-based [12] and learning-based [16] approaches. In the former case, the robot behavior can be explicitly formalized over a model of the environment, which results in immediate solutions and robust action policies. Nevertheless, planning-based techniques usually lack generalization to similar states and could require expert domain knowledge to be correctly designed. Instead, the latter category aims to learn an optimal action policy through continuous interaction with the environment. In particular, reinforcement learning (RL) techniques have been widely adopted to determine robust action policies that can generalize to unknown situations. However, learning approaches are sample inefficient, and to converge to competitive policies, they require a tremendous amount of iterations and training samples [2], which make their use less appealing in robotics.

In this paper, we attack the problem of sample efficiency by combining the main advantages both of planning and learning paradigms for obtaining a robust decision-making process. We introduce SERP (Sample Efficient Robot Policies), which is specifically designed to enable and support the focused exploration of learning algorithms by exploiting a sub-optimal planner. In particular, SERP is an iterative algorithm that uses a sub-optimal planner based on an incomplete transition model as an expert for (1) initializing the robot policy; and for (2) preventing the exploration of state-action space portions that do not contribute to task completion. SERP is configured to follow the planner policy throughout its first iterations, accumulate training samples, and then gradually switch to the action policy learned at training time. Importantly, the planner remains active during the entire learning routine, and it is exploited to maintain a focused and safe exploration. Such a configuration allows the robot to achieve competitive performance with a reduced number of training samples and to avoid the choice of actions that are not focused on the current task.

The goal of this work is to introduce a novel methodology in robot learning that improves sample efficiency, making learning approaches more practical and safe in robotics. In Figure 1 is shown the selection mechanism of the planner, which marks with utilities the possible future actions, assigning low utilities to potentially useless ones (highlighted in red) and with good ones the promising useful actions (highlighted in blue). Another key feature of this algorithm is assigning the lowest possible utilities to actions that could negatively impact robot safety. To confirm this feature we assign extremely negative reward signals to actions that can impact the robot's safety, in order to have an high impact of such actions in the overall collected reward. It is important to remember that we consider robot safety during learning as the learner's ability to ensure research space configurations that do not damage the robot embodiment, the environment and nearby people. In fact the planner contains information that regulates the behavior of the robot to prevent these situations. We validate SERP in two different robotic scenarios: a cart-pole task and robot soccer task. In the first

(a) (b)

**Fig. 1.** The figure shows SERP sample efficiency that avoids the exploration of red actions and promotes portions of the state space that are meaningful to the task (blue).

scenario, the algorithm is used to balance the cart-pole within a reduced number of iterations. In the second scenario, the proposed approach is exploited to learn a defender robot's action policy when opposing an attacker. The proposed tasks have been learned using simulated environments; however, the action policy has also been deployed in real settings in the RoboCup scenario.

## 2 Related work

SERP relies on both planning and learning approaches. Accordingly, we survey existing literature, and we focus on approaches that use RL or Automated Planning to face decision-making problems for robots.

Automated planning techniques have been widely used in the recent years to implement articulated decision-making processes; for example, in [8], authors formalize all the robot behaviors are by defining the tuple of static axioms, effect axioms, action preconditions, and initial state. Due to such a formalization, the authors can promptly trace the entire robot behavior and recover from the planners' failures. In a different context, [13] propose a formal language $\mathcal{KL}$ to express planning domains with incomplete knowledge and to generate plans in partially observable environments. Also, in the context of soccer robots, there are some planning-based solutions for robot behaviors; in [20] authors propose a work based on the exploration of future states. They introduce a method for fast decision-making. The outcome of each possible action is simulated based on the estimated state of the situation. The simulation of a single action is split into several simple deterministic simulations, considering the uncertainties of the action model's estimated state. However, planning-based approaches cannot generalize to unknown situations, and the contributions mentioned above all rely on solid domain expertise to define the planning procedure. On the other hand, learning-based approaches are receiving increasing interest in the last years. Several steps have been taken in this direction in robotics, although, due to demanding computational requirements, many RL solutions cannot be easily adopted in robotics [25]. To alleviate computational constraints, learning algorithm are used for system identification of high-dimensional problems [29] and learning control policies in dynamic environments [5]; advanced manipulation [26]; and

