Qualitatively Constrained Control Policy Learning

Domen Šoberl,[1] **Jure Žabkar** [2]

[1] Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, Slovenia
[2] Faculty of Computer and Information Science, University of Ljubljana, Slovenia
domen.soberl@famnit.upr.si, jure.zabkar@fri.uni-lj.si

Abstract

This paper introduces a novel approach that leverages qualitative reasoning to enhance reinforcement learning in physical domains. Traditional reinforcement learning methods often suffer from sample inefficiency and lack of explainability, especially in complex, physics-driven environments. Our proposed approach addresses these challenges by integrating qualitative induction and qualitative planning to learn a control strategy. Our method enables faster convergence of the learning process and yields an interpretable and physically plausible model of the environment. It employs a unique feedback loop mechanism that iteratively improves the qualitative model of the environment based on the observed outcomes of the executed plans, allowing continual refinement of the system's understanding and actions. Through an extensive set of experiments, we demonstrate a superior performance of our method compared to a state-of-the-art deep reinforcement learning method.

Introduction

Reinforcement Learning (RL), especially when used in Deep Learning (DL), has excelled in various environments, from discrete-action games to continuous control in robotics (Sutton and Barto 2018; Mnih et al. 2015; Silver et al. 2017; Vinyals et al. 2019; Silver et al. 2014; Lillicrap et al. 2015; Ibarz et al. 2021). However, deep architectures typically require large training data and consequently take a long time to converge. The obtained models offer minimal insight into their internal control policies. Model-based RL methods partly address these issues by combining learning with planning and using state transition models to enhance efficiency and explainability (Moerland et al. 2023; Plaat, Kosters, and Preuss 2023; Milani et al. 2022).

When training an agent within a physical domain (e.g. in robotics - either real-world or simulated), integrating some degree of physical knowledge into the training mechanism can significantly speed up the training process. In RL, domain knowledge is typically coded in the reward function that guides the agent faster towards the goal (Grzes 2017). Reward shaping is limited to the form of a real-valued function and cannot encode generalized laws of physics. Instead of dealing with physics directly, the agent tries to maximize the received rewards, handcrafted by a domain expert that considered certain laws of physics when designing it.

Qualitative representations (Forbus 2019) show a promising direction towards sample efficient learning (Šuc 2003; Žabkar et al. 2011), learning explainable models (Bratko 2008, 2011; Košmerlj, Bratko, and Žabkar 2011), and learning explainable control strategies (Šoberl and Bratko 2019). These representations can bridge the gap between numerical and symbolic machine learning, and can therefore be used for symbolic learning and planning in continuous robotic environments (Žabkar, Bratko, and Mohan 2008; Wiley, Bratko, and Sammut 2018; Šoberl and Bratko 2020).

In this paper, we propose a qualitative approach to learning continuous control policies, where instead of a rewarding mechanism, qualitative reasoning about the physical environment is used to constrain the search space and guide the system toward the goal state. A qualitative model is built and a qualitative state space determined during the first few episodes of training and then being continuously refined over new observations as the training continues. The training is guided along qualitative plans, which are devised from the observed state transitions, and so avoiding spurious solutions. We evaluate the performance of our proposed method on the swing-up problem for the inverted pendulum and compare it to the performance of the Deep Deterministic Policy Gradient (DDPG) on the same problem (Lillicrap et al. 2015).

In comparison to similar qualitative control methods, the contributions of this paper are the following:

- The learned qualitative model, the inferred qualitative state space and the execution parameters are being continuously refined, while the existing methods refine only the execution parameters.
- The structure of the qualitative state space is inferred from the observed numerical behaviors instead of from the rules of qualitative simulation, which eliminates the possibility of spurious plans and confines the computational complexity of planning to the complexity of the breadth-first search algorithm.
- Qualitative control is demonstrated on the swing-up benchmark domain and its performance is compared to the performance of the deep reinforcement learning algorithm DDPG.

Related work

First attempts at solving physical and mechanical problems qualitatively date back to the 1980s, when the concept of *Qualitative Differential Equations* (QDE) as a simplified alternative to ordinary differential equations (ODE) was introduced (Forbus 1984; De Kleer and Brown 1984). Kuipers devised a way to use QDEs to simulate dynamic systems and introduced an algorithm for *qualitative simulation* called QSIM (Kuipers 1986). Forbus formalized the concept of *action* in the context of qualitative simulation (Forbus 1989), which complied with the paradigm of classical planning, where actions are deterministic and instantaneous. Such a formulation was used by Sammut and Yik (Sammut and Yik 2010) to devise a qualitative plan for a bipedal robot walking, which was executed through trial-and-error (Sammut and Yik 2010). A similar approach was used by Wiley et al. to train a multi-tracked vehicle to climb the stairs (Wiley, Bratko, and Sammut 2018).

