MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Barycentric Quantization for Planning in Continuous Domains

Nikovski, D.N.

TR2015-151 July 2015

Abstract

We consider the class of planning and sequential decision making problems where the state space has continuous components, but the available actions come from a discrete set, and argue that a suitable approach for solving them could involve an appropriate quantization scheme for the continuous state variables, followed by approximate dynamic programming. We propose one such scheme based on barycentric approximations that effectively converts the continuous dynamics into a Markov decision process, and demonstrate that it can be viewed both as an approximation to the continuous dynamics, as well as a value function approximator over the continuous domain. We describe the application of this method to several hard industrial problems, and point out additional candidate problems that could be amenable to it.

AI Communications

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Copyright © Mitsubishi Electric Research Laboratories, Inc., 2015
201 Broadway, Cambridge, Massachusetts 02139

AI Communications 28 (2015) 539–551
DOI 10.3233/AIC-150675
IOS Press

Barycentric quantization for planning in continuous domains

Daniel Nikolaev Nikovski
Mitsubishi Electric Research Labs, 201 Broadway, Cambridge, USA
E-mail: nikovski@merl.com

Abstract. We consider the class of planning and sequential decision making problems where the state space has continuous components, but the available actions come from a discrete set, and argue that a suitable approach for solving them could involve an appropriate quantization scheme for the continuous state variables, followed by approximate dynamic programming. We propose one such scheme based on barycentric approximations that effectively converts the continuous dynamics into a Markov decision process, and demonstrate that it can be viewed both as an approximation to the continuous dynamics, as well as a value function approximator over the continuous domain. We describe the application of this method to several hard industrial problems, and point out additional candidate problems that could be amenable to it.

Keywords: Planning, Markov decision processes

1. Introduction

The problem of optimal sequential decision making arises in many practical systems, and a number of distinct scientific disciplines such as AI planning, control systems engineering, operations research and decision science are concerned with computationally efficient methods for its solution. The commonality and connections between these disciplines have been recognized, and many useful correspondences between concepts from these fields have been drawn [3,4,12]. Although each of these fields has generated highly original theories and computational methods that address one aspect of optimal sequential decision making or another, a clearly dominant set of computational methods has neither emerged yet, nor is likely to emerge in the near future. Rather, the characteristics of a particular problem often suggest the most appropriate set of tools and methods that can be deployed to successfully solve it. For example, control systems engineering has developed reliable methods for building robust controllers for regulating linear time invariant dynamical systems with continuous variables (inputs, outputs and states). In contrast, AI planning algorithms have excelled at solving sequential decision making problems in domains with discrete variables, whose dynamics are described in situation calculus by logical sentences [31]. Other formalisms, often used in operations research and decision science, include Markov decision processes, influence diagrams, and dynamic Bayesian networks, which effectively represent the uncertainty in the problem domain and can be solved by means of dynamic programming, can be very effective for large stochastic domains. These tools and approaches are not mutually exclusive; on the contrary, many intelligent systems would naturally employ two or more methods, properly organized into a decision architecture. For example, a robot could use an AI planner to decide an overall plan of action in terms of high-level actions and goals described in the form of logical statements, and then execute the plan by regulating its actuators by means of low-level controllers. Similarly, a queuing Markov model of a transportation system can be used to decide where to dispatch vehicles in order to satisfy demand driven by stochastic processes, and then regulating controllers can be employed to ensure that the vehicles are following the desired trajectories to their destinations.

A more interesting, and far more difficult choice of tools and methods appears when a target problem domain does not clearly favor one of these solution methodologies. One arguably large and important class of sequential decision making problems is concerned with optimizing decisions where the decision variables (actions, control inputs) are discrete, but at least some of the state variables are continuous. Some examples of such decision problems are described in Section 2, along with a discussion of why they are hard to solve.

0921-7126/15/$35.00 © 2015 – IOS Press and the authors. All rights reserved

Section 3 argues that a suitable solution approach for this class of problems might be to quantize the continuous state variables into a discrete-valued state descriptor, and proposes one such quantization scheme based on barycentric coordinates. This scheme effectively turns the problem into a large factored Markov decision problem (MDP) that can be solved by means of approximate dynamic programming. We demonstrate that the application of dynamic programming on the resulting discrete MDP can be viewed both as an approximation to the original continuous dynamics of the continuous domain, as well as a form of value function approximation over that continuous domain. Section 4 discussed the application of the method to the problems previously introduced in Section 2, and discusses remaining issues that are needed to be resolved in order for this approach to scale to even larger problems. Finally, Section 5 concludes and discusses the general applicability of the method to problems in this class.

