Machine learning for dynamic incentive problems*

Philipp Renner
Department of Economics
University of Lancaster
p.renner@lancaster.ac.uk

Simon Scheidegger
Department of Finance
HEC Lausanne, University of Lausanne
simon.scheidegger@unil.ch

November 11, 2018

Abstract

We propose a generic computational method for solving large-scale infinite-horizon, discrete-time dynamic incentive problems with hidden states. We first combine set-valued dynamic programming techniques with unsupervised machine learning to determine irregularly shaped feasible sets. Second, we generate training data from those pre-computed feasible sets to recursively solve the dynamic incentive problem by a supervised machine learning algorithm. Third, to speed up the time-to-solution process, we propose a generic parallelization scheme for dynamic incentive problems that allows an efficient use of contemporary high-performance computing hardware. This combination enables us to analyze models of a complexity that was previously considered to be intractable. To demonstrate the broad applicability of our computational framework, we study an insurance-like dynamic adverse selection problem with up to ten different, persistent types. Unlike the previous literature, we allow the agent to overreport his type. We find that the agent has to pay into the insurance for a shorter amount of time. This effect occurs because the agent now has access to the policies of a higher type than his own, which in turn forces the principal to prevent him from overreporting. Moreover, we observe that the overall time the agent has to pay into the insurance until he can file claims is reduced as the number of types increases.

Keywords: Dynamic Contracts, Principal-Agent Model, Dynamic Programming, Machine Learning, Gaussian Processes, High-Performance Computing.

JEL Classification: C61, C73, D82, D86, E61

*We thank Ben Brooks, Johannes Brumm, Rick Evans, Ken Judd, Felix Kubler, Luca Mazzone, John Rust, Karl Schmedders, Tony Smith, Chris Sleet, Aleh Tsyvinski, Sevin Yeltekin, and seminar participants at Yale University, the University of Lausanne, the University of Zurich, EPFL Lausanne, ETH Zurich, CEF 2018 in Milan and LGTC 2018 for their valuable comments. This work was supported by grants from the Swiss National Supercomputing Centre (CSCS) under project IDs s790, s885, and the platform for advanced scientific computing (PASC). Simon Scheidegger gratefully acknowledges support from the Cowles Foundation at Yale University.

1 Introduction

Dynamic incentive problems are of key importance in model-based economics. They occur whenever two parties form a contract with asymmetric information, and encompass questions such as manager remuneration, insurance contracts, and optimal taxation (see, e.g., Golosov et al. (2016) and references therein for a thorough review). Solving these models is a formidable task, as standard recursive techniques (see, e.g., Stokey et al. (1989a)) are often unworkable in these situations. To this end, previous research[1] extensively studied methods of recursively representing incentive-compatible contracts in order to make them formally tractable. Two obstacles arise in these models in general. First, unobservable continuous actions lead to incentive constraints that are themselves optimization problems. In the literature, they are commonly referred to as *moral hazard* problems. Second, in so-called *adverse selection* problems, unobserved, persistent discrete types make it necessary to look at a high-dimensional dynamic program with an unknown domain.

In the work presented below, we focus on solving infinite-horizon, discrete-time dynamic adverse selection models with hidden states and history-depend shocks. Two major bottlenecks create substantial difficulties in solving such problems numerically. The first issue is the determination and approximation of multi-dimensional and (possibly) non-convex equilibrium correspondences—that is, feasible sets of a dynamic program. The second one is performing dynamic programming on them. Thus, we also have to repeatedly and efficiently approximate functions on irregularly shaped domains. To the best of our knowledge, there exists at present no solution framework in the dynamic incentive context that can cope with all these issues at once.

To remedy those shortfalls, the contribution of this paper is four-fold: First, we present a novel, generic solution method for dealing with dynamic incentive problems. In particular, we focus on solving infinite-horizon, discrete-time dynamic incentive models with hidden states and with shocks that follow a Markov chain.[2] Second, we recast a classical model from the adverse selection literature towards more realism by removing a simplifying assumption. Third, in order to speed up the solution process, we introduce a generic hybrid parallelization scheme for value function algorithms with the aim to solve mixed high-dimensional continuous/discrete state dynamic incentive problems. Finally, we solve dynamic adverse selection problems of unprecedented complexity—that is to say, up to ten types.

Finding the feasible set is a standard issue in models with repeated agency, since they require full history dependence (see Lambert (1983) and Rogerson (1985)). This, in turn, leads to time-inconsistent dynamic programs. A way of dealing with this issue is to introduce *promised utilities* as a bookkeeping mechanism, which has led to several recursive formulations for various types of hidden information (see, e.g., Spear and Srivastava (1987), Fernandes and Phelan (2000), and Doepke and Townsend (2006)). Once models with history dependence are written in a recursive form, the set of feasible utility vectors is not known in advance and can be of irregular—that is, non-hypercubic geometry. Performing value function iteration on these sets is subject to the *curse of dimensionality* (Bellman, 1961), since for every individual type of agent, the dimensionality of the state space increases. Hence, if standard, Cartesian grid-based algorithms are applied in the solution process, the computational effort as well as the storage requirements grow exponentially and render even models of only moderate complexity computationally intractable. The existing literature, therefore, is limited to at most two-

[1] For an incomplete list of research in discrete-time settings, see, e.g., Spear and Srivastava (1987), Fernandes and Phelan (2000), Cole and Kocherlakota (2001), Werning (2002), Doepke and Townsend (2006), Abraham and Pavoni (2008), and Pavoni et al. (2017), and for models in continuous time, see, e.g., DeMarzo and Sannikov (2006), Sannikov (2008), Williams (2009, 2011), and He et al. (2017).

[2] Note that our framework is capable of handling more general shock structures. This, however, is beyond the scope of the work presented.

dimensional models (see, e.g., Broer et al. (2017), Doepke and Townsend (2006), and Abraham and Pavoni (2008)), as in them, the curse of dimensionality is avoided from the outset. Being numerically substantially restricted is an unfortunate situation, as several model formulations thus may have to oversimplify real-world situations.

In this paper, we propose to pre-compute the time-invariant feasible set for the continuous state variables by combining ideas from Abreu et al. (1986, 1990) (henceforth APS[3]) with Bayesian Gaussian mixture models (see, e.g., Rasmussen (2000))—a method from unsupervised machine learning.[4] Next, we solve the recursively formulated dynamic incentive problem on an irregularly shaped domain by applying a massively parallelized dynamic programming algorithm that uses Gaussian process regression—a method from supervised machine learning (see, e.g., Rasmussen and Williams (2005))—to approximate high-dimensional value and policy functions on the entire feasible set. This combination enables us to analyze dynamic incentive models that were previously considered to be intractable.

The seminal work by APS introduced a constructive procedure for computing feasible sets. In particular, the authors showed that there exists a monotone set-valued operator whose fixed point is the feasible set, similar to the Bellman operator in dynamic programming. In practical applications, we, therefore, have to repeatedly approximate (potentially non-convex) equilibrium correspondences. To do so, we apply Bayesian Gaussian mixture models to construct a classifier[5] to divide the computational domain into a feasible and an infeasible region. This classification then leads to an outer approximation that iteratively shrinks toward the feasible set. We terminate the iteration once two successive classifiers become sufficiently close. Our approach has several desirable features. First, since sampling achieves the approximation of feasible sets through Bayesian Gaussian mixture models, it does not suffer from the curse of dimensionality (Dasgupta, 1999) and thus can deal with problems involving many types. Second, the numerical approximation of feasible sets is independent of solving the dynamic incentive problem and thus does not directly add to the computational complexity. Third, it can approximate both convex and non-convex sets. Thus, the work presented here is a substantial improvement on the previous literature. Judd et al. (2003) and Yeltekin et al. (2017), for example, provide a numerical scheme for determining the feasible sets of discrete state supergames by using polygons. Their approach, however, relies on the convexification of the payoff set and suffers from the curse of dimensionality. Similarly, Sleet and Yeltekin (2016) provide an extension to this method for the case of continuous state variables, but their extension has the same issues.

Several authors proposed alternative ways for dealing with dynamic incentive problems. Marcet and Marimon (2017) for example look at a planners problem with forward-looking constraints. In particular, they introduce Lagrange multipliers as state variables, which then leads to a recursive saddle point problem. As a result, they can avoid the procedure of finding the feasible set. Pavoni et al. (2017) recently extended the approach by Marcet and Marimon (2017) towards more general constraints. Their state space is the positive orthant. However, both in Pavoni et al. (2017) and Marcet and Marimon (2017), the state space is non-compact. Thus, it is not possible to apply standard numerical dynamic programming techniques because the latter requires a compact domain (Stokey et al., 1989b). Pavan et al. (2014) go a different route by looking at a continuum of types. They use a first-order approach to consider a relaxed problem, i.e., a simplified version of the optimization problem. If the solution to the relaxed

[3]For the remainder of this paper, APS is used to refer both to Abreu et al. (1986, 1990) and to the method introduced by these authors.

[4]For an introduction to machine learning, see, e.g., Murphy (2012) or Goodfellow et al. (2016).

[5]Note that in the machine learning literature, classification can—loosely speaking—be considered to be the problem of identifying to which of a set of categories a new data observation belongs (see, e.g., Murphy (2012) and Goodfellow et al. (2016)).

problem can be computed by hand, the method by Pavan et al. (2014) yields an easy and elegant way to deal with infinitely many types. Their approach, however, requires strong assumptions, e.g., equi-Lipschitz continuity, on the fundamentals of the problem. In case those assumptions are not met, one has to carry out an ex-post verification of the computed solution. The latter, however, cannot guarantee optimality, but instead can only validate feasibility.

After pre-computing the feasible sets, we have to solve the recursively formulated dynamic adverse selection model on the irregularly shaped domains. For this purpose, we propose and apply a massively parallelized discrete-time dynamic programming algorithm that uses Gaussian process regression to approximate the value and policy functions. Gaussian process regression is a form of supervised machine learning that has successfully been applied to a variety of applications in data science, engineering, and other fields to perform approximation and classification tasks. In economics, Scheidegger and Bilionis (2017) recently applied Gaussian processes to solve very-high-dimensional dynamic stochastic growth models as well as to perform uncertainty quantification. A defining feature of Gaussian processes is that they combine the best of two worlds—namely, those of grid-free methods such as Monte Carlo (see, e.g., Press et al. (2007), and references therein) and of function approximation theory. Gaussian processes learn a function based on the observations available at so-called design points, and do so without any geometric restriction. Gaussian processes, therefore, stand in stark contrast to ordinary, grid-based approximation schemes for high-dimensional state spaces such as Smolyak's method (see, e.g., Malin et al. (2010) and Judd et al. (2014)), adaptive sparse grids (see, e.g., Brumm and Scheidegger (2017), Brumm et al. (2017), and Scheidegger et al. (2018)), high-dimensional model reduction (see, e.g., Eftekhari et al. (2017), and references therein), or projection methods (see, e.g., Judd (1992)). Those grid-based approximators are restricted to hyper-rectangular state spaces and thus are not a natural modeling choice in the context of solving dynamic adverse selection models.

Note that the method proposed in this paper has a far broader scope: it can as well be applied, for example, to moral hazard problems or dynamic games, where one of the major difficulties also lies in finding the equilibrium sets (see, e.g., Wang (1995) and Judd et al. (2003)). For discrete actions and lotteries over payoffs, the correspondences are convex valued. However, for other assumptions they are not, which demands a more general approach such as the one proposed in this paper.

To demonstrate the capabilities of the framework proposed in this paper, we solve a dynamic adverse selection model very similar to the one discussed in Fernandes and Phelan (2000). It consists of a risk-averse agent who has unobserved, persistent income shocks, and a risk-neutral planner who wants to provide optimal incentive-compatible insurance against income shocks. The agent reports his income shock to the principal, who then transfers consumption or charges a fee to the agent dependent on the reported shock. Since the principal can only see the reports, the observed shock process is fully history dependent.[6] The reason for this history dependence is that the principal has to condition his actions with respect to the agent's reports and not with respect to the actual shocks.

The original model by Fernandes and Phelan (2000) consists of a low and a high endowed type and does not allow the agent to report a type which is higher than his own. This measure rules out that the agent has to pay more than he can afford, which drastically simplifies the numerical treatment of the problem. However, given the timing of events in the model, the agent is fully aware of the contract. Hence, he would not overreport if that behavior would lead to an infeasible situation. Thus, we recast the model by Fernandes and Phelan (2000) towards more realism and permit the agent to overreport if his means allow it. By doing so, we

[6]Previous research suggests that private information in the economic environments we are interested in is highly persistent (see, e.g., Meghir and Pistaferri (2004) and Storesletten et al. (2004)).

obtain quantitatively very different results compared to the baseline model. Our reformulation introduces additional constraints which in turn have the effect that the principal is forced to offer comparably better insurance terms to the agent. In particular, the agent has to pay for a significantly shorter amount of time until he can file a claim. Hence, to be able to look at policy implications or to take this model to data, it is necessary to deal with a non-simplified version of the problem.

Besides, we solve models of up to ten types, which is—to the best of our knowledge—unprecedented in the literature. In our large-scale models, we choose a Markov chain such that the ergodic distribution is uniform over the types. For models larger than two types, we find that in the intermediately endowed states, the agent pays into an insurance scheme where he has lower levels of coverage compared to the highest state. Moreover, we observe that the time the agent has to pay into the insurance decreases with every additional type we add. This effect occurs because the endowments of the agent vary less drastically over time in comparison to models with fewer types. The agent, in direct consequence, now values insurance less, which in turn forces the principal to offer better contractual terms.

The remainder of this paper is organized as follows: In Section 2, we specify the dynamic incentive environment we are targeting with our framework. In Section 3, we outline our proposed, generic solution method. We first discuss how we construct an APS-style algorithm by using Bayesian Gaussian mixture models; then, we provide a short review Gaussian process machine learning. Finally, we show how to embed APS and Bayesian Gaussian mixture models into a hybrid parallel dynamic programming algorithm that is based on Gaussian process regression. In Section 4, we discuss the performance of our method via a variety of illustrative test cases. Section 5 concludes.

2 A dynamic incentive model

To demonstrate the scalability and flexibility of the method we introduce in this paper, we consider an infinitely repeated, dynamic adverse selection problem in discrete time as described in Fernandes and Phelan (2000).[7]

Time is indexed by $t = 1, 2, \ldots \in \mathbb{N}$. In every period t, an agent observes his[8] taste shock $\theta_t \in \Theta$, where Θ is a finite set of cardinality N. He then reports his shock to the principal. Subsequently, the principal offers a contract c_t, which depends on the entire history of reports, to the agent. Since we need to keep track of the history of types, we define the type history at time t to be the (t)-tuple—that is, an ordered list $h^t = (\theta_0, \theta_1, \ldots, \theta_{t-1})$ for $t \geq 1$. Next, we define the set of possible histories at time t by $H^t = \Theta^t = \Theta \times \Theta \times \ldots \times \Theta$ and set $h^1 \in \Theta$ to be the initial history at time 1. We assume without loss of generality that the initial history is public knowledge. An optimal solution to the problem is a transfer scheme that maximizes the principal's utility while delivering a predetermined lifetime utility to the agent. This scheme will be a sequence of conditional transfers that depend on all past realizations of types. Furthermore, we impose the following assumptions on the primitives of the model:

Assumption 1.

1. *The set of types $\Theta = \{1, \ldots, N\}$ is of cardinality $N \in \mathbb{N}$.*
2. *The set $C \subset \mathbb{R}$ of compensations is a non-empty and compact interval.*
3. *The transition probabilities $\pi(\theta_t|\theta_{t-1})$ are defined by a probability matrix $\Pi \in \mathbb{R}^{N \times N}$.*

[7] We follow the formalism of Golosov et al. (2016) to describe the model.

[8] In this paper, for the sake of brevity, both the agent and the principal are male.

4. *The principal's utility function is given by* $v : C \times \Theta \to \mathbb{R}$ *and the agent's utility function by* $u : C \times \Theta \to \mathbb{R}$*. They are both twice continuously differentiable in* C*.*

5. *The principal and the agent have a common discount factor*[9] $\beta \in (0, 1)$*.*

For each period t and history $h^t = (h_0, \ldots, h_{t-1}) \in H^t$ the probability distribution on the set H^t of histories is given by

$$\pi\left((h^t, i)|h^t\right) = \Pi_{\theta_{t-1},i},\ \forall i \in \{1, \ldots, N\} \tag{1}$$

and by recursion for $\tau \geq 1$,

$$\pi\left(\left(h^{t+\tau}, i\right)|h^t\right) = \Pi_{\theta_{t+\tau-1},i} \cdot \pi\left(h^{t+\tau}|h^t\right), \tag{2}$$

where (h^s, i) generally denotes the history h^{s+1} where i has happened at time s, and h^s before that. The principal's compensation strategy is a function of the history of reports up to period t. His strategy, therefore, is a function $c_t : H^t \to C$. We denote the principal's infinite horizon strategies by the sequences of strategy functions $\mathfrak{c} = (c_1, c_2, \ldots)$.