optimizing robot behaviors [23]. Moreover, several model-based RL techniques for decision making have been used over time, such as [9] and [6]. However, these applications require a complete model of the environment to be provided to the learning algorithm. Even for soccer robots, a lot of work has been done using reinforcement learning for learning players' skills. In [1] authors discuss an imitation learning [27] system. The agent has to learn the dribble and searching skills, imitating the motion of the Striker agent (model-based) of the former world champions of the RoboCup@Soccer SPL competition. However, this approach still suffers the requirement of a previously modeled and efficient player to imitate. The dribble with the ball problem is also addressed in [19]. The paper describes and compares several hierarchical learning strategies for designing robot skills. However, the results show that there is a trade-off between sample efficiency and the final performance of the model. More closely related to our approach, exploiting both planning and learning approaches, [18] propose DARLING, a method that uses planning to bound the agent's behavior to reasonable choices during reinforcement learning to adapt it to the environment. However, DARLING does not evaluate the actions of the planner during training. Thus, the learner agent can explore undesired portions of the state-action space and does not preserve a focused exploration. Differently, [14] introduces TMP-RL to integrate Task Motion Planning TMP and RL, while [10] exploits a planning algorithm to model the rewards function of a learning algorithm in a multi-agent system. However, these contributions' focus is to improve policy generalization and not to guarantee sample efficiency. [4] present R-Max, a model-based RL algorithm that learns a complete transition model of the environment that can be used to define planning algorithms. However, differently from our solution, R-Max is randomly initialized, and thus, does not guarantee a focused and safe exploration since the first iterations. The problem of learning from sub-optimal teachers has been addressed in several recent works. In [17], authors present a learning algorithm that uses an ensemble of sub-optimal teachers. The robot successfully manages to outperform the teacher performances and to complete the task. Nevertheless, in this work, the teachers are meant to solve part of the problem. In our work, the teacher cannot solve any part of the problem itself, but it is only used for deciding which action to avoid or to explore in the initial part of the training phase. This allows the teacher to be way simpler to model both with a handcrafted or a learning-based technique. Similarly, in [15], the authors introduce a residual method to correct the robot control performed by a sub-optimal controller using an RL-based approach. In this case, the method differs strongly from the one that we propose. The cited work, in fact, acts on the correction of a continuous control signal, which is not applicable to a decision-making process like the ones addressed in this paper. In [7] a method that alternates Imitation Learning phases to Reinforcement Learning ones starting the imitation from a sub-optimal teacher is introduced. This work manages to outperform the teacher performances and speed up the RL algorithm's learning process. Differently, our work also introduces a method for avoiding actions that are risky for robot safety keeping the planner always active in the background.

Summarizing, several procedures for efficient decision-making processes have been created using automated planning and reinforcement learning, and competitive results can be achieved with both. This work investigates the use of both planning and learning to improve sample efficiency by exploiting a planner for guaranteeing focused exploration of the state-action space. Hence, the contribution of SERP is to generate competitive action policies already since the first iterations of the algorithm using just a partial model of the environment, then use this to prevent the learner from exploring invalid portions of the search space.

## 3 SErP

In order to iteratively refine an action policy, SERP relies upon two main building blocks: a planner and a learner. These two components are used to achieve focused search-space exploration and to learn effective action policies with a reduced number of training samples. To formalize SERP we adopt the MDP notation, where decision-making is expressed as a tuple

$$MDP = (S, A, \mathcal{T}, R, \gamma), \tag{1}$$

where $S$ and $A$ represent the set of states and actions respectively, $\mathcal{T} : S \times A \times S \rightarrow [0, 1]$ is a stochastic transition model from state $s \in S$ to $s' \in S$, when taking action $a \in A$, $R : S \times A \rightarrow R$ is the reward function, and $\gamma$ is a discount factor in $[0, 1)$. At each iteration, the output of the algorithm is a policy $\pi_i$ that is used to determine the next action to perform.

### 3.1 Algorithm

During the first phase of the algorithm, SERP achieves sample efficiency by exploiting a planner, as an expert, to collect training samples. In fact, the planner is used to explore the search-space with a sub-optimal action policy and provide prior knowledge to the learner, which is refined at each time a state-action transition is observed. During in the initial phase, the agent only considers the planner's actions and disregards the learner's output. Afterward, once enough samples are collected, the agent's focus gradually shifts toward the learner policy. To guarantee safety during the learning process the planner is always active and available to validate the learner's choices. It is worth noting that the concept of safety expressed in this paper refers to the robot embodiment, the environment, and the people involved in the task.