2. Sequential decision problems with discrete actions and continuous states

2.1. Problem definition and possible solution methods

We are considering the class of sequential decision problems where the objective is to optimize a performance measure over multiple discrete time periods by selecting appropriate inputs a from a finite discrete set A, usually of low cardinality, in order to steer a dynamical system with given dynamics, and at least some components of the state space of that system are continuous, that is, they can be represented by a real-valued vector $x^{(c)} \in R^d$. Other components $x^{(b)}$ of this system can have Boolean or integer variables, turning it into a hybrid dynamical system. The dynamics of the controlled system are described by the general form $x_{k+1} = f(x_k, a_k)$, where $x_k = [x_k^{(c)}, x_k^{(b)}]$ is the entire state of the system at time t_k, a_k is the control applied at time t_k, f is an arbitrary non-linear, possibly stochastic, and possibly discontinuous function, and the system evolves in discrete time such that $t_k = k\Delta t$ for a suitably chosen constant interval Δt. The goal of the sequential decision maker is to find the optimal sequence of actions $a_0, a_1, a_2, \ldots$ that minimizes a suitable performance measure defined over the states traversed by the system and the actions applied to it. One possible performance measure is the cumulative cost over a finite horizon of K steps $J = \sum_{k=0}^{K-1} g(x_k, a_k) + h(x_K)$, where $g(x_k, a_k)$ is a suitably chosen running cost, and $h(x_K)$ is a terminal cost associated with the final state x_K. Another possible performance measure is the discounted cumulative cost over an infinite horizon $J = \sum_{k=0}^{\infty} \gamma^k g(x_k, a_k)$, where $0 < \gamma < 1$ is a discounting factor.

The disciplines of control systems engineering, AI planning and operations research have developed very effective methods for special cases of this or related problems, but direct applicability of these methods to the general problem described above is not straightforward. For example, in control engineering, the subfield of optimal control solves this problem for systems whose state spaces and control inputs are continuous. When the systems dynamics function f is linear, and the functions g and h are quadratic in the state x and control a, the problem is known as the linear quadratic (LQ) problem, and an analytical solution known as the linear quadratic regulator (LQR) is available [35]. It can also be extended to the case when the function f is not deterministic, but stochastic, and the system state is not directly observable: when system dynamics are disturbed by additive white Gaussian noise, the solution of the resulting linear quadratic Gaussian (LQG) problem can be reduced to a combination of a linear quadratic estimator (LQE) in the form of a Kalman filter, and an LQR on the estimated state.

Although important and useful for many applications, the LQR and LQG methods are clearly not applicable to the class of problems defined above. When the function f is not linear, and/or the cost functions g and h are not quadratic in the state and control, numerical methods must be tried instead. The method of model predictive control (MPC) can solve such decision problems by continuously optimizing the performance criterion over a truncated future window, in a rolling horizon fashion, using the system dynamics function to predict the change in state variables as a consequence of changes in the input control variables, and the resulting effect on the performance criterion. This approach effectively reduces the decision problem into a constrained non-linear optimization problem, where system dynamics define a major part of the constraints, and its success depends critically on how well and how fast this problem can be solved. Beyond the case when the control actions are continuous, dynamics are linear, and costs are quadratic, the resulting non-linear optimization problem can be non-convex, but still various Newton-type optimization methods, for example the primal-dual interior point method, can find solutions very effectively, provided that certain conditions of differentiability are satisfied. This is the approach taken

by direct optimal control methods, some of which have been implemented in modeling and optimization environments [1]. However, when some of the state variables are discrete, and the corresponding parts of the system dynamics are described by discontinuous functions, for example propositional logic statements, these differentiability conditions are clearly violated, and for such problems the MPC and direct optimal control approaches quickly become inapplicable or very suboptimal.

The sub-field of mixed integer programming in operations research has devised techniques for dealing with domains where at least some of the decision and state variables are discrete, resulting in hybrid discrete/continuous dynamics. For ordinal integer variables, various kinds of relaxation techniques to continuous approximations can be attempted, and some of them, such as the cutting plane method, can be very effective, when the objective function of the optimization problem is linear. However, when it is non-linear, and the domain has two-valued Boolean variables that represent logical components of the system state, such techniques cannot be expected to perform very well. Exact solutions, such as the branch-and-bound algorithm and its variant, the branch-and-cut algorithm, can also be very effective, especially when combined with suitable branching heuristics, but the fact still remains that mixed-integer programming (MIP) is an NP-hard problem, and insisting on optimality might be infeasible and impractical. Various heuristic methods have been applied as well, such as hill climbing, simulated annealing, ant colony optimization, etc., but handling hybrid discrete/continuous domains could be difficult for them, in general.