When first introduced, QDEs were abstracted from ODEs manually. However, when a numerical model of the domain is not known or cannot be easily devised by intuition, there is a tendency to learn it from numerical traces. This was largely made possible with the introduction of Multivariate Monotonic Qualitative Constraints (MMQC) (Wellman 1991). These types of constraints can be induced directly from numerical data and represented in the form of decision trees (Bratko and Šuc 2003), or abstracted as qualitative partial derivatives and constructed into a model with any chosen classifier (Žabkar et al. 2011). Because these qualitative models abstract away most of the numerical information and present the learned theory in the form of increasing and decreasing intervals, they better comply with the human intuition that the traditional numerical models (Bratko 2008).

Practical applications demonstrated that by autonomously interacting with the physical world, a robot can learn qualitative representations that could intuitively be interpreted as *obstacle*, *stability* and *movability* (Leban, Žabkar, and Bratko 2008; Košmerlj, Bratko, and Žabkar 2011). The robots were interacting with high-level actions and observations. Mugan and Kuipers (Mugan and Kuipers 2012) proposed learning on the motor level and developed an unsupervised learning algorithm QLAP (Qualitative Learner of Action and Perception) that uses qualitative representations as a way to discretize the input data. The learned models are represented as Dynamic Bayesian Networks (DBNs).

Using learned qualitative models on the motor level to execute robotic tasks requires a method for resolving the effects of actions on a qualitative level. This approach alone allows a reactive execution of tasks, simple enough to be solved without planning. This was demonstrated on the problems of pursuing a goal, avoiding collision and pushing a box (Šoberl, Žabkar, and Bratko 2015; Šoberl and Bratko 2017). Employing qualitative planning, explainable control strategies can be devised, which was demonstrated on the problem of balancing the inverted pendulum (Šoberl and Bratko 2019). Combining qualitative planning with reactive execution, a quadcopter was able to learn an explainable controller to navigate through space and plan its way around a simple maze (Šoberl and Bratko 2020). A domain-independent framework was proposed soon after (Šoberl 2021).

Learning qualitative control vs. Reinforcement learning

Existing qualitative approaches to learning motor-level control learn a qualitative model first, then devise a qualitative plan, which is finally refined through execution. The learned model and the qualitative plan remain unchanged for the remainder of the execution. The solution is either a quantified qualitative plan (Sammut and Yik 2010; Wiley, Bratko, and Sammut 2018) or a qualitative plan with a numerically fine-tuned execution policy (Šoberl and Bratko 2019, 2020). In both cases, learning is model-based and goal-oriented. This is fundamentally different from reinforcement learning, where the learned policy is refined continuously as new observations are collected. In the case of the model-free Q-learning method, no model of the environment is learned, while the learned policy aims to maximize the reward function, typically without any notion of a goal state.

Each of the two approaches has certain advantages over the other: Reinforcement learning can be used with a wider range of discrete and continuous environments and requires a large set of training samples, obtained over a large number of training episodes. The learned control policy is purely numerical, typically in the form of a deep neural network. The latter methods do not provide any explanation of why a certain action is taken in a certain state. The policy is numerically bound to the training environment and requires retraining in case the environmental parameters change.

The fundamental idea behind learning qualitative control is to narrow down the search space by constraining it in two ways: (i) with qualitative constraints that arise from the laws of physics, and (ii) with domain-specific qualitative constraints learned through experimentation with the environment. The first type of constraints restrict the use of qualitative control methods to continuous real-world environments — typically to robotic domains. The reasons to constrain the search space qualitatively instead of numerically are: (i) the generality of qualitative physics, and (ii) sample efficient learning of qualitative models. Qualitative physics is more general than conventional physics in the sense that it abstracts away numerical constants and works with symbolic time. It, therefore, predicts a succession of qualitative states instead of exact numerical states at precise times. This makes the devised qualitative solutions transferable to environments with the same qualitative dynamics, but different numerical parameters. The execution parameters need to be re-tuned, but qualitative models and plans remain unchanged. Moreover, qualitative abstractions comply with the way humans reason about the physical world and are therefore a feasible basis for explainability (Bratko 2008; Šoberl and Bratko 2019). The key differences between reinforcement learning and learning qualitative control are summarized in Table 1.

The presumption taken in previous research on learning qualitative control is that constraining the search space qual-

	Reinforcement learning	Learning qualitative control
Environment types	Arbitrary mechanics	Real-world physics
Trainable entity	(Deep) neural networks	Qualitative constraints
Background knowledge	Reward function (explicit)	Qualitative physics (implicit)
Reward function	Crucial for success	Used as performance metric
Goal	Maximize the reward	Reach a goal state
Sample efficientcy	Low	High
Explainable policy	No	Yes, through qualitative behaviors
Transferable policy	No	Yes, qualitative models and plans

Table 1: Key differences between reinforcement learning and learning qualitative control.

itatively reduces the training time. It is a reasonable premise, considering the fact that model-free reinforcement learning usually attempts many absurd and non-intuitive actions, before finding a working solution. However, this has — to the best of our knowledge — not yet been demonstrated and evaluated on a common benchmark domain. One of the reasons for the lack of such a comparison is the fundamentally different ways in which the two methods approach a problem and hence the lack of a common evaluation metric. In this paper, we introduce certain adaptations to the qualitative control method, that bring it closer to the paradigm of reinforcement learning. Training is executed over multiple episodes of limited duration and the received rewards are used as a performance metric, although they are not utilized by the qualitative method for training.

