ECAI 2020
G.D. Giacomo et al. (Eds.)

© 2020 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA200266

Fleet Control Using Coregionalized Gaussian Process Policy Iteration

Timothy Verstraeten[1] and **Pieter J.K. Libin**[1] and **Ann Nowé**[1]

Abstract. In many settings, as for example wind farms, multiple machines are instantiated to perform the same task, which is called a fleet. The recent advances with respect to the Internet of Things allow control devices and/or machines to connect through cloud-based architectures in order to share information about their status and environment. Such an infrastructure allows seamless data sharing between fleet members, which could greatly improve the sample-efficiency of reinforcement learning techniques. However in practice, these machines, while almost identical in design, have small discrepancies due to production errors or degradation, preventing control algorithms to simply aggregate and employ all fleet data. We propose a novel reinforcement learning method that learns to transfer knowledge between similar fleet members and creates member-specific dynamical models for control. Our algorithm uses Gaussian processes to establish cross-member covariances. This is significantly different from standard transfer learning methods, as the focus is not on sharing information over tasks, but rather over system specifications. We demonstrate our approach on two benchmarks and a realistic wind farm setting. Our method significantly outperforms two baseline approaches, namely individual learning and joint learning where all samples are aggregated, in terms of the median and variance of the results.

1 INTRODUCTION

Reinforcement Learning (RL) is a framework for optimizing control policies through trial-and-error [35]. A learning agent operates within an environment and optimizes its actions based on gathered experiences. As state-of-the-art techniques typically require a large amount of experiences, a key challenge in RL is to increase sample-efficiency, such that RL techniques become viable in real-life applications [40].

In this work, our objective is to improve sample efficiency in the context of a *fleet*[2], i.e., a set of machines instantiated to perform the same task and managed as a single system. Fleets are prominent in many industrial applications, such as wind farms [23, 37] and autonomous vehicles [11], because fleets are cheaper to maintain and operate. To this end, we present a method that aggregates the experiences of the distinct fleet members, in contrast to the experiences of a single machine. The time is right for such a method, as the recent advances in the *Internet of Things* allow fleet members to share data from modern wireless sensors using a cloud-based architecture, rapidly providing a complete overview of the problem [16].

As fleet members carry out the same task, they typically share the same design. In reality, fleet members differ slightly in terms of dynamics, for example due to production errors or degradation [34]. Thus, naively aggregating data over all members can be detrimental to the learning process. Therefore, information should only be shared between fleet members that are sufficiently similar.

We propose a new RL method for fleet control where knowledge transfer over dynamics models of similar devices is possible without compromising the specificity of an individual's model. More specifically, we create a Bayesian RL method that uses a Gaussian process (GP) to model the dynamics for a single member and aims to estimate correlations with other members using a novel sparse coregionalization method.

GPs are Bayesian models known to successfully capture complex non-linear surfaces using only a limited amount of data. They have previously been used in the context of RL [30, 9, 6] and are popular when high sample-efficiency is necessary. Coregionalization was originally introduced in geostatistics to generate valid covariance matrices for modeling multivariate data sets [14]. It has later been used in the context of multi-task learning to describe correlations between a set of tasks [4].

In this work, we develop a fleet-wide policy iteration method based on coregionalized GPs. We start by positioning our research within the literature (see Section 2) and provide background on GPs (Section 3) and RL (Section 4). Then, we describe the Bayesian fleet transition model and explain how a fleet member can access its specific predictive statistics (Section 5). Next, we analyze the sample efficiency and performance of our method on fleet-variants of well-known benchmark settings, namely, the continuous mountain car and the underactuated cart-pole (Section 6). Additionally, we demonstrate the practical benefits of our method on a state-of-the-art wind farm simulator (Section 6.3). Finally, we discuss the results and identify future work (Section 7).

2 RELATED WORK

The type of learning we consider in our work is related to multi-task (or inductive transfer) reinforcement learning [28], where a set of control tasks is jointly learned, leveraging potential similarities between them. In contrast to our setting, most work on multi-task RL considers different task parameters, while the system specifications remain the same [17, 22, 36, 38, 20, 21, 7]. Specifically, [22] and [38] construct a Bayesian (hierarchical) structure of tasks, where the task parameters are assumed to be drawn from a set of priors shared among similar tasks. Recent work has focussed on MDPs for

[1] Artificial Intelligence Lab Brussels, Vrije Universiteit Brussel, Belgium, email: {tiverstr, plibin, anowe}@vub.be
[2] Although the word "fleet" usually refers to a group of ships or cars, it recently has been adopted by other fields in the context of any type of machinery.

which the system specifications are different, but the reward function remains the same [8, 19, 32]. Typically, a latent embedding of the system specifications is learned in order to share information among various machines.

Our work is different as it concerns fleet settings in which we assume that the system specifications are nearly identical except for degraded parts or small design discrepancies. This means that a more targeted approach is possible. Rather than having a single latent embedding from which all members originate, a more directed peer-to-peer transfer method can be obtained through correlations. Such direct transfer is sample-efficient, as estimating the fleet-wide correlations for a given target is limited to learning a set of parameters linear in the size of the fleet.

Fleet settings are inherently multi-agent systems. While multi-agent reinforcement learning deals with control and coordination and control in multi-agent systems [5], it focuses on coordination problems, rather than information exchange between agents.

3 GAUSSIAN PROCESS

Gaussian processes (GPs) [31] are an extension of multivariate normal distributions. Similar to the latter, a GP describes a set of normally distributed random variables that are potentially correlated, i.e., knowledge about one variable gives information about another. However, the difference with multivariate normal distributions is that a GP is defined over arbitrary sets of annotated random variables. In a regression context, these random variables are the outputs of an unknown function, and their annotations are the inputs to that function.

