Adaptive Sparse Grids in Reinforcement Learning

Jochen Garcke and Irene Klompmaker

Abstract We propose a model-based online reinforcement learning approach for continuous domains with deterministic transitions using a spatially adaptive sparse grid in the planning stage. The model learning employs Gaussian processes regression and allows a low sample complexity. The adaptive sparse grid is introduced to allow the representation of the value function in the planning stage in higher dimensional state spaces. This work gives numerical evidence that adaptive sparse grids are applicable in the case of reinforcement learning.

1 Introduction

We consider function approximation techniques for reinforcement learning (RL). Reinforcement learning is a computational approach to learning, where an agent tries to maximise the total amount of reward it receives when interacting with a complex, uncertain environment [34]. The setting is very closely related to solving optimal control problems using Hamilton-Jacobi Bellman (HJB) equations, but in contrast to that only a partial amount of the data describing the system is known. For example the state dynamics describing the evolution of a system are unknown and can only be observed by performing actions.

Formally the evolution of the problem in the control space is determined by the differential equation

$$\frac{\partial x(t)}{\partial t} = f(x(t), \beta(t)),$$

Jochen Garcke
Universität Bonn, Institut für Numerische Simulation, Wegelerstr. 6, D-53115 Bonn
e-mail: garcke@ins.uni-bonn.de

Irene Klompmaker

where $x(t)$ is the *state*, $\beta(t)$ the *action* and f is called *state dynamics*. The latter describes the effect of an action β taken in a particular state x, and gives the new state $f(x,\beta)$ after the action is taken. Although we consider deterministic dynamics in this work, they could also be stochastic, to which situation our approach can be extended. For an initial state x_0 the choice of actions β therefore leads to a unique trajectory $x(t)$. Further, there is the *reinforcement* or *reward* function $r(x,\beta)$, which assigns each state (or state-action pair) a numerical value indicating the intrinsic desirability of that state. The aim is to find a policy which maximises the total reward in the long run, where rewards in states reached by a trajectory through the state space are taken into account. For simplicity we consider a deterministic *policies* $\pi(x)$, which assign each state a unique action, i.e., $\beta = \pi(x)$; it is a mapping from perceived states of the environment to actions to be taken when in those states.

There are many types of reinforcement learning problems: state dynamics known or not, discrete or continuous case, model-based or model-free, deterministic or stochastic [34]. What they all have in common is that they solve an optimal control problem, at least implicitly. The difference of reinforcement learning in comparison to optimal control problems is that the state dynamics and the reinforcement function are, a priori, at least partially unknown. Nevertheless, it is a problem of optimal control and the dynamic programming method is usually employed to estimate the best future cumulative reinforcement.

In this work we consider a deterministic *model-based* reinforcement learning approach in a continuous state space with unknown state dynamics, but known rewards. As in [9, 23] and related methods our approach consists of two ingredients, a *model-learner* and a *planner*. By performing an action β in a state x the algorithm interacts with the environment and observes a sample $f(x,\beta)$ of the state dynamics. Based on such sample transitions $\{x_k,\beta_k,f(x_k,\beta_k)\}_{k=1,\ldots,K}$ the model-learner then estimates the state dynamics. On the other hand, given the current model the planner aims to find the best possible action β in a state x, i.e. those which is part of the trajectory starting at x with the highest total reward, and thereby determines an approximation π to the optimal policy π^*. With more and more samples of the state dynamics the model-learner is assumed to become more accurate, while the derived actions are supposed to get closer to the optimal ones from π^*.

For model-learning we use Gaussian process regression as in [23], while for the planner we employ adaptive sparse grid interpolation. The discretization technique of sparse grids allows to cope with the curse of dimensionality to some extent. It is based on a hierarchical multilevel basis [36] and a sparse tensor product construction. The underlying idea was first used for numerical integration and interpolation [33]. Subsequently, the sparse grid method has been developed for the solution of partial differential equations [37]. By now, it is also successfully used for, e.g., integral equations, stochastic differential equations, machine learning, or approximation, see the overview articles [10, 17] and the references cited therein.

For the representation of a function f defined over a d-dimensional domain, the conventional sparse grid approach employs $\mathcal{O}(h_n^{-1}\cdot\log(h_n^{-1})^{d-1})$ grid points in the discretization process, where $h_n := 2^{-n}$ denotes the mesh width. It can be shown that the order of approximation to describe a function f, provided that cer-

tain mixed smoothness conditions hold, is $\mathcal{O}(h_n^2 \cdot \log(h_n^{-1})^{d-1})$. This is in contrast to conventional grid methods, which need $\mathcal{O}(h_n^{-d})$ for an accuracy of $\mathcal{O}(h_n^2)$, albeit for less stringent smoothness conditions. Thus, the curse of dimensionality of full grid methods arises for sparse grids to a much smaller extent. In case the smoothness conditions are not fulfilled, spatially adaptive sparse grids have been used with good success [6, 10, 15, 31]. There, as in any adaptive grid refinement procedure, the employed hierarchical basis functions are chosen during the actual computation depending on the function to be represented. In regard to adaptivity, closely related work in reinforcement learning was presented in [28, 30], in contrast to these approaches we investigate sparse grids in the planner and use a model-based setting.

The presented sparse grid approach for reinforcement learning is an extension of a semi-Lagrangian scheme for HJB-equations on an adaptive sparse grid, which was introduced in [6]. There it was empirically shown that for problems related to the front propagation model, the number of grid points needed in higher dimensions to approximately represent the involved functions with a given threshold error can be small. Thus, the approach is able to circumvent the curse of dimensionality of standard grid approaches for Hamilton-Jacobi Bellman equations to some extent. This work now shows numerical results for the case of reinforcement learning and gives evidence that adaptive sparse grids can be used there as well.