The proposed method

Any qualitative method of acting in a numerical environment would inevitably assume at least the following four phases: (1) data collection, (2) qualitative abstraction, (3) qualitative reasoning, and (4) numerical implementation. Qualitative abstraction lifts numerical data to a qualitative level so that qualitative reasoning can take place, while numerical implementation acts in reverse: it quantifies a qualitative solution so that it can be executed in the numerical environment. Our method denotes these four phases as:

1. *Data collection*. Sensory data is collected either by random or systematic experimentation or by motor babbling.
2. *Qualitative abstraction*. A qualitative model is induced from the collected numerical data.
3. *Qualitative planning*. A qualitative plan is found from the current state to a goal state.
4. *Plan execution*. The obtained qualitative plan is executed reactively — one action at a time in a closed control loop.

Existing approaches to qualitative control employ similar four phases, but mostly in linear succession, with the execution phase being the only one done in a closed loop. We propose expanding the sensory feedback also to the phases of qualitative induction and qualitative planning. In such a non-linear setting, these can no longer be deemed *phases*, but rather *levels* of acting. *Experimentation* is the only phase of acting done separately from the rest. It is performed during the first few episodes of training to collect the minimum required samples to induce a qualitative model. In reinforcement learning, the first few episodes are often also purely explorational. Samples are then collected for the remainder of the training and used to refine the parameters on each level.

Data collection

The objective of the initial data collection is to provide enough numerical samples to induce a qualitative model. The goal is to model how actions affect the observable numerical state. For example, modeling the behavior of a simple pendulum, we model how the force $F \in \mathcal{X}$, applied to the pendulum, affects its radial acceleration $\ddot{\theta} \in \mathcal{Y}$ at a certain position and radial velocity $\theta, \dot{\theta} \in \mathcal{C}$. When parts of the state space exhibit different operational principles than others, samples should be collected in all operating regions (as in (Šoberl, Žabkar, and Bratko 2015; Wiley, Bratko, and Sammut 2018)).

Qualitative induction

Qualitative induction is a process of generating qualitative models from numerical data (Bratko and Šuc 2003). In this paper, we consider qualitative models that capture monotonically increasing and decreasing intervals of continuous functions. For instance, function $y = x^2$ has two such intervals — it is monotonically decreasing in all $x < 0$ and monotonically increasing in all $x > 0$. Point $x = 0$ is considered a *critical point*, a border between two *operating regions*. In this paper, we use Padé (Žabkar et al. 2011), which allows us to systematically introduce into the modeling the action variables $\mathcal{X}$, dependant variables $\mathcal{Y}$, and conditional variables $\mathcal{C}$. We do this in the following way:

$$\begin{tikzcd} \begin{array}{c}\text{numerical}\\\text{samples}\end{array} \arrow[r, "\mathcal{X},\mathcal{Y}"] & \text{Padé} \arrow[r, "\mathcal{C}"] & \text{classifier} \arrow[r] & \begin{array}{c}\text{qualitative}\\\text{model}\end{array} \end{tikzcd}$$

In the case of only one operating region, the resulting qualitative model would consist of qualitative constraints of the form $\mathcal{Y} = Q^{\{+,-\}}(\mathcal{X})$, which are valid everywhere within the domain. If there is more than one operating region, the model would contain multiple sets of such constraints, each set conditioned by values from $\mathcal{C}$.

Let us recall the pendulum example from the previous section. Since we only have direct control over the output variable $F \in \mathcal{X}$, we want to model the effect of F on the radial acceleration $\ddot{\theta} \in \mathcal{Y}$. Assuming that this effect may be qualitatively different depending on $\theta, \dot{\theta} \in \mathcal{C}$, we use Padé to find qualitative dependencies of the form $\ddot{\theta} = Q^{\{+,-\}}(F)$ and a classifier (typically a decision tree learner) to learn operating regions determined by θ and $\dot{\theta}$.

Qualitative planning

Qualitative planning considers a search through the *qualitative state space* from an initial to a goal qualitative state; a *qualitative plan* is a qualitative behavior, permitted by the given qualitative model (Wiley, Bratko, and Sammut 2018; Šoberl and Bratko 2020). The definition of the qualitative state space has been adopted from Kuipers' QSIM (Kuipers 1986) and adapted for planning by extending it with the concept of action. We adopt the same definition of the qualitative state space, but base its internal structure on qualitatively abstracted observations, instead on the QSIM's theoretical framework. For instance, suppose that a robot is moving uphill toward some critical point. The robot's path can in such case be qualitatively abstracted as *the interval before the critical point* and *the interval after the critical point*. Because of the continuity of the path, QSIM would presume the possibility of transitioning between the two intervals. However, the hill may in reality be too steep for the robot to reach the critical point and transition to the next qualitative state. Our method, therefore, presumes the possibility of transitioning between two qualitative states only if the transition has already been observed.