Formally, assuming a zero-mean GP prior, i.e.,

$$f(\mathbf{x}) \sim \mathcal{GP}\left(0, k(\mathbf{x}, \mathbf{x}')\right), \tag{1}$$

and any arbitrary set of inputs X, we can model the associated latent function values $\mathbf{f}$ as

$$\mathbf{f} \mid X \sim \mathcal{N}(\mathbf{0}, K) \tag{2}$$

where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is the covariance between variables f_i and f_j. When regressing over a training set $(X_{\text{tr}}, \mathbf{y}_{\text{tr}})$, we can compute the posterior statistics of $(\mathbf{f} \mid X, X_{\text{tr}}, \mathbf{y}_{\text{tr}})$ to obtain the predictive outputs $\mathbf{f}$ for inputs X. For the zero-mean GP described in Equation 2, we have:

$$\begin{aligned} \mathbb{E}\left[\mathbf{f} \mid X, X_{\text{tr}}, \mathbf{y}_{\text{tr}}\right] &= K_{X,X_{\text{tr}}} C^{-1}_{X_{\text{tr}},X_{\text{tr}}} \mathbf{y}_{\text{tr}} \\ \mathbb{V}\left[\mathbf{f} \mid X, X_{\text{tr}}, \mathbf{y}_{\text{tr}}\right] &= K_{X,X} - K_{X,X_{\text{tr}}} C^{-1}_{X_{\text{tr}},X_{\text{tr}}} K_{X_{\text{tr}},X} \\ C_{X_{\text{tr}},X_{\text{tr}}} &= K_{X_{\text{tr}},X_{\text{tr}}} + \sigma^2 I, \end{aligned} \tag{3}$$

where $K_{X,X_{\text{tr}}}$ is a matrix containing the pair-wise covariances between sets X and X_{tr} according to the covariance kernel and σ^2 is observational noise.

The choice of covariance kernel $k(\cdot,\cdot)$ is important, as it defines the various characteristics about how the model should generalize from the training set. We use the squared exponential (SE) kernel, defined as:

$$k^{SE}_{\theta^{SE}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{2l_d^2}\right) \tag{4}$$

where θ^{SE} contains the hyperparameters l_d, which denotes the length scale along dimension d and characterizes the smoothness of the unknown function. This kernel has several properties, including continuity, differentiability and stationarity, rendering it a popular choice for general modeling purposes.

4 REINFORCEMENT LEARNING

Consider the Markov decision process (MDP) $\mathcal{M} = (S, A, \tau, \gamma, R)$ [29]. S and A are state and action spaces, respectively. The transition function $\tau(\mathbf{s}, \mathbf{a})$ returns the state $\mathbf{s}'$ when executing action $\mathbf{a}$ in state $\mathbf{s}$. The reward function $R : S, A, S \to \mathbb{R}$ returns the immediate reward. The discount factor $\gamma \in [0, 1)$ determines the importance of future rewards. Additionally, consider a policy $\pi : S \to A$, which defines how an agent behaves given a particular state.

We specify the reward function as a square-exponential[3] centered around a goal state with width σ_R, i.e.,

$$R(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \frac{1}{\sqrt{2\pi\sigma_R^2}} \exp\left(-\frac{||\mathbf{s}' - \mathbf{s}_{\text{goal}}||_2^2}{2\sigma_R^2}\right). \tag{5}$$

The transition function is unknown. Thus, we define the outputs of the transition function as samples from GPs, i.e.,

$$\tau_e(\mathbf{s}, \mathbf{a}) \sim \mathcal{GP}\left(0, k^{SE}_{\theta^{SE}_e}\right), \tag{6}$$

for each output feature e.

The expected long-term reward when following a policy π is defined by a value function V^π. This function can be written recursively as the sum of the expected immediate reward and future reward, i.e.,

$$V^\pi(\mathbf{s}) = \mathbb{E}\left[R(\mathbf{s}, \pi(\mathbf{s}), \mathbf{s}') + \gamma V^\pi(\mathbf{s}') \mid \mathbf{s}' = \tau(\mathbf{s}, \pi(\mathbf{s}))\right]. \tag{7}$$

This is the sum of all possible long-term rewards weighted by their probability of occurrence when executing a policy π. The goal of an agent is then to find the optimal policy $\pi^* : S \to A$ which maximizes this expression.

The expectation in Equation 7 typically has no closed-form expression for arbitrary continuous reward and transition functions. In order to approximate the value function, we use the Gaussian Process Reinforcement Learning (GPRL) method [30], which is a policy iteration method that iteratively evaluates a policy π on a discrete set of states and improves it until convergence.

During the policy evaluation step, GPRL computes the values of a finite, but dense, vector of support points $\mathbf{s}_{\text{supp}} = \langle \mathbf{s}^{(i)} \rangle_{i=1}^{N}$. We use Latin hypercube sampling [24] to generate this vector, such that the state space is sufficiently covered. The values of these points can be computed analytically when the transition model and value function are described by a GP, and the reward function is bell-shaped. Formally, given a policy π, a reward function centered around $\mathbf{s}_{\text{goal}}$ with width σ_R^2, and an initial GP over the value function, the support values $\mathbf{v}_{\text{supp}}$ have the recursive form:

$$\begin{aligned} \mathbf{v}_{\text{supp}} &= \mathbf{r} + \gamma P \mathbf{v}_{\text{supp}} \\ \mathrm{r}_i &= \frac{1}{\sqrt{|2\pi C^{(i)}|}} \exp\left(-\frac{1}{2}(\mathbf{s}_{\text{goal}} - \boldsymbol{\mu}^{(i)})^T C^{(i)^{-1}} (\mathbf{s}_{\text{goal}} - \boldsymbol{\mu}^{(i)})\right) \\ C^{(i)} &= \Sigma^{(i)} + \sigma_R^2 I, \end{aligned} \tag{8}$$

with the statistics of the transition model,

$$\begin{aligned} \boldsymbol{\mu}^{(i)} &= \mathbb{E}\left[\mathbf{s}' \,\middle|\, \mathbf{s}' = \tau\left(\mathbf{s}^{(i)}, \pi\left(\mathbf{s}^{(i)}\right)\right)\right] \\ \Sigma^{(i)} &= \mathrm{Var}\left[\mathbf{s}' \,\middle|\, \mathbf{s}' = \tau\left(\mathbf{s}^{(i)}, \pi\left(\mathbf{s}^{(i)}\right)\right)\right], \end{aligned} \tag{9}$$

[3] The reward function can be learned using a GP without jeopardizing the analytical benefits of our method. However, as our work focuses on learning over multiple transition models, we assume a known reward function centered around a prespecified goal state.

and P a matrix that depends on the transition model and the value function. The equation for the support values can be rewritten as a closed-form expression:

$$\mathbf{v}_{\text{supp}} = (I - \gamma P)^{-1}\mathbf{r}. \tag{10}$$

We refer the reader to the work of Rasmussen and Kuss [30] for more information about the exact form of the matrix P.