But note that the sparse grid scheme is not monotone as the interpolation with sparse grids is not monotone [29, 31]. Thus neither convergence towards the viscosity solutions of Hamilton-Jacobi Bellman equations nor stability of the scheme can presently be guaranteed, even for the linear advection equation. Consequently, these properties do not necessarily hold in the case of reinforcement learning either; numerically divergent behaviour of the adaptive sparse grid approach can be observed in certain situations. To this end, further analytical work on the scheme, both for the HJB and the RL case, is necessary.

2 Reinforcement Learning

Our reinforcement learning approach is based on the procedure presented in [23], a model-based online reinforcement learning approach for continuous domains with deterministic transitions. It separates function approximation in the model learner from the interpolation in the planner. For model-learning we use Gaussian process regression as in [23], but we replace the equidistant grid in the planner by an adaptive sparse grid procedure similar to the one used for HJB equations [6]. The overall approach assumes some properties of the reinforcement learning problems under consideration: We consider discrete actions, a smooth transition function, i.e. an action performed on states which are close in state space must lead to successor states that are close, deterministic transitions, and known reward functions. The latter two are mainly for simplicity, the ingredients of the approach can be extended to the non-deterministic case, and learning a reward function would just be one more function to be learned. Since our goal is to investigate the applicability of adaptive sparse

grids in the planning stage of a reinforcement learning setting, a simple setting is advantageous to concentrate on the effect of, and interplay with, unknown and only approximately learned state dynamics, which is the extension in comparison to [6].

We assume that the *state space* $\mathscr{X}$ is a hyperrectangle in $\mathbb{R}^d$, which is justified for many applications, and that we have a finite *action space* $\mathscr{B}$, this might involve discretizations of continuous controls. For simplicity we assign each action a unique number $1,\ldots,|\mathscr{B}|$. Note that we use here a setting where all actions involve the same time horizon τ, which therefore can be omitted from the exposition for simplification. In general, temporal aspects need to be taken into account, see e.g. [26, 30]. The function $f : \mathscr{X} \times \mathscr{B} \to \mathscr{X}$ describes the *state dynamics*. In our setting the state dynamics are (at least partially) unknown, only an approximate model $\hat{f} : \mathscr{X} \times \mathscr{B} \to \mathscr{X}$, which will be learneded from samples, is available with $f \approx \hat{f}$. Finally $r : \mathscr{X} \times \mathscr{B} \to \mathbb{R}$ is the *reward* function.

For a state $x \in \mathscr{X}$ one is interested in determining a sequence of actions $\beta_0, \beta_1, \ldots$ such that the accumulated reward is maximised, this is given by the optimal value function $v^*(x)$

$$v^*(x) := \max_{\beta_0,\beta_1,\ldots} \left\{ \sum_{t=0}^{\infty} \gamma^t \cdot r(x_t,\beta_t) \mid x_0 = x, x_{t+1} = f(x_t,\beta_t) \right\},$$

where $0 < \gamma < 1$ is the discount factor, which determines the importance of future rewards [5, 34].

The value iteration, the employed basic numerical scheme, is based on the dynamic programming principle and can be formulated as

$$v^{n+1}(x) = \max_{\beta \in \mathscr{B}} \left[\gamma \cdot v^n(\hat{f}(x,\beta)) + r(x,\beta) \right], \tag{1}$$

which computes the value function $v^*(x)$ in the limit $n \to \infty$, see e.g. [1, 25, 26]. Note that in a similar fashion the value function v^π for a fixed policy π can be computed. This formulation for the computation of the value function, the planning, is valid for both situations, a known model f and a to be learned model $\hat{f}$. In addition, a numerical discretization of the value function is necessary, in particular for continuous domains.

For any given value function v, e.g. a suitable numerical approximation $\hat{v}$ of the optimal value function v^* computed by value iteration using a discretization approach, the corresponding control policy $\pi(x)$ determined by v can easily be obtained, since at each state the optimal action can be chosen depending on the given value function v as follows

$$\pi(x) \in \arg\max_{\beta \in \mathscr{B}} \left[\gamma \cdot v(\hat{f}(x,\beta)) + r(x,\beta) \right].$$

The overall reinforcement learning approach now consists of three parts as outlined in the following algorithm.

Algorithm: Generic Model-based Reinforcement Learning Approach

while *learning* **do**
  interact with system and store observed transitions
  learn model $\hat{f}$ based on observed transitions
  for planning use model $\hat{f}$ to determine $\hat{v}^{\pi}$

In the following we will describe the model learning using Gaussian process regression and the planning procedure in the next sections, where for the representation of the value function v we use a finite element approach, similar to [1, 25, 26], but based on a sparse grid.

2.1 Model Learning with Gaussian Processes

We now describe how we, following [23], learn the model from samples which are obtained by interactions with the environment. In its core, model learning is a regression problem. In this work we aim to concentrate on evaluating sparse grids for the planning stage, and therefore apply for the model learning a regression approach successfully used before in reinforcement learning, namely Gaussian processes (GPs) [13, 23, 32]. As the kernel the squared exponential is employed

$$k(x,x';v_0,b,\theta) = v_0 \exp\left\{-0.5(x-x')^2\theta\right\},$$

where v_0, b, θ are the to be determined hyperparameters. In our view, a main advantage of Gaussian processes in this application is the possible automatic determination of the hyperparameters in a controlled fashion using a maximum likelihood approach. This is here in particular relevant since, as we will see, one needs to repeatedly compute, or update, regression models. Nevertheless, it would be interesting to investigate sparse grid regression [16, 31] in the model learning phase as well.