In particular, SERP follows the policy generated by the planner in its first iterations. In fact, the planner guarantees that the learner collects only training samples from useful portions of the search space. This is especially important during policy initialization since the majority of learning algorithms require demanding exploration phases. Then, with a $\rho$ probability, increasing over time, selects actions in accordance with the policy $\pi$ of the learner. Algorithm 1 reports the SERP procedure: (1) policy $\pi_0$ with a random dataset $d_0$; (2) sample

**Algorithm 1:** SErP.

**Input:** $D_0$ dataset of random state-action pairs, $\pi_P$ planner policy
**Output:** $\pi_I$ learner policy
**Data:** $I$ number of iterations, $\rho$ learner probability, $\Delta\rho$ learner probability increment, $R_{min}$ reward minimum value

```
begin
1[  Train π_0 on D_0
    for i = 1 to I do
        p ← UniformRandom(0, 1)
2[      if p > ρ then
            Get action a from π_P(s_{t−1}).
        end
        else
            Get action a from π_{i−1}(s_{t−1}).
            s' ←T(a, s_{t−1})
3[          if R(s') < R_min then
4[              Get action a from π_P(s_{t−1}).
            end
        end
        s' ←τ(a, s_{t−1})
        r ←R(s')
5[      D^{0:i} ← {s, a, r, s'} ∪ D^{0:i−1}
6[      Train π_i on D^{0:i}
        ρ ←ρ + Δρ
    end
    return π_I
end
```

an action from the learner or the planner according to a probability $\rho$; (3) if the action is sampled from the learner, it is evaluated with the minimum reward threshold $R_{min}$; (4) if it is lower than $R_{min}$, then the action is selected from the planner policy; (5) upon action completion, a new sample is generated and aggregated to the dataset; (6) finally the learner policy is updated with a gradient descent algorithm and the $\rho$ probability is increased. It is important to highlight that SErP is agnostic to the planner and the learner implementation. This is an important feature that allows an effective instantiation to various domains that require specific planners and/or learners. Here, we provide a description of SErP when the planner and the learner are instantiated with a monitor replanning algorithm [24] and with DQN [21] respectively.

### 3.2 Planner

On-line planning techniques have demonstrated satisfactory results when physical agents are deployed in dynamic and unpredictable environments. In the literature, monitor-replanning (MrP) [24] has shown promising results being re-

sponsive and easy to configure. MrP algorithms replan the agent actions at each iteration and after each action is executed. This guarantees that each action of the agent is informed, and the plan is generated by observing the current state of the environment. The MrP algorithm implemented in SERP executes four steps at each iteration: (1) monitors the environment to get the current state; (2) expands the planning graph in accordance to predefined actions effects, and preconditions; (3) visits the graph to select the path that maximizes the reward of the agent $R$ – usually defined by an expert of the domain; (4) executes the first action of the path. It is important to highlight that we define a planning graph as a structure in which nodes represent states of the environment, and edges represent actions that transition from a state to another.

### 3.3 Learner

Deep RL has shown super-human performance in various applications demonstrating remarkable generalization capability and competitive action policies. In this work, we take advantage of the success of such approaches, and we implement the learner of SERP with a basic Deep RL algorithm – DQN [21]. In this work, we formalize the state of the agent as a feature vector characterizing meaningful landmarks of the search space. While the output of the learner is a policy $\pi$ that is refined iteratively. The DQN algorithm implemented executes the following three steps: (1) aggregates training samples to the original dataset (a reply-buffer) each time a transition is observed; (2) randomly samples state-action pairs from the dataset to compose mini-batches of i.i.d samples; (3) updates the learner policy by minimizing the loss function. Such a loss function is defined as the mean squared error (MSE) between the Q-value obtained by the Bellman equation and the network output (Equation 2).

$$L\left(\theta_i\right) = E_{s,(\cdot)}\left[\left(y_i - Q\left(s, a; \theta_i\right)\right)^2\right] \tag{2}$$

where $\theta$ is the set of parameters of the approximator and $y_i$ us defined as in Equation 3.

$$y_i = \begin{cases} R_T & \text{for terminal state } s_T \\ R_{t+1} + \gamma \max_{a'} Q\left(s_{t+1}, a'\right) & \text{for non-terminal state } s_t \end{cases} \tag{3}$$