At time t, the agent has to decide what to report to the principal; his strategy is thus a function on the domain H^t—that is, $a_t : H^t \to \Theta$. A strategy $\mathfrak{a} = (a_t)_{t=1}^{\infty}$ is called incentive compatible iff

$$\mathbb{E}\left[\sum_{t=1}^{\infty} \beta^{t-1} u\left(c_t(h^t, a_t(h^t, X_t)), X_t\right)\middle| h^1\right] \geq \mathbb{E}\left[\sum_{t=1}^{\infty} \beta^{t-1} u\left(c_t(h^t, \tilde{a}_t(h^t, X_t)), X_t\right)\middle| h^1\right],\ \forall \tilde{\mathfrak{a}}, \tag{3}$$

where X_t is a random variable with values in Θ and distribution π, and $\tilde{\mathfrak{a}} = (\tilde{a}_t)_{t=1}^{\infty}$ is a feasible strategy. We can get rid of the reporting strategy $\mathfrak{a}$ of the agent by applying the following theorem:

Theorem 1 (Revelation principle)**.** *(Golosov et al., 2016, Th. 1) For every contract* $\mathfrak{c}$ *and incentive compatible strategy* $\hat{\mathfrak{a}}$*, there is a* $\hat{\mathfrak{c}}$ *with the same payoff to the principal such that truth-telling is incentive compatible.*

The revelation principle allows us to only look at compensation schemes that induce truth-telling and thus to define the *dynamic adverse selection problem*. Given an initial history $h^1 = \{\theta_0\}$, the principal faces the following infinite horizon utility maximization problem:

$$\max_{\mathfrak{c}} \mathbb{E}\left[\sum_{t=1}^{\infty} \beta^{t-1} v\left(c_t(h^t, X_t), X_t\right)\middle| h^1\right] \tag{4}$$

subject to the truth-telling constraint

$$\mathbb{E}\left[\sum_{t=1}^{\infty} \beta^{t-1} u\left(c_t(h^t, X_t), X_t\right)\middle| h^1\right] \geq \mathbb{E}\left[\sum_{t=1}^{\infty} \beta^{t-1} u\left(c_t(h^t, a_t(X_t, h^t)), X_t\right)\middle| h^1\right],\ \forall \mathfrak{a} \tag{5}$$

and the reservation utility vector $\tilde{w}$ yields the constraints

$$\tilde{w}_\theta \leq \mathbb{E}\left[\sum_{t=1}^{\infty} \beta^{t-1} u\left(c_t(h^t, X_t), X_t\right)\middle| h^1\right],\ \forall \theta \in \Theta. \tag{6}$$

As a next step, we simplify the incentive constraints by applying a classic theorem.

[9] This last assumption, while being standard, is not necessary for our computational framework.

Theorem 2 (One-shot deviation principle)**.** *(Golosov et al., 2016, Lem. 3) The truth-telling constraint given by Eq. (5) holds if and only if*

$$\mathbb{E}\left[\sum_{t=1}^{\infty}\beta^{t-1}u\left(c_t(h^t,X_t),X_t\right)\Big|h^1\right]\geq \\ \mathbb{E}\left[u\left(c_s(h^s,a(X_s)),X_s\right)+\sum_{t=1,t\neq s}^{\infty}\beta^{t-1}u\left(c_t(h^t,X_t),X_t\right)\Big|h^1\right],\ \forall s\in\mathbb{N},\forall a:\Theta\to\Theta. \tag{7}$$

Note that in the one-shot deviation constraint (see Eq. (7)), we do not allow for all reporting strategies. Instead, only strategies that remain feasible for the agent are admissible. As a concrete example, assume that an agent overreports his income when filling out his tax declaration. In such a case, he would be charged income tax that he might be unable to pay. All such reports are excluded by Eq. (7).

However, even after all these simplifications the dynamic adverse selection problem is still unsolvable in its present form. To address this, we follow Fernandes and Phelan (2000) and introduce the utility promise w_θ as a continuous state variable.

Theorem 3. *(Golosov et al., 2016, Eqs. (44–47)) There exist $W(\theta)\subset\mathbb{R}^N$, the feasible sets of utility promises for type θ, such that the problem defined by Eqs. 4,6, and 7 can be written recursively as*

$$\begin{aligned} V(w,\theta)=\max_{c,w^+}&\sum_{\tilde\theta\in\Theta}\Pi_{\theta,\tilde\theta}(v(c_{\tilde\theta},\tilde\theta)+\beta V^+(w^+_{\tilde\theta},\tilde\theta)) \\ s.t.\ &w_\nu=\sum_{\tilde\theta\in\Theta}\Pi_{\nu,\tilde\theta}(u(c_{\tilde\theta},\tilde\theta)+\beta w^+_{\tilde\theta,\tilde\theta})\ \forall\nu\in\Theta, \\ &u(c_{\tilde\theta},\tilde\theta)+\beta w^+_{\tilde\theta,\tilde\theta}\geq u(c_\nu,\tilde\theta)+\beta w^+_{\nu,\tilde\theta}\ \forall\tilde\theta\in\Theta,\ \forall\nu\in\Theta\setminus\{\tilde\theta\}, \\ &c\in[0,\overline{c}]^N, \\ &w^+_{\tilde\theta}\in W(\tilde\theta)\ \textit{with}\ w^+_{\tilde\theta,\nu}=(w^+_{\tilde\theta})_\nu\ \forall\tilde\theta\in\Theta,\ \forall\nu\in\Theta. \end{aligned} \tag{8}$$

Note that we arranged w^+ as an $N\times N$ matrix where the θ-th row of the matrix represents next period's utility promise conditional that θ happens. The first constraint is called "promise keeping constraint". It ensures that the principal follows up on his utility promises w^+_θ. The second constraint is the "truth-telling constraint".

This measure allows us at the same time to keep track of the full history and to obtain a recursive formulation. The latter point is critical in making the model tractable for the computational method we propose in Sec. 3. Fig. 1 depicts the timeline for the recursively formulated problem. At period t, the principal "learns" last period's report θ and the utility promises w. He then offers the agent a consumption menu $c\in[0,\overline{c}]^N$ for each possible report, which satisfy the truth-telling and promise-keeping constraints. Next, the agent chooses the truth-telling reporting strategy. Finally, the shock θ_{true} is realized, the agent reports θ_r according to his strategy, and both the principal and the agent receive their respective utilities, $v(c_{\theta_r},\theta_r)$ and $u(c_{\theta_r},\theta_{true})$.

It is not straightforward to see how the recursive problem stated in Eq. (8) relates to the original one (see Eq. (4), Eq. (5), and Eq. (6)). To obtain a solution for Eqs. 4, 5, and 6 from Eq. (8), we select a starting utility promise by maximizing over the recursive value function given the constraint that we provide at least the reservation utility. Starting at this utility promise and an initial history, we can now extract the current consumption transfer as well as the next period's utility promises from the recursive problem. In this way, we can

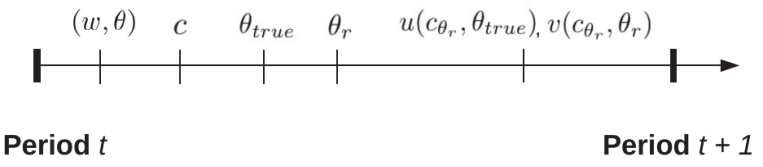

Figure 1: The sequence of events in period t.

iteratively obtain the optimal policies for the non-recursive problem stated in Eq. (4), Eq. (5), and Eq. (6). The detailed procedure is summarized in the following Corollary:

Corollary 1. *Let V be the value function of Eq. (8), $W(\theta)$ its domain, and c^*, w^* the optimal policies. Assume that the initial state is $h^1 = \{\tilde{\theta}\}$. Then the optimal value $\tilde{V}$ of the original problem Eq. (4), Eq. (5), and Eq. (6) is given by*

$$\tilde{V} = \max_{w \in W(\tilde{\theta})} V(w, \tilde{\theta}) \text{ subject to } w_\theta \geq \tilde{w}_\theta \ \forall \theta. \tag{9}$$

In particular we can find an optimal policy by selecting

$$\hat{w} \in \arg \max_{w \in W(\tilde{\theta})} V(w, \tilde{\theta}) \text{ subject to } w_\theta \geq \tilde{w}_\theta \ \forall \theta, \tag{10}$$

and define

$$\begin{aligned} w_1(h^1) &= \hat{w}, \\ c_1(h^1, \theta) &= c^*_\theta(w_1(h^1), \tilde{\theta}), \ \forall \theta, \\ w_t(h^{t-1}, \theta) &= w^*(w_{t-1}(h^{t-1}), \theta), \ \forall \theta, t > 1, \\ c_t((h^{t-1}, \overline{\theta}), \theta) &= c^*_\theta(w_t(h^{t-1}, \overline{\theta}), \overline{\theta}), \ \forall \theta, \end{aligned} \tag{11}$$

where $(h^{t-1}, \overline{\theta}) = h^t$.

3 Dynamic programming on irregularly shaped domains

Solving dynamic adverse selection models as described in Sec. 2 numerically is a formidable task, since we have to deal with a variety of complex issues at the same time: First, we have to numerically determine the feasible sets $W(\theta) \subset \mathbb{R}^N$ of utility promises for the different types θ. In most of the interesting cases, such sets have a non-trivial—that is, a non–hypercubic geometry (see, e.g., Fernandes and Phelan (2000), Broer et al. (2017)). Second, we have to solve a dynamic incentive problem recursively, for example with value function iteration (see, e.g., Judd (1998) and Ljungqvist and Sargent (2000)). Thus, we need to repeatedly approximate and evaluate (potentially) high-dimensional functions at arbitrary coordinates within domains of interest—that is to say, the irregularly shaped feasible sets. To meet all these challenging modeling demands, we apply set-valued dynamic programming techniques in combination with Bayesian Gaussian mixture models—a method from unsupervised machine learning—to determine irregularly shaped equilibrium value correspondences. Subsequently, we use a massively parallel value function iteration algorithm that applies Gaussian process regression—a tool from supervised machine learning—in each iteration step to learn the value function and, if needed, the policy functions on the pre-computed, time-invariant feasible sets.

In this section, we, therefore, proceed in three steps. In Sec. 3.1, we present a novel way of efficiently computing equilibrium correspondences by combining ideas from Abreu et al.

(1986, 1990) with Bayesian Gaussian mixture models (see Rasmussen (2000)). Subsequently, we summarize–in Sec. 3.2–how Gaussian process machine learning can be used to approximate value and policy functions. Sec. 3.3 finally combines all these components into a generic value function iteration framework for adverse selection problems with persistent shocks. Besides, we characterize in Appendix A the general structure of the models we aim to solve by dynamic programming, whereas in Appendix C, we describe its hybrid parallelization.

3.1 On the iterative approximation of irregularly shaped feasible sets

One major complication we are facing in the models under consideration is the fact that the feasible set—that is, the state space for dynamic adverse selection problems is unknown and potentially of irregular geometry. If the problem is simple enough, it can be possible to find an analytical solution (see, e.g., Mailath et al. (2002)). However, we are usually at most able to give an estimate of the hypercubic domain that contains it, namely–

$$\underline{w}_\theta = \min_{c \in C} \frac{u(c,\theta)}{1-\beta}, \quad \overline{w}_\theta = \max_{c \in C} \frac{u(c,\theta)}{1-\beta}, \tag{12}$$

where $\underline{w}_\theta$ and $\overline{w}_\theta$ are the lower and upper bounds on promised utilities, respectively. Hence, in most of the interesting settings, a numerical procedure for approximating the equilibrium correspondences is required.

One possible way of getting around this issue is to relax the recursive model formulation (see Eq. (8)) by introducing slack variables on the constraints. Whenever they are nonzero, one adds a penalty term to the objective function.[10] Since the penalty quickly becomes large outside the original feasible region, the actual solution can be found by restricting the relaxed problem to the feasible set. The advantage of this procedure is that one can apply highly tuned dynamic programming algorithms for hypercubic domains (see, e.g., Brumm and Scheidegger (2017)) to solve dynamic incentive models. The major disadvantage of this brute-force approach, however, lies in the fact that we need to approximate a computational domain of which large parts might be infeasible, which in turn can result—particularly in multi-dimensional settings—in a massive waste of computational resources.

We, therefore, propose a novel method for determining irregularly shaped feasible sets that is based on unsupervised machine learning. This approach will enable us to concentrate the computational resources where they are needed and thus to be highly efficient (cf., Secs. 3.3 and 4). To this end, we follow the classical work by APS, who provide set-valued dynamic programming techniques for determining feasible sets. In particular, they show that the feasible sets $W(\theta)$ can be found by repeatedly applying a set operator $\mathcal{B}$ to an initial "candidate" set until the resulting sequence of sets converges in some metric. In our case (cf. Sec. 2), this means that we can define the set-valued operator as

$$\begin{aligned}\mathcal{B}(S) = \Big\{ w \,|\exists (c, w^+) \text{ with } w_\nu = \sum_{\tilde{\theta}\in\Theta} \Pi_{\nu,\tilde{\theta}} (u(c_{\tilde{\theta}}, \tilde{\theta}) + \beta w^+_{\tilde{\theta},\tilde{\theta}}) \;\forall \nu \in \Theta, \\ u(c_{\tilde{\theta}}, \tilde{\theta}) + \beta w^+_{\tilde{\theta},\tilde{\theta}} \geq u(c_\nu, \tilde{\theta}) + \beta w^+_{\nu,\tilde{\theta}} \;\forall \tilde{\theta} \in \Theta,\; \forall \nu \in \Theta \setminus \{\tilde{\theta}\}, \\ c \in [0, \overline{c}]^N, \\ w^+_{\tilde{\theta}} \in S_{\tilde{\theta}} \text{ with } w^+_{\tilde{\theta},\nu} = (w^+_{\tilde{\theta}})_\nu \;\forall \tilde{\theta} \in \Theta,\; \forall \nu \in \Theta \Big\},\end{aligned} \tag{13}$$

[10] Note that in Appendix B, we will apply this procedure in combination with a highly-tuned adaptive sparse grid code (see Brumm and Scheidegger (2017), Brumm et al. (2015), and Scheidegger et al. (2018)) to verify the method we propose in this paper.

where $S = \prod_{\theta=1}^{N} S_\theta, \quad S_\theta \subset \mathbb{R}^N$, and where the operator $\mathcal{B}(S)$ contains the constraints of the original problem stated in Eq. (8). This is a monotone operator—that is to say, $\mathcal{B}(S) \subset S$, and its fixed point will be the feasible set for type θ—namely $\mathcal{B}(W(\theta)) = W(\theta)$.

A numerical implementation of the APS method requires the repeated approximation of candidate equilibrium value correspondences—that is, a finite collection of sets. To do so, we propose applying Bayesian Gaussian mixture models to construct a classifier to divide the computational domain into a feasible and an infeasible region. Bayesian Gaussian mixture models are usually applied to approximate probability distributions from observed data. Suppose that we have m data samples $\mathbf{X} = \{\mathbf{x}_i : 1 \leq i \leq m\}$. We then can approximate $\rho_{estimated}$ as a mixture of Gaussians:

$$\rho_{estimated}(\mathbf{x}) = \sum_{l=1}^{L} \pi_l \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l), \tag{14}$$

where the mean vectors $\boldsymbol{\mu}_l \in \mathbb{R}^N$, the covariance matrices $\boldsymbol{\Sigma}_l \in \mathbb{R}^{N\times N}$, the weights π_l with $\sum_{l=1}^{L} \pi_l = 1$, and the number of components L are fitted to $\mathbf{X}$ (see, e.g., Rasmussen (2000) and Blei and Jordan (2005)). This classification then leads to an outer approximation that iteratively shrinks toward the feasible set. We terminate the iteration once two successive classifiers become sufficiently close.

In practice, we start the APS iteration by uniformly drawing, for every type, m sample points

$$\mathbf{X}_{test} = \left\{ w^{(1)}, \ldots, w^{(m)} \right\} \tag{15}$$

from a hypercube that contains the feasible set (see Eq. (12)) and fit a Bayesian Gaussian mixture model to the points $\mathbf{X}_{\text{feas}} \subset \mathbf{X}_{\text{test}}$ that were deemed to be feasible. We denote the log-likelihood that corresponds to the Bayesian Gaussian mixture model as f_θ. Then, we start with the set-valued dynamic programming procedure. To do so, we define f_θ^+ as the log-likelihood from the Bayesian Gaussian mixture model in iteration $i-1$, and denote ℓ_θ^+ to be the smallest log-likelihood that corresponded to a feasible point—that is, $\ell_\theta^+ = \min\left\{ f_\theta^+(w^{(j)}) \mid w^{(j)} \in (\mathbf{X}_{\text{feas}})_\theta \right\}$, and generate another m sample points $\tilde{\mathbf{X}}$ from inside the updated feasible set via the respective Bayesian Gaussian mixture models. At a sample point $w^{(k)} \in \tilde{\mathbf{X}}$ that satisfies $f_\theta^+(w^{(k)}) \geq \ell_\theta^+$ within an iteration step i, we transform Eq. (13) into an optimization problem[11], namely–

$$\begin{aligned}
\varphi = \max_{c, w^+} & \sum_{\theta\in\Theta} -\log(\exp(-\alpha(f_\theta^+(w_\theta^+) - \ell_\theta^+)) + 1)/\alpha + N\log(2)/\alpha, \\
\text{s.t. } & w_\nu = \sum_{\tilde{\theta}\in\Theta} \Pi_{\nu,\tilde{\theta}}(u(c_{\tilde{\theta}}, \tilde{\theta}) + \beta w_{\tilde{\theta},\tilde{\theta}}^+) \ \forall \nu \in \Theta, \\
& u(c_{\tilde{\theta}}, \tilde{\theta}) + \beta w_{\tilde{\theta},\tilde{\theta}}^+ \geq u(c_\nu, \tilde{\theta}) + \beta w_{\nu,\tilde{\theta}}^+ \ \forall \tilde{\theta} \in \Theta, \ \forall \nu \in \Theta \setminus \{\tilde{\theta}\}, \\
& c \in [0, \bar{c}]^N, \\
& w_{\tilde{\theta},\nu}^+ = (w_{\tilde{\theta}}^+)_\nu \ \forall \tilde{\theta} \in \Theta, \ \forall \nu \in \Theta.
\end{aligned} \tag{16}$$

The objective function $F(w_\theta^+, \theta, f_\theta^+, \ell_\theta^+)$ of the constrained optimization problem stated in Eq. (16) is a smooth relaxation of a classifier that returns approximately 0 for feasible points and a large negative number for infeasible ones. Furthermore, the constraints are the same ones as those in the description of the operator $\mathcal{B}$ in Eq. (13). The parameter α controls how strict the

[11] In the present form, Eq. (13) is numerically un-tractable due to the constraint $w_{\tilde{\theta}}^+ \in W(\tilde{\theta})$—there is no explicit description of the set $W(\theta)$ that is given by inequalities. To circumvent this issue, we get rid of this constraint by introducing a penalty function p. For any point in the feasible set, this function p will be approximately 0 and will tend to minus infinity when we leave the boundary of the set (see Eq. (16)).

Data: For all θ: initial set of points $\{w^{(1)}, \ldots, w^{(m)}\}$, uniformly drawn from a domain that contains the feasible set (see Eq. (12)). Approximation accuracy $\bar{\epsilon}$.

Result: The approximate feasible sets $W(\theta)$, determined by the log-likelihood f_θ and cut-off ℓ_θ that define the classifier C_θ, where $C_\theta(w^{(j)}) = 1$ iff $f_\theta(w^{(j)}) \geq \ell_\theta$.