In its turn, the field of AI planning has devised algorithms that can deal very effectively with Boolean and other discrete variables, especially when the objective is to reach a final goal (system state), regardless of the trajectory followed. Non-linearity and discontinuity are natural aspects of such domains, and rarely present problems. Very large state spaces can be explored by casting the planning problem as that of non-exhaustive search, aided by various heuristics. System dynamics involving many variables can be represented conveniently by specialized languages such as STRIPS and PDDL. Extensions to these languages such as PDDL+ [17] can also represent continuous and hybrid discrete/continuous dynamics, and importantly, also handle continuous time by considering actions of varying length, instead of immediately discretizing it at a constant time step, as is customary in control engineering. As a result, PDDL+ permits actions to start at their most appropriate time, as opposed to the closest multiple of a sampling time step. However, the problems of plan validation and plan optimization are solvable only for some subsets of the domains that can be described by means of these languages, that is, there is a mismatch between the class of problems that can be represented by means of PDDL+, and the classes of problems that can be effectively solved by various planners.

Multiple approaches to solving hybrid planning problems encoded in PDDL+ have been explored by a significant number of planners. One approach is to reduce the program to a propositional satisfiability problem augmented with linear constraints arising from continuous state or decision variables, and solve it by means of mature and powerful satisfiability solvers [34]. However, the ability to handle arbitrary continuous dynamics has been reported to result in slow computational speed; for cases when the continuous dynamics are linear (i.e., all numeric-valued fluents vary at a constant rate with time), the method proposed in the planner COLIN can find solutions much more quickly by formulating a set of linear constraints arising from continuous dynamics, and applying linear programming solvers [11]. Another planner that can solve hybrid planing problems with linear continuous dynamics after suitable discretization of the continuous state space is described in [24].

A possible method for planning with arbitrary nonlinear continuous dynamics is discretization of the continuous variables followed by heuristic search in the discretized domain, as implemented in the planner UPMurphy [13,14]. However, the resulting approximation error can be very large, if discretization is coarse, or the computational time can be prohibitively high, if discretization is at a finer scale. One particular unfavorable effect is the compounding of errors in dynamics across multiple sequential steps, such that the predicted final state resulting from a long sequence of actions calculated by means of the discretized model can be very different from that computed by means of the original continuous dynamics. Another disadvantage of this approach is that by relying on heuristic search, some relatively good solutions might be found, but the truly optimal solution would typically not be reached. Yet another approach to solve problems with hybrid dynamics is to translate the domain description to a set of hybrid automata, and perform model checking on the translated representation [5].

There have also been effective methods for combining the computational power of MIP solvers with the

representational power of planning problem description languages. Li and Williams [23] proposed a hybrid systems planner that combines a planning graph similar to the one used by GraphPlan for representing the effect of discrete actions with a compact representation for an infinite set of trajectories involving continuous states and actions called Flow Tubes. The resulting hybrid plan representation called Hybrid Flow Graph can then be solved by a general-purpose MIP solver.

Finally, a modeling formalism shared between the fields of AI planning, operations research, and decision science, and widely employed in sequential decision making, especially under uncertainty, is the framework of Markov decision processes (MDP) [30]. An MDP is defined by a finite set of states, a finite set of actions, a transition probability function that specified the probability of ending up in a particular state if the system has been in another given state, and an action is applied, and an immediate reward (or cost) function that specifies the expected reward obtained during a transition in some units. The objective is to find a policy that maps states to actions, such that some cumulative measure of the reward is optimized. In its basic form, MDPs are not very suitable for representing planning problems, but when the state of the system can be factored over multiple discrete variables, various graphical representations such as dynamic Bayesian networks can be used to represent planning problems under uncertainty in a natural manner. Although MDPs of relatively small size (up to several million states) can be solved exactly by means of algorithms such as value iteration and policy iteration, most practical planning problems have state spaces with far larger sizes, and in such cases approximate solution methods are necessary, such as stochastic dynamic programming [6]. When continuous variables exist in the problem domain, they can be discretized, but discretization on a fine uniform grid is again likely to lead to a corresponding drastic increase in the size of the state space, and might lead to loss of optimality or completely intractability. This is even more valid when the state is not directly observable, but has to be inferred from observations; although it is well understood how to model such domains by means of partially-observable MDPs (POMDPs), the resulting state spaces are huge, and by necessity continuous, because they are defined in terms of beliefs over the set of underlying MDP states. One approach to handling the large state space of planning problems expressed as MDPs and POMDPs is to combine their expressive power with the efficient mechanisms for heuristic search available in classical planners, as proposed in languages such as RDDL [32]. The FF-Replan and FF-Hindsight planners also apply classical planning techniques such as heuristic search and action selection to problems expressed as MDPs, and the POND-Hindsight planner extends this approach to POMDP planning problems, too [28].

Next, we discuss two examples of hard industrial problems that belong to this class of sequential decision problems, and describe concretely why they are difficult to solve by means of the solution methods mentioned above.