The input data for the regression algorithm are the taken actions and their resulting state, i.e. $\mathscr{S} = \{x_k, \beta_k, x_{k+1}\}_{k=1,2,\ldots}$, where $x_{k+1} = f(x_k, \beta_k)$. As noted before, the transition function $f : \mathscr{X} \times \mathscr{B} \to \mathscr{X}$ is d-dimensional: $f(x,\beta) = [f_1(x,\beta),\ldots,f_d(x,\beta)]^T$. One can either estimate f directly, i.e. the absolute transitions, or the relative change $x_{k+1} - x_k$, the latter we do as in [23]. We train multiple Gaussian processes, one for each action and output dimension, and use the combined predictions afterwards. In other words, a GP_{ij} is trained for the output in the i-th dimension of the j-th action, using the data when that action was taken, i.e. the input data for GP_{ij} is $\mathscr{S}_{ij} = \left\{x_k, x_{k+1}^{(i)} - x_k^{(i)}\right\}_{\beta_k=j,k=1,2,\ldots}$. The hyperparameters are computed for each individual GP_{ij} by optimizing the marginal likelihood. For any test point x the GP_{ij} gives a distribution over target values $\mathscr{N}(\mu_{ij}, \sigma_{ij})$ with mean

$\mu_{ij}(x)$ and variance $\sigma_{ij}^2(x)$. The change in the i-th coordinate of the state under the j-th action is predicted by the mean μ_{ij}, where the variance can be interpreted as the uncertainty of the prediction. The full learned model consists of $d \cdot |\mathscr{B}|$ Gaussian processes and the predicted successor state $\hat{f}(x,\beta)$ for any action β taken in state x is then

$$\hat{f}(x,\beta) := \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(d)} \end{bmatrix} + \begin{bmatrix} \mu_{1\beta}(x) \\ \vdots \\ \mu_{d\beta}(x) \end{bmatrix},$$

see [23] for more details on model learning with Gaussian processes.

3 Planning with Sparse Grids

We consider in the following the discretization needed for the planner in the case of a continuous state-space and concentrate on the function approximation aspect. These techniques are used in the context of Hamilton-Jacobi Bellman equations as a device for having a numerical representation of the value function. They discretize an HJB-equation (with a resolution ε) into a dynamic programming (DP) problem for some stochastic Markov Decision Process (MDP). Using DP techniques the MDP can then be solved. The convergence of the solution V^ε of the discrete MDP to the value function V of the continuous problem for $\varepsilon \to 0$ can be proven under assumptions on the discretization scheme, namely it being (using suitable definitions) a consistent, monotone and uniformly continuous numerical procedure to solve the underlying HJB-equation. Generalizing these proofs (in regard to the deterministic or stochastic setting, the regularity of the value function and the properties of the discretization scheme) is an active field of research [2, 3, 4, 24, 26, 28]. Typical discretization schemes are of finite-difference type [8, 7, 25], (operator) splitting methods [35], or control schemes based on the dynamic programming principle (e.g. [11]).

The idea of a discretized HJB-equation was adopted to the field of reinforcement learning [26, 28, 30]. Since the state dynamics and the reinforcement function are not perfectly known, the original convergence proof [4] for DP was generalized to the case where only approximations of these are known. Convergence was shown when the number of iterations of the RL scheme (for the approximation of the state dynamics and the reinforcement function) goes to infinity and the discretization step tends to zero. The result applies in a general form to model-based or model-free RL algorithms, for off-line or on-line methods, for deterministic or stochastic dynamics and finite element or finite difference discretization [26].

Adaptive finite difference grids and a posteriori error estimates using the methodology from the numerical solution of partial differential equations were studied for the deterministic HJB-equation [19] and later generalized to the stochastic case [20]. Adaptive schemes are particularly important since often the value function is non-smooth. In [28] different spatial refinement strategies were studied, in particular a heuristic is proposed which refines the grid mostly where there is a transition in

the optimal control. These discretization approaches are limited in the number of dimensions due to the curse of dimensionality, i.e. the complexity grows exponentially with the number of dimensions.

An important aspect of these grid based approaches is the inherent locality in the schemes and its properties. Although, for example, formulated using a finite element representation, one does not solve a 'global' Galerkin-type problem, but the convergence is due to local properties of a function defined on a simplex (or box). This local view unfortunately does not hold for sparse grids, which renders the usual theoretical justifications of these grid approaches for HJB-equations invalid. Nevertheless, the empirical observations in [6] and this work give evidence that spatial adaptive sparse grids can be used in this setting, although further detailed investigations are necessary to provide criteria when this is the case.

In case of a suitably chosen, for now fixed, discretization grid Ω with corresponding function space V the planning step based on dynamic programming and using a model $\hat{f}$ can be written as

- Suitably initialize $v_0 \in V$.
- Iterate for $n = 0, 1, 2, \ldots$ until convergence, e.g. $|v^{n+1}(x) - v^n(x)| < tol \quad \forall x \in \Omega$,

$$v^{n+1}(x) = \max_{\beta \in \mathscr{B}} [\gamma \cdot v^n(\hat{f}(x,\beta)) + r(x,\beta)] \quad \forall x \in \Omega. \tag{2}$$

Here, $v^n \in V$ is the numerical solution computed by the scheme at DP step n, which in some cases can be interpreted as a time step, and the value $v^n\left(\hat{f}(x,\beta)\right)$ denotes the interpolation of v^n at point $\hat{f}(x,\beta)$. Note that for our later experiment the minimization over the set $\mathscr{B}$ can be done in a straightforward way by evaluating the function values for each possible action $b \in \mathscr{B}$. In case this number is too large, or if the set $\mathscr{B}$ is infinite, then a minimisation procedure which uses only evaluations of the objective function, and not its derivatives, could be performed without changing the main steps of the scheme.