During the policy improvement step, a new GP is fitted over the value function $V(\cdot)$ using the support values to generalize over the state space. This function is used to optimize π:

$$\pi(\mathbf{s}) \leftarrow \text{argmax}_{\mathbf{a}} \mathbb{E}\left[R(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma V^{\pi}(\mathbf{s}') \mid \mathbf{s}' = \tau(\mathbf{s},\mathbf{a})\right]. \tag{11}$$

An expression similar to the one presented in Equation 8 can be obtained for arbitrary actions using the vector of support states. This expression can be maximized using standard stochastic optimizers for continuous action spaces or enumeration for discrete action spaces.

Throughout this manuscript, we only deal with deterministic transition models, and we thus set the observational noise σ^2 of the GP to 10^{-8} to ensure numerical stability. Other hyperparameters are optimized using evidence maximization on the training set [31].

5 COREGIONALIZATION OVER MULTIPLE TRANSITION MODELS

To transfer knowledge between fleet members, we leverage the statistical properties of GPs. Specifically, we assert that fleet members should only share information when they are correlated. This means that, for a given state-action pair, there exists a linear transformation between the outputs of the members' transition models. Intuitively, the GP's covariance kernel allows to generalize in the regression model by correlating unobserved outputs to observed ones. Coregionalization extends this concept to the outputs of different GPs, suggesting that information from one process can be generalized to another. The main contribution in this work is the introduction of coregionalization to capture similarities between multiple transition models in order to decide whether and how knowledge should be transferred.

Formally, we define a fleet MDP as:

$$\begin{aligned} \mathcal{M}_F &= (S, A, T^F, \gamma, R), \text{with} \\ T^F &= \{\tau_m\}_{m=1}^{M}. \end{aligned} \tag{12}$$

Compared to the definition in Section 4, all properties are the same, except that T^F is now a set of M transition models, one for each fleet member m.

Consider a single member from the fleet, which we refer to as the *target* and the rest of the fleet as the *sources*. We denote the target's index as t, and the index of a source as s. In order to achieve knowledge transfer from the sources to the target, we must define a medium through which sources can communicate. Specifically, for inputs $\mathbf{x} = [\mathbf{s}, \mathbf{a}]$, we consider this model for the transition functions:

$$\begin{aligned} &\tau_t(\mathbf{x}) = \sum_{s \neq t} w_{t,s} g_s(\mathbf{x}) + \alpha_t l_t(\mathbf{x}) \\ &\forall s \neq t: \\ &\quad \tau_s(\mathbf{x}) = w_{s,s} g_s(\mathbf{x}) + \alpha_s l_s(\mathbf{x}). \end{aligned} \tag{13}$$

The transition function of the target is a linear combination of $M-1$ global functions g_s shared with every source s, and member-specific local functions (i.e., l_t for the target and l_s for the sources) to model the member's specific behavior. This ensures that all sources can exchange information with the target without compromising the specifics of a member's dynamics. The parameters $w_{t,s}$, $w_{s,s}$, α_t and α_s weigh the contribution of the different components in the transition functions. For example, when source s has relevant information for the target t, the parameter $w_{t,s}$ should be high in order to transfer knowledge through function g_s. In contrast, when the sources have no relevant information for the target, the parameter $w_{t,s}$ should be zero in order to model independence between the source and the target.

We define the unknown components g_s, l_s and l_t as independent samples from a zero-mean GP with covariance kernel $k^{SE}_{\theta_{SE}}$ (see Equation 6). This entails that each of the transition functions is a linear combination of GP-distributed random variables, and is thus also a GP-distributed random variable [31]. The mean functions of the transition models will be zero, due to the linearity of expectation property and the fact that all components have a mean of zero. Moreover, as the components are independently sampled, covariance can only exist within a single component, which is defined through the kernel $k^{SE}_{\theta_{SE}}$. The resulting cross-covariance equations for the transition functions are as follows:

$$\begin{aligned} \text{Cov}\left[\tau_t(\mathbf{x}), \tau_t(\mathbf{x}')\right] &= \left(\sum_{s \neq t} w_{t,s}^2 + \alpha_t^2\right) k^{SE}_{\theta_{SE}}(\mathbf{x}, \mathbf{x}') \\ \text{Cov}\left[\tau_s(\mathbf{x}), \tau_s(\mathbf{x}')\right] &= \left(w_{s,s}^2 + \alpha_s^2\right) k^{SE}_{\theta_{SE}}(\mathbf{x}, \mathbf{x}') \\ \text{Cov}\left[\tau_t(\mathbf{x}), \tau_s(\mathbf{x}')\right] &= \left(w_{t,s} w_{s,s}\right) k^{SE}_{\theta_{SE}}(\mathbf{x}, \mathbf{x}') \\ \text{Cov}\left[\tau_s(\mathbf{x}), \tau_{s'}(\mathbf{x}')\right] &= 0, \end{aligned} \tag{14}$$

where $s, s' \neq t$ and $s \neq s'$.

From these statistics, we can reformulate the target's transition function as a sample from a GP with the following fleet-wide kernel:

$$\begin{aligned} k^F_{\theta^{(t)}}\left([\mathbf{x}, m], [\mathbf{x}', m']\right) &= k^{SE}_{\theta_{SE}}(\mathbf{x}, \mathbf{x}') G_{m,m'} \\ G &= W + \left(\boldsymbol{\alpha}^2\right)^T I \\ W &= \sum_{s \neq t} \mathbf{w}_s \mathbf{w}_s^T \end{aligned} \tag{15}$$

where $\mathbf{w}_s$ only has non-zero elements at indices t and s, and m, m' are the indices of two fleet members. The matrix W encodes relationships between the target and the sources, while $\boldsymbol{\alpha}$ contains independent terms for each member.