2.2. *Contingent unit commitment of thermal generators*

Unit commitment is a classical sequential decision problem of high economic importance. Its objective is to find the optimal schedule for the operation of multiple electric power generators that have operating constraints spanning multiple time steps, given an estimate of the demand for electrical power that will be placed on the whole group of generators. Typical examples of such generators are fossil-burned thermal generators that often need time to warm up, before they can produce their maximum amount of power, and once they have warmed up, it is highly desirable to keep them operating over several hours, in order to avoid structural damage due to frequent thermal expansion and contraction. For the same reason, when they have been turned off, it is desirable to keep them off for several hours before switching them on again. If the decision step is one hour, as typical, these constraints would span multiple time steps. This makes the problem sequential, and typical solution horizons are one day ($T = 24$ time steps) or one week ($T = 168$ time steps).

If there are N thermal generators, the number of possible schedules is $2^{N \cdot T}$; clearly, for most practical instances of the problem, where power utility companies can have tens of units, and require solutions over a day or a week, exhaustive enumeration of all possible schedules is not possible. Moreover, the operating states $u_{i,t}$ of each generator i (on/off) at time step t are only part of the operating schedule – the other part is the actual amounts $o_{i,t}$ that each generator should produce at any given moment in time.

To complicate the decision problem even further, the rapidly increasing adoption of renewable power sources has increased significantly the uncertainty in the net demand to the thermal generators, and that uncertainty must be taken into consideration, if the util-

ity is to achieve power supply reliability as mandated by government regulators. To begin with, even without renewable power sources, the total demand D_t to the utility at time period t is a random variable, typically estimated by means of time series forecasting methods. These methods usually produce estimates not only of the expected demand, but also of its variance. Currently, mean average percentage errors (MAPE) in estimating power demand over short horizons, such as one day, are around 2%. In its turn, supply reliability is often defined in terms of loss of load probability (LOLP), that is, the probability that a loss of load (failure to match demand with the available capacity of generators that are on) would occur for a particular combination of demand and generator configuration. Typical schedules target reliability levels such that loss of load duration should not exceed one day in ten years, approximately equal to LOLP of 3×10^{-4}. In the past, reliability has been ensured by adding a safety margin to the expected demand; a typical number is 3% of total demand, implying that for the usual levels of variability in demand (2%), this kind of margin would be sufficient.

However, this approach only works if variability in demand is constant and known. Arguably, with the increased adoption of renewable power sources, such as wind turbines and solar panels, neither condition is true anymore. The output of a wind turbine can drop suddenly, if the wind dies down; similarly, the output of a solar panel can be reduced significantly, if the sun hides behind a cloud. However, this can happen only if the day was windy or sunny, respectively; if there was no output from these power sources, variability in the output might be a lot lower. Planning for the highest possible variability might not be a good idea: if safety margins increase to 10%–20% of expected demand, the planner would be committing generators to run idly, just for the possibility of countering a sudden drop in renewable generator output. Because such idly running generators do consume fuel and do produce greenhouse gases, just like an idly running car would, this would defeat one of the main purposes of introducing renewable power sources, that of prevention of greenhouse emissions and global warming.

There might be a better way to handle the increased uncertainty in demand to thermal generators resulting from the impact of renewable power sources, if the problem is formulated as a contingent planning problem. Most scheduling algorithms for unit commitment are not contingent: given a sequence of expected demands $\{D_t\}, t \in [1, \ldots, T]$, they will produce a single sequence of generator configurations $\{u_{i,t}\}$ and their expected outputs $\{o_{i,t}\}, t \in [1, \ldots, T], i \in [1, \ldots, N]$ over the entire planning horizon. This sequence will then be executed in an open loop, regardless of what happens while it is being executed. However, as time advances, the expected demands can be revised, and it might become clear that overall demand might be substantially lower or higher than originally expected. It would be beneficial to have instead a contingent plan that can monitor the demand, and tailor the operating plan accordingly.

Moreover, if the power utility owns some of the renewable power generators in its distribution network (the other possibility is that they are distributed generators installed by customers at their own locations), the utility can model explicitly the power output of these generators, with much higher accuracy. In that case, the operating plan can be contingent not only on the current demand, but also on the current output of these generators.

Due to the high economic significance of this problem, a very large number of methods for its solution have been proposed. Within the taxonomy of optimization problems, unit commitment can be recognized as a mixed-integer non-linear program (MINLP), and if the operating cost of a generator is a quadratic function of the power it outputs, it is a mixed-integer quadratic program (MIQP). Due to the size of the problem, exhaustive enumeration of all schedules is typically not possible. Some successful solution methods that have been attempted include Lagrangian relaxation [9,36], heuristic search [22] and dynamic programming [37].

However, these solution methods are applicable mainly to the basic scheduling problem, where a schedule is computed for the entire duration of the planning period, and executed in open loop. For a contingent scheduler, there is at least one more component of the domain dynamics – the evolution of the total demand for electricity. In addition, there might be other components, such as the evolution of the output of renewable power sources. It has been demonstrated that these components are largely predictable, and there is active ongoing research on predicting the demand for electrical power [2] and the output of photovoltaic panels and wind turbines [15]. When such predictive models are included in the dynamics of the domain used for unit commitment, hybrid dynamics emerge, while the actions available at teach time steps are discrete (which generators to turn on/off). Thus, the problem is of the type described in Section 2.1.