Fit Bayesian Gaussian mixture model to $\mathbf{X}_{\text{feas}}$ for all θ to get f_θ and $\ell_\theta = \min\left\{f_\theta(w^{(j)}) \mid w^{(j)} \in (\mathbf{X}_{\text{feas}})_\theta\right\}$.
$f_\theta^+ = f_\theta$.
$\ell_\theta^+ = \ell_\theta$.
while $\epsilon > \bar{\epsilon}$ **do**
  for $\theta \in \Theta$ **do**
    Generate m sample points $\tilde{\mathbf{X}}$ from the most recent "candidate" feasible set via last iteration's Bayesian Gaussian mixture model and set $\mathbf{X} = \{w^{(k)} \in \tilde{\mathbf{X}} : f_\theta^+(x) \geq \ell_\theta^+\}$.
    Set $n = |\mathbf{X}|$.
    Set $(\mathbf{X}_{\text{feas}})_\theta = \emptyset$.
    while $|(\mathbf{X}_{feas})_\theta| < n$ **do**
      Solve optimization problem given by Eq. (16) at point $w^{(k)} \in \mathbf{X}$ and get φ.
      if $\varphi \geq 0$ **then**
        $(\mathbf{X}_{\text{feas}})_\theta = (\mathbf{X}_{\text{feas}})_\theta \cup \{w^{(j)}\}$.
      end
    end
    Fit a Bayesian Gaussian mixture model to $(\mathbf{X}_{\text{feas}})_\theta$ to obtain f_θ and $l_\theta = \min\left\{f_\theta(w) \mid w^{(j)} \in (\mathbf{X}_{\mathbf{feas}})_\theta\right\}$.
  end
  $f_\theta^+ = f_\theta$.
  $\ell_\theta^+ = \ell_\theta$.
  Compute the average L_1-error ϵ.
end

Algorithm 1: Overview of the critical steps for computing the equilibrium correspondences.

relaxation is: the smaller α is, the quicker the objective in Eq. (16) will go to $-\infty$ when leaving the feasible set.[12] If φ is above 0, we label the point $w^{(j)}$ as feasible and add it to a set of feasible points $\mathbf{X}_{\text{feas}}$. We repeat this until the set $\mathbf{X}_{\text{feas}}$ has sufficiently large cardinality.[13] Subsequently, we fit another Bayesian Gaussian mixture model to $\mathbf{X}_{\text{feas}}$ to obtain updated values for f_θ and ℓ_θ. This procedure is repeated until convergence. Alg. 1 summarizes the detailed steps of the set-valued dynamic programming procedure for determining the feasible sets $W(\theta)$ in a more formal way.

Note that it is non-trivial to measure whether the set-valued dynamic programming method has converged. To this end, we propose the following procedure: In every iteration step I and every state θ, we construct a binary classifier $C_{I,\theta}$ by labeling infeasible sample points as 0, and feasible ones by 1, and compute the average L_1-error across subsequent candidate equilibrium correspondences. In practice, this amounts to drawing a uniform sample of points from a hypercube that contains the feasible set. Subsequently, we use the classifiers from steps $I-1$ and I to find the label of each sample point. We then determine the percentage of points that

[12] In practical applications (see Sec. 4), we set $\alpha = 0.1$.
[13] Note that in our computations (see Sec. 4), $|(\mathbf{X}_{\text{feas}})_\theta| \approx 1,000$ led to satisfactory results.

get different labels from the classifiers at $I-1$ and I, resulting in an approximation of the error.

The set-valued dynamic programming approach proposed here has three highly desirable features that yield a substantial improvement over the previous literature. First, since sampling achieves the approximation of feasible sets through Bayesian Gaussian mixture models, it does not suffer from the curse of dimensionality (Dasgupta, 1999) and thus can deal with problems involving many types of agents. Second, our scheme of approximating equilibrium correspondences is independent of solving the dynamic incentive problem and thus does not directly add to the computational complexity of solving the actual model. Finally, it can approximate both convex and non-convex sets.

In contrast, Judd et al. (2003) and Yeltekin et al. (2017), for example, approximate feasible sets by linear hyperplanes. Their approach is limited to models with discrete states and convex sets. Moreover, it suffers from the curse of dimensionality. Thus, it is not useful for models with many types and potentially non-convex, state spaces. Abreu and Sannikov (2014) as well as Abreu et al. (2018) provide methods for computing feasible sets in discrete state, two-player games that are restricted to convex sets and also suffers from the curse of dimensionality. Sleet and Yeltekin (2016) provide an extension to Judd et al. (2003) for the case of continuous state variables, but their method is also restricted to convex sets in low-dimensional state spaces.

Once the set-valued dynamic programming procedure for finding the equilibrium correspondences has converged, we can use the equilibrium Bayesian Gaussian mixture models to generate training data of value functions from within these feasible sets $W(\theta)$ to train the Gaussian processes (see Secs. 3.2 and 3.3). This focuses the computational resources where needed. Note that for the same reasons as in Eq. (16), we have to restate the original recursive problem (see Eq. (8)) and replace the constraints on the future utility promises with a penalty $F(\cdot,\cdot,\cdot,\cdot)$, resulting in

$$
\begin{aligned}
V(w,\theta) = \max_{c,w^+} & \sum_{\tilde{\theta}\in\Theta} \Pi_{\theta,\tilde{\theta}}(v(c_{\tilde{\theta}},\tilde{\theta}) + \beta V^+(w^+_{\tilde{\theta}},\tilde{\theta})) + F(w^+_{\tilde{\theta}},\tilde{\theta},f_{\tilde{\theta}},\ell_{\tilde{\theta}})) \qquad (17)\\
\text{s.t. } & w_\nu = \sum_{\tilde{\theta}\in\Theta} \Pi_{\nu,\tilde{\theta}}(u(c_{\tilde{\theta}},\tilde{\theta}) + \beta w^+_{\tilde{\theta},\tilde{\theta}}) \ \forall \nu\in\Theta,\\
& u(c_{\tilde{\theta}},\tilde{\theta}) + \beta w^+_{\tilde{\theta},\tilde{\theta}} \geq u(c_\nu,\tilde{\theta}) + \beta w^+_{\nu,\tilde{\theta}} \ \forall\tilde{\theta}\in\Theta,\ \forall\nu\in\Theta\setminus\{\tilde{\theta}\},\\
& c\in[0,\overline{c}]^N,\\
& w^+_{\tilde{\theta},\nu} = (w^+_{\tilde{\theta}})_\nu \ \forall\tilde{\theta}\in\Theta,\ \forall\nu\in\Theta,
\end{aligned}
$$

where $F(\cdot,\cdot,\cdot,\cdot)$ is the objective function from the constrained optimization problem stated in Eq. (16). This penalty will be close to zero for $f_\theta \geq \ell_\theta$, but will diverge to $-\infty$ for $f_\theta < \ell_\theta$. Moreover, it is smooth. Hence, we can use it in the optimization problem.

3.2 Gaussian process regression

In the following, we provide a very brief introduction to Gaussian process regression based on Rasmussen and Williams (2005) and Scheidegger and Bilionis (2017), with references therein. Gaussian process regression is a nonparametric regression method from supervised machine learning and addresses the problem of learning input-output mappings from observed data—the so-called training set. Observations can for example stem from a computer code (as in our case) or from empirical experiments. More abstractly, given a data set $\mathcal{D} = \{(\mathbf{x}^{(i)}, t^{(i)}) \,|i = 1,...,\mu\}$ consisting of μ input vectors $\mathbf{x}^{(i)} \in W \subset \mathbb{R}^N$ and corresponding, potentially noisy, observations $t^{(i)} = V(\mathbf{x}^{(i)}) + \epsilon, \epsilon \sim \mathcal{N}(0,\sigma_\epsilon^2)$, we want to deduce a model of the unknown function V that generated the data such that we then can make predictions for new inputs $\mathbf{x}^*$ that we have not

seen in the training set. In the literature, the matrix

$$\mathbf{X} = \left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\mu)}\right\} \tag{18}$$

is commonly referred to as *training inputs*, whereas

$$\mathbf{t} = \left\{t^{(1)}, \ldots, t^{(\mu)}\right\} \tag{19}$$

is the vector of the corresponding *training targets* (observations). To enable predictions based on information contained in $\mathcal{D}$, we must make assumptions about the characteristics of the underlying functions, as Gaussian process regression is a Bayesian regression method. We start by defining a probability measure on the function space, where $V(\cdot)$ lives corresponding to our beliefs. Before seeing any data, we model our state of knowledge about $V(\cdot)$ by assigning a Gaussian process prior to it. We say that $V(\cdot)$ is a Gaussian process with *mean function* $m(\cdot;\boldsymbol{\phi})$ and *covariance function* $k(\cdot,\cdot;\boldsymbol{\phi})$, and write

$$V(\cdot)|\boldsymbol{\phi} \sim \mathrm{GP}(V(\cdot)|m(\cdot;\boldsymbol{\phi}), k(\cdot,\cdot;\boldsymbol{\phi})), \tag{20}$$

where $\boldsymbol{\phi} \in \boldsymbol{\Theta} \subset \mathbb{R}^{d_\theta}$ and $d_\theta \in \mathbb{N}$ are the so-called *hyper-parameters* of the model. The prior beliefs about the response are reflected in our choice of the mean and covariance functions. The prior mean function is required to model any general trends of the response surface and can have any functional form.[14] The covariance function, also known as the *covariance kernel*, is the most crucial part of Gaussian process regression: it defines a measure of similarity on the input space. That is to say, given two input points, their covariance models how close we expect the corresponding outputs to be. A valid choice for a covariance function must be positive semi-definite and symmetric. A very popular covariance function is the *square exponential* (SE)

$$k_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}';\boldsymbol{\phi}) = s^2 \exp\left\{-\frac{1}{2}\sum_{i=1}^{N}\frac{(x_i - x_i')^2}{\ell_i^2}\right\}, \tag{21}$$

where $\boldsymbol{\phi} = \{s, \ell_1, \ldots, \ell_N\}$, with $s > 0$ being the variability of the latent function V, and $\ell_i > 0$ the characteristic lengthscale of the i-th input.[15] We will use the SE kernel in all our numerical experiments below (see Sec. 4).

Given an arbitrary set of training inputs $\mathbf{X}$, Eq. (20) induces a Gaussian prior on the corresponding response outputs:

$$\mathbf{V} = \left\{V\left(\mathbf{x}^{(1)}\right), \ldots, V\left(\mathbf{x}^{(\mu)}\right)\right\}. \tag{22}$$

In particular, $\mathbf{V}$ is distributed as

$$\mathbf{V}|\mathbf{X}, \boldsymbol{\phi} \sim \mathcal{N}\left(\mathbf{V}|\mathbf{m}, \mathbf{K}\right), \tag{23}$$

where $\mathcal{N}\left(\cdot|\mathbf{m}, \mathbf{K}\right)$ is the PDF of a multivariate Gaussian random variable with $\mathbf{m} := \mathbf{m}(\mathbf{X};\boldsymbol{\phi}) \in \mathbb{R}^\mu$ being the mean function evaluated at all points in $\mathbf{X}$, and $\mathbf{K} \in \mathbb{R}^{\mu\times\mu}$ is the covariance matrix with $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)};\boldsymbol{\phi})$ (see, e.g., Eq. (21)).

In the Bayesian framework we are operating in, we have to model explicitly the measurement process that gives rise to the observations $\mathbf{t}$. We do so by assuming that measurements are independent of one another, and that they are normally distributed about $V(\cdot)$ with variance s_n^2:

$$t^{(i)}|V\left(\mathbf{x}^{(i)}\right), s_n \sim \mathcal{N}\left(t^{(i)}\middle|V\left(\mathbf{x}^{(i)}\right), s_n^2\right), \tag{24}$$

[14] We set the prior mean to 0 throughout this paper.

[15] Note that the hyper-parameters of the covariance function are typically estimated by maximizing the likelihood.

where $s_n > 0$ is an additional hyper-parameter that must be determined from the training targets. Using the independence of the observations, we get

$$\mathbf{t}|\mathbf{V}, s_n \sim \mathcal{N}\left(\mathbf{t}\middle|\mathbf{V}, s_n^2\mathbf{I}_N\right). \tag{25}$$

In direct consequence, the *likelihood* of the observations is, given the inputs,

$$\mathbf{t}|\mathbf{X}, \boldsymbol{\phi}, s_n \sim \mathcal{N}\left(\mathbf{t}\middle|\mathbf{m}, \mathbf{K} + s_n^2\mathbf{I}_N\right). \tag{26}$$

Bayes's rule combines the prior Gaussian process (see Eq. (20)) with the likelihood (see Eq. (26)) and yields the *posterior* Gaussian process

$$V(\cdot)|\mathbf{X}, \mathbf{t}, \boldsymbol{\phi}, s_n \sim \mathrm{GP}\left(V(\cdot)\middle|\tilde{m}(\cdot), \tilde{k}(\cdot,\cdot)\right), \tag{27}$$

where the *posterior* mean and covariance functions are given by

$$\tilde{m}(\mathbf{x}) := \tilde{m}(\mathbf{x}; \boldsymbol{\phi}) = m(\mathbf{x}; \boldsymbol{\phi}) + \mathbf{K}(\mathbf{x}, \mathbf{X}; \boldsymbol{\phi})\left(\mathbf{K} + s_n^2\mathbf{I}_N\right)^{-1}(\mathbf{t} - \mathbf{m}) \tag{28}$$

and

$$\begin{aligned} \tilde{k}(\mathbf{x}, \mathbf{x}') &:= \tilde{k}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\phi}, s_n) \\ &= k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\phi}) - \mathbf{K}(\mathbf{x}, \mathbf{X}; \boldsymbol{\phi})\left(\mathbf{K} + s_n^2\mathbf{I}_N\right)^{-1}\mathbf{K}(\mathbf{X}, \mathbf{x}; \boldsymbol{\phi}), \end{aligned} \tag{29}$$

respectively. In order to carry out interpolation tasks when performing value function iteration (see Secs. 3.3 and 4), one has to operate with the predictive (marginal) distribution of the function value $V(\mathbf{x}^*)$ for a single test input $\mathbf{x}^*$ conditional on the hyper-parameters $\boldsymbol{\phi}$ and s_n—namely,

$$V(\mathbf{x}^*)|\mathbf{X}, \mathbf{t}, \boldsymbol{\phi}, s_n \sim \mathcal{N}\left(V(\mathbf{x}^*)|\tilde{m}(\mathbf{x}^*), \tilde{\sigma}(\mathbf{x}^*)\right), \tag{30}$$

where $\tilde{m}(\mathbf{x}^*) = \tilde{m}(\mathbf{x}^*; \boldsymbol{\phi})$ is the *predictive mean* given by Eq. (28), and $\tilde{\sigma}^2(\mathbf{x}^*) := \tilde{k}(\mathbf{x}^*, \mathbf{x}^*; \boldsymbol{\phi}, s_n)$ is the *predictive variance*. The predictive mean can be used as a point-wise surrogate of the response surface—that is, the interpolation value.

Note that the predictive mean $\tilde{m}(\mathbf{x})$ of Eq. (28) can also be written as

$$\tilde{m}(\mathbf{x}) = m(\mathbf{x}) + \sum_{i=1}^{N} a_i k(\mathbf{x}_i, \mathbf{x}), \tag{31}$$

where $\boldsymbol{a} = (a_1, \ldots, a_N) = \left(\mathbf{K} + s_n^2\mathbf{I}_N\right)^{-1}(\mathbf{t} - \mathbf{m})$. We can think of the Gaussian process posterior mean as an approximation of $V(\cdot)$ using N symmetric radial basis functions (RBFs) centered at each observed input. Thus, by choosing a covariance function $k(\mathbf{x}, \mathbf{x}')$ that vanishes when $\mathbf{x}$ and $\mathbf{x}'$ are separated by a lot, for example the SE covariance function (see Eq. (21)), we see that an observed input-output will only affect the approximation locally. This observation also establishes a connection between Gaussian process regression and reproducing-kernel Hilbert spaces, which is discussed in Rasmussen and Williams (2005). Moreover, the radial basis function has universal approximation and regularization capabilities. Theoretically, the RBF can approximate any continuous function arbitrarily well (see, e.g., Poggio and Girosi (1990), Park and Sandberg (1991), and Wu et al. (2012), and references therein).

3.3 A solution algorithm for dynamic incentive problems

Given the generic dynamic programming procedure described in Appendix A, we now outline how to solve a dynamic adverse selection model with persistent shocks recursively (cf. Sec. 2)

by combining the pre-computation of feasible sets (cf. Sec. 3.1) with value function iteration and Gaussian process regression (cf. Sec. 3.2).[16]

The algorithm that we propose for computing the optimal decision rules in dynamic incentive problems proceeds as follows: For every discrete state present in the model (that is, every individual type θ), we first determine the static—that is, time-invariant feasible set $W(\theta)$ (see Sec. 3.1) by Bayesian Gaussian mixture models, from which we can sample training inputs for the Gaussian process regression. Next, we make an initial guess for a value function $V_0(\cdot,\cdot)$ from which we can instantiate the value function iteration procedure. Then, we generate at every iteration step $j+1$ of the value function iteration and for every discrete state θ a set of μ training inputs from within the feasible set $W(\theta)$–namely, $\mathbf{x}_{1:\mu}^{j+1}$, on which we evaluate the Bellman operator (see Eqs. 46 and 17)[17]

$$\mathbf{t}_{1:\mu}^{j+1} = \{\mathbf{t}_1^{j+1}, \ldots, \mathbf{t}_\mu^{j+1}\}, \tag{32}$$

where

$$\mathbf{t}_i^{j+1} = T(V^j)(\mathbf{x}_{1:\mu}^{j+1}, \theta^{j+1}). \tag{33}$$

We use this training data to learn the surrogate of the updated value function for type θ. Note that every individual evaluation of the Bellman operator is carried out by using the predictive mean of $V^j(\cdot,\cdot)$ as the interpolator. The value function iteration procedure is continued until numerical convergence is reached (cf. Eq. (48)). Alg. 2 summarizes the detailed steps of the Gaussian process value function iteration on multi-dimensional, irregularly shaped feasible sets $W(\theta)$ in a more formal way. Note that at convergence, one can not only learn the approximate value functions $V^*(\cdot,\cdot) = V^\tau(\cdot,\cdot)$ but also, if desired, the equilibrium policy functions $\chi^*(\cdot,\cdot)$, using an individual Gaussian process per policy.