Note that many reinforcement learning algorithms in continuous state-space, e.g. the ones employing approximate value iteration (AVI) [27, 28, 34] for the dynamic programming part, use supervised machine learning methods (i.e. regression algorithms [21]) to achieve the function approximation, also called *generalization* in this context [5, 34]. For example neural networks [5] or decision trees [12] can be used. But there is no general convergence of the algorithms, and the combination of DP methods with function approximation may produce unstable or divergent results even when applied to very simple problems [34]. Nevertheless, for schemes which are linear in the model parameters, a convergence proof is available (see [34]). Finally note that there are recent results indicating that the contribution of the error due to the approximation of the Bellman operator at each iteration is more prominent in later iterations of AVI and the effect of an error term in the earlier iterations decays exponentially fast [14]. To put more emphasis on having a lower Bellman error at later iterations one could increase the number of samples during the scheme or use more powerful function approximators in the end.

Having set the background for function approximation for the value function in the planning stage we now describe our adaptive sparse grid procedure, which is based on [6].

3.1 Sparse Grids

For ease of presentation we will consider the domain $\Omega = [0,1]^d$ in this section. Let $\underline{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$ denote a multi-index. We define the anisotropic grid $\Omega_{\underline{l}}$ on Ω with mesh width $h_{\underline{l}} := (h_{l_1}, \ldots, h_{l_d}) := (2^{-l_1}, \ldots, 2^{-l_d})$. It has, in general, different but equidistant mesh widths h_{l_t} in each coordinate direction t, $t = 1, \ldots, d$. The grid $\Omega_{\underline{l}}$ thus consists of the points $x_{\underline{l},\underline{j}} := (x_{l_1,j_1}, \ldots, x_{l_d,j_d})$, with $x_{l_t,j_t} := j_t \cdot h_{l_t} = j_t \cdot 2^{-l_t}$ and $j_t = 0, \ldots, 2^{l_t}$. For any grid $\Omega_{\underline{l}}$ we define the associated space $V_{\underline{l}}$ of piecewise d-linear functions

$$V_{\underline{l}} := \operatorname{span}\{\phi_{\underline{l},\underline{j}} \mid j_t = 0, \ldots, 2^{l_t},\ t = 1, \ldots, d\}, \tag{3}$$

which is spanned by the conventional basis of d-dimensional piecewise d-linear hat functions

$$\phi_{\underline{l},\underline{j}}(\underline{x}) := \prod_{t=1}^{d} \phi_{l_t,j_t}(x_t). \tag{4}$$

The one-dimensional functions $\phi_{l,j}(x)$ are defined by

$$\phi_{l,j}(x) = \begin{cases} 1 - |x/h_l - j|, & x \in [(j-1)h_l, (j+1)h_l] \cap [0,1], \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The multi-index $\underline{l} \in \mathbb{N}^d$ denotes the level, i.e. the discretization resolution, be it of a grid $\Omega_{\underline{l}}$, of a space $V_{\underline{l}}$, or of a function $f_{\underline{l}}$, whereas the multi-index $\underline{j} \in \mathbb{N}^d$ gives the position of a grid point $x_{\underline{l},\underline{j}}$ or its corresponding basis function $\phi_{\underline{l},\underline{j}}$.

We now define a hierarchical difference space $W_{\underline{l}}$ via

$$W_{\underline{l}} := V_{\underline{l}} \setminus \bigoplus_{t=1}^{d} V_{\underline{l}-\underline{e}_t}, \tag{6}$$

where $\underline{e}_t$ is the t-th unit vector. In other words, $W_{\underline{l}}$ is spanned by all $\phi_{\underline{k},\underline{j}} \in V_{\underline{l}}$ which are not included in any of the spaces $V_{\underline{k}}$ smaller[1] than $V_{\underline{l}}$. To complete the definition, we formally set $V_{\underline{l}} := \emptyset$, if $l_t = 0$ for at least one $t \in \{1, \ldots, d\}$. As can be easily seen from (3) and (6), the definition of the index set

[1] We call a discrete space $V_{\underline{k}}$ smaller than a space $V_{\underline{l}}$ if $\forall_t k_t \leq l_t$ and $\exists t : k_t < l_t$. In the same way a grid $\Omega_{\underline{k}}$ is smaller than a grid $\Omega_{\underline{l}}$.

$$\mathsf{B}_{\underline{l}} := \left\{ \underline{j} \in \mathbb{N}^d \,\middle|\, \begin{array}{ll} j_t = 1,\ldots,2^{l_t}-1, & j_t \text{ odd}, t = 1,\ldots,d, \text{ if } l_t > 1, \\ j_t = 0,1,2, & t = 1,\ldots,d, \text{ if } l_t = 1 \end{array} \right\} \tag{7}$$

leads to

$$W_{\underline{l}} = \text{span}\{\phi_{\underline{l},\underline{j}} | \underline{j} \in \mathsf{B}_{\underline{l}}\}. \tag{8}$$

The family of functions

$$\left\{ \phi_{\underline{l},\underline{j}} \,\middle|\, \underline{j} \in \mathsf{B}_{\underline{l}} \right\}_{\underline{l}=(1,\ldots,1)}^{(n,\ldots,n)} \tag{9}$$

is just the hierarchical basis [36] of $V_n (:= V_{(n,\ldots,n)})$, which generalizes the one-dimensional hierarchical basis to the d-dimensional case with a tensor product ansatz. Observe that the supports of the basis functions $\phi_{\underline{l},\underline{j}}(\underline{x})$, which span $W_{\underline{l}}$, are disjoint for $\underline{l} > 1$.

Zenger [37] introduced so-called *sparse grids*, where hierarchical basis functions with a small support, and therefore a small contribution to the function representation, are not included in the discrete space of level n any more.