Solving the contingent unit commitment problem as a classical optimal control problem does not ap-

pear to be feasible, due to the discrete actions and discrete components of the state space (generator status). A mixed-integer solution approach, although appealing due to its possible eventual true optimality, would have an enormous combinatorial complexity. Classical AI planning algorithms are similarly difficult to apply, because of the continuous components of the state space. In Section 4.1, we will describe a method for solving the problem by reducing it to a discrete-state MDP that can be solved by means of dynamic programming. Other recent work on addressing this problem by means of AI planning methods has focused on applying existing planners for hybrid domains, such as the UPMurphy planner described above [7,8].

2.3. Run-curve optimization

Many transportation problems, such as the energy efficient operation of electrical trains, guided transport systems at airports, or hybrid cars can be reduced to optimizing the velocity profile of a moving vehicle along a one-dimensional path. This velocity profile is called *run curve*, and if the distance along the path is denoted by z, then the desired velocity $v(z)$ at position z describes the run curve. The run curve must obey the legal and mechanical constraints of the route (e.g. speed limits, safety margins), and must be physically realizable by the motors and mechanisms of the vehicle. A good run curve would result in short running times between an origin and destination points, located at $z = 0$ and $z = Z$, respectively, and would also need as little energy as possible to execute. Usually, these two requirements are contradictory: the shorter the running time, the more energy is needed, and vice versa. The problem of computing good run curves thus reduces to optimizing these two criteria (time and energy) simultaneously.

The dynamics of the vehicle can typically be represented by a simplified set of ordinary differential equations concerning the relative position $z(t)$ of the vehicle along the path at time t, and its velocity $v(t)$:

$$\dot{v} = F_a(z, v, a),$$
$$\dot{z} = v,$$

where the function $F_a(z, v, a)$ describes what acceleration would be experienced by the vehicle if action a is applied to it at position z while moving at speed v. This function incorporates the inertia of the vehicle, as represented by its mass and velocity, the slope (gradient) of the path at location z, as well as the air resistance at velocity v. If we represent the state of the vehicle as the vector $x = [z, v]^T$, then we can represent the dynamics by the vector-valued equation $\dot{x} = F(x, a)$. We also assume that the dynamic function F incorporates all existing constraints; for example, if the speed limit is reached, velocity will not exceed it, but remain equal to it.

The instantaneous power consumed by the vehicle is represented by the function $p(z, v, a)$, which we assume depends on position, velocity, and applied control, but is otherwise time independent. When regenerative brakes are used, the function $p(z, v, a)$ can also be negative, representing energy that is generated by the vehicle and returned to the energy source, when it decelerates. (For example, for the case of electrical trains, the source is the catenary system above the tracks.) A given control trajectory $a(t)$, $0 \leqslant t \leqslant T$ would then result in total energy expenditure of

$$E(T) = \int_0^T p\big[z(t), v(t), a(t)\big]\,\mathrm{d}t,$$

where the terminal (end) time T is usually fixed. The goal is to find a function $a(t)$, $0 \leqslant t \leqslant T$, that minimizes the cost functional $J[a(t)]$, subject to the dynamics of the vehicle $\dot{x} = F(x, a)$, and the constraints/conditions $z(0) = 0$, $z(T) = Z$, $v(0) = v(T) = 0$ and $0 \leqslant v(t) < v_{\max}(t)$, where Z is the distance between the origin and destination points, and $v_{\max}(t)$ is the speed limit for location $z(t)$.

This formulation represents an optimal control problem, and it is well known that the optimal function $a(t)$ can be found by solving Eq. (1), known as the Hamilton–Jacobi–Bellman (HJB) equation. If we define the instantaneous cost incurred if control a is applied at state x as $C(x, a)$, and the optimal cumulative cost-to-go until the end destination as $V(x, t)$, the HJB equation allows us to relate the time derivative of V to the instantaneous cost C and the gradient of V in state space [35,38]:

$$\frac{\partial V(x,t)}{\partial t} + \min_u \big\{C(x,a) + \nabla V(x,t) \cdot F(x,a)\big\} = 0. \tag{1}$$

Here the gradient $\nabla V(x, t)$ is the vector of all spatial derivatives of $V(x, t)$ with respect to the state variables, in this case z and v. The instantaneous cost is equal to the consumed power, $C(x, a) \doteq p(x, a)$.