Note that one of the defining features of Gaussian process regression is that it is a grid-free method of constructing a surrogate—that is, it allows the modeler to steer the content of the training set closely $\{\mathbf{X}, \mathbf{t}\}$ (see Sec. 3.2) and thus to construct interpolators on irregularly shaped geometries. This has two significant practical advantages when addressing dynamic incentive models numerically. First, if an individual optimization problem at some particular point $\mathbf{x}_i$ does not converge, one does not need to deal with tuning the optimizer until it converges at this location of the state space. Instead, this training input (and the corresponding, nonsensical training target) can be discarded—that is to say, it is not added to the training set. This is in stark contrast to grid-based methods such as "Smolyak" (see, e.g., Krueger and Kubler (2004) and Judd et al. (2014)) or "adaptive sparse grids" (see, e.g., Brumm and Scheidegger (2017), Brumm et al. (2015, 2017), and Scheidegger and Treccani (2018)), where the construction of the surrogate breaks down if not every optimization problem required by the algorithm can be solved. Second, computing solutions solely on a domain of relevance—that is, $W(\theta)$, allows one to carry out value function iteration on complex, high-dimensional geometries without suffering from massive inefficiencies, as the computational resources are concentrated where needed. Particularly in high-dimensional settings, this can potentially speed up the time-to-solution process by orders of magnitude, as the feasible set might have a negligibly small volume compared to the computational domain that standard approximation methods require. Finally, note that to solve large-scale problems in a reasonably short time, we make use of hybrid parallel computation. For the details of the generic parallelization scheme we propose and apply in this paper, see Appendix C.

[16] The value function iteration algorithm proposed in this section is loosely related to the one proposed by Keane and Wolpin, 1994, who use Monte Carlo simulations and linear regression to solve and estimate discrete choice dynamic programming problems, a setting that substantially differs from the one we are targeting here.

[17] At every single training point, the individual optimization problems are solved with `Ipopt` (see Waechter and Biegler (2006)), which is a high-quality open-source software for solving nonlinear programs (`http://www.coin-or.org/Ipopt/`).

Data: Initial guess $V_0(\cdot,\cdot)$ for the value function of every type θ. Approximation accuracy $\bar{\epsilon}$.

Result: The M (approximate) equilibrium policy functions $\chi^*(\cdot,\cdot)$ and the corresponding value functions $V^*(\cdot,\cdot)$ for every type $\theta \in \mathbf{\Theta}$, where $|\mathbf{\Theta}| = N$.

Determine for every type θ the respective feasible set $W(\theta)$.
Set iteration step $j = 0$.
while $\epsilon > \bar{\epsilon}$ **do**
  for $\theta^{j+1} \in \mathbf{\Theta}$ **do**
    Generate μ training inputs $\mathbf{X} = \{\mathbf{x}_i^{j+1} : 1 \leq i \leq \mu\} \in W(\theta)$.
    for $\mathbf{x}_i^{j+1} \in \mathbf{X}$ **do**
      Evaluate the Bellman operator $T(V^j)(\mathbf{x}_i^{j+1}, \theta^{j+1})$ (see. Eq. (17)).
      Set the training targets for the value function: $t_i = T(V^j)(\mathbf{x}_i^{j+1}, \theta^{j+1})$.
    end
    Set $\mathbf{t} = \{t_i : 1 \leq i \leq \mu\}$.
    Given $\{\mathbf{X}, \mathbf{t}\}$, learn the surrogate $V^{j+1}(\cdot, \theta^{j+1})$.
    Compute an error measure, e.g.: $\epsilon_\theta = \|V^{j+1}(\cdot, \theta^{j+1}) - V^j(\cdot, \theta^{j+1})\|_2$.
  end
  Set $j = j + 1$. $\epsilon = \max(\epsilon_{\theta_1}, ..., \epsilon_{\theta_D})$
end
$\tau = j - 1$
$V^*(\cdot,\cdot) = V^\tau(\cdot,\cdot)$.
$\chi^*(\cdot,\cdot) = \{\chi_1(\cdot,\cdot), \cdots, \chi_M(\cdot,\cdot)\}$.

Algorithm 2: Overview of the critical steps of the value function iteration algorithm that operates on equilibrium correspondences $W(\cdot)$.

4 Numerical experiments

To demonstrate the broad applicability and versatility of the computational framework introduced in this paper, we solve three distinct versions of the dynamic adverse selection environment outlined in Sec. 2. In Sec. 4.1, we solve the model by Fernandes and Phelan (2000) as a basic verification test for our method. In Sec. 4.2, we generalize their model towards more realism by allowing for a broader range of reporting strategies. Third, we solve in Sec. 4.3 the model introduced in Sec. 4.2 for increasingly many types θ to show that we can handle dynamic adverse selection problems with state spaces of irregular geometry and at least ten dimensions.

4.1 Baseline model

To gain a systematic understanding of how our computational framework behaves in real applications, we apply it to a privately observed endowment model by Fernandes and Phelan (2000), for which numerical solutions are known. First, we briefly summarize the model and its parameterization. Second, we report on the performance of our proposed set-valued dynamic programming method. Finally, we discuss the solution to the dynamic incentive problem that was obtained by performing value function iteration on the pre-computed, irregularly shaped state spaces.

We consider an environment that consists of a risk-neutral principal who *minimizes* his

Parameter	Value
β	0.9
h_H	0.35
h_L	0.1
$[\underline{c}, \overline{c}]$	$[0, 1]$

Table 1: Parameterization of the privately observed endowment model by Fernandes and Phelan (2000).

cost, and a risk-averse agent. We choose the agent's utility over consumption c to be

$$u(c) = \sqrt{c}, \tag{34}$$

where c is restricted to the finite range $[\underline{c}, \overline{c}]$.[18] There are two types in the model—namely a "low" (state 1; also denoted as L) and a "high" (state 2; also denoted as H) one. The respective endowments are given by h_L and h_H. The agent "learns" his private type in each period and then reports it to the principal. Subsequently, the principal transfers consumption to agent dependent on what the agent reported (see Fig. 1). Since the problem depends on the full history of reports, we use the recursive reformulation that we introduced in Theorem 3. Concretely, we look at the following problem:

$$\begin{aligned} V(w, \theta) = \min_{\tau, w^+} & \sum_{\tilde{\theta} \in \{L,H\}} \Pi_{\theta,\tilde{\theta}} (\tau_{\tilde{\theta}} + \beta V^+(w^+_{\tilde{\theta}}, \tilde{\theta})) \\ \text{s.t. } & w_\nu = \sum_{\tilde{\theta} \in \{L,H\}} \Pi_{\nu,\tilde{\theta}} (\sqrt{\tau_{\tilde{\theta}} + \tilde{\theta}} + \beta w^+_{\tilde{\theta},\tilde{\theta}}) \ \forall \nu \in \{L, H\}, \\ & \sqrt{\tau_H + h_H} + \beta w^+_{H,H} \geq \sqrt{\tau_L + h_H} + \beta w^+_{L,H}, \\ & \sqrt{\tau_L + h_L} + \beta w^+_{L,L} \geq \sqrt{\tau_H + h_L} + \beta w^+_{H,L}, \\ & -h_L \leq \tau_L \leq \overline{c}, \ -h_H \leq \tau_H \leq \overline{c}, \\ & w^+_{\tilde{\theta}} \in W(\tilde{\theta}) \text{ with } w^+_{\tilde{\theta},\nu} = (w^+_{\tilde{\theta}})_\nu \ \forall \tilde{\theta} \in \{L, H\}, \ \forall \nu \in \{L, H\}, \end{aligned} \tag{35}$$

where $\tau_{i \in \{L,H\}}$ denotes the transfer from the principal to the agent. Furthermore, we follow Fernandes and Phelan (2000) for now and assume that the agent cannot claim to be a higher type than he is. In particular, if the principal charges more than the lower type can afford, then we have to prevent the agent from overreporting his type. This is usually done in the literature because the agent only has h_L level of endowment in the low state, but the transfer τ_H can be less than $-h_L$ if the principal thinks he deals with the high type agent. Therefore, the principal might charge a fee that is higher than what the agent can afford.

The resulting state space of promised utilities in this model is 2-dimensional. The Markov process governing the endowments in the model specification here is such that the agent has a 90 percent chance of receiving the endowment he received in the previous period. This yields the following transition probabilities across the different types:

$$\Pi_{\theta,\tilde{\theta}} = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}. \tag{36}$$

The remaining parametrization is reported in Tab. 1.

In Fig. 2, we display approximations of the equilibrium value correspondences for the low state at various iteration steps when performing set-valued dynamic programming with

[18] Note that our computational framework can easily deal with preferences other than then ones used here.

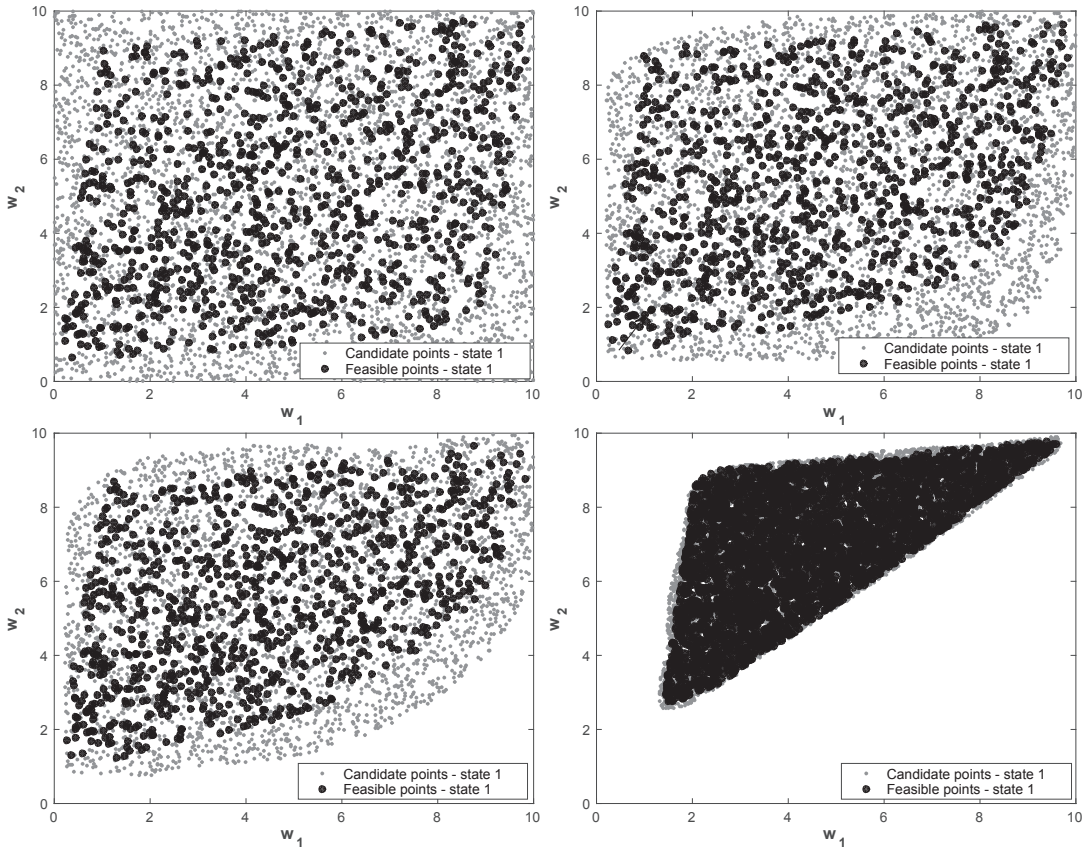

Figure 2: The above four panels display the convergence of the (candidate) feasible set for state 1 as a function of w_1 and w_2 (see Alg. 1). The top-left panel shows the first step in the set-valued dynamic programming procedure; the top-right shows step 2. The lower-left panel is a result from step 15, and the lower-right is based on iteration 35.

Bayesian Gaussian mixture models (see Alg. 1). In particular, we show the candidate points that were generated from within the current approximation of the feasible set as well as the sample points that are deemed feasible in the respective APS iteration step. This sequence of figures indicates that our proposed method—that is, to merge the set-valued dynamic programming with Bayesian Gaussian mixture models leads to converging results. In Fig. 3, we show the feasible sets for both the low and the high state once the set-valued dynamic programming iteration has converged—that is, after about 50 iteration steps, where the approximation error for the sets reaches $\mathcal{O}(10^{-3})$ percent.[19]

Having determined the equilibrium value correspondences for the two states, we turn our attention now to the recursive solution of the model. Fig. 4 depicts the decreasing, aggregated L_1- and L_2-errors for the value functions when performing value function iteration with Gaussian processes on the feasible sets (see Alg. 2). The graph indicates that our proposed solution framework—that is, the combination of APS, Bayesian Gaussian mixture models, and Gaussian process regression—can successfully solve dynamic incentive problems.[20] In Fig. 5, we show the value functions for the low and high states at convergence.

To gain some economic insights into the optimal contracts computed, we now carry out a

[19] We cross-validate the findings here by an alternative method that is based on adaptive sparse grids. The affirmative results are summarized in Appendix B.

[20] In this application, generating 100 observations per state and iteration step from the approximate equilibrium correspondences to train the Gaussian processes led to satisfactory results.

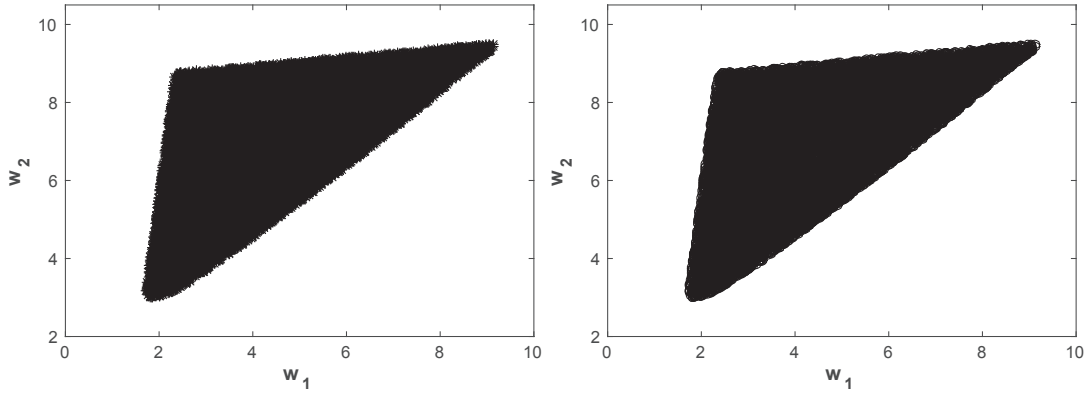

Figure 3: The left panel shows the approximate feasible set for the low state at convergence as a function of w_1 and w_2. The right panel shows the respective set for the high state.

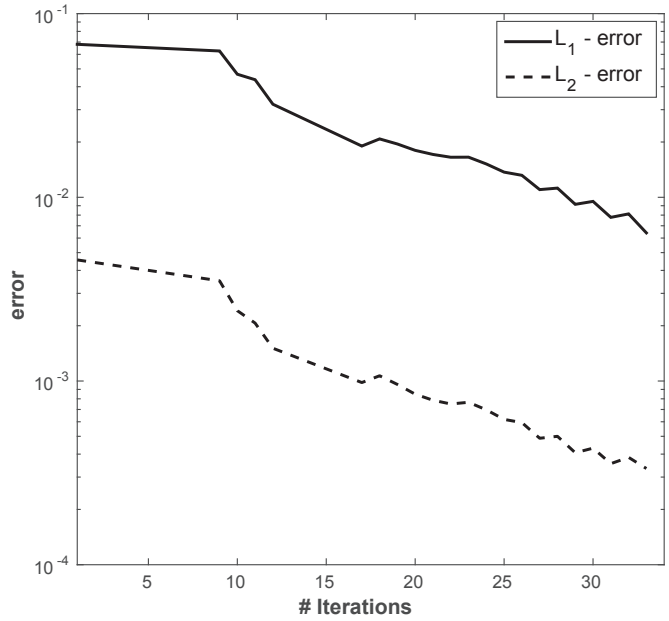

Figure 4: Aggregated, decreasing L_1- and L_2-errors for the 2-dimensional dynamic incentive model that was solved by training Gaussian process interpolators with 100 observations that were generated from within the feasible sets. The errors were computed by randomly drawing $1,000$ points from the feasible sets.

collection of simulation experiments. In the left panel of Fig. 6, we show simulation patterns for the principal's optimal value in the low (V_1) and high (V_2) states. The simulation was launched from the optimal value in the promised utility space—that is, at $w_k^* \in \text{argmin} V_{k\in\{1,2\}}$. Furthermore, we kept the shocks constant in the respective state. We find that in the high state, the optimal value is increasing, whereas in the low state, the value remains constant. Analogously, the right panel of Fig. 6 displays the utility promise in state i to type i.

The left and right panels of Fig. 7 display the corresponding utility transfers τ_1 and τ_2 with the same shock pattern as in Fig. 6. In the left panel, for example, the agent is in state 1, and τ_1 represents the transfer if he reports being in state 1, whereas τ_2 is the transfer if state 2 is reported. For state 1, we can see that the transfer the principal charges from a truthfully reporting agent is the maximum amount within the constraints given (cf. Eq. (12)). τ_2 has no

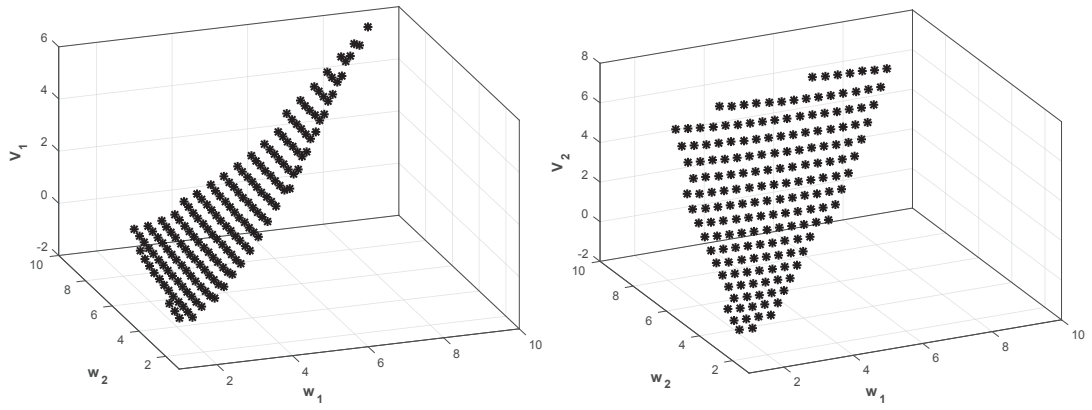

Figure 5: Left panel—value function of the cost-minimization problem in the low state (V_1) as a function of w_1 and w_2. Right panel—value function of the high state (V_2) as a function of w_1 and w_2. In both panels, we evaluated the respective value functions at 180 equally spaced points for illustrative purposes.