Formally, the sparse grid function space $V_n^s \subset V_n$ is defined as

$$V_n^s := \bigoplus_{|\underline{l}|_1 \leq n+d-1} W_{\underline{l}}. \tag{10}$$

Every $f \in V_n^s$ now can be represented as

$$f_n^s(\underline{x}) = \sum_{|\underline{l}|_1 \leq n+d-1} \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}}(\underline{x}). \tag{11}$$

The resulting grid which corresponds to the approximation space V_n^s is called sparse grid and is denoted by Ω_n^s.

The sparse grid space V_n^s has a size of order $\dim V_n^s = \mathscr{O}(2^n \cdot n^{d-1})$, see [10]. It thus depends on the dimension d to a much smaller degree than a standard full grid space whose number of degrees of freedom is $\mathscr{O}(2^{nd})$. Note that for the approximation of a function f by a sparse grid function $f_n^s \in V_n^s$ the error relation

$$\|f - f_n^s\|_2 = \mathscr{O}\left(2^{-2n} \cdot n^{d-1}\right)$$

holds, provided that f fulfils the smoothness requirement $|f|_{H^2_{mix}} < \infty$ [10]. Therefore, sparse grids need much fewer points in comparison to a full grid to obtain an error of the same size.

3.2 Spatially Adaptive Sparse Grids

The sparse grid structure (10) defines an a priori selection of grid points that is optimal if certain smoothness conditions are met, i.e. if the function has bounded second mixed derivatives, and no further knowledge of the function is known or used. If the aim is to approximate functions which either do not fulfil this smoothness condition at all or show strongly varying behaviour due to finite but nevertheless locally large derivatives, adaptive refinement may be used. There, depending on the characteristics of the problem and the function at hand, adaptive refinement strategies decide which points and corresponding basis functions should be incrementally added to the sparse grid representation to increase the accuracy.

In the sparse grid setting, usually an error indicator stemming directly from the hierarchical basis is employed [15, 18, 31]: depending on the size of the hierarchical surplus $\alpha_{\underline{l},\underline{j}}$ it is decided whether a basis function is marked for further improvement or not. This is based on two observations: First, the hierarchical surplus indicates the absolute change in the discrete representation at point $x_{\underline{l},\underline{j}}$ due to the addition of the corresponding basis function $\phi_{\underline{l},\underline{j}}$, i.e. it measures its contribution to a given sparse grid representation (11) in the maximum-norm. And second, a hierarchical surplus represents discrete second mixed derivatives and hence can be interpreted as a measure of the smoothness of the considered function at point $x_{\underline{l},\underline{j}}$.

In the adaptive procedure we use an index set $\mathscr{I}$ to track the indices of the employed basis functions and denote the corresponding sparse grid by $\Omega_{\mathscr{I}}$ and the associated sparse grid space by $V_{\mathscr{I}}$, respectively. We start with a coarse initial sparse grid function $f_n^s \in V_n^s$ for some given small n as in (11). The index set is thus initialized as $\mathscr{I} := \{(\underline{l},\underline{j}) \mid |\underline{l}|_1 \leq n+d-1\}$. We proceed as follows: If, for any given index $(\underline{l},\underline{j}) \in \mathscr{I}$, we have

$$|\alpha_{\underline{l},\underline{j}}| \cdot \|\phi_{\underline{l},\underline{j}}\| > \varepsilon \tag{12}$$

for some given constant $\varepsilon > 0$, then the index will be *marked*. Here, $\|\cdot\|$ is typically either the L^∞- or L^2-norm, but other norms or weighted mixtures of norms are used in practice as well. If an index is marked, all its $2d$ so-called *children* will be added to the index set $\mathscr{I}$ to refine the discretization, i.e. all $(\tilde{\underline{l}},\tilde{\underline{j}})$ with $\tilde{\underline{l}} = \underline{l} + \underline{e}_t$ and $\tilde{\underline{j}} = \underline{j} + j_t\ \underline{e}_t \pm 1$ will be added to $\mathscr{I}$ for $t = 1,\ldots,d$. For the indices added that way it is possible that not all *parents* in all dimensions are already contained in the grid; note that in such cases, for algorithmic and consistency reasons, these missing parents have to be added to $\mathscr{I}$ as well. Thus for any $(\underline{l},\underline{j}) \in \mathscr{I}$ all parents $(\tilde{\underline{l}},\tilde{\underline{j}})$ with $\tilde{\underline{l}} \leq \underline{l}$ and $supp(\phi_{\tilde{\underline{l}},\tilde{\underline{j}}}) \cap supp(\phi_{\underline{l},\underline{j}}) \neq \emptyset$ are also in the index set $\mathscr{I}$. In other words, "holes" in the hierarchical structure are not allowed. In Algorithm 1 we give the adaptive refinement procedure. If the function values at the newly added grid points are easily available, the refinement step can be repeated until no indices are added any more [6]. Note that if a global error criterion is available one can perform an additional outer loop with successively decreasing ε until the measured global error falls below a given threshold ε_{glob}.

In a similar way one can use the value $|\alpha_{\underline{l},\underline{j}}| \cdot \|\phi_{\underline{l},\underline{j}}\|$ to *coarsen* the grid in case of over-refinement. If this value is smaller than some coarsening constant η, and