The HJB equation is a partial differential equation (PDE) that is seldom possible to solve analytically. Specifically for run-curve optimization, analytical solutions do not appear to be available, and numerical methods must be applied instead. The traditional method of solving PDEs is to perform numerical discretization by employing either finite differences or finite elements, followed by solution procedures such as Galerkin, Rayleigh–Ritz or collocation [19]. In general, implementing and verifying direct solutions to the HJB equation is quite difficult, and results in slow computation.

One practical simplification then is to limit the set of available actions to only three or four, such as accelerating (a_1), decelerating (a_2), running at a constant speed (a_3), and coasting (moving due to the vehicle's own momentum, a_4). Furthermore, dynamics can be converted to discrete time by the usual transformation

$$x_{k+1} = f(x_k, a_k) = \int_{t_k}^{t_k+\Delta t} F(x, a_k)\,\mathrm{d}t,$$

$$g(x_k, a_k) = \int_{t_k}^{t_k+\Delta t} C(x, a_k)\,\mathrm{d}t$$

which results in dynamics of the type described in Section 2. In discrete time and with discrete actions, the HJB equation (1) reduces to the much simpler Bellman equation:

$$V(x,t) = \min_u \big\{ g(x,a) + V\big[f(x,a), t+\Delta t\big]\big\}. \tag{2}$$

Unfortunately, this equation cannot be solved easily when the state variable x is continuous, because the value function $V(x,t)$ would have to be computed for an infinite number of states. The following section describes how this problem can be addressed by a discrete-state approximation of the dynamics and value function.

3. Barycentric value function approximation

We have been experimenting with a method for solving sequential decision making problems with discrete actions and mixed discrete/continuous states that reduces the problem's dynamics to a Markov decision process. It is based on similarities in the mathematical properties of probability functions and convex combinations. A probability mass function specifies the probability that a random variable is equal to some specified value. For the case of MDPs, the transition function is such a (conditional) probability mass function, conditioned on the starting state s_k and the applied control a_k. The random variable for which the probability function is specified is the successor state s_{k+1}. If the size of the state set S is N, let $s^{(1)}, s^{(2)}, \ldots, s^{(N)}$ be an enumeration of all states. The elements of the transition function can then be defined as $p_{ijl} \doteq \Pr(s_{k+1} = s^{(j)} | s_k = s^{(i)}, a_k = a^{(l)}) = p(s^{(j)} | s_k, a_k)$. From the axiomatic properties of probability mass functions, then, it is always true that $\sum_{j=1}^{N} p_j = 1$, and $p_j \geqslant 0$, $j \in [1, \ldots, N]$.

On the other hand, a convex combination of N vectors y_j, $j \in [1, \ldots, N]$ is defined as $\sum_{j=1}^{N} c_j y_j$, such that $\sum_{j=1}^{N} c_j = 1$, and $c_j \geqslant 0$, $j \in [1, \ldots, N]$. By comparing the two definitions, it can be observed that probability mass functions and the set of coefficients defining a convex combination obey exactly the same mathematical constraints, and a valid probability function can be used as coefficients of a valid convex combination, and vice versa. We use this fact to construct all transition functions of the MDP as sets of coefficients for suitably defined convex combinations, using the procedure described in Algorithm 1.

The algorithm starts with selecting N states $s^{(1)}$, $s^{(2)}, \ldots, s^{(N)}$ such that each corresponds to a state $x \in R^d$ (lines 1–5). We denote the continuous state that corresponds to MDP state $s^{(i)}$ by $x^{(i)}$. Every $x^{(i)}$ is a point (vector) in d-dimensional Euclidean space (Fig. 1). Call the indexed set of points $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$, where the set $\{1, 2, \ldots, N\}$ indexes X. Any selection method is acceptable, for example sampling the continuous state space uniformly, or imposing a kind of a grid and using its vertices as $x^{(i)}$. The extreme points of the continuous domain must also be included in the set X.

The next step is to find the Delaunay triangulation of the set of points X (line 6). The Delaunay triangulation [29] consists of simplices, each of which has $d+1$ vertices, such that each of these vertices is a member of X. In two-dimensional space ($d = 2$, the plane), the simplices are triangles; in three-dimensional space ($d = 3$), the simplices are tetrahedra, etc. Let there be M such simplices. Store the Delaunay triangulation in a suitable data structure, for example a $(d+1)$-by-P matrix D, where each column corresponds to a simplex, and the $d+1$ entries in the column contain indices to the states in the indexed set X.