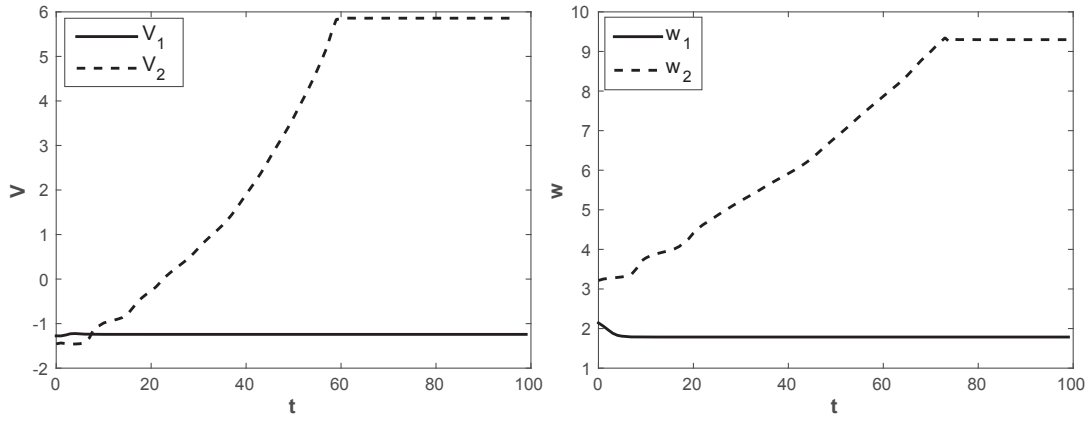

Figure 6: Left panel—optimal value of the cost-minimization problem at simulation step t in states 1 (V_1) and 2 (V_2). Right panel—promised utility in the low (w_1) and high states (w_2) as a function of the simulation step t.

meaning for the low type here in this subsection since he is prohibited from overreporting. The right panel of Fig. 7 shows what would happen if one was always in the high state: At first, the principal imposes a negative transfer for either of the reports—that is, the agent has to pay. After about 35 periods, both the potential transfer for a low type and the high type become positive, i.e., the principal gives consumption to the agent. In other words, the agent pays into the insurance, and after a sufficiently long time period receives a promise of payment in case he drops into the low state. Furthermore, if the agent pays in long enough into the insurance, then the principal transfers consumption back to the agent by an increasing amount, regardless of the state.

The left and right panels of Fig. 8 depict the simulation paths within the respective feasible sets. In the low state, we almost start at the steady state, whereas in the high state, we traverse almost the entire set.

Note that the transfer policies in Fig. 7 are not monotone. However, if we look at the pattern in the feasible set (see Fig. 8), we see that the utility promises are monotone—that is, the promise decreases in state 1 and increases in state 2. Hence, the principal has three ways

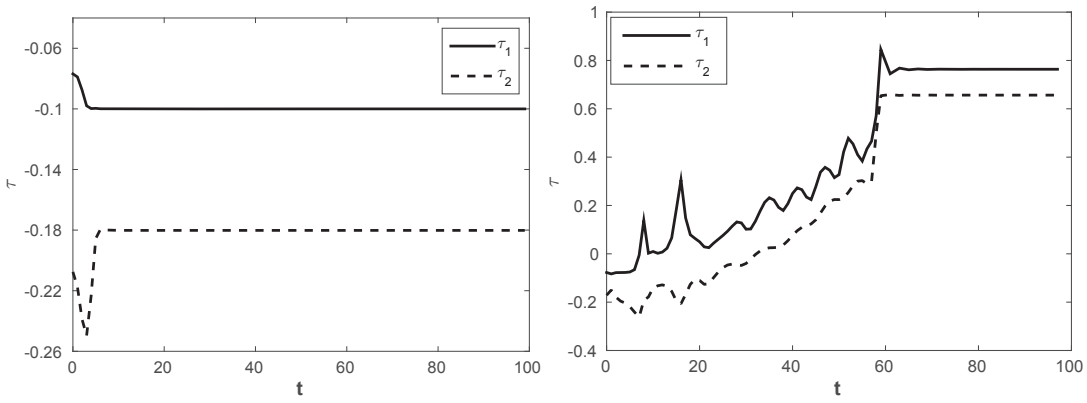

Figure 7: Left panel—transfer policy τ_1, τ_2 while constant state 1 (low state); Right panel—transfer policy τ_1, τ_2 while constant state 2 (high state).

to fulfill his promise to the agent, namely by a utility transfer and by promises of future utility for either state.

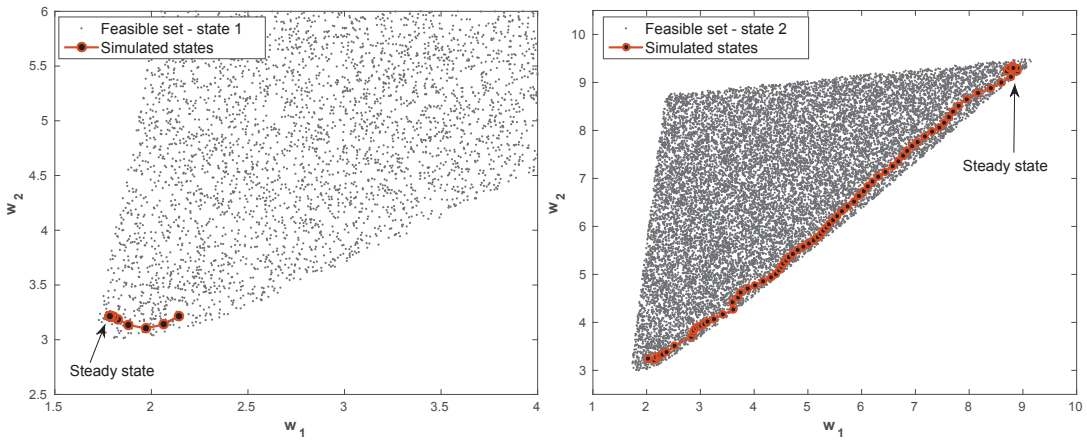

Figure 8: Left panel—simulation path in the feasible set for the low state, state 1. Right panel—the same, for the high state, state 2.

In summary, we note that the results shown in this section replicate those reported in Fernandes and Phelan (2000).

4.2 Generalizing Fernandes and Phelan (2000)

Fernandes and Phelan (2000) assume that the agent cannot report to be of high type if he is in truth of low type to ensure that their model is well defined. Otherwise, the principal might charge the agent a premium he cannot afford. However, this is a gross simplification of the economic problem. Technically, the principal announces the transfer policy before the agent signs the contract (cf. Fig. 1). Thus, the agent has full knowledge of whether a certain report would be infeasible for him or not. In this section, we, therefore, relax the assumptions by Fernandes and Phelan (2000) towards a more general setting, but keep the parameterization as before (cf. Tab. 1).

The basic idea to generalize the model by Fernandes and Phelan (2000) is to introduce two new variables, τ_H^+, τ_H^-, and additional constraints that replace one of the incentive constraints

in Eq. (35). The new variable τ_H^- deals with the part of the transfer that would be infeasible for a low type agent, and τ_H^+ deals with the part that is feasible. Thus, if τ_H is infeasible for the low type, then τ_H^+ is zero, and if τ_H^+ is nonzero, then τ_H^- is equal to $h_H - h_L$.[21] Taken together, these modifications lead to the following complementarity problem:

$$\begin{aligned}(\sqrt{\tau_L + h_L} + \beta w_{L,L}^+)\tau_H^+ &\geq (\sqrt{\tau_H^+} + \beta w_{H,L}^+)\tau_H^+, \\ (h_H - h_L - \tau_H^-)\tau_H^+ &= 0, \\ \tau_H^- + \tau_H^+ &= h_H + \tau_H, \\ 0 \leq \tau_H^- &\leq h_H - h_L, \\ 0 \leq \tau_H^+.&\end{aligned} \tag{37}$$

To clarify how these changes work, we look at few situations. First, suppose that $\tau_H < -h_L$ holds. Then, the agent cannot afford to overreport. In this case, $\tau_H^- < h_H - h_L$ and $\tau_H^+ = 0$ hold because of the second constraint in Eq. (37). Therefore, the first constraint in Eq. (37) becomes irrelevant. Second, if $\tau_H \geq -h_L$ holds, then $\tau_H^- = h_H - h_L$ and $\tau_H^+ \geq 0$ is true. Plugging the latter statements into the third constraint of Eq. (37) and solving for τ_H^+ leads to $\tau_H^+ = h_L + \tau_H$—that is, the consumption for a misreporting agent of low type. Since $\tau_H^+ > 0$, it cancels out and the first constraint is just the standard incentive constraint.

The introduced changes yield the following optimization problem with complementary constraints:

$$\begin{aligned}V(w,\theta) = \min_{\tau,\tau_H^+,\tau_H^-,w^+} & \sum_{\tilde{\theta}\in\{L,H\}} \Pi_{\theta,\tilde{\theta}}(\tau_{\tilde{\theta}} + \beta V^+(w_{\tilde{\theta}}^+, \tilde{\theta})) \\ \text{s.t. } w_\nu = & \sum_{\tilde{\theta}\in\{L,H\}} \Pi_{\nu,\tilde{\theta}}(\sqrt{\tau_{\tilde{\theta}} + \tilde{\theta}} + \beta w_{\tilde{\theta},\tilde{\theta}}^+)\ \forall \nu \in \{L,H\}, \\ & \sqrt{\tau_H + h_H} + \beta w_{H,H}^+ \geq \sqrt{\tau_L + h_H} + \beta w_{L,H}^+, \\ & (\sqrt{\tau_L + h_L} + \beta w_{L,L}^+)\tau_H^+ \geq (\sqrt{\tau_H^+} + \beta w_{H,L}^+)\tau_H^+, \\ & (h_H - h_L - \tau_H^-)\tau_H^+ = 0, \\ & \tau_H^- + \tau_H^+ = h_H + \tau_H, \\ & -h_L \leq \tau_L \leq \bar{c},\ -h_H \leq \tau_H \leq \bar{c},\ 0 \leq \tau_H^- \leq h_H - h_L, 0 \leq \tau_H^+ \\ & w_{\tilde{\theta}}^+ \in W(\tilde{\theta}) \text{ with } w_{\tilde{\theta},\nu}^+ = (w_{\tilde{\theta}}^+)_\nu\ \forall \tilde{\theta} \in \{L,H\},\ \forall \nu \in \{L,H\}.\end{aligned} \tag{38}$$

In Fig. 9, we display approximations of the equilibrium value correspondences for the low state at various iteration steps when performing set-valued dynamic programming with Bayesian Gaussian mixture models. As in the previous section, we show the candidate points that were generated from within the current approximation of the feasible set as well as the sample points that were deemed feasible in the respective APS iteration step. As before, this sequence of figures indicates that our proposed computational method leads to nicely-converging results. However, comparing the feasible sets from the original problem by Fernandes and Phelan (2000) (see Fig. 2) to the ones display here in Fig. 9 yields substantial differences. In the generalized model formulation presented in this section, we find substantially "smaller" sets. This finding is due to the fact that we added several additional constraints to the original model (cf. Eq. (35) and Eq. (38)).

To gain insights into the optimal contracts computed in the generalized model presented in this section, we carry out again a collection of numerical experiments. In the left panel of

[21] The procedure outlined here is a technique commonly applied in linear programming (see, e.g., Chvatal (2016)). There, one splits a variable x in a positive and a negative part—that is, $x = x^+ - x^-$.

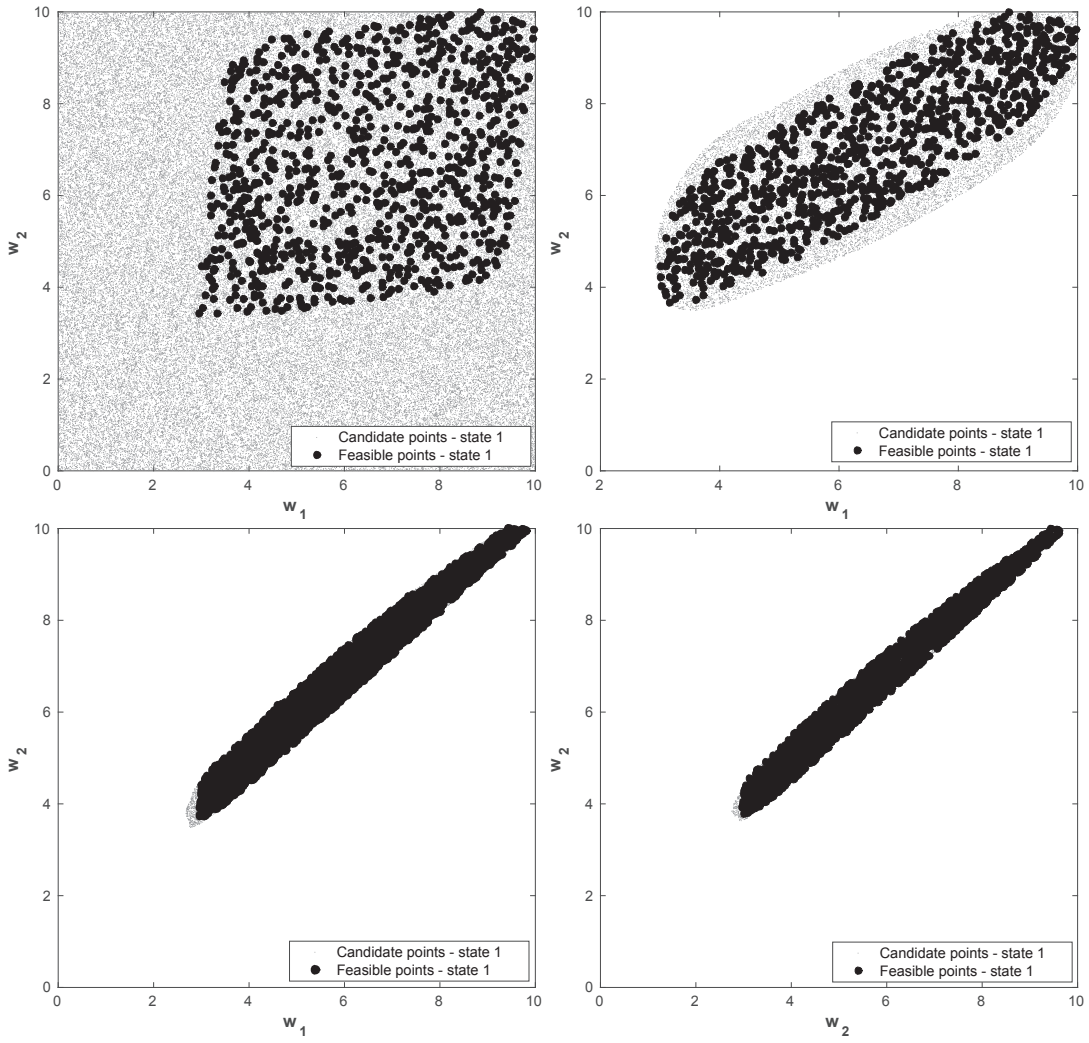

Figure 9: The above four panels display the convergence of the (candidate) feasible set for state 1 as a function of w_1 and w_2 (see Alg. 1). The top-left panel shows the first step in the set-valued dynamic programming procedure; the top-right shows step 5. The lower-left panel is a result from step 15, and the lower-right is based on iteration 20.

Fig. 10, we depict simulation patterns for the principal's optimal value in the low (V_1) and high (V_2) states. The simulation was launched from the optimal value in the promised utility space. As in the previous section, we kept the shocks constant in the respective state. We find that in the high state, the optimal value is increasing, while in the low state, the value remains constant. This is the same trend as in the original model. Analogously, the right panel of Fig. 10 displays the utility promise in state i to type i.

In analogy to Fig. 8, the left and right panels of Fig. 11 depict the simulation patterns from within the feasible sets of the low and high states. The simulations were launched from the optimal value in the promised utility space. After that, we kept the shocks constant in the respective state. Similarly to the findings in the original model (see Sec. 4.1), the simulation barley moves away from the optimal value in the low state but traverses the entire set in the high state.

In Fig. 12, we depict impulse–response experiments on the equilibrium policies. In particular, starting from the steady state, we induce the complementary state and then return to

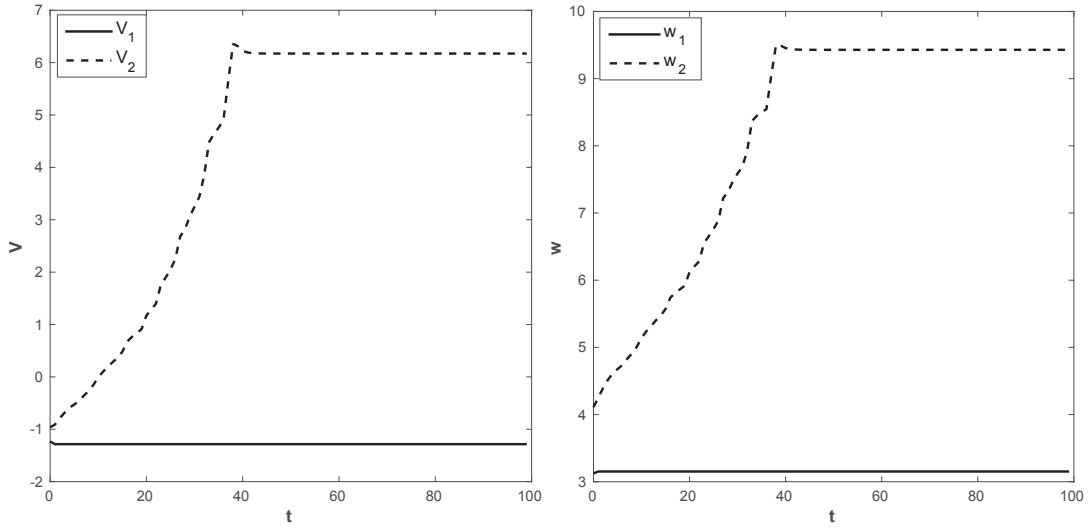

Figure 10: Left panel—optimal value of the cost-minimization problem at simulation step t in states 1 (V_1) and 2 (V_2). Right panel—promised utility in the low (w_1) and high states (w_2) as a function of the simulation step t.