Algorithm 1: Spatially Adaptive Refinement Step

Data: initial index set $\mathscr{I}$, to be refined function $v_{|\mathscr{I}}$, refinement threshold ε
Result: refined index set $\mathscr{I}$, refined function $v_{|\mathscr{I}}$
for $(\underline{l},\underline{j}) \in \mathscr{I}$ **do** ▷ look at all indices
  if $|\alpha_{\underline{l},\underline{j}}| \cdot \|\phi_{\underline{l},\underline{j}}\| > \varepsilon$ **then** ▷ hierarchical surplus is large
    for $t = 1,\ldots,d$ **do**
      if $(\tilde{\underline{l}},\tilde{\underline{j}}) \notin \mathscr{I}$ *for* $\tilde{\underline{l}} = \underline{l} + \underline{e}_t$ *and* $\tilde{\underline{j}} \in \{\underline{j} + j_t\,\underline{e}_t \pm 1\}$ **then**
        $\mathscr{I} = \mathscr{I} \cup (\tilde{\underline{l}},\tilde{\underline{j}})$ ▷ add children which are not in $\mathscr{I}$

check $\forall (\underline{l},\underline{j}) \in \mathscr{I}$ holds: $(\tilde{\underline{l}},\tilde{\underline{j}}) \in \mathscr{I}$ for $\tilde{\underline{l}} \le \underline{l}$ and $supp(\phi_{\tilde{\underline{l}},\tilde{\underline{j}}}) \cap supp(\phi_{\underline{l},\underline{j}}) \neq \emptyset$
for *all added indices* $(\underline{l},\underline{j}) \in \mathscr{I}$ **do**
  initialize $\alpha_{\underline{l},\underline{j}} = 0$

no children of $(\underline{l},\underline{j})$ are in $\mathscr{I}$, the index will be removed from this set. In Algorithm 2 we give the coarsening step, where the procedure is repeated until no indices are being removed. The coarsening will in particular be relevant once we consider problems where the region in need of a higher resolution changes during the computation. This is relevant for time-dependent problems, but also for the planning considered in this work, which in some sense can be viewed as a time-dependent problem. More importantly, the value function to be represented can change during the computation due to changes in the learned model.

Algorithm 2: Spatially Adaptive Coarsening

Data: index set $\mathscr{I}$, coarsening threshold η, and $\alpha_{\underline{l},\underline{j}}\ \forall (\underline{l},\underline{j}) \in \mathscr{I}$
Result: coarsened index set $\mathscr{I}$
while *indices are removed from* $\mathscr{I}$ **do**
  for $(\underline{l},\underline{j}) \in \mathscr{I}$ **do** ▷ look at all indices
    if $|\alpha_{\underline{l},\underline{j}}| \cdot \|\phi_{\underline{l},\underline{j}}\| < \eta$ **then** ▷ hierarchical surplus is small
      if $\forall t = 1,\ldots,d$: $(\tilde{\underline{l}},\tilde{\underline{j}}) \notin \mathscr{I}$ *for* $\tilde{\underline{l}} = \underline{l} + \underline{e}_t$ *and* $\tilde{\underline{j}} \in \{\underline{j} + j_t\,\underline{e}_t \pm 1\}$ **then**
        $\mathscr{I} = \mathscr{I} \backslash (\underline{l},\underline{j})$ ▷ remove if no children in $\mathscr{I}$

4 Sparse Grid Based Scheme for Reinforcement Learning

With these ingredients we now can formulate our approach, which in the end is quite similar to the semi-Lagrangian scheme for Hamilton-Jacobi Bellman equations using spatial adaptive sparse grids introduced in [6], but with only an approximate model for the state dynamics.

The planning scheme is given in Algorithm 3. We perform p steps of the DP equation (2) and then do one refinement step. Note that the initial steps are randomly chosen since no infomation is exploitable, i.e. no samples from the state dynamics exist. This is repeated a certain number of times, or stops early in case the change in a DP step is small. Observe that it is not useful to aim for a convergence of the DP scheme in the initial stages of the overall RL procedure [14]. Since the learned model will take some time to be a reasonable approximation of the state dynamics, and can be quite coarse in the beginning, aiming for convergence of the planning step can even lead to wrong behaviour. Therefore we limit the number of DP steps per planning stage. The same reasoning is behind the idea of calling the refinement algorithm only after every p DP steps, the resolution of the discretization of the value function shall only grow slowly. Once the value iteration is finished we coarsen the obtained sparse grid to reduce the further computational effort.

Algorithm 3: Adaptive SG-planning in Reinforcement Learning

Data: initial index set $\mathscr{I}(0)$, initial function v^0, refinement constant ε, coarsening constant η, model $\hat{f}$, $k = 0$
Result: adaptive sparse grid solution $v^{k_e} \in V_{\mathscr{I}(k_e)}$
repeat ▷ iterate until convergence or max. number of steps
  k = k+1
  $v^k(x) = \max_{\beta \in \mathscr{B}} [\gamma \cdot v^{k-1}(\hat{f}(x,\beta)) + r(x,\beta)] \quad \forall x \in \Omega_{\mathscr{I}}$ ▷ DP step (2)
  if k mod $p = 0$ **then**
    call Alg. 1 with $\mathscr{I}(k-1)$, v^k, ε ▷ refine $v^k \in V_{\mathscr{I}(k)}$ every p steps
until $|v^k(x) - v^{k-1}(x)| < tol \quad \forall x \in \Omega_{\mathscr{I}(k)}$ *and* $k < k_{max}$
$k_e = k$
call Alg. 2 with $\mathscr{I}(k_e)$, η and v^{k_e} ▷ coarsen v^{k_e}

The overall reinforcement learning procedure is presented in Algorithm 4. The algorithm performs a number of interactions with the environment and thereby observes new state transitions $(x, \beta, f(x,\beta))$. Every T actions first the model is updated to integrate the newly observed data, and then the value function is updated in the planning step. Updating after every new transition information would be a quite costly procedure, which we hereby avoid. Like in [23] the experiment we are investigating is performed in episodes, i.e. the trajectory through the state space is re-started once the target state, if defined, is reached or after a certain number of steps, i.e. $T \cdot \#ep_{max}$. The latter prevents the algorithm from investigating uninteresting regions of the state space when the transition path does not follow a 'good' trajectory after 'bad' choices of actions due to model uncertainty or early stages of convergence to the value function.