The decomposition of G yields a valid covariance matrix, as it is symmetric positive semidefinite, i.e.,

$$\forall \mathbf{z} \neq \mathbf{0} \;:\; \mathbf{z}^T G \mathbf{z} = \sum_{s \neq t} ||\mathbf{z}^T \mathbf{w}_s||_2^2 + ||\mathbf{z}^T \boldsymbol{\alpha}||_2^2 \geq 0. \tag{16}$$

The set $\theta^{(t)}$ contains now both the hyperparameters θ_{SE} of the SE kernel and of the matrix G, i.e., $\mathbf{w}$ and $\boldsymbol{\alpha}$. These parameters can be optimized using the training set $(X^F_{\text{tr}}, \mathbf{y}^F_{\text{tr}})$ of the entire fleet, annotated with the indices of its members.

Note that the matrix G contains $3M - 2$ parameters per target (i.e., M in $\boldsymbol{\alpha}$ and $2(M-1)$ in W). Therefore, the computational complexity is linear per target, and the method can be executed in a distributed manner over the fleet. Moreover, because of the sparsity of the defined covariances (see Equation 14), it is possible to significantly reduce the computational complexity of the matrix inversion in Equation 3. Specifically, the computational complexity of

the inversion operation can be reduced from $O\left(\left(\sum_{m=1}^{M} N_m\right)^3\right)$ to $O\left(\sum_{m=1}^{M} N_m^3\right)$, where N_m is the number of samples of member m. The derivation of this complexity can be found in the Supplementary Information[4]. Both properties render the method scalable to larger fleets.

Using the new fleet covariance kernel, we can describe a single GP jointly over the outputs of all fleet members (Equation 2) and compute the target's posterior statistics (Equation 3) for regression. Note that even though the new model uses the whole fleet data set, the target can predict using its own transition function by computing the posterior statistics using index t, i.e.,

$$\boldsymbol{\tau}_t \mid X^{(t)}, X_{\text{tr}}^F, \mathbf{y}_{\text{tr}}^F. \tag{17}$$

We can define such a model for each member in the fleet independently by setting that member as the target, and thus construct the set T_θ^F by defining the transition model of each member m as a fleet-wide GP:

$$\tau_m(\mathbf{x}) \sim \mathcal{GP}\left(0, k_{\theta^{(m)}}^F\left([\mathbf{x}, m], [\mathbf{x}', m']\right)\right). \tag{18}$$

GPRL can be used for policy iteration to learn the optimal value function and policy. A high-level description of the complete fleet-wide policy iteration method for a given target is provided in Algorithm 1.

Algorithm 1: Fleet-Wide Policy Iteration

Input: Reward function R, set of support points $\mathbf{s}_{\text{supp}}$, fleet-wide dynamics training data $(X_{\text{tr}}^F, \mathbf{y}_{\text{tr}}^F)$, target index t

Output: Learned policy $\pi(\mathbf{s})$

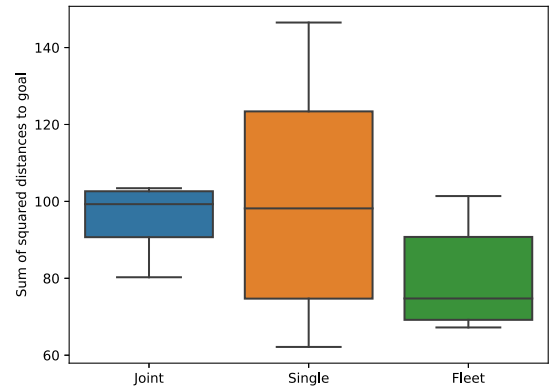
Initialize:
  $\pi(\mathbf{s}) \leftarrow$ random policy;
  $\mathbf{v}_{\text{supp}} \leftarrow$ Apply reward function R on $\mathbf{s}_{\text{supp}}$;
  Define $V \sim \mathcal{GP}\left(0, k_{\theta_V}^{SE}\right)$;
  Fit V using $(S_{\text{supp}}, \mathbf{v}_{\text{supp}})$;
Train fleet-wide transition model: (Section 4)
  $\theta^{(t)} \leftarrow \underset{\theta_{SE}, G}{\operatorname{argmax}}\, p(\mathbf{y}_{\text{tr}}^F \mid X_{\text{tr}}^F, \theta_{SE}, G)$;
  Define $\tau \sim \mathcal{GP}(0, k_{\theta^{(t)}}^F)$;
  Fit τ using $(X_{\text{tr}}^F, \mathbf{y}_{\text{tr}}^F)$;
Policy iteration (GPRL): (Section 2)
  while $\mathbf{v}_{\text{supp}}$ not converged **do**
    $\mathbf{v}_{\text{supp}} \leftarrow$ Policy evaluation using S_{supp}, R, τ, V and π;
    Fit V using $(S_{\text{supp}}, \mathbf{v}_{\text{supp}})$;
    $\pi \leftarrow$ Policy improvement using S_{supp}, R, τ and V;
  end

6 EXPERIMENTS

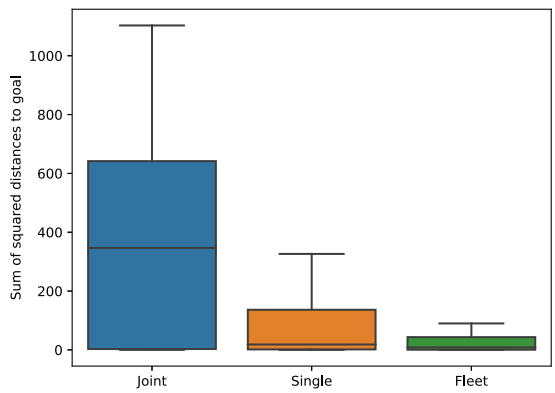
First, we experimentally analyze our method on the well-known mountain car [26] and cart-pole [2] benchmark problems. Next, we apply our method on a state-of-the-art wind farm simulator [10] to demonstrate our method's real-world benefits on larger fleets.[5]

[4] Supplementary Information: https://timo-verstraeten.github.io/fwpi-experiments/supplement.pdf

[5] The source code for the methods and experiments is publicly available (see Section 9).