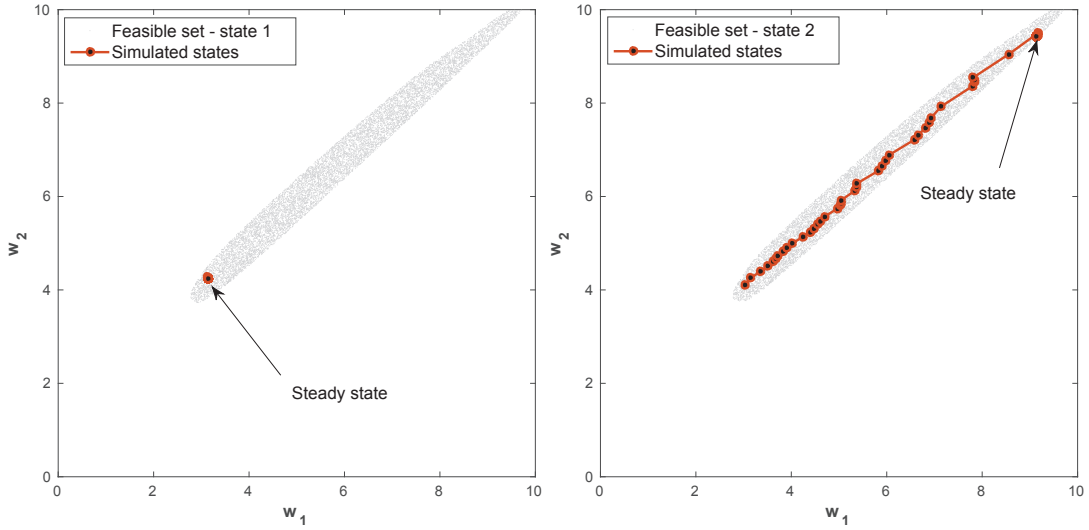

Figure 11: Left panel—simulation path in the feasible set for the low state, state 1. Right panel—the same, for the high state, state 2.

the original one. We find that for both states, the utility promises return after a few steps back to the steady state. If we now look again at the corresponding transfer payments (see Fig. 13), we see that the overall behavior very similar to the one found in the previous section (cf. Fig. 7): In the long run in state 1—the low state—the principal charges the agent a fee. In the second state (the high state), the principal at first imposes a negative transfer—that is, the agent pays part of his income to the principal. After about ten steps, he again pays the agent if he falls into the low state. Moreover, after about 18 simulation steps, the principal transfers consumption to the agent even in the high state.

There is a notable difference to the original model by Fernandes and Phelan (2000) when it comes to the magnitude of the payments: In the low state, the negative transfer to the agent is about half of the size compared to one found in the original, non-generalized model.

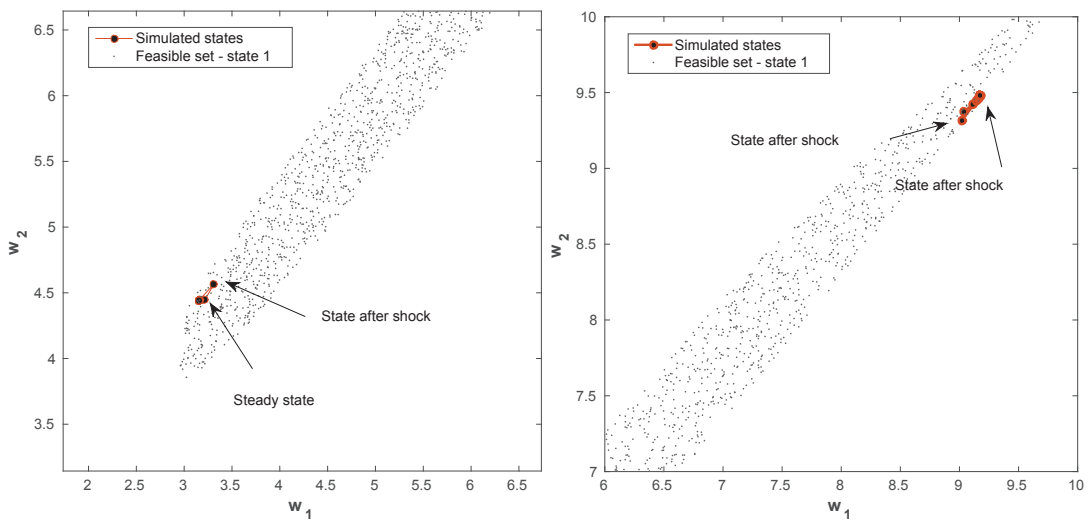

Figure 12: Path of the utility promises after a shock to the steady state has occurred, depicted in the left panel for state 1 and in the right panel for state 2. In both panels, the steady state, as well as the utility promises right after the shock, are labeled by arrows.

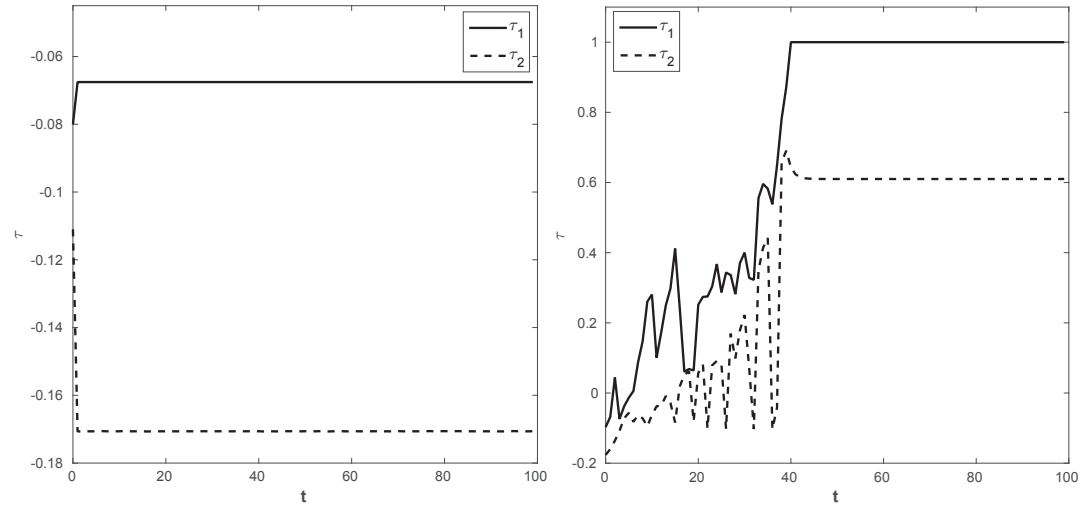

Figure 13: Left panel – Transfer policy τ_1, τ_2 while constant state 1 (low state); Right panel – Transfer policy τ_1, τ_2 while constant state 2 (high state).

Furthermore, in the high state, the amount of time the agent needs to pay into the insurance is shorter by about twenty simulation steps, and with a payout if he should fall into the low state (cf. Figures 7 and 13). Note that our discount factor being 0.9 is quite low (see Tab. 1), and therefore these time differences are even more significant.

We conduct now again an impulse–response experiment. When looking at the simulations in Fig. 14, we can see that the agent pays a premium to the principal in the high state. The promised utility to the agent rises monotonically while being in the high state. If a low shock hits the agent, and he has paid into the insurance for around twenty steps (see Fig. 14), the principal gives a transfer to the agent. As long as the agent is in the low state, the promised utility decreases, and with it, also the transfer decreases. Please note that even in the high state, the principal starts to pay the agent money back, even after he has paid into the insurance

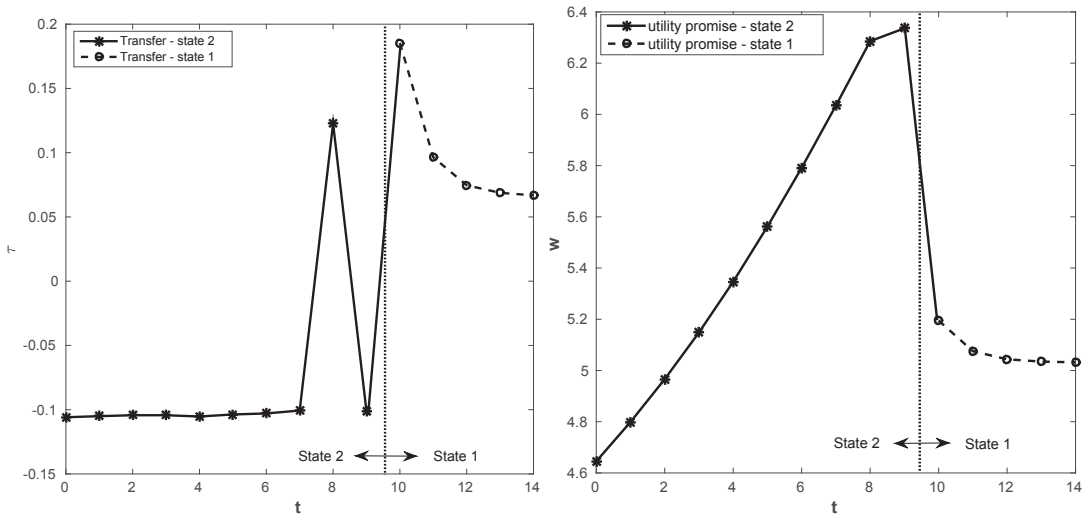

Figure 14: The left panel shows the consumption transfer to the agent while he is in the higher state 2, and then after he has dropped into the lower state 1. The right panel shows the utility the agent receives in this situation.

for twenty time steps, as one can see in Fig. 14 by the spike in transfer just before we switch states.

The behavior reported here replicates the qualitative behavior of an insurance contract: The agent pays into the insurance for a sufficiently long period until he can file a claim in case of an emergency. Note that the level of the insurance payoff depends on how long he has paid the premium. Besides, if the agent does not make a claim for a while, then the insurance will pay back money, even if a low shock does not hit the agent.

4.3 Many types

In this section, we demonstrate the dimensional scalability of our method. To this end, we expand the two-dimensional model outlined in Sec. 4.2 first to three types—a "low" (state 1), a "middle" (state 2), and a "high" (state 3) one. An agent's endowment is either $h_{low} = 0.1, h_{middle} = 0.2$, or $h_{high} = 0.35$, and the discount factor is again set to $\beta = 0.9$ (cf. Tab. 1). Moreover, we choose the transition probabilities across the different types as

$$\Pi_{\theta,\tilde{\theta}} = \begin{bmatrix} 0.9 & 0.1 & 0.0 \\ 0.1 & 0.8 & 0.1 \\ 0.0 & 0.1 & 0.9 \end{bmatrix}. \tag{39}$$

The choice of the Markov chain given by Eq. (39) follows a similar pattern as the one from the original problem (see Sec. 4.1) and thus allows some comparison across different dimensions.

In Fig. 15, we display approximations of the equilibrium value correspondences for the low state, state 1 as a function of w_1, w_2, and w_3 at various iteration steps when performing set-valued dynamic programming with Bayesian Gaussian mixture models. We show the candidate points that were generated from within the current approximation of the feasible set as well as the sample points that were deemed feasible in the respective APS iteration step. This sequence of figures clearly indicates that our proposed computational method also leads to nicely-converging results in models with more than two agents.

As the next step, we simulate the optimal policies. In line with Secs. 4.1 and 4.2, we launch the simulations from the optimal value in the promised utility space and keep the shocks

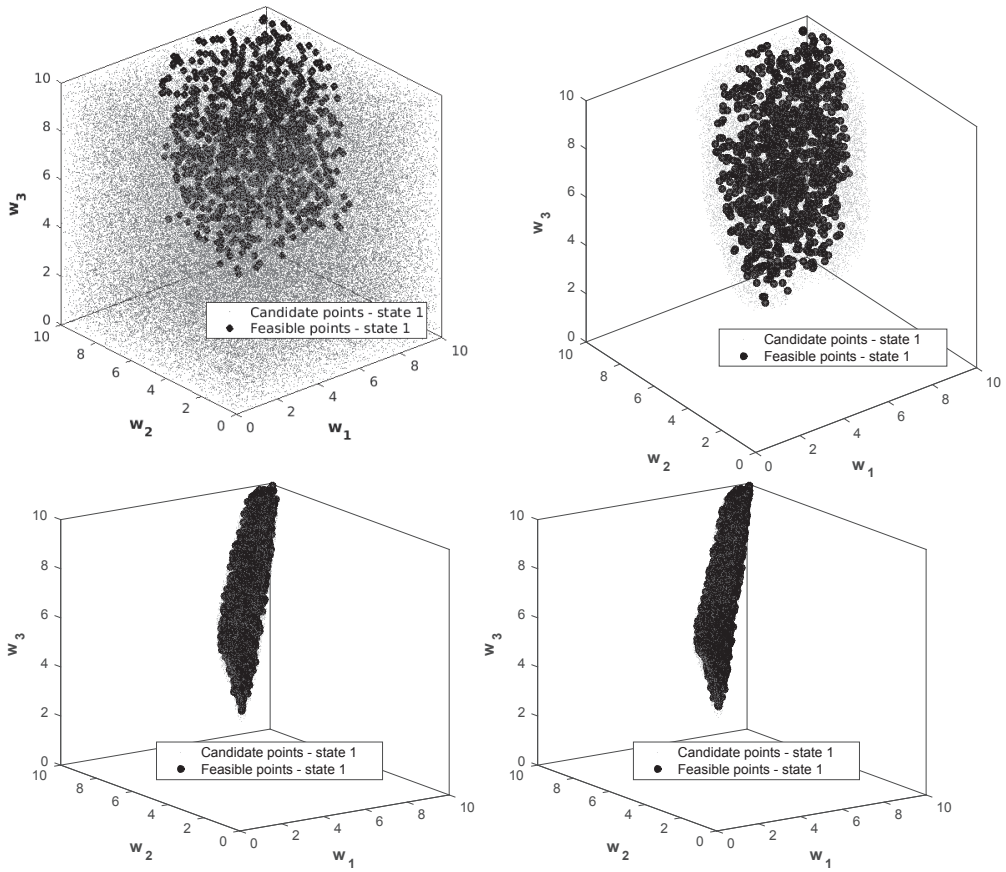

Figure 15: The above four panels display the convergence of the three-dimensional (candidate) feasible set for state 1 as a function of w_1, w_2, and w_3 (see Alg. 1). The top-left panel shows the first step in the set-valued dynamic programming procedure; the top-right shows step 3. The lower-left panel is a result from step 15, and the lower-right is based on iteration 20.

constant in the respective state. The corresponding paths of the utility promises within the feasible sets are depicted in Fig. 16.

The results we find are in line with the ones from the two-dimensional model (cf. Fig. 11). The simulation pattern in the low state, state 1 is that the utility promise tends to the lowest promise. This behavior mirrors the result from the low state of the two-dimensional model (see Sec. 4.2). In the middle state, state 2 the simulated promises rise to a central point in the state space and remain there. Finally, the simulation in the high state, state 3 behaves like the one in the high state of the two-dimensional model: it rises to the highest possible promises.

Our findings suggest that there are now two levels of insurance coverage, one with intermediate and one with a high one. If the agent is in the middle state, he buys less insurance compared to the situation where he is in the highest state, since he receives less income. In direct consequence, he cannot afford a fully covered insurance scheme, as can be seen in the middle panel of Fig. 16, where the steady state is only part way up in the promised utility space. If the agent, on the other hand, is in the high state 3, he can afford full coverage, which is reflected by a steady state in the upper corner of the promised utility space (see the right panel of Fig. 16). If the agent happens to drop from the highest state 3 to the middle state 2, he collects the payments from the high coverage insurance until it levels off when he

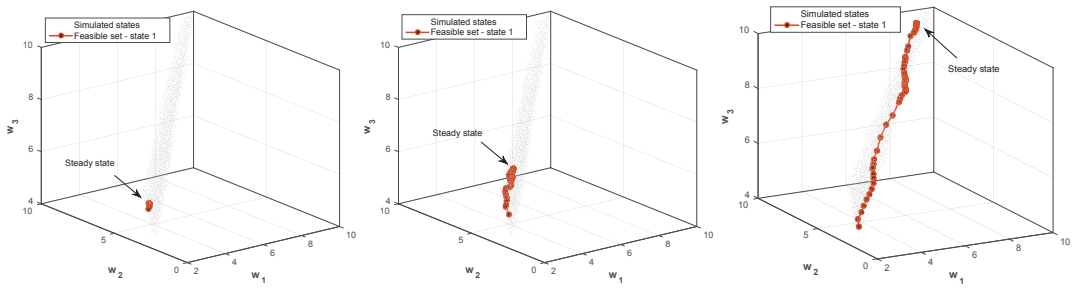

Figure 16: Left panel—simulation path in the feasible set for the low state, state 1 as a function of w_1, w_2, and w_3. Middle panel—the same, for the middle state, state 2. Right panel—the same, for the high state, state 3.

reaches the new steady state. What happens technically is that the promised utility changes from the high steady state (see the right panel of Fig. 16) to the middle steady state (see the middle panel of Fig. 16). In contrast, if the agent is in the lowest state, he cannot afford to pay any additional insurance premia and thus only receives transfers until the state arrives at the lowest promised utility point, as implied by the left panel of Fig. 16.