## 4 Experimental Evaluation

In this section, we aim to demonstrate the performances of the SERP algorithm in terms of sample efficiency. Hence, we configured two applications: a cart-pole robot environment and a RoboCup@Soccer environment. In both experiments, the learner is implemented as DQN. The training is configured with an Adam optimizer with learning rate = 0.001 and decay factor = 0.9. The $\epsilon$ for the DQN $\epsilon - greedy$ policy is set to 0.3. The learner probability $\rho$ takes values in $(0, 1]$, and increase at each iteration by a $\Delta\rho$ selected depending on the application. In this evaluation, the CartPole scenario is configured with $\Delta\rho = 0.01$ ,the robot

soccer environment $\Delta\rho = 0.1$ this is given by the different amount of samples that the two environments have. To validate SERP, we compare its performance with vanilla-MrP and vanilla-DQN algorithms, as well as PPO, which represent a state-of-the-art baseline [28,11].

### 4.1 CartPole

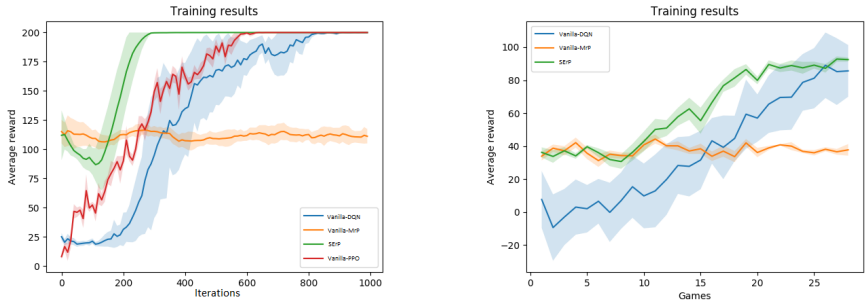
The CartPole scenario is composed of an under-actuated platform made by a pole with an un-actuated joint attached to a cart that moves over a track (see Figure 1, bottom). The robot can execute two actions: Left and Right. The former applies a force to the cart pushing it to the left while the latter to the right. The pendulum starts upright, and the goal is to prevent it from falling. An episode ends when (1) the distance between the pole and its upright position is greater than 15 degrees or (2) when the cart is pushed beyond a certain limit on the track. To formalize the cartpole scenario as a planning problem, we define the planner's state as a tuple of four elements: pole position, pole velocity, cart position, and cart velocity.

In this scenario, SERP is compared with OpenAI baselines[1] of the vanilla-DQN and PPO and with a vanilla-MrP implementation. All algorithms are let run for 1000 episodes, and the cumulative average reward of each episode is reported in Figure 2(a). On the y-axis, the average cumulative reward value while on the x-axis the training iterations. The solid line represents the average value, while the light area represents their standard deviation. The figure shows that the prior knowledge allows SERP to rapidly increase during the time, due to the exploration of meaningful portions of the search space. In fact, PPO and DQN (that are initialized randomly) require a larger amount of training samples and result to be sample inefficient. Moreover, DQN standard deviation is considerably larger, which suggests that random factors in the initial exploration can significantly affect the action policy's overall performance and accuracy. The plot confirms our insights and shows that SERP converges to competitive results with a reduced number of training samples being extremely sample efficient. Moreover, it is worth noticing that the reward signal never decays also in the first iterations that confirms that the planner prevents the learner from exploring not useful portions of the search space.

### 4.2 Soccer Defender

RoboCup@Soccer is a challenging test-bed for decision-making algorithms due to unpredictability, partial observability, unstructured environments, and highly dynamic events. In this experimental evaluation, we validate sample efficiency and confirm that SERP is practical in complex robotic environments such as multi-robot adversarial setting. Specifically, we task SERP to learn a defensive robot's behavior in fighting for the ball with an opponent attacker. Hence, the experiment configures two robots, a defender that runs SERP, and an attacker

[1] `https://github.com/openai/baselines`




(a) (b)

**Fig. 2.** a) Training results on CartPole. (b) Training results on Robocup@Soccer. Full line: average, Dull area: Standard deviation

that plays-out a predefined action policy. The goal of the defender is to contrast the attacker and to prevent it from scoring. We configured three possible actions that the defender can perform, and we let SErP decide which is the appropriate depending on the current state of the environment. Namely, in order to block the attacker, the defender can: cover the goal not allowing the opponent to move towards or kick (Figure 3)(a); advance and push the ball on the side of the attacker (Figure 3)(b); make a frontal contrast (Figure 3)(c). In order to configure the planner in this scenario, the state is represented as a tuple of 3 elements: robot pose, ball position, and opponent pose. It is important to notice that in this environment, some robot actions could damage the platforms. For example, while learning, the robot could learn to kick other robots to prevent them from scoring. Conversely, the SErP planner not only guarantees focused exploration but also prevents the learner from exploring such actions.