To promote some level of exploration, maximum depth d is given as a parameter and all qualitative plans up to depth d are constructed. A randomly chosen plan is then selected for execution. If no plan is found, a random action is executed and planning is repeated from the new state, hopefully with new observations allowing for a wider set of state transitions. In contrast to reinforcement learning, where at every step a single action is chosen either randomly or by the learned policy, our method commits to plans rather than to single actions.

Plan execution

Execution of qualitative plans is the problem of implementing a continuous transition between two consecutive qualitative states $S_i \to S_{i+1}$ of the plan within a particular numerical environment. We consider reactive control: the agent selects and executes an action several times per second after observing the current numerical state with the same frequency. At each reactive step, the executor is solving two problems: (i) determining which qualitative action will produce the most desirable effect towards state S_{i+1}, and (ii) translating the qualitative action to output numerical signals.

The question of how to resolve the effects of qualitative actions through qualitative models has been addressed in (Šoberl and Bratko 2017). *Qualitative action* has been defined as *an instruction on the directions of change of control variables*. Formally, let $c_{\{1\ldots m\}} \in \mathbb{R}$ be control variables that represent output signals (e.g., signals to control the motors). Then a qualitative action A formalized as $A = [c_1 : \mathrm{dir}_1, \ldots, c_m : \mathrm{dir}_m]$, where $\mathrm{dir}_{\{1\ldots m\}} \in \{\mathrm{inc}, \mathrm{dec}, \mathrm{std}\}$. Action $A = [c_1 : \mathrm{inc}, c_2 : \mathrm{dec}, c_3 : \mathrm{std}]$ would therefore instruct the value of c_1 to be increased, the value of c_2 decreased, and c_3 kept at its current value. There is no information on the rates of change at this stage.

The process of action prioritization is mathematically formulated as follows. Let variables x_i for all $i \leq k$ and some k have a target value, and let variables x_i for all $i > k$ be numerically constrained. Numerical constraints are optional[1] and determine the fail states, e.g. numerical constraint $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$ specifies the allowed interval for variable θ. Let e_i be the respective time estimates of variables x_i. Define the function:

$$f(e_1, \ldots, e_n) = \sum_{i=1}^{k} \frac{{e_i}^2}{2} + \\ + \sum_{i=k+1}^{n} \Big((1 + e_i)^{-1} \cdot (1 - e_i)^{-1} - 1 \Big) \tag{1}$$

and denote its gradient as:

$$\nabla f = \left(\frac{\partial f}{\partial e_1}, \ldots, \frac{\partial f}{\partial e_n} \right). \tag{2}$$

Let $E = [\mathrm{dir}_{x_1}, \ldots, \mathrm{dir}_{x_n}]$ be the vector of qualitative effects of action A on variables $x_1, \ldots, x_n$ as deduced through the qualitative model, where qualitative directions $\mathrm{dir}_{x_i} \in \{\mathrm{inc}, \mathrm{std}, \mathrm{dec}\}$ are mapped to integers as $\mathrm{inc} \mapsto 1, \mathrm{std} \mapsto 0, \mathrm{dec} \mapsto -1$. The priority $Q(A)$ of action A is then computed as

$$Q(A) = -\nabla f \cdot E. \tag{3}$$

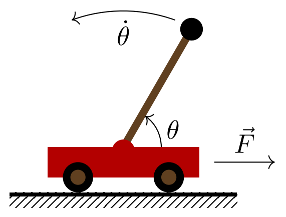
In previous work (Šoberl and Bratko 2020; Šoberl 2021), it was presumed that acceleration is constant over the entire domain. If the agent was observed to accelerate at a certain rate, it was presumed that it can decelerate at the same rate at any location. However, there are domains where this is not the case. A pendulum, for instance, will exhibit different accelerations under the same applied force when in different positions, due to the force of gravity. Therefore, in this paper, we propose predicting the maximum possible deceleration through linear regression. From the collected numerical samples, we select the points near the goal state to model the relation between the control variables and the observed accelerations. Using linear regression, we then predict the maximum deceleration rates at the goal position. The predictions get refined as new samples are collected throughout training episodes.

The experimental domain

To demonstrate our approach, we simulated a classic benchmark control problem of swinging up the inverted pendulum. A freely rotating pole is attached to a cart as shown in

[1] In this paper we do not use numerical constraints, but they can be useful in other control domains, e.g. (Šoberl, Žabkar, and Bratko 2015; Šoberl and Bratko 2019; Šoberl 2021).