Next, we expand the model outlined in Sec. 4.2 to five and ten types, respectively. In the case of 5 types—that is, a model with 5 dimensions, we have a "low" (state 1), three "middle" ones (state 2, 3, and 4), and a "high" one (state 5). In presence of 10 types, i.e., in a model with a 10-dimensional state space, we have a "low" (state 1), eight "middle" ones (states 2 to 9), and a "high" one (state 10). An agent's endowment in the 5-type case is $h_1 = 0.1, h_2 = 0.16, h_3 = 0.22, h_4 = 0.28$, or $h_5 = 0.35$, whereas in the 10-type case, it is $h_1 = 0.1, h_2 = 0.13, h_3 = 0.16, h_4 = 0.19, h_5 = 0.21, h_6 = 0.24, h_7 = 0.27, h_8 = 0.3, h_9 = 0.33$, or $h_{10} = 0.35$. In both situations, the discount factor is again set to $\beta = 0.9$ (cf. Tab. 1). We choose the transition probabilities across the different types for the 5-dimensional example as

$$\Pi_{\theta,\tilde{\theta}} = \begin{bmatrix} 0.9 & 0.1 & 0.0 & 0.0 & 0.0 \\ 0.1 & 0.8 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.8 & 0.1 & 0.1 \\ 0.0 & 0.0 & 0.1 & 0.8 & 0.1 \\ 0.0 & 0.1 & 0.0 & 0.1 & 0.9 \end{bmatrix}, \tag{40}$$

whereas for the 10-dimensional example, we choose the transition probabilities to be

$$\Pi_{\theta,\tilde{\theta}} = \begin{bmatrix} 0.9 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.1 & 0.8 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.8 & 0.1 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.8 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.1 & 0.8 & 0.1 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.8 & 0.1 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.8 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.8 & 0.1 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.8 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.9 \end{bmatrix}. \tag{41}$$

Note that the stationary distribution of the Markov chain for n states will be the uniform distribution. In particular, the expected type under the ergodic distribution remains the same. However, the probability that we are in one of the states goes down and is $\frac{1}{n}$. Therefore, the ergodic distribution will converge to the uniform distribution as n goes to infinity with the

two extreme states as the bounds of the support—that is, $[0.1, 0.35]$ in our parameterization (cf. Tab. 1).[22] Thus, the variance in the endowments decreases with an ascending number of types.

In Figs. 17 and 18, we show three-dimensional projections of the equilibrium value correspondences for the five and ten-dimensional models at various iteration steps when performing set-valued dynamic programming. This sequence of figures again clearly indicates that our proposed computational method also leads to nicely-converging results in models with many agents.

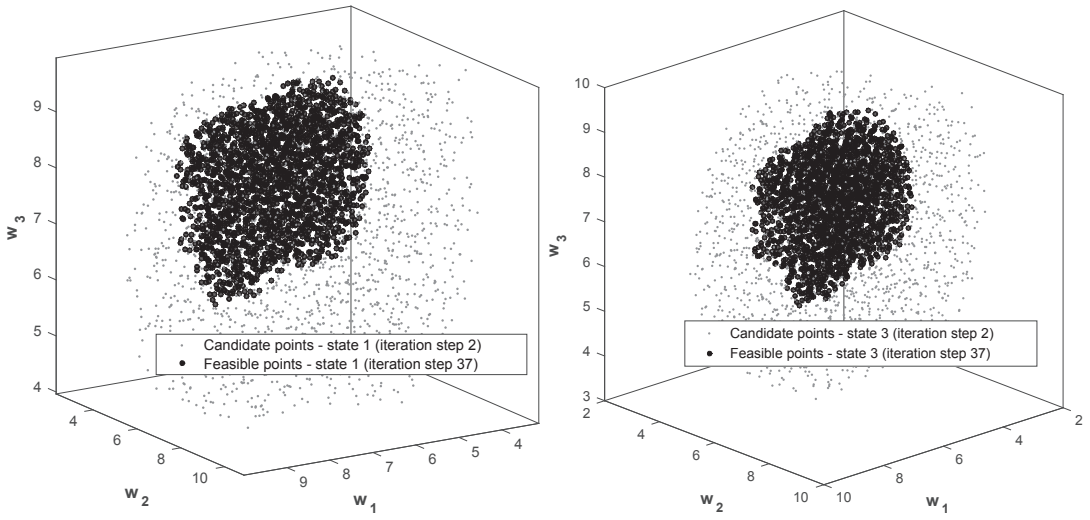

Figure 17: The above two panels display the convergence of the five-dimensional (candidate) feasible set for states 1 and 3 as a function of w_1, w_2, and w_3, while keeping $w_4 = 9.27$ and $w_5 = 6.20$ constant. Iteration steps 2 and 37 are displayed (see Alg. 1).

In Fig. 19, we display a concrete example of a simulated consumption transfer in a model with 5 types. We can see that the agent first has to pay the principal a decreasing insurance premium while being in the high state 5. Once he is hit by a type shock that moves him down to state 4, he receives a payment that is decreasing over time (see Fig. 19). Besides, please note that the payments to the agent briefly go down around simulation step $t = 35$ for a short period, right after the agent switches to the lower state 4. This effect occurs because some of the compensation the principal offers to the agent is in the form of promised utilities instead of today's consumption. Besides, we find in our computations that the overall time the agent has to pay into the insurance until he can file claims is reduced as we increase the number of types. In Fig. 13, one can see that it takes about 20 simulation steps for the transfer to reach a level well above zero. In contrast, this effect occurs already after about four iteration steps in the 3-dimensional model, and after only two simulation steps in the 5-dimensional case (see Fig. 19). This phenomenon is caused by the reduction of the standard deviation in the endowments of the ergodic distribution with an increasing number of types. One further interesting feature of our high-dimensional models is that we can study agents in intermediate states. In those settings, the agent pays into different insurance schemes with different levels of coverage–another effect that could not be studied prior to this work (see, e.g., 16).

[22] For the example of a two-type case, this implies that the agent will spend 50% of the time in the low state, and 50% of the time in the high state.

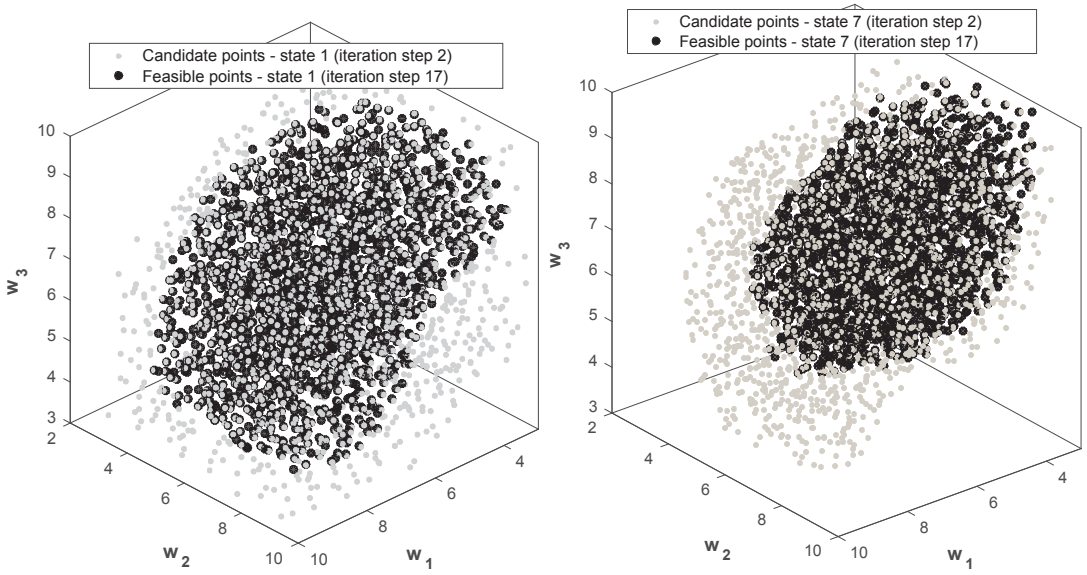

Figure 18: The above two panels display the convergence of the ten-dimensional (candidate) feasible set for states 1 (left panel) and 7 (right panel) as a function of w_1, w_2, and w_3, while keeping $w_4 = 6.17$, $w_5 = 4.48$ $w_6 = 4.39$, $w_7 = 8.38$, $w_8 = 8.56$, $w_9 = 4.59$, $w_{10} = 6.11$ (left panel) and $w_4 = 9.31, w_5 = 6.08, w_6 = 3.53, w_7 = 4.52, w_8 = 8.91, w_9 = 7.97$ and $w_{10} = 9.53$ (right panel) constant. Iteration steps 2 and 17 are displayed (see Alg. 1).

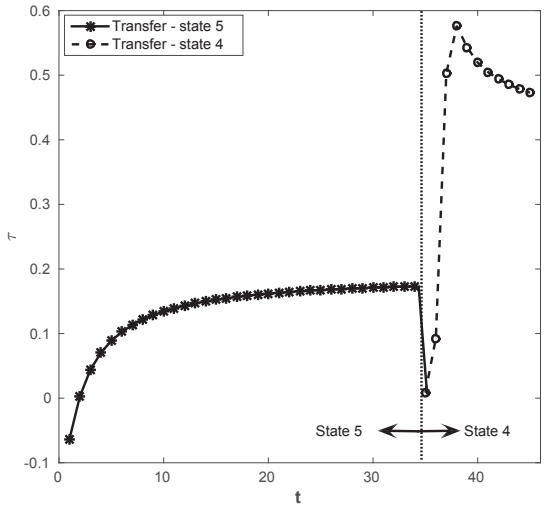

Figure 19: Consumption transfer τ (in a 5-dimensional model) from the principal to the agent while he is in the high state 5, and then after he has dropped into the lower state 4.

5 Conclusion

In this paper, we introduce a scalable and flexible framework for solving infinite-horizon, discrete time dynamic incentive problems with hidden, persistent states and of unprecedented complexity. Addressing such dynamic adverse selection models is a formidable task since two major computational bottlenecks create difficulties in the solution process. Issue one is related to the fact that the feasible sets of utility promises are not known in advance and have to be determined numerically. In most of the interesting economic applications, such sets have a

non-hypercubic geometry and might even be non-convex. Issue two results from solving the dynamic adverse selection problem with value function iteration. To do so, we need to repeatedly approximate and evaluate high-dimensional functions at arbitrary coordinates within the domain of interest—that is, the feasible sets.

We address the computation of the equilibrium value correspondences, combining the set-valued dynamic programming techniques by APS with Bayesian Gaussian mixture models—a method from unsupervised machine learning. To deal with the value function iteration, we use a massively parallel algorithm that applies Gaussian process machine learning in every iteration step to approximate the value functions on the pre-computed, time-invariant feasible sets.

Our approach provides several desirable features that yield a substantial improvement over the previous literature. First, since sampling achieves the approximation of feasible sets through Bayesian Gaussian mixture models, it does not directly suffer from the curse of dimensionality and thus can deal with problems involving many types of agents. Second, our scheme of approximating equilibrium value correspondences is independent of solving the dynamic incentive problem, and thus does not directly add to the computational complexity of solving the actual model. Third, it can approximate both convex and non-convex sets. Moreover, the equilibrium Bayesian Gaussian mixture models can be used to generate sample data from within the approximate equilibrium value correspondences. This directly plays to the strength of Gaussian process regression: a form of supervised machine learning that can—given a training set that is generated from within the feasible region—be used to approximate and interpolate functions on irregularly shaped state spaces. Furthermore, since the construction of Gaussian process interpolators is achieved by sampling from a domain of interest such as a feasible set, they can handle the curse of dimensionality to some extent. Thus, our proposed method—that is, using APS and Bayesian Gaussian mixture models in conjunction with Gaussian processes—has the potential of handling highly complex dynamic incentive models at a relatively low computational cost, as it focuses the resources where needed.

To demonstrate the capabilities of our framework, we compute solutions to models of repeated agency with history dependence, and up to ten types. In addition, and unlike the previous literature, we allow the agent to overreport his type. We find that the agent has to pay into the insurance for a shorter amount of time until he can claim insurance benefits when compared to the models commonly used. Moreover, we observe that the overall time the agent has to pay into the insurance is decreasing even further if we increase the number of types, a quantitative effect that could not be studied prior to this work.

Besides, we emphasize that while the focus of the work presented in this paper lies in solving dynamic adverse selection problems, the method proposed here—being generic—has a far broader scope: it can also be applied, for example, to moral hazard problems, mechanism design, or dynamic games, where one of the significant difficulties also lies in finding the equilibrium correspondences.

In summary, this all suggests that our proposed framework will enable researchers to think of dynamic incentive problems of greater richness than was possible prior to this work, as they no longer need to drastically restrict their modeling choices from the outset.

A Background on dynamic programming

Throughout this paper, the abstract class of models we consider are discrete-time, infinite-horizon stochastic optimal decision-making problems. We briefly characterize them here by the subsequent general description: let $x_t \in W \subset \mathbb{R}^N$ denote the state of the economy at time $t \in \mathbb{N}^+$ of dimensionality $N \in \mathbb{N}$. Controls (actions) are represented by a *policy function* $\upsilon : W \to \zeta$, where ζ is the space of possible controls. The discrete-time transition function of the economy from one period to the next is given by the distribution of x_{t+1}, which depends on the current state and policies

$$x_{t+1} \sim f\left(\cdot | x_t, \upsilon(x_t)\right). \tag{42}$$

The transition function f that stochastically maps a state-action pair to a successor state is assumed to be given, whereas the policy function υ needs to be determined from equilibrium or optimality conditions. The standard way to do so is to use dynamic programming (see, e.g., Bellman (1961), Stokey et al. (1989a), Judd (1998), Ljungqvist and Sargent (2000)), where the task is to find an infinite sequence of *controls* $\{\chi_t\}_{t=0}^{\infty}$ to maximize the *value function*

$$V(x_0) := \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t r(x_t, \chi_t)\right] \tag{43}$$

for an initial state $x_0 \in W$, $r(\cdot, \cdot)$ is the so-called *return function*, and $\chi_t \in \Gamma(x_t) \subset W$, with $\Gamma(x_t)$ being the set of feasible choices given x_t. The discount factor $\beta \in (0, 1)$ weights future returns. Dynamic programming seeks a time-invariant policy function υ mapping the state x_t into the action χ_t, such that for all $t \in \mathbb{N}$

$$\chi_t \quad = \quad \upsilon(x_t) \in \Gamma(x_t), \tag{44}$$

and $\{\chi_t\}_{t=0}^{\infty}$ solves the original problem. The *principle of optimality* states that we can find such a solution by solving the *Bellman equation*

$$V(x) = \max_{\chi}\{r(x, \chi) + \beta \mathbb{E}\left[V(\tilde{x})\right]\}, \tag{45}$$

where the successor state is distributed as $\tilde{x} \sim f\left(\cdot | x, \chi\right)$. The solution is a fixed point of the Bellman operator T, defined by

$$(TV)(x) = \max_{\chi}\{r(x, \chi) + \beta \mathbb{E}\left[V(\tilde{x})\right]\}. \tag{46}$$

Under appropriate conditions (see, e.g., Stokey et al. (1989a)) the Bellman operator is a contraction mapping. In this case, iteratively applying T provides a sequence of value functions that converges to a unique fixed point. This procedure is called *value function iteration* (see, e.g. Bertsekas (2000), Judd (1998), or Ljungqvist and Sargent (2000)) and is motivated by this theoretical justification and numerically implements the iterative application of the Bellman operator to successive approximations of the value function. The corresponding dynamic programming recursion thus starts from any bounded and continuous guess for the value function, and the solution is approached in the limit as $j \to \infty$ by iterations on

$$V^{j+1}(x) = T(V^j)(x) := \max_{\chi^{j+1}}\{r(x, \upsilon) + \beta \mathbb{E}\left[V^j(\tilde{x})\right]\}. \tag{47}$$

In practice, we say that value function iteration has converged if numerical convergence in some norm, for example

$$\|V^{\tau} - V^{\tau - 1}\|_{\infty} \leq \epsilon, \quad \epsilon \in \mathbb{R}^+, \tag{48}$$

and some at some iteration step τ, is reached. The (approximate) equilibrium value function shall be denoted as $V^* = V^\tau$ and the corresponding policy functions as $\chi^* = \chi^\tau$. From Eq. (47), it becomes apparent that we need to repeatedly approximate and evaluate (potentially multi-dimensional) value functions. An additional complication stems from the fact that domain W for the models we are targeting (cf. Sec. 2) is often highly complex—that is, irregularly shaped and not known a priori (see, e.g., Fernandes and Phelan (2000) and Broer et al. (2017)). We, therefore, describe in Sec. 3.1 how we determine W numerically, whereas in Sec. 3.2 we show how we approximate value and policy functions on irregularly shaped feasible sets.

B Approximating feasible sets with adaptive sparse grids

To verify the functionality of the framework proposed in this paper for determining the equilibrium correspondences (see Sec. 3.1), we cross-validate it by applying adaptive sparse grids (see, e.g., Brumm and Scheidegger (2017)). In particular, we solve—as briefly mentioned in Sec. 3.1—an auxiliary problem of Eq. (13) for which we know the true solution on the feasible set. More precisely, we relax the feasibility via a penalty function such that the resulting value function will be zero on the feasible set and less otherwise. Strictly speaking, we are looking for the fixed point of the following dynamic program:

$$
\begin{aligned}
F(w,\theta) = \max_{c,w^+,\xi} \sqrt{\varepsilon} - \sqrt{\varepsilon + \sum_{\tilde{\theta}} \xi_{\tilde{\theta}}^2} + \beta \sum_{\tilde{\theta}\in\Theta} \Pi_{\theta,\tilde{\theta}} F^+(w_{\tilde{\theta}}^+, \tilde{\theta}) \qquad (49)\\
\text{s.t. } w_\nu = \xi_\nu + \sum_{\tilde{\theta}\in\Theta} \Pi_{\nu,\tilde{\theta}} (u(c_{\tilde{\theta}}, \tilde{\theta}) + \beta w_{\tilde{\theta},\tilde{\theta}}^+) \ \forall \nu \in \Theta,\\
u(c_{\tilde{\theta}}, \tilde{\theta}) + \beta w_{\tilde{\theta},\tilde{\theta}}^+ \geq u(c_\nu, \tilde{\theta}) + \beta w_{\nu,\tilde{\theta}}^+ \ \forall \tilde{\theta} \in \Theta, \ \forall \nu \in \Theta \setminus \{\tilde{\theta}\},\\
c \in [0, \overline{c}]^N,\\
w_{\tilde{\theta}}^+ \in \mathbb{R}^N \text{ with } w_{\tilde{\theta},\nu}^+ = (w_{\tilde{\theta}}^+)_\nu \ \forall \tilde{\theta} \in \Theta, \ \forall \nu \in \Theta,
\end{aligned}
$$

where $\varepsilon > 0$ is a relaxation parameter in

$$
\sqrt{\varepsilon} - \sqrt{\varepsilon + \sum_{\tilde{\theta}} \xi_{\tilde{\theta}}^2}, \qquad (50)
$$

and where Eq. (50) is a smooth approximation of the norm $\|\cdot\|_1$. $F(\cdot,\cdot)$ is approximated by piecewise linear basis functions on an adaptive sparse grid. Eq. (49) is an ordinary dynamic programming problem with a fixed point and therefore can be solved by value function iteration. We know that for all feasible w, the value function will be 0. To see this, note that the payoff of Eq. (49) in some iteration t is $\sqrt{\varepsilon} - \sqrt{\varepsilon + \sum_{\tilde{\theta}} \xi_{\tilde{\theta}}^2}$—that is to say, less than or equal to zero. Moreover, any feasible point will have an optimal value of zero, since there we will not need to relax the bounds on the equality constraints. Thus, the resulting value function has to be zero for feasible values and less than zero otherwise. Once the value function iteration for Eq. (49) has converged, we can use $F(\cdot,\cdot)$ as a penalty for the dynamic incentive problem (see Eq. (17)).