In this scenario, SErP and baseline algorithms are compared in different games lasting ten minutes. A training episode is started each time the agent robot and the nearest opponent enter within a radius of 70 centimeters of the ball. It is considered finished when at least one of the robots exits such an area, or the ball is secured by the defender, or the attacker scores. Each episode is then given a cumulative reward depending on: (1) the robots' positions with respect to the ball, (2) the coverage of the goal, and (3) the final outcome of the contrast. To best simulate learning on a real robot, the simulations have been run in real-time. Each training cycle is set to last 28 games. Thus, the time required to complete a training session of about 280 minutes. Algorithms are tested by repeating the training session five times and reporting their cumulative average rewards and standard deviation. Due to PPO long training time, the algorithm proved itself to be not a practical solution in such scenarios, and therefore it has been not included in this evaluation.

As for the cartpole scenario, Figure 2(b) reports the cumulative average reward of the soccer scenario. A similar analysis can also be conducted in this


**Fig. 3.** a) Goal cover action, (b) Side advance action (c) Frontal contrast action

experiment. SERP outperforms both vanilla-MrP and vanilla-DQN algorithms since first iterations and demonstrates to be a more practical solution that easily generates competitive action policies. In the end, both models converge on similar cumulative reward values even though SERP is more sample-efficient. The avoidance of execution of critical actions for the robot's safety in this scenario is also confirmed by the obtained results in terms of cumulative reward, in fact, very negative rewards have been assigned to this type of situations.

## 5 Conclusion

In this work, we introduce SERP, a new methodology for performing robot behavior learning using a partial model. The SERP algorithm is based on the use of a model-free RL algorithm and a model-based planning one. To realize the required online planner, only a basic model of the environment is needed; in fact, the planner itself does not need to completely solve the problem. This implies that it is not required the figure of an expert to define the transition model of this kind of planning system. With this simple shaping in the initial action selection, we can significantly improve the sample efficiency of the model-free DQN algorithm. Also, using elementary modeling of harmful actions for robot safety, it is possible to avoid selecting them and preventing potential damages on robots.

There are several directions to extend the work presented in this paper. One overall is to scale this algorithm over different behaviors and demonstrate its applicability to several environments and situations. For example, in the soccer domain, different situations can benefit from the use of our approach. Corner kick

situations, in fact, can be easily addressed with SERP. Differently, another way to extend the proposed approach is to generalize it to a hierarchical problem formulation. In this way, the problem can be divided, and the learner can be challenged to learn smaller tasks in a more efficient way.

## References


1. Aşık, O., Görer, B., Akın, H.L.: End-to-end deep imitation learning: Robot soccer case study. In: Holz, D., Genter, K., Saad, M., von Stryk, O. (eds.) RoboCup 2018: Robot World Cup XXII. pp. 137–149. Springer International Publishing (2019)
2. Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., Mordatch, I.: Emergent tool use from multi-agent autocurricula (2019)
3. Ben-Ari, M., Mondada, F.: Robots and Their Applications, pp. 1–20. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-62533-$1_1$, https://doi.org/10.1007/978-3-319-62533-1_1
4. Brafman, R.I., Tennenholtz, M.: R-max - a general polynomial time algorithm for near-optimal reinforcement learning. J. Mach. Learn. Res. **3**, 213–231 (Mar 2003). https://doi.org/10.1162/153244303765208377, https://doi.org/10.1162/153244303765208377
5. Böhmer, W., Springenberg, J.T., Boedecker, J., Riedmiller, M., Obermayer, K.: Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. KI - Künstliche Intelligenz **29**(4), 353–362 (2015). https://doi.org/10.1007/s13218-015-0356-1, this work was partially funded by the German science foundation (DFG) within the priority program SPP 1527.
6. Chatzilygeroudis, K., Rama, R., Kaushik, R., Goepp, D., Vassiliades, V., Mouret, J.: Black-box data-efficient policy search for robotics. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 51–58 (2017)
7. Cheng, C.A., Yan, X., Wagener, N., Boots, B.: Fast policy learning through imitation and reinforcement. arXiv preprint arXiv:1805.10413 (2018)
8. De Giacomo, G., Iocchi, L., Nardi, D., Rosati, R.: Planning with sensing for a mobile robot. In: Steel, S., Alami, R. (eds.) Recent Advances in AI Planning. pp. 156–168. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
9. Deisenroth, M., Rasmussen, C.E.: Pilco: A model-based and data-efficient approach to policy search. In: Proceedings of the 28th International Conference on machine learning (ICML-11). pp. 465–472 (2011)
10. Devlin, S., Kudenko, D.: Plan-based reward shaping for multi-agent reinforcement learning. The Knowledge Engineering Review **31**(1), 44–58 (2016). https://doi.org/10.1017/S0269888915000181
11. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
12. Ingrand, F., Ghallab, M.: Deliberation for autonomous robots: A survey. Artificial Intelligence **247**, 10 – 44 (2017). https://doi.org/https://doi.org/10.1016/j.artint.2014.11.003, http://www.sciencedirect.com/science/article/pii/S0004370214001350, special Issue on AI and Robotics
13. Iocchi, L., Nardi, D., Rosati, R.: Generation of strong cyclic plans with incomplete information and sensing **2**(4), 58–65 (2005)