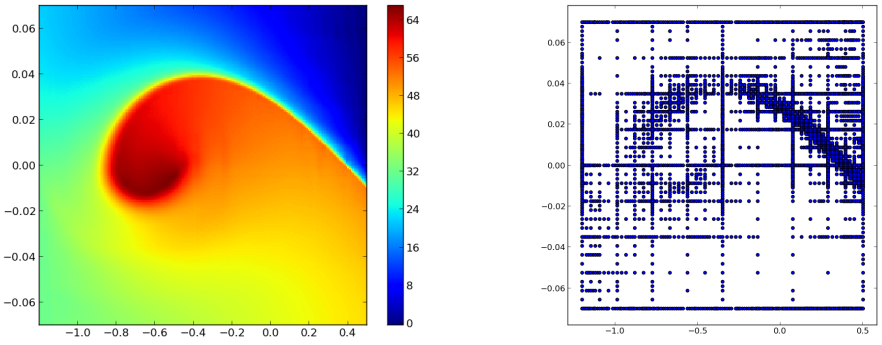
Algorithm 4: SG-GP-scheme in reinforcement learning

Data: suitable initial index set $\mathscr{I}(0)$, refinement constant ε, coarsening constant η, initial state x_{init}, and episode length $\#ep_{max}$
Result: adaptive sparse grid solution $v^{n_e} \in V_{\mathscr{I}(n_e)}$
$\#ep = 0, n = 0, x_0 = x_{init}$
repeat ▷ iterate
  for $t = 1, \ldots, T$ **do** ▷ interact with system in chunks
    action β_t is chosen according to $\beta_t = \arg\max_{\beta \in \mathscr{B}} [\gamma \cdot v^n(\hat{f}(x_{t-1}, \beta)) + r(x_{t-1}, \beta)]$
    interact with system by executing action β_t
    observe next state x_t and store transition $\mathscr{S} = \mathscr{S} \cup \{x_{t-1}, \beta_t, x_t\}$
  based on $\mathscr{S}$ learn model $\hat{f} = \{\mathsf{GP}_{ij}\}_{i=1,\ldots,d, j=1,\ldots,|\mathscr{B}|}$
  call Alg. 3 with $\mathscr{I}(n)$, v^n, ε, η, $\hat{f}$
    ▷ for planning compute value function v^{n+1} on $\Omega_{\mathscr{I}(n+1)}$
  $n = n+1, \#ep = \#ep+1$ ▷ count number of chunks
  if $\#ep \bmod \#ep_{max} = 0$ *or* x_T *is target state* **then** ▷ limit length of episode
    $x_0 = x_{init}$
  else
    $x_0 = x_T$
until $|v^n(x) - v^{n-1}(x)| < tol \quad \forall x \in \Omega_{\mathscr{I}(n)}$
$n_e = n$

5 Experiments

We evaluate our approach on a well-known reinforcement learning benchmark, the mountain car problem [34]. The goal is to drive a car from the bottom of a valley to the top of a mountain. Since the car is underpowered it cannot climb the mountain directly, but instead has to build up momentum by first going in the opposite direction. The state space is two-dimensional, the position of the car is described by $x_1 \in [-1.2, 0.5]$ and $x_2 \in [-0.07, 0.07]$ is its velocity. The actions are accelerating by a fixed value to the left, right, or no acceleration, encoded by actions $\beta \in \{-1, +1, 0\}$. Every step gives a reward of -1, until the top of the mountain on the right satisfying $x_1 \geq 0.5$ is reached, the goal is therefore to have as few steps as possible to reach the top. This experimental setup is the same as in [23], which in modification to [34] sets a maximal episode length of 500, while model and value function are relearned every 50 steps, uses a discount factor $\gamma = 0.99$ and every episode starts at $x_0 = (\pi/6, 0)$. The parameters of the sparse grid approach were $\varepsilon = 0.01$ and $\eta = 0.002$, while an initial grid of $n = 3$ was used. In Algorithm 3 we used $p = 20$ and $k_{max} = 200$.

In Figure 1 we give the final value function v^{n_e} and the employed adaptive sparse grid $\Omega_{\mathscr{I}(n_e)}$, which consists of 4054 grid points. We observe that, as expected, the majority of the grid points are where the value function has a large gradient. In context of the problem there is a high gradient, where on one side the mountain can be reached directly, on the other side only with first gaining momentum by going in



(a) value function v^{n_e} (b) adaptive sparse grid $\Omega_{\mathscr{I}(n_e)}$

Fig. 1 Results for the mountain car problem.

the opposite direction. The algorithm converged after 5 episodes, with 479 different states visited in total. The goal was reached in each episode as follows:

episode	1	2	3	4	5
steps to goal	159	105	103	104	104

Note that the optimal number of steps for this problem is 103 [23] and the number steps we observe per episode is essentially (only a graph is shown) as in that paper, which uses a full grid. The standard online model-free RL algorithm Sarsa(λ) with tile coding [34] needs many more episodes to get below 150 steps and still is a couple of steps away from 103 even after 1000 episodes [23]. During the course of the Sarsa(λ) algorithm many more states are visited than in our procedure.

6 Conclusion

The combination of the Gaussian processes approach for model learning, which achieves a low sample complexity, with the adaptive sparse grid procedure, which breaks the curse of dimensionality to some extent and allows function representation in higher dimensional state spaces, is applicable for reinforcement learning.

However, due to the lack of monotonicity of the sparse grid approach, the stability of the scheme presently cannot be guaranteed. In practise, we do observe divergent behaviour of the proposed scheme with unfavourable settings of the parameters of the refinement algorithm. Additionally, the initial randomly chosen steps can lead the algorithm off-track, although this can also happen for other reinforcement learning approaches. In such a case the combination of adaptivity and a too wrong model of the state dynamics can lead to divergence, even with otherwise, i.e. for other initial samples, suitable refinement parameters. In any case, theoretical investigations

are needed to provide criteria, in particular useable for the actual computation, under which conditions on the problem and which settings of the algorithm the scheme can successfully be used in reinforcement learning.