(a) Mountain car



(b) Cart-Pole

Figure 1: Boxplot of the total sum of squared distances to the goal state for the mountain car (a) and cart-pole (b) benchmarks during 200 time steps. The experiment is repeated 50 times for each benchmark.

In the first two benchmarks, we consider a fleet of 3 members. The fleet consists of a target, a similar source member A and a significantly different source member B. We sample the environments of sources A and B sufficiently, such that the dynamics are well-represented by the transition model of the respective fleet member. However, we provide the target with only a limited amount of data sampled from its own environment. This means that the target cannot sufficiently estimate its transition model based on these samples, making it challenging to find the optimal policy. Therefore, transferring knowledge from member A will assist the target in finding the optimal policy. However, the dynamics in which member B operates are different from the target's dynamics, and sharing samples with it would misinform the target's transition model. Therefore, the objective of the target is to estimate a sufficiently accurate transition model by estimating the correlations with all sources and use the sources' knowledge proportional to the estimated correlation. In the wind farm control task, we consider a fleet of 8 members. Again, we have a single target and sample the environments of the other fleet members.

Once the transition model is learned, we compute[6] the optimal value function and policy using the GPRL method presented in Section 4. We consider an off-line batch RL setting and provide the learner with a random batch of transition samples. The discount factor γ is set to 0.99 and the observational noise of the value GP is set to 0.1.

In all experiments, we compare our method against two baselines, i.e., learning with a single target type and learning with a joint target type. The single target only uses its own samples to learn a transition model, while the joint target considers all fleet data jointly, assuming it is fully correlated with the sources. Specifically, we construct transition models that use the SE kernel described in Equation 4, fitted only on the target's own samples for the single target type, or using all fleet samples for the joint target type. For the fleet target type, we use our method to fit a transition model, using the fleet kernel described in Equation 15, based on all fleet samples.

6.1 Mountain Car

To illustrate our method, we set up the continuous mountain car domain [25]. The car is positioned in a valley and its objective is to reach the top of the right-most hill. However, the slope is too steep for the car to simply accelerate to the top. Thus, it has to first drive up the opposite side of the valley and then accelerate from there to reach the top.

In this problem, a state $\mathbf{s}$ consists of the position of the car (in $[-1.2, 0.6]$) and the velocity of the car (in $[-0.07, 0.07]$), while an action is a force applied to either side of the car (in $[-1, 1]$ times a power parameter). The start and goal state are, respectively, given by $\mathbf{s}_{\text{start}} = [-0.5, 0]$ and $\mathbf{s}_{\text{goal}} = [0.45, 0]$, i.e., the bottom and top of the hill. The standard deviation of the reward function σ_R is set to 0.05. We use 200 support points.

We consider a fleet of three mountain cars: a target with a power of $1.5 \cdot 10^{-3}$ units, source A with power 10^{-3} and source B with power 10^{-4}. For each source, we provide a batch of 100 transitions sampled uniformly random from its environment. We do the same for the target, but only sample 20 times, resulting in a total of 220 samples. We run the experiment 50 times for the three target types: single, joint and fleet.

We measure the performance of the methods by reporting the total sum of squared distances to the goal state during 200 time steps.[7] The results are shown in Figure 1a. We observe that the joint target (i.e., learning from the full data set without the fleet kernel) rarely reaches the goal. This is because the target uses the data of source B, which has a low power parameter and is incapable of reaching the goal. Therefore, the target does not expect to reach the goal and is often remaining at the bottom of the hill during runs. The single target (i.e., learning from own experiences) can sometimes achieve good results, but is unable to accurately represent its dynamics, due to the limited amount of data it can learn from. Because of the uncertainty in the transition model, the car is often incapable of finding a suitable policy. The fleet target consistently achieves good results, as the target is able to figure out which source is most useful to share data with through the fleet kernel.

In Figure 2, we plot the resulting GP of the value function during the best performant run for each of the target types. We can see that that the region with highest value matches the goal state for the fleet target, while the single and joint targets misidentify this region. The average standard deviation over the surface of the value GPs is 115 for the fleet target, 590 for the joint target and 3906 for the single target. This indicates that insufficient data is available to the single target to accurately represent the value function. In contrast, the fleet target has a low standard deviation, which means that it is sufficiently confident about its value function.

Next, we plot the correlation matrices learned by the fleet target, averaged over all runs. Given the optimized cross-covariance matrix G from Equation 15, we can compute the correlation matrix:

$$\text{corr}(G) = (\text{diag}(G))^{-0.5} G (\text{diag}(G))^{-0.5}. \tag{19}$$

The element-wise means of these matrices over all runs are given in Figure 3. The fleet target successfully identifies source A to be similar, while assigning a notably lower correlation value to source B.

6.2 Cart-Pole

In the cart-pole domain, the goal is to keep a pole balanced on top of a controllable cart. Cart-pole is an underactuated system, as the system has more degrees of freedom than actuators. Balancing the pole is challenging, as its equilibrium is highly unstable.

In this problem, a state $\mathbf{s}$ consists of the position of the cart (in $[-4.8, 4.8]$), the angular position of the pole (in $[-0.42, 0.42]$), the velocity of the cart (in $[-2, 2]$) and the angular velocity of the pole (in $[-2, 2]$). The start and goal state are the same, namely, at the equilibrium, i.e., $\mathbf{s}_{\text{start}} = \mathbf{s}_{\text{goal}} = [0, 0]$. The standard deviation of the reward function σ_R is set to 0.2. We set the number of support points to 300.

We consider a fleet of three carts: a target with a pole mass of 0.1 units, source A with mass 0.2 and source B with mass 0.5. For each source, we provide a batch of 50 transitions sampled uniformly random from its environment. We do the same for the target, but only sample 5 times, resulting in a total of 105 samples. We run the experiment 50 times for the three target types: single, joint and fleet.

We measure the performance of the methods by reporting the total sum of squared distances from the equilibrium during 200 time steps.[7] The results are shown in Figure 1b. Due to the instability of the equilibrium, it is necessary to accurately represent the transition model. The joint target fails to achieve this, as it aggregates samples over different dynamics. The single target achieves better results, but is often uncertain about its transition model, leading to suboptimal behavior. The fleet target consistently manages to keep the pole around its equilibrium.