We now recompute the test case outlined in Sec. 4.1. In Fig. 20, we display the equilibrium correspondences for the low and high states, computed by applying adaptive sparse grids. Comparing Fig. 20 to Fig. 3, it becomes apparent that the two methods yield numerically "identical" results and thus confirm each other.

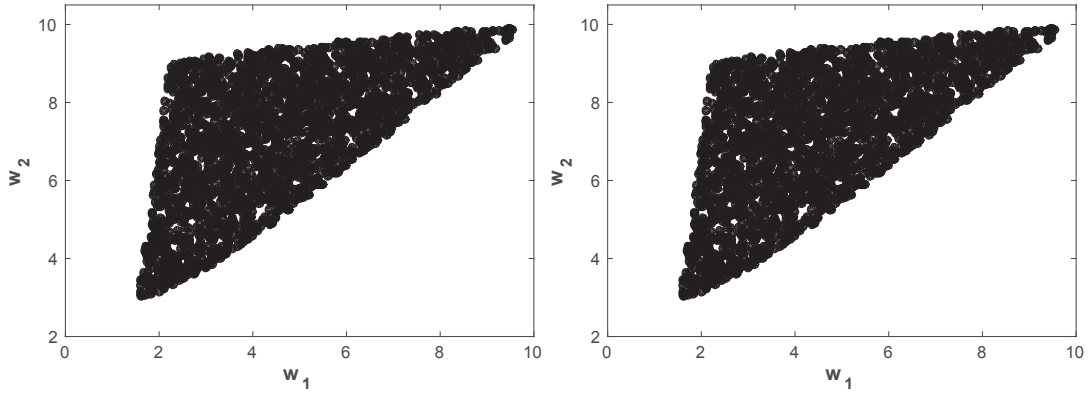

Figure 20: The left panel shows the approximate equilibrium correspondences for the low state as a function of w_1 and w_2, whereas the right panel shows the respective set for the high state. Both feasible sets were computed by applying adaptive sparse grids and set-valued dynamic programming (see Eq. (49)).

C A generic parallelization scheme for dynamic incentive problems

To speed up the solution process of the computational method introduced in this paper, we propose a highly-scalable parallelization scheme that is generic to dynamic incentive problems with persistent, discrete shocks (cf. Sec. 3.3 and Alg. 2). In particular, we exploit the generic structure of the economic model under consideration to implement a hybrid parallelization scheme that uses shared memory (see, e.g, Jost et al. (2007)) and distributed memory (see, e.g., Gropp et al. (2014)) parallelization paradigms. This will allow us to make efficient use of state-of-the-art high-performance computing (HPC) facilities (see, e.g., Dongarra and Steen (2012)), whose performance nowadays can reach up to hundreds of Petaflop/s (see, e.g., `https://www.top500.org`).

Conceptually, the top layer of parallelism present in our problem is the N_s discrete states—that is, the different types of the dynamic incentive problem, which are entirely independent of each other within a value function iteration step. Hence, the `MPI_COMM_WORLD` communicator is split into N_s sub-communicators of sizes $J, ..., T$, each of them representing an individual discrete state—that is, an independent Gaussian process which updates its share of the total value function $V = (V(\cdot, \Theta = 1), \ldots, V(\cdot, \Theta = N_s))$. After that, every `MPI_Group` obtains a fraction from all the MPI processes available in the `MPI_COMM_WORLD` communicator assigned such that an optimal workload balance across different discrete states is guaranteed (see the first three rows of Fig. 21). Inside every `MPI_Group`, the training targets for the Gaussian process regression are computed in a massively parallel fashion. The points $\omega \in G_i$ that are generated within the feasible set of state $\Theta = i$ are distributed via MPI among multiple, multi-threaded processes. The sample points that are sent to one particular compute node are then further distributed among different threads. Multithreading on compute nodes is implemented with OpenMP. To guarantee efficient use of any of the compute nodes, the threads leverage OpenMP's dynamic workload balancing. In general, each OpenMP thread has to solve an independent optimization problem for every single training point assigned to it.[23] These optimization problems are in our case solved with `Ipopt` (Waechter and Biegler, 2006). The generic parallelization scheme outlined here has excellent strong scaling properties, and thus allows us to accelerate the time to solution by two to three orders of magnitude.

[23]Note that the computation of the feasible sets (cf. Tab. 1) is parallelized analogously.

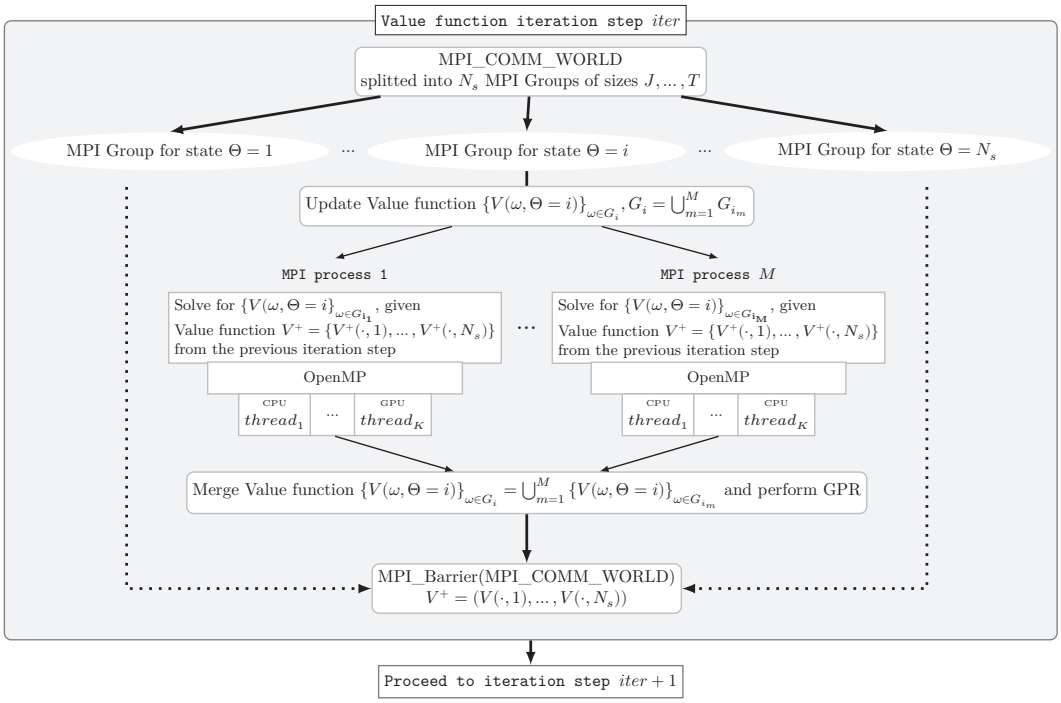

Figure 21: Schematic representation of the hybrid parallelization scheme in a single value function iteration step *iter*. Every `MPI` process within an `MPI_Group` is using OpenMP.

References

Abraham, Arpad and Nicola Pavoni. "Efficient allocations with moral hazard and hidden borrowing and lending: a recursive formulation". *Review of Economic Dynamics* 11.4 (2008), pp. 781–803.

Abreu, Dilip, Ben Brooks, and Yuliy Sannikov. "An algorithm for stochastic games with perfect monitoring". *Working paper* (2018).

Abreu, Dilip, David Pearce, and Ennio Stacchetti. "Optimal cartel equilibria with imperfect monitoring". *Journal of Economic Theory* 39.1 (1986), pp. 251–269.

Abreu, Dilip, David Pearce, and Ennio Stacchetti. "Toward a Theory of Discounted Repeated Games with Imperfect Monitoring". *Econometrica* 58.5 (1990), pp. 1041–1063.

Abreu, Dilip and Yuliy Sannikov. "An algorithm for two-player repeated games with perfect monitoring". *Theoretical Economics* 9.2 (2014), pp. 313–338.

Bellman, R. *Adaptive Control Processes: A Guided Tour*. 'Rand Corporation. Research studies. Princeton University Press, 1961.

Bertsekas, Dimitri P. *Dynamic Programming and Optimal Control*. 2nd. Athena Scientific, 2000.

Blei, David M. and Michael I. Jordan. "Variational inference for dirichlet process mixtures". *Bayesian Analysis* 1 (2005), pp. 121–144.

Broer, Tobias, Marek Kapicka, and Paul Klein. "Consumption risk sharing with private information and limited enforcement". *Review of Economic Dynamics* 23 (2017), pp. 170–190.

Brumm, Johannes, Felix Kubler, and Simon Scheidegger. "Computing equilibria in dynamic stochastic macro-models with heterogeneous agents". *Advances in Economics and Econometrics, Eleventh World Congress* (2017), pp. 185–230.

Brumm, Johannes, Dmitry Mikushin, Simon Scheidegger, and Olaf Schenk. "Scalable high-dimensional dynamic stochastic economic modeling". *Journal of Computational Science* 11 (2015), pp. 12–25.

Brumm, Johannes and Simon Scheidegger. "Using adaptive sparse grids to solve high-dimensional dynamic models". *Econometrica* 85.5 (2017), pp. 1575–1612.

Chvatal, Vasek. *Linear Programming*. Series of books in the mathematical sciences. BEDFORD BOOKS, 2016.

Cole, Harold L. and Narayana R. Kocherlakota. "Efficient allocations with hidden income and hidden storage". *The Review of Economic Studies* 68.3 (2001), pp. 523–542. eprint: `/oup/backfile/content_public/journal/restud/68/3/10.1111/1467-937x.00179/2/68-3-523.pdf`.

Dasgupta, S. "Learning mixtures of gaussians". *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*. 1999, pp. 634–644.

DeMarzo, Peter M. and Yuliy Sannikov. "Optimal security design and dynamic capital structure in a continuous-time agency model". *The Journal of Finance* 61.6 (2006), pp. 2681–2724.

Doepke, Matthias and Robert M. Townsend. "Dynamic mechanism design with hidden income and hidden actions". *Journal of Economic Theory* 126.1 (2006), pp. 235–285.

Dongarra, J. J. and A. J. van der Steen. "High-performance computing systems: status and outlook". *Acta Numerica* 21 (2012), pp. 379–474. eprint: `http://journals.cambridge.org/article_S0962492912000050`.

Eftekhari, Aryan, Simon Scheidegger, and Olaf Schenk. "Parallelized dimensional decomposition for large-scale dynamic stochastic economic models". *Proceedings of the Platform for Advanced Scientific Computing Conference*. PASC '17. Lugano, Switzerland: ACM, 2017, 9:1–9:11.

Fernandes, Ana and Christopher Phelan. "A recursive formulation for repeated agency with history dependence". *Journal of Economic Theory* 91.2 (2000), pp. 223–247.

Golosov, Mikhail, Aleh Tsyvinski, and Nicolas Werquin. "Recursive contracts and endogenously incomplete markets". *Handbook of Macroeconomics* 2 (2016), pp. 725–841.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

Gropp, William, Torsten Hoefler, Rajeev Thakur, and Ewing Lusk. *Using Advanced MPI: Modern Features of the Message-Passing Interface*. The MIT Press, 2014.

He, Zhiguo, Bin Wei, Jianfeng Yu, and Feng Gao. "Optimal long-term contracting with learning". *The Review of Financial Studies* 30.6 (2017), pp. 2006–2065.

Jost, Gabriele, Barbara Chapman, and Ruud van der Pas. *Using OpenMP - Portable Shared Memory Parallel Programming*. MIT Press, 2007.

Judd, Kenneth L. "Projection methods for solving aggregate growth models". *Journal of Economic Theory* 58.2 (1992), pp. 410–452.

Judd, Kenneth L. *Numerical methods in economics*. The MIT press, 1998.

Judd, Kenneth L, Lilia Maliar, Serguei Maliar, and Rafael Valero. "Smolyak method for solving dynamic economic models: lagrange interpolation, anisotropic grid and adaptive domain". *Journal of Economic Dynamics and Control* 44 (2014), pp. 92–123.

Judd, Kenneth, Sevin Yeltekin, and James Conklin. "Computing supergame equilibria". *Econometrica* 71.4 (2003), pp. 1239–1254.

Keane, Michael P. and Kenneth I. Wolpin. "The solution and estimation of discrete choice dynamic programming models by simulation and interpolation: monte carlo evidence". *The Review of Economics and Statistics* 76.4 (1994), pp. 648–672.

Krueger, Dirk and Felix Kubler. "Computing equilibrium in OLG models with stochastic production". *Journal of Economic Dynamics and Control* 28.7 (2004), pp. 1411–1436.

Lambert, Richard A. "Long-term contracts and moral hazard". *Bell Journal of Economics* 14.2 (1983), pp. 441–452.

Ljungqvist, L. and T.J. Sargent. *Recursive macroeconomic theory*. Mit Press, 2000.

Mailath, George J., Ichiro Obara, and Tadashi Sekiguchi. "The maximum efficient equilibrium payoff in the repeated prisoners' dilemma". *Games and Economic Behavior* 40.1 (2002), pp. 99–122.

Malin, Benjamin, Dirk Krueger, and Felix Kuebler. "Solving the multi-country real business cycle model using a smolyak-collocation method". *Journal of Economic Dynamics and Control* 35.2 (2010), pp. 229–239.

Marcet, Albert and Ramon Marimon. "Recursive contracts" (2017).

Meghir, Costas and Luigi Pistaferri. "Income variance dynamics and heterogeneity". *Econometrica* 72.1 (2004), pp. 1–32.

Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

Park, J. and I. W. Sandberg. "Universal approximation using radial-basis-function networks". *Neural Computation* 3.2 (1991), pp. 246–257.

Pavan, Alessandro, Ilya Segal, and Juuso Toikka. "Dynamic mechanism design: a myersonian approach". *Econometrica* 82.2 (2014), pp. 601–653.

Pavoni, Nicola, Christopher Sleet, and Matthias Messner. "The dual approach to recursive optimization: theory and examples". *Forthcoming, Econometrica* (2017).

Poggio, T. and F. Girosi. "Networks for approximation and learning". *Proceedings of the IEEE* 78.9 (1990), pp. 1481–1497.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. New York, NY, USA: Cambridge University Press, 2007.

Rasmussen, Carl Edward. "The infinite gaussian mixture model". *In Advances in Neural Information Processing Systems 12*. MIT Press, 2000, pp. 554–560.

Rasmussen, Carl Edward and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

Rogerson, William P. "Repeated moral hazard". *Econometrica* 53.1 (1985), pp. 69–76.

Sannikov, Yuliy. "A continuous- time version of the principal: agent problem". *The Review of Economic Studies* 75.3 (2008), pp. 957–984.

Scheidegger, S., D. Mikushin, F. Kubler, and O. Schenk. "Rethinking large-scale economic modeling for efficiency: optimizations for gpu and xeon phi clusters". *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2018, pp. 610–619.

Scheidegger, Simon and Ilias Bilionis. "Machine learning for high-dimensional dynamic stochastic economies". *Preprint available at SSRN: https://ssrn.com/abstract=2927400* (2017).

Scheidegger, Simon and Adrien Treccani. "Pricing american options under high-dimensional models with recursive adaptive sparse expectations". *Journal of Financial Econometrics* (2018).

Sleet, Christopher and Sevin Yeltekin. "On the computation of value correspondences for dynamic games". *Dynamic Games and Applications* 6.2 (2016), pp. 174–186.

Spear, Stephen E. and Sanjay Srivastava. "On repeated moral hazard with discounting". *The Review of Economic Studies* 54.4 (1987), pp. 599–617.

Stokey, Nancy, Robert Lucas, Jr., and Edward Prescott. *Recursive Methods in Economic Dynamics*. Cambridge, MA: Harvard University Press, 1989.

Stokey, Nancy, Robert Lucas, and Edward Prescott. "Recursive methods in economic dynamics". *Cambridge MA* (1989).

Storesletten, Kjetil, Christopher Telmer, and Amir Yaron. "Consumption and risk sharing over the life cycle". *Journal of Monetary Economics* 51.3 (2004), pp. 609–633.

Waechter, Andreas and Lorenz T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". *Math. Program.* 106.1 (2006), pp. 25–57.

Wang, Cheng. "Dynamic Insurance with Private Information and Balanced Budgets". *The Review of Economic Studies* 62.4 (1995), pp. 577–595.

Werning, Ivan. *Optimal unemployment insurance with unobservable savings*. 2002.

Williams, Noah. "On dynamic principal-agent problems in continuous time" (2009).

Williams, Noah. "Persistent private information". *Econometrica* 79.4 (2011), pp. 1233–1275.

Wu, Yue, Hui Wang, Biaobiao Zhang, and K.-L. Du. "Using radial basis function networks for function approximation and classification". *ISRN Applied Mathematics* (2012).

Yeltekin, Sevin, Yongyang Cai, and Kenneth L. Judd. "Computing equilibria of dynamic games". *Operations Research* 65.2 (2017), pp. 337–356.