Figure 1. The cart can be moved either left or right by applying a force in the direction of motion. Consequently, the pole accelerates rotationally either clockwise (CW) or counterclockwise (CCW). The pendulum is initially in the downward position ($\theta = -180°$). The goal is to lift the pendulum upward ($\theta = 0°$) and maintain it in this state. We do not impose any goals or constraints on the position of the cart. We orient the coordinate system so that the force F is positive in the right direction, and the pole's angle θ increases in the CCW direction. The length of the cart is $1\,\mathrm{m}$ and weighs $10\,\mathrm{kg}$. The pole is $1\,\mathrm{m}$ long and weighs $1\,\mathrm{kg}$. The force F is applied to the cart with the frequency of $50\,\mathrm{Hz}$ and can be between $-100\,\mathrm{N}$ and $100\,\mathrm{N}$. We limit the maximum speed of the pole to $360°/s$ in any direction. We did not simulate noise.



θ — The angle of the pole.

$\dot{\theta}$ — The angular velocity of the pole.

$\vec{F}$ — The force applied to the cart.

Figure 1: The state $(\theta, \dot{\theta})$ and the action $(\vec{F})$ in our cart-pole domain.

The observable state of the system is $(\theta, \dot{\theta})$, while $\ddot{\theta}$ can be derived from the observed $\Delta\dot{\theta}$ and Δt. The initial state is $(\theta = -180°, \dot{\theta} = 0)$ with $F = 0$, and the goal state is defined as $(\theta = 0°, \dot{\theta} = 0)$. We use the reward function

$$R(\theta, \dot{\theta}) = -\left(\theta^2 + 0.1 \cdot \dot{\theta}^2\right), \tag{4}$$

which is similar to the reward function used in the OpenAI Gym pendulum environment (Brockman et al. 2016), except for the torque component $(+0.001 \cdot u^2)$, which we here omit. In the Gym pendulum domain, force F is applied directly to the tip of the pole, so the translation of the force to the torque is straightforward. In our inverted pendulum domain, the force is transferred via the cart to the base of the pole, which makes the torque not directly observable. Moreover, the impact of the torque on reward values is small due to a low coefficient of 0.001. The two methods compared in this paper – reinforcement learning and our qualitative control method — are hence both driven by the same observable quantities θ and $\dot{\theta}$, the former through optimizing the received rewards and the latter by pursuing the goal state. Recall that our qualitative method does not employ a reward function; we use it here only to be able to compare the performance of both methods.

Reinforcement learning is usually done over multiple *episodes*. After each episode, the system is reset to its initial state and the training is repeated with the updated policy. The DDPG method updates the weights and biases of the actor and the critic neural networks (as described in (Lillicrap et al. 2015)), while our qualitative control method updates (i) the qualitative model, (ii) the set of state transitions and (iii) the linear regression model to predict accelerations. We set the length of each episode to 6 seconds, which is 300 steps with the $50\,\mathrm{Hz}$ action frequency. We identify no fail states in this domain, although, in general, a fail state would also terminate an episode.

Results

In this section, we separately present the results of qualitative induction, qualitative planning, and execution that we obtained in our simulated inverted pendulum domain. We compare the results of execution with those obtained by the DDPG algorithm that we ran in the same simulator and under the same conditions.

Qualitative induction

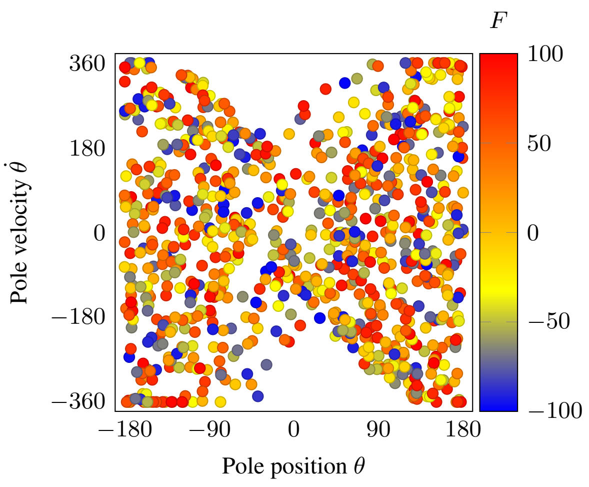
The training of the inverted pendulum started purely experimentally. Random forces F were applied to the cart with random durations between 20 and 500 ms. Numerical samples $(F, \theta, \dot{\theta})$ were captured 50 times per second, and $\ddot{\theta}$ computed as $\Delta\dot{\theta}$ under the constant time step $\Delta t = 20$ ms between two consecutive observations. What we aim to model is how the force F affects $\ddot{\theta}$ in any given state $(\theta, \dot{\theta})$. Since $\ddot{\theta}$ is the time derivative of $\dot{\theta}$ and the latter of θ, the effect of F on those higher integrals would be simulated by the qualitative planner.