Furthermore, a bottleneck in the time complexity of the algorithm is the evaluation of the adaptive sparse grid function, which at this stage prevents us from detailed numerical experiments in higher dimensions, although an adaptive sparse grid would cope with the higher dimensional setting. Recently it was shown that using a specific reordering of the steps of the point evaluations together with a GPU-based parallelisation can achieve speed-ups of almost 50 in comparison to the standard implementation of adaptive sparse grid [22], employing this procedure in our scheme would improve the runtime significantly and allow higher dimensional examples to be finished in reasonable time.

References

1. M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Systems and Control: Foundations and Applications. Birkhäuser, Boston, 1997.
2. G. Barles and E. R. Jakobsen. On the convergence rate of approximation schemes for Hamilton-Jacobi-Bellman equations. *M2AN Math. Model. Numer. Anal.*, 36(1):33–54, 2002.
3. G. Barles and E. R. Jakobsen. Error bounds for monotone approximation schemes for parabolic Hamilton-Jacobi-Bellman equations. *Math. Comp.*, 76(240):1861–1893, 2007.
4. G. Barles and P. Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic Anal.*, 4(3):271–283, 1991.
5. D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming.* Belmont, MA: Athena Scientific., 1996.
6. O. Bokanowski, J. Garcke, M. Griebel, and I. Klompmaker. An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations. *Journal of Scientific Computing*, 55(3):575–605, 2013. also available as INS Preprint No. 1207.
7. J. Bonnans, E. Ottenwaelter, and H. Zidani. A fast algorithm for the two dimensional HJB equation of stochastic control. *M2AN, Math. Model. Numer. Anal.*, 38(4):723–735, 2004.
8. J. F. Bonnans and H. Zidani. Consistency of generalized finite difference schemes for the stochastic HJB equation. *SIAM J. Numer. Anal.*, 41(3):1008–1021, 2003.
9. R. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.
10. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:1–123, 2004.
11. F. Camilli and M. Falcone. An approximation scheme for the optimal control of diffusion processes. *RAIRO, Modélisation Math. Anal. Numér.*, 29(1):97–122, 1995.
12. D. Chapman and L. P. Kaelbling. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of the 12th International Joint Conference on Artific ial Intelligence*, pages 726–731, 1991.
13. M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, Mar. 2009.
14. A.-M. Farahmand, R. Munos, and C. Szepesvári. Error Propagation for Approximate Policy and Value Iteration. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 568–576, 2010.
15. C. Feuersänger. *Sparse Grid Methods for Higher Dimensional Approximation*. Dissertation, Institut für Numerische Simulation, Universität Bonn, Sept. 2010.

16. J. Garcke. Regression with the optimised combination technique. In W. Cohen and A. Moore, editors, *Proceedings of the 23rd ICML '06*, pages 321–328, New York, NY, USA, 2006. ACM Press.
17. J. Garcke. Sparse grids in a nutshell. In J. Garcke and M. Griebel, editors, *Sparse grids and applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*, pages 57–80. Springer, 2013.
18. M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.
19. L. Grüne. An adaptive grid scheme for the discrete Hamilton-Jacobi-Bellman equation. *Numer. Math.*, 75(3):319–337, 1997.
20. L. Grüne. Error estimation and adaptive discretization for the discrete stochastic Hamilton-Jacobi-Bellman equation. *Numer. Math.*, 99(1):85–112, 2004.
21. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.
22. A. Heinecke and D. Pflüger. Multi- and many-core data mining with adaptive sparse grids. In *Proceedings of the 8th ACM International Conference on Computing Frontiers*, CF '11, pages 29:1–29:10, New York, NY, USA, 2011. ACM.
23. T. Jung and P. Stone. Gaussian Processes for Sample Efficient Reinforcement Learning with RMAX-Like Exploration. In J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, editors, *ECML/PKDD 2010 (1)*, volume 6321 of *Lecture Notes in Computer Science*, pages 601–616. Springer, 2010.
24. N. V. Krylov. The rate of convergence of finite-difference approximations for Bellman equations with Lipschitz coefficients. *Appl. Math. Optimization*, 52(3):365–399, 2005.
25. H. Kushner and P. Dupuis. *Numerical methods for stochastic control problems in continuous time (2nd Ed.)*. Number 24 in Applications of Mathematics. Springer, New York, 2001.
26. R. Munos. A study of reinforcement learning in the continuous case by the means of viscosity solutions. *Mach. Learn.*, 40(3):265–299, 2000.
27. R. Munos. Performance bounds in L_p-norm for approximate value iteration. *SIAM Journal on Control and Optimization*, 46(2):541–561, 2007.
28. R. Munos and A. Moore. Variable resolution discretization in optimal control. *Mach. Learn.*, 49(2-3):291–323, 2002.
29. J. Noordmans and P. Hemker. Application of an adaptive sparse grid technique to a model singular perturbation problem. *Computing*, 65:357–378, 2000.
30. S. Pareigis. Adaptive Choice of Grid and Time in Reinforcement Learning. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *NIPS*. MIT Press, 1997.
31. D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München, Aug. 2010. Dissertation.
32. C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
33. S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk SSSR*, 148:1042–1043, 1963. Russian, Engl. Transl.: Soviet Math. Dokl. 4:240–243, 1963.
34. R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
35. A. Tourin. Splitting methods for Hamilton-Jacobi equations. *Numer. Methods Partial Differ. Equations*, 22(2):381–396, 2006.
36. H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.
37. C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990*, volume 31 of *Notes on Num. Fluid Mech.*, pages 241–251. Vieweg-Verlag, 1991.