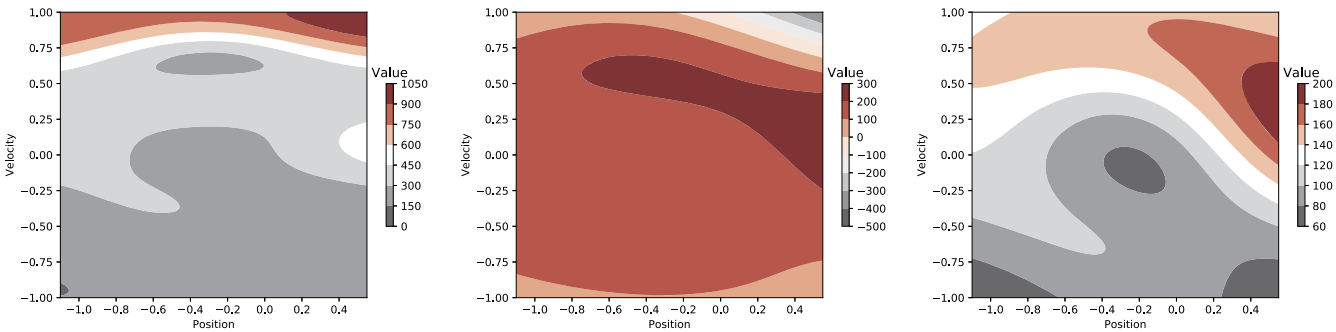
6.3 Wind Turbines

To demonstrate the need for our method in practical applications, we introduce a new setting in the context of wind energy. Current wind turbine controllers position the rotors toward the measured incoming wind vector to ensure high productivity [3]. However, as wind passes through upstream turbines, the wind speed is reduced and the energy extracted by downstream turbines is significantly lower, which is referred to as the wake effect. When considering wind farms (i.e., groups of wind turbines), it is essential to take this effect in consideration [12].

In recent work, steering wake through rotor misalignment is investigated [10, 1]. For example, in a setting with two wind turbines, the upstream turbine slightly misaligns its rotor to deflect the wake away from the downstream turbine. Therefore, although the upstream

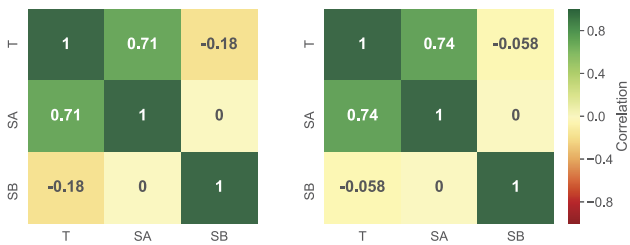
[6] We use the GPy library for all GP functionality [15].

[7] The success rates of solving the task for each of the target types are reported in the Supplementary Information.



(a) Single (b) Joint (c) Fleet

Figure 2: Mountain car – Contour plots of the learned value functions (GPs) during the best runs of each target type. Each of the learner types are depicted; single (a), joint (b) and fleet (c) learning.



(a) Position (b) Velocity

Figure 3: Mountain car – Optimized correlation matrix between the fleet members, i.e., target (T), source A (SA) and source B (SB) for both state dimensions.

turbine itself produces less energy, the group's total productivity is increased.

Because of the complexity of the wake effect and incomplete knowledge about a turbine's condition, it is necessary to gather data in the field about potential control policies, rendering it a reinforcement learning problem. As learning policies from scratch could result in potential revenue loss, fleet-wide policy iteration can improve the learning speed. Moreover, our batch RL setting makes sense, as wind farm service providers first need to thoroughly assess the performance of the acquired policy before implementing it [3].

We demonstrate our method on a fleet of two-turbine rows in a wind farm that consists of 8 rows and show how information exchange between transition models can improve the learning speed. We use the state-of-the-art open source WISDEM FLORIS simulator to model the wind farm dynamics and wake effect [27], and use the 5-megawatt (MW) reference turbine description from the National Renewable Energy Laboratory to model the individual wind turbines [18].

In this environment, the state consists of the orientations of both turbines (values in $[-45, 45]$ degrees with respect to the wind vector) and the associated total power production (values in $[0.5, 1.05]$ MW). The actions are changes in orientation with values of either -1, 0 or 1 degrees. The start state is both turbines aligned with the wind vector, which is current practice in wind turbine control [3]. The goal $\mathbf{s}_{\text{goal}}$ is centered around a power production of 1.07 MW with a scale σ_R of 0.05 to encourage high productivity. The number of support points is set to 300.

Each two-turbine row represents a fleet member. Again, we report the results for one target. This is considered to be a new row of which the generator efficiency is set to 1. However, we set the generator efficiencies to 0.9 for 3 source members, and to 0.8 for the remaining 4 source members, which is a realistic configuration that could be the result of aging [34]. The turbines are positioned 100 m apart on a line parallel to the incoming wind vector. The wind speed is set to 6 m/s. We assume independence between turbine rows, which is a reasonable assumption given the specified wind vector, since wake generated by one row will not influence the other turbine rows.

To each turbine row, we provide a batch of 50 transitions randomly sampled from its environment. We measure the performance of the methods by reporting the power production (MW) achieved at the end of the run. We compare the targets to the performance achieved under the optimal policy and to the performance under the policy used in current practice, i.e., aligning all turbines with the incoming wind vector. The results are shown in Figure 4.

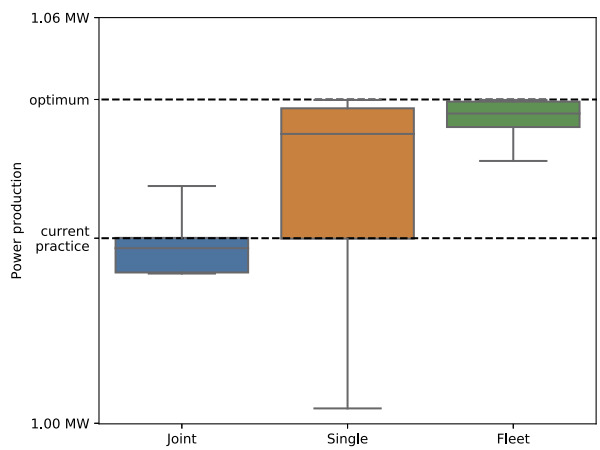


Figure 4: Wind Farm – Boxplot of the power productions for each target type over 50 runs. The optimal performance, as well as the performance achieved when using the control policy used in current practice, are given (dashed lines).