After 3 episodes, 900 samples were obtained, which sufficed for inducing a qualitative model with an average numerical error below 4° for θ. Considering that the maximum speed of the pole $\dot{\theta}$ is $360\,°\,\mathrm{s}^{-1}$ and that observations are collected with the frequency of 50 Hz, up to $7.2°$ input error is possible just from temporal resolution. The collected numerical samples and the induced qualitative model are shown in Figure 2. It can be seen from the model that two operating regions have been learned: $\ddot{\theta} = Q^{+}(F)$ for $-89.94° \geq \theta < 93.53°$ and $\ddot{\theta} = Q^{-}(F)$ outside that interval. The theoretically correct boundaries are $-90.0°$ and $90.0°$ respectively. The error decreased with further sampling, dropping below $1.2°$ after 30 episodes. The learner determined that $\dot{\theta}$ plays no role in specifying the operating regions.

The obtained qualitative model can be interpreted in the following way:

If the pole is in the upright position (above the horizontal line), increasing the force on the cart will increase the acceleration of the pole, and vice-versa.

If the pole is in the downright position (below the horizontal line), increasing the force on the cart will decrease the acceleration of the pole, and vice-versa.



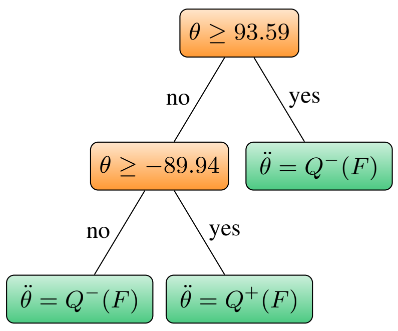


Figure 2: Qualitative induction from numerical samples. Above: Samples collected during the experimentation episodes. Below: A qualitative model induced after processing the numerical samples with Padé.

Qualitative planning

The qualitative model was passed to the qualitative planner, together with the initial state ($\theta = -180°, \dot{\theta} = 0$) and the goal state ($\theta = 0°, \dot{\theta} = 0$), the planner immediately determines the landmarks of the configuration space $(\theta, \dot{\theta})$:

$$\begin{aligned} \theta &: \{-180, -90, 0, 90, 180\}, \\ \dot{\theta} &: \{\text{min}, 0, \text{max}\}. \end{aligned} \tag{5}$$

For clarity, we here write the theoretical boundaries -90 and 90 for the two operating regions, although the actual values with the induced model are -89.94 and 93.53. The *min* and *max* landmarks are symbolic representations of the minimum and maximum values, which at this point are not yet known.

These landmarks determine 24 possible qualitative states. The possible qualitative values for θ are:

$$[-180], [-180..-90], [-90], [-90..0], [0], [0..90], [90], [90..180],$$

and for $\dot{\theta}$:

$$[\text{min}..0], [0], [0..\text{max}],$$

to which we will also respectively refer to as *neg*, *zero*, and *pos*.

Transitions between the qualitative states are deduced from numerical observations. Most of the possible transitions were abstracted from the 900 numerical samples used to induce the qualitative model. From these, the circular topology ($-180° = 180°$) of θ was also discovered and incorporated into the initial space partitioning (5). As seen from Figure 2, the 900 samples are more densely concentrated around the initial position $\theta = -180°$, while being sparse at the goal position $\theta = 0°$. For this reason, the transitions abstracted from these samples were also denser around the initial position, while most transitions around the goal position were discovered in later episodes. A typical scenario occurring during the early episodes of training would be making the pole rotate a full circle after overshooting the goal, because the possibility of stopping the pole close to the goal and reversing its direction had not yet been observed. Still, the early plans did tend to bring the pole closer to the goal more often than random exploration, hence new transitions were eventually discovered and with them the possibilities of new plans.

Figure 3 shows how a qualitative plan was found to swing up the pendulum. The shorter plan, leading from the initial state $[-180/\text{zero}]$ to goal state $[0/\text{zero}]$ failed to transition from $[-180..-90/\text{zero}]$ to $[-90/\text{zero}]$ due to the lack of momentum to overcome the force of gravity. A longer plan was then deduced which takes a different path after the point of failure. This way the concept of swinging to build up the momentum was discovered.

Figure 4 shows a plan deduced after the goal position $\theta = 0°$ is overshot. Instead of transitioning from $[0..90/\text{neg}]$ to $[0/\text{zero}]$, the pendulum would overshoot to state $[-90..0/\text{neg}]$. A plan would then be devised to reverse the direction of the pendulum back to the goal position.