14. Jiang, Y., Yang, F., Zhang, S., Stone, P.: Integrating task-motion planning with reinforcement learning for robust decision making in mobile robots. ArXiv **abs/1811.08955** (2018)
15. Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J.A., Solowjow, E., Levine, S.: Residual reinforcement learning for robot control. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 6023–6029. IEEE (2019)
16. Kober, J., Peters, J.: Reinforcement Learning in Robotics: A Survey, pp. 9–67. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-03194-1$_2$, https://doi.org/10.1007/978-3-319-03194-1_2
17. Kurenkov, A., Mandlekar, A., Martin-Martin, R., Savarese, S., Garg, A.: Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. arXiv preprint arXiv:1909.04121 (2019)
18. Leonetti, M., Iocchi, L., Stone, P.: A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. Artifical Intelligence **241**, 103–130 (2016). https://doi.org/10.1016/j.artint.2016.07.004, http://dx.doi.org/10.1016/j.artint.2016.07.004
19. Leottau, D.L., Ruiz-del Solar, J., MacAlpine, P., Stone, P.: A study of layered learning strategies applied to individual behaviors in robot soccer. In: Almeida, L., Ji, J., Steinbauer, G., Luke, S. (eds.) RoboCup 2015: Robot World Cup XIX. pp. 290–302. Springer International Publishing, Cham (2015)
20. Mellmann, H., Schlotter, B., Blum, C.: Simulation based selection of actions for a humanoid soccer-robot. In: Behnke, S., Sheh, R., Sarıel, S., Lee, D.D. (eds.) RoboCup 2016: Robot World Cup XX. pp. 193–205. Springer International Publishing, Cham (2017)
21. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
22. Mohanan, M., Salgoankar, A.: A survey of robotic motion planning in dynamic environments. Robotics and Autonomous Systems **100**, 171 – 185 (2018). https://doi.org/https://doi.org/10.1016/j.robot.2017.10.011, http://www.sciencedirect.com/science/article/pii/S0921889017300313
23. Noda, K., Arie, H., Suga, Y., Ogata, T.: Multimodal integration learning of robot behavior using deep neural networks. Robotics and Autonomous Systems **62**(6), 721 – 736 (2014). https://doi.org/https://doi.org/10.1016/j.robot.2014.03.003, http://www.sciencedirect.com/science/article/pii/S0921889014000396
24. Onaindia, E., Sapena, O., Sebastia, L., Marzal, E.: Simplanner: An execution-monitoring system for replanning in dynamic worlds. In: Brazdil, P., Jorge, A. (eds.) Progress in Artificial Intelligence. pp. 393–400. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
25. Pierson, H.A., Gashler, M.S.: Deep learning in robotics: A review of recent research. CoRR **abs/1707.07217** (2017), http://arxiv.org/abs/1707.07217
26. Polydoros, A., Nalpantidis, L., Krüger, V.: Real-time deep learning of robotic manipulator inverse dynamics (09 2015). https://doi.org/10.1109/IROS.2015.7353857
27. Schaal, S.: Is imitation learning the route to humanoid robots? Trends in cognitive sciences **3**(6), 233–242 (1999)
28. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
29. Watter, M., Springenberg, J.T., Boedecker, J., Riedmiller, M.A.: Embed to control: A locally linear latent dynamics model for control from raw images. CoRR **abs/1506.07365** (2015), http://arxiv.org/abs/1506.07365