Algorithm 1. Convert dynamical system to MDP. Algorithm for converting a continuous dynamical system to an MDP with a specified number of discrete states

1: $X \leftarrow \emptyset$
2: **for** $i = 1$ to N **do**
3: $\quad x^{(i)} \leftarrow Sample()$
4: $\quad X \leftarrow X \cup x^{(i)}$
5: **end for**
6: $[D, M] \leftarrow DelauneyTriangulation(X)$
7: **for** $i = 1$ to N **do**
8: $\quad$ **for** $l = 1$ to L **do**
9: $\quad\quad y \leftarrow f(x^{(i)}, a^{(l)})$
10: $\quad\quad m \leftarrow 0$
11: $\quad\quad$ **repeat**
12: $\quad\quad\quad m \leftarrow m + 1$
13: $\quad\quad\quad q \leftarrow x^{(D_{d+1,m})}$
14: $\quad\quad\quad$ **for** $j = 1$ to d **do**
15: $\quad\quad\quad\quad v \leftarrow x^{(D_{j,m})}$
16: $\quad\quad\quad\quad$ **for** $k = 1$ to d **do**
17: $\quad\quad\quad\quad\quad E_{kj} \leftarrow v_k - q_k$
18: $\quad\quad\quad\quad$ **end for**
19: $\quad\quad\quad$ **end for**
20: $\quad\quad\quad c_{1:d} \leftarrow SolveLinear[Ec = (y - q)]$
21: $\quad\quad\quad c_{d+1} \leftarrow 1 - \sum_{j=1}^{d} c_j$
22: $\quad\quad$ **until** $\forall_j c_j \geqslant 0$, $j = 1$ to $d + 1$
23: $\quad\quad$ **for** $j = 1$ to N **do**
24: $\quad\quad\quad p_{i,j,l} \leftarrow 0$
25: $\quad\quad$ **end for**
26: $\quad\quad$ **for** $j = 1$ to $d + 1$ **do**
27: $\quad\quad\quad p_{i,D_{j,m},l} \leftarrow c_j$
28: $\quad\quad$ **end for**
29: $\quad$ **end for**
30: **end for**

Then, for every starting state $s^{(i)}$ (line 7) and each control $a^{(l)}$ (line 8), the algorithm retrieves the point $x^{(i)}$ that corresponds to state $s^{(i)}$, and uses the system function f of the continuous dynamical system to find the successor point y of $x^{(i)}$ under control $a^{(l)}$: $y = f(x^{(i)}, a^{(l)})$ (line 9). In general, the successor point y does not coincide with any of the pre-selected points $x^{(i)}$, $i \in [1, \ldots, N]$. It is assumed that the successor point is within the limited domain bounded by the points in X; if necessary, the dynamics should be restricted to keep y within bounds.

After that, the algorithm finds the simplex in the Delaunay triangulation that contains the point y (lines 11–22). To this end, it traverses all M simplices in the Delaunay triangulation and repeats the following steps for every simplex m, $m \in [1, \ldots, M]$. First, it retrieves the last, $(d + 1)$-st vertex of simplex m, and stores it

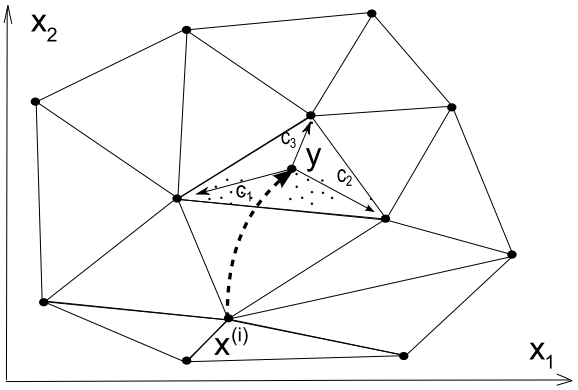

Fig. 1. Representation of continuous dynamics over a tessellation of the continuous state space obtained by means of Delaunay triangulation. The barycentric coordinates of the successor state with respect to the vertices of the enclosing simplices are used as the transition probabilities of the approximating MDP.

in vector q (line 13). Then, it creates a d-by-d matrix E, whose column j contains the difference $v_{m,j} - q$ between the jth vertex of simplex m (denoted here by $v_{m,j}$) and the vector q, for $j \in [1, \ldots, d]$ (lines 14–19). Then, it finds the d-dimensional vector c such that $Ec = (y - q)$ by solving the set of d simultaneous linear equations (line 20). The last, $(d + 1)$-st element of the vector c is computed as $c_{d+1} = 1 - \sum_{j=1}^{d} c_j$ (line 21). For every element c_j, $j \in [1, \ldots, d + 1]$, if $c_j < 0$, then simplex m does not contain point y; in that case, increase m by one and test the next simplex. If all $c_j \geqslant 0$, $j \in [1, \ldots, d + 1]$, then this simplex contains point y (line 22).

At this point, the $(d + 1)$-dimensional vector c contains coefficients that define a valid convex combination such that $y = \sum_{j=1}^{d+1} c_j v_{m,j}$. Moreover, it defines a valid probability transition function, since all of its entries are positive and sum up to unity. In order to construct a complete transition probability distribution over all possible N successor states, we perform the following step for each state $s^{(j)}$, $j \in [1, \ldots, N]$. If $s^{(j)}$ corresponds to one of the vertices of the simplex m, that is, $x^{(j)} = v_{m,r}$ for some $r \in [1, \ldots, d+1]$, then the corresponding transition probability of the MDP is $p_{ijl} = \Pr(s_{k+1} = s^{(j)} | s_k = s^{(i)}, a_k = a^{(l)}) = c_r$ (lines 26–28); otherwise, $p_{ijl} \doteq 0$ by default (lines 23–25).