Execution

We compared the performance of our qualitative execution with the performance of DDPG, which we configured as follows: the *actor* and the *critic* neural networks both contained two hidden layers of 64 and 32 nodes, both using the ReLu activation function. The actor took two inputs, θ and $\dot{\theta}$, and output via the Tanh activation function a continuous action within $[-1, 1]$, which was scaled to $F \in [-100, 100]$ before being executed. The critic took the same input as the actor, together with the actor's output, and output the Q-value via the linear activation function. Both networks used the Adam optimizer with the learning rate $\alpha = 10^{-3}$. The Q-learning discount factor was set to $\gamma = 0.99$. The training was done in batches of 32 samples with unlimited replay memory size. We tried various other DDPG configurations with different neural network architectures but achieved the best results using the described settings.

The training performance of both algorithms is shown in Figure 5. The plots were obtained by running each algorithm 100 times and averaging the episode rewards obtained in each training episode. The episode reward itself is the average of the 300 rewards received during an episode (recall that the duration of each episode is 300 steps). Both algorithms eventually converged toward the same strategy — build up the momentum by swinging and then balance

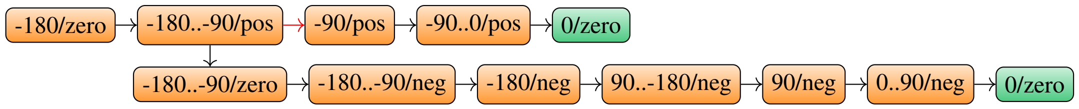


Figure 3: A qualitative plan found to swing up the pendulum. The alternative (lower) branch is deduced when the transition marked with the red arrow fails to execute.

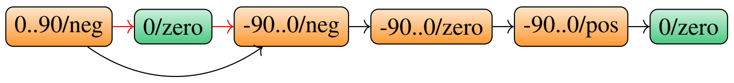


Figure 4: A qualitative plan found to correct a goal overshoot. The alternative (lower) branch is deduced when the pendulum fails to stop exactly at the goal position.

at the goal position. However, our qualitative algorithm converged significantly faster. The level of performance reached by DDPG after about 150 episodes, was achieved by our qualitative executor around episode 50 in the worst case.

Conclusion

This paper aims to bridge the gap between reinforcement learning and learning qualitative models, which have previously been researched separately and, to the best of our knowledge, never evaluated on the same control problem. The type of qualitative modeling and reasoning that we focus on in this paper is based on *monotonic qualitative constraints*, which are more sample efficient to learn, have better noise resiliency and offer a higher level of explainability than traditional numerical models.

We demonstrated our method on a single benchmark control problem and compared it to state-of-the-art deep reinforcement learning method. The approach should easily be transferrable to other control domains. We find it somewhat more difficult to implement than a typical reinforcement learning algorithm, since it contains a three-layered feedback loop, the complete Padé learner, a qualitative physics engine, and a non-trivial execution mechanism that is capable of dynamically adapting to the environment. It also cannot be used in (simulated) environments that do not comply with Newtonian physics.

References

Bratko, I. 2008. An assessment of machine learning methods for robotic discovery. *Journal of Computing and Information Technology - CIT*, 16(4): 247–254.

Bratko, I. 2011. Autonomous Discovery of Abstract Concepts by a Robot. In *Adaptive and Natural Computing Algorithms*, 1–11.

Bratko, I.; and Šuc, D. 2003. Learning qualitative models. *AI magazine*, 24(4): 107–119.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. arXiv:1606.01540.

De Kleer, J.; and Brown, J. S. 1984. A qualitative physics based on confluences. *Artificial Intelligence*, 24(1): 7–83.

Forbus, K. D. 1984. Qualitative Process Theory. *Artificial Intelligence*, 24(1–3): 85–168.

Forbus, K. D. 1989. Introducing Actions into Qualitative Simulation. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'89, 1273–1278. Morgan Kaufmann Publishers Inc.

Forbus, K. D. 2019. *Qualitative Representations: How People Reason and Learn about the Continuous World*. MIT Press, 1st edition.

Grzes, M. 2017. Reward Shaping in Episodic Reinforcement Learning. In *Adaptive Agents and Multi-Agent Systems*.

Ibarz, J.; Tan, J.; Finn, C.; Kalakrishnan, M.; Pastor, P.; and Levine, S. 2021. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5): 698–721.

Košmerlj, A.; Bratko, I.; and Žabkar, J. 2011. Embodied Concept Discovery through Qualitative Action Models. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 19: 453–475.

Kuipers, B. 1986. Qualitative simulation. *Artificial Intelligence*, 29(3): 289–338.

Leban, G.; Žabkar, J.; and Bratko, I. 2008. An Experiment in Robot Discovery with ILP. In *Proceedings of the 18th International Conference on Inductive Logic Programming*, 77–90.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N. M. O.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971.

Milani, S.; Topin, N.; Veloso, M.; and Fang, F. 2022. A Survey of Explainable Reinforcement Learning.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518: 529–533.

Moerland, T. M.; Broekens, J.; Plaat, A.; and Jonker, C. M. 2023. Model-based Reinforcement Learning: A Survey. *Foundations and Trends® in Machine Learning*, 16(1): 1–118.