To each member, we provide a batch of 50 transitions sampled uniformly random from their environment, resulting in 400 fleet samples. We run the experiment 50 times for each of the target types: single, joint and fleet. We show the results in Figure 4.

We observe that the single target has a wide variance on its performance. The uncertainty about its transition model is high due to the limited amount of data it has access to. The joint target has lower variance, but has the worst performance, close to the performance of current practice policies. As all data is aggregated, many transition samples are not representative for the true dynamics of the target. The fleet target consistently achieves results that closely follow the optimal performance, as it has the ability to differentiate between relevant and irrelevant samples over the entire fleet data set.

7 DISCUSSION

Fleet-wide policy iteration outperforms the two baselines over all of our experiments. The joint target often fails to reach the goal while the single target is often uncertain about its own transition model. This reflects a trade-off in bias and variance, where we have to decide to either use all data at the risk of misrepresenting the transition model (bias) or only use representative data while remaining uncertain about the model (variance). Our method successfully balances both by properly weighing each source with their correlation with the target. This is reflected in the learned value functions. The fleet target finds the region of highest reward, which is around the goal state, while the single and joint targets misrepresent their value function. We further validated the ability of our method to balance between bias and variance through a sensitivity analysis on the mountain car setting. Specifically, we varied the power parameter of source A between $5 \cdot 10^{-3}$ and $15 \cdot 10^{-3}$ to simulate a range of similarities between source A and the target. The fleet target outperforms both baselines and exhibits similar performance to the single target when the target is significantly different from source A, and thus no information transfer is possible. More information on this analysis can be found in the Supplementary Information.

The successful use of data exchange in fleets has strong implications for the real world applications. The wind farm experiment shows that close-to-optimal performance can be achieved when using fleet-wide policy iteration, while the alternatives (i.e., single and joint learning) often yield performances close to current practice or worse.

The fleet-wide transition model is a sparse variant of the intrinsic coregionalization model (ICM) [14, 4]. The ICM captures cross-covariances between multiple functions, and thus improves the accuracy of those functions jointly. However, as we consider multiple sources and a single target, the target's transition model will be tailored toward improving its own accuracy, rather than the joint accuracy over all fleet members. Additionally, the computational burden when using our sparse coregionalization model is significantly lower. Our method can be executed in a distributed manner over the fleet which reduces the quadratic complexity of the coregionalization matrix in the ICM model to a linear complexity per target member. Moreover, as the covariances in Equation 14 are sparse, the inversion operation can be made linear in the number of fleet members as well. This renders our method scalable to larger fleets. We perform additional experiments to compare our sparse model against ICM in the Supplementary Information.

In future work, we will further improve the scalability of our method by using sparse GP approximations [33, 39] to reduce the computational burden of the matrix inversion. As many of these methods are independent with respect to the covariance kernel or require a factorable kernel, our model is extensible to many of these approximations. By using a sparse GP approximation, our fleet RL method can handle even larger data sets and fleets. Additionally, in a real-world wind farm, dependencies exist between the fleet members [13]. In previous work, we investigated that coordination among agents in the farm can significantly increase the farm's power production and propose a coordinated exploration-exploitation mechanism [1]. In future work, we aim to connect this work with the fleet-wide policy iteration method, which allows for coordinated exploration in fleet settings, while retaining the ability of transferring knowledge among the fleet members.

8 CONCLUSION

In this work, we introduced a novel sample-efficient fleet reinforcement learning method, called *fleet-wide policy iteration*, based on Gaussian processes and coregionalization. It estimates cross-covariances between a fleet of machines and transfers knowledge between them.

We provided experimental results on two benchmark problems: mountain car and cart-pole. In these settings, a target has to share data with a similar and dissimilar fleet member. While the two baselines (i.e., no transfer and single model learning) perform poorly, the learner that uses our fleet-wide policy iteration method manages to solve both tasks consistently. Additionally, we provided a real-world example of how fleet-wide policy iteration can be used in wind farms. The method shows overall improvement with respect to the wind farm's power production.

9 STATEMENT OF REPRODUCIBILITY

The code for the method and experiments is available at `https://github.com/timo-verstraeten/fwpi-experiments`.

ACKNOWLEDGEMENTS

The authors would like to acknowledge FWO (Fonds Wetenschappelijk Onderzoek) for their support through the SB grant of Timothy Verstraeten (#1S47617N) and the SB grant of Pieter J.K. Libin (#1S31916N). This research was supported by funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

REFERENCES

[1] Eugenio Bargiacchi, Timothy Verstraeten, Diederik Roijers, Ann Nowé, and Hado van Hasselt, 'Learning to coordinate with coordination graphs in repeated single-stage multi-agent decision problems', in *International Conference on Machine Learning (ICML)*, pp. 491–499, (2018).

[2] Andrew G Barto, Richard S Sutton, and Charles W Anderson, 'Neuronlike adaptive elements that can solve difficult learning control problems', *IEEE transactions on systems, man, and cybernetics*, (5), 834–846, (1983).

[3] Sjoerd Boersma, BM Doekemeijer, Pieter MO Gebraad, Paul A Fleming, Jennifer Annoni, Andrew K Scholbrock, JA Frederik, and Jan-Willem van Wingerden, 'A tutorial on control-oriented modeling and control of wind farms', in *2017 American Control Conference (ACC)*, pp. 1–18. IEEE, (2017).

[4] Edwin V. Bonilla, Kian Ming A. Chai, and Christopher K. I. Williams, 'Multi-Task Gaussian Process Prediction', *Advances in Neural Information Processing Systems*, **20**, 153–160, (2008).

[5] L. Busoniu, R. Babuska, and B. De Schutter, 'Multi-agent reinforcement learning: A survey', in *The 9th International Conference on Control, Automation, Robotics and Vision*, pp. 1–6, (2006).