Conceptually, we can think of this algorithm as a way of converting the system dynamics represented by the function f to an equivalent probabilistic representation involving only a small set of points $s^{(i)}$ embedded into the original continuous state space of the system. If the system starts in one of these few points, the

successor state y, in general, will not coincide with another one of these points. However, we can identify the $d+1$ points that define a simplex that completely encloses the successor state y, and can think that the system has transitioned not to point y itself, but to the vertices of this simplex with various probabilities, instead. The probabilities are equal to the convex decomposition of point y with respect to the vertices of the simplex, also known as the barycentric coordinates of that point within the simplex. The similarities between convex combinations (barycentric coordinates) and probability mass functions required by the MDP formalism make this conversion possible.

Once the continuous domain dynamics have been converted to a discrete MDP, we can find a suitable policy for the MDP by a number of different methods, depending on the size of the state space and the structure of the transition function of the MDP. When the number of states is sufficiently small (e.g., several million), an optimal policy $a = \pi^*(s)$ for that MDP can be found by means of policy iteration or value iteration [30]. For example, for the discounted case, we can evaluate the value function $V(s^{(i)})$ of the MDP by means of repeated applications of Bellman backups of the form

$$\begin{aligned} V\left(s^{(i)}\right) &= \min_l \left\{ g\left(s^{(i)}, a^{(l)}\right) \right. \\ &\qquad \left. + \gamma \sum_{j=1}^{N} \Pr\left(s^{(j)} \middle| s^{(i)}, a^{(l)}\right) V\left(s^{(j)}\right) \right\} \\ &= \min_l \left\{ g\left(s^{(i)}, a^{(l)}\right) + \gamma \sum_{j=1}^{N} p_{ijl} V\left(s^{(j)}\right) \right\} \end{aligned}$$

for each state $s^{(i)}$, $i \in [1, \ldots, N]$, until convergence of the value function $V(s^{(i)})$. The optimal policy for the MDP is then $\pi^*(s^{(i)}) = \arg\min_a Q(s^{(i)}, a)$, where we make use of the auxiliary function $Q(s^{(i)}, a) \doteq g(s^{(i)}, a) + \gamma \sum_{j=1}^{N} p_j V(s^{(j)})$. The computational complexity of this solution is only O(Nd), because in each iteration of the value iteration algorithm, a Bellman backup is performed for each of the N anchor states, and although the sum is over all N successor states $s^{(j)}$, there are non-zero transition probabilities to only $d+1$ of them. These non-zero values can either be stored explicitly during conversion, or the entire transition probabilities can be placed in sparse matrices.

It is, however, often the case that the continuous dynamics are only one part of the whole dynamics of a domain; there can easily be one or more discrete components, plus other continuous sub-systems, too. For such cases, a suitable formalism is a factored MDP (fMDP), where the system state is factored over multiple variables, and dynamics consist of relatively independent parts that interact weakly or not at all. In many cases, the coupling can be only through the cost structure of the fMDP. When such structure is available, it can be leveraged computationally by decision-theoretic planning methods to find optimal policies for fMDPs whose total state space size would otherwise make policy and value iteration intractable [6].

Furthermore, the ultimate goal when solving sequential decision problems with continuous components of state space is to find a control law $a = \mu^*(x)$ that is a mapping from the *continuous* (or *hybrid*) state x, as opposed to the discrete state of the MDP s. By recognizing that our method introduces uncertainty about the state the system is in, we can use several control strategies from the field of partially observable Markov decision processes (POMDP) [10]:

Nearest anchor point: find the closest anchor point $x^{(i)}$ to x in the embedding continuous space in terms of Euclidean distance, and use the optimal action for the corresponding MDP state $s^{(i)}$: $a = \pi^*(s^{(i)})$.

Largest vote: find the simplex m that contains x, and compute the barycentric coordinates c of x with respect to the $d+1$ vertices $v_{m,j}$, $j \in [1, \ldots, d+1]$ of that simplex, identically to the procedure described in lines 10–22 in Algorithm 1. Then, if $a_j = \pi^*(s^{(j)})$, where $s^{(j)}$ is the state corresponding to vertex $v_{m,j}$, we can use c_j as an individual vote for action a_j, and execute the action that has the highest cumulative vote over all $d+1$ vertices.

Highest expected merit: use the barycentric coordinates to estimate the merit $\hat{Q}(x,a)$ of the individual action a taken in state x as $\hat{Q}(x,a) = \sum_