Mugan, J.; and Kuipers, B. 2012. Autonomous Learning of High-Level States and Actions in Continuous Environ-

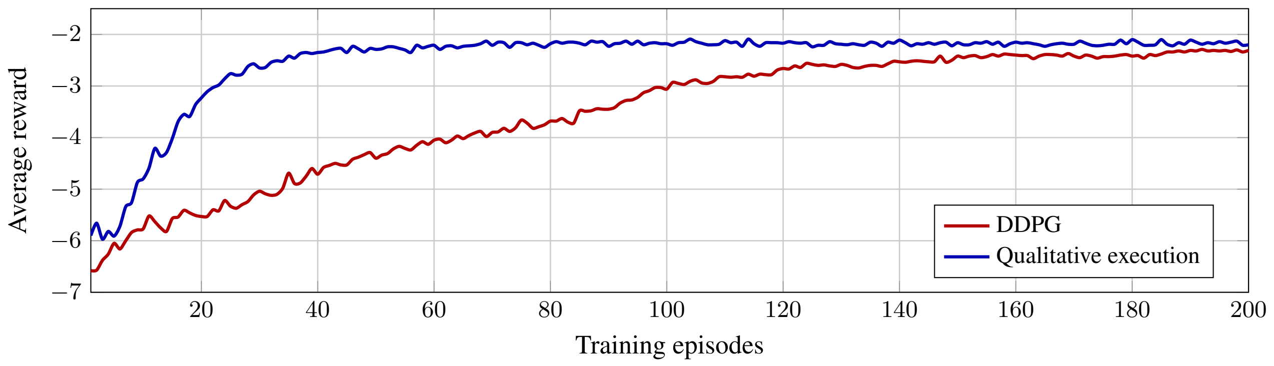


Figure 5: The performance of the two evaluated algorithms. The plots show how the reward converges over the number of training episodes. The plotted values are the averages of 100 repetitions.

ments. *IEEE Transactions on Autonomous Mental Development*, 4(1): 70–86.

Plaat, A.; Kosters, W.; and Preuss, M. 2023. High-accuracy model-based reinforcement learning, a survey. *Artificial Intelligence Review*.

Sammut, C.; and Yik, T. F. 2010. Multistrategy Learning for Robot Behaviours. In *Advances in Machine Learning I: Dedicated to the Memory of Professor Ryszard S.Michalski*, 457–476. Berlin, Heidelberg: Springer Berlin Heidelberg.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, 387—395. JMLR.org.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T.; Hui, F.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017. Mastering the game of Go without human knowledge. *Nature*, 550(7676): 354–359.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J. P.; Jaderberg, M.; Vezhnevets, A. S.; Leblond, R.; Pohlen, T.; Dalibard, V.; Budden, D.; Sulsky, Y.; Molloy, J.; Paine, T. L.; Gulcehre, C.; Wang, Z.; Pfaff, T.; Wu, Y.; Ring, R.; Yogatama, D.; Wünsch, D.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, T. P.; Kavukcuoglu, K.; Hassabis, D.; Apps, C.; and Silver, D. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 1–5.

Wellman, M. P. 1991. Qualitative Simulation with Multivariate Constraints. In *Second International Conference on Principles of Knowledge Representation and Reasoning*, 547–557.

Wiley, T.; Bratko, I.; and Sammut, C. 2018. A Machine Learning System for Controlling a Rescue Robot. In *RoboCup 2017: Robot World Cup XXI*, 108–119.

Šoberl, D. 2021. *Automated planning with induced qualitative models in dynamic robotic domains*. Ph.D. thesis, University of Ljubljana.

Šoberl, D.; and Bratko, I. 2017. Reactive Motion Planning with Qualitative Constraints. In *Advances in Artificial Intelligence: From Theory to Practice*, 41–50. Springer International Publishing.

Šoberl, D.; and Bratko, I. 2019. Learning Explainable Control Strategies Demonstrated on the Pole-and-Cart System. In *Advances and Trends in Artificial Intelligence. From Theory to Practice*, 483–494. Springer International Publishing.

Šoberl, D.; and Bratko, I. 2020. Learning to Control a Quadcopter Qualitatively. *Journal of Intelligent & Robotic Systems*.

Šoberl, D.; Žabkar, J.; and Bratko, I. 2015. Qualitative Planning of Object Pushing by a Robot. In *Foundations of Intelligent Systems*, 410–419. Springer International Publishing.

Šuc, D. 2003. *Machine Reconstruction of Human Control Strategies*. Frontiers in Artificial Intelligence and Applications. IOS Press, Inc.

Žabkar, J.; Bratko, I.; and Mohan, A. C. 2008. Learning qualitative models by an autonomous robot. In *22th International Workshop on Qualitative Reasoning*, 150–157.

Žabkar, J.; Možina, M.; Bratko, I.; and Demšar, J. 2011. Learning qualitative models from numerical data. *Artificial Intelligence*, 175(9–10): 1604–1619.