[6] M. P. Deisenroth and C. E. Rasmussen, 'PILCO: A model-based and data-efficient approach to policy search', in *Proc. of the International Conference on Machine Learning (ICML)*, (2011).

[7] Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox, 'Multi-task policy search for robotics', in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3876–3881. IEEE, (2014).

[8] Finale Doshi-Velez and George Konidaris, 'Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations', in *Proc. of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1432–1440, (2016).

[9] Yaakov Engel, Shie Mannor, and Ron Meir, 'Reinforcement Learning with Gaussian Processes', in *Proc. of the 22nd International Conference on Machine Learning*, pp. 201–208. ACM Press, (2005).

[10] Pieter Gebraad, Jared J Thomas, Andrew Ning, Paul Fleming, and Katherine Dykes, 'Maximization of the annual energy production of wind power plants by optimization of layout and yaw-based wake control', *Wind Energy*, **20**(1), 97–107, (2017).

[11] Mario Gerla, Eun-Kyu Lee, Giovanni Pau, and Uichin Lee, 'Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds', in *2014 IEEE world forum on internet of things (WF-IoT)*, pp. 241–246. IEEE, (2014).

[12] F González-Longatt, P Wall, and V Terzija, 'Wake effect in wind farm performance: Steady-state and dynamic behavior', *Renewable Energy*, **39**(1), 329–338, (2012).

[13] Francisco González-Longatt, P Wall, and V Terzija, 'Wake effect in wind farm performance: Steady-state and dynamic behavior', *Renewable Energy*, **39**(1), 329–338, (2012).

[14] Pierre Goovaerts, 'Geostatistics for natural resource evaluation', **42**, (1997).

[15] GPy. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy, since 2012.

[16] J. Helsen, C. Peeters, T. Verstraeten, J. Verbeke, N. Gioia, and A. Nowé, 'Fleet-wide condition monitoring combining vibration signal processing and machine learning rolled out in a cloud-computing environment', in *International Conference on Noise and Vibration Engineering (ISMA)*, (2018).

[17] Auke J Ijspeert, Jun Nakanishi, and Stefan Schaal, 'Learning attractor landscapes for learning motor primitives', in *Advances in neural information processing systems*, pp. 1547–1554, (2003).

[18] Jason Jonkman, Sandy Butterfield, Walter Musial, and George Scott, 'Definition of a 5-MW reference wind turbine for offshore system development', Technical report, National Renewable Energy Laboratory (NREL), Golden, CO, USA, (2009).

[19] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez, 'Robust and efficient transfer learning with hidden parameter markov decision processes', in *Advances in Neural Information Processing Systems (NIPS)*, eds., I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, volume 30, 6250–6261, Curran Associates, Inc., (2017).

[20] Jens Kober, Erhan Oztop, and Jan Peters, 'Reinforcement learning to adjust robot movements to new situations', in *Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, (2011).

[21] George Konidaris, Ilya Scheidwasser, and Andrew Barto, 'Transfer in reinforcement learning via shared features', *Journal of Machine Learning Research (JMLR)*, **13**, 1333–1371, (2012).

[22] Alessandro Lazaric and Mohammad Ghavamzadeh, 'Bayesian multi-task reinforcement learning', in *Proc. of the 27th International Conference on Machine Learning (ICML)*, pp. 599–606. Omnipress, (2010).

[23] Rebecca Martin, Iraklis Lazakis, Sami Barbouchi, and Lars Johanning, 'Sensitivity analysis of offshore wind farm operation and maintenance cost and availability', *Renewable Energy*, **85**, 1226–1236, (2016).

[24] Michael D McKay, Richard J Beckman, and William J Conover, 'Comparison of three methods for selecting values of input variables in the analysis of output from a computer code', *Technometrics*, **21**(2), 239–245, (1979).

[25] A. Moore, *Efficient Memory-Based Learning for Robot Control*, Ph.D. dissertation, University of Cambridge, 1990.

[26] Andrew W Moore, 'The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces', in *Proc. of the 6th International Conference on Neural Information Processing Systems (NIPS)*, pp. 711–718, (1993).

[27] NREL. FLORIS. Version 1.0.0, 2019.

[28] Sinno Jialin Pan and Qiang Yang, 'A survey on transfer learning', *IEEE Transactions on knowledge and data engineering*, **22**(10), 1345–1359, (2009).

[29] Martin L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, Inc., 1st edn., 1994.

[30] Carl Edward Rasmussen and Malte Kuss, 'Gaussian processes in reinforcement learning', **16**, 751–758, (2003).

[31] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, Cambridge, MA, USA, 2006.

[32] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth, 'Meta reinforcement learning with latent variable gaussian processes', in *Proc. of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 642–652, (2018).

[33] Edward Snelson and Zoubin Ghahramani, 'Sparse gaussian processes using pseudo-inputs', in *Advances in neural information processing systems*, pp. 1257–1264, (2006).

[34] Iain Staffell and Richard Green, 'How does wind farm performance decline with age?', *Renewable energy*, **66**, 775–786, (2014).

[35] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press Cambridge, 1998.

[36] Matthew E Taylor and Peter Stone, 'Cross-domain transfer for reinforcement learning', in *Proc. of the 24th International Conference on Machine Learning (ICML)*, pp. 879–886. ACM, (2007).

[37] Timothy Verstraeten, Ann Nowe, Jonathan Keller, Yi Guo, Shuangwen Sheng, and Jan Helsen, 'Fleetwide data-enabled reliability improvement of wind turbines', *Renewable and Sustainable Energy Reviews*, **109**, 428–437, (2019).

[38] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli, 'Multi-task reinforcement learning: A hierarchical bayesian approach', in *Proc. of the 24th International Conference on Machine Learning (ICML)*, pp. 1015–1022, (2007).

[39] Andrew Wilson and Hannes Nickisch, 'Kernel interpolation for scalable structured gaussian processes (KISS-GP)', in *International Conference on Machine Learning (ICML)*, pp. 1775–1784, (2015).

[40] Yang Yu, 'Towards sample efficient reinforcement learning', in *Proc. of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5739–5743. International Joint Conferences on Artificial Intelligence Organization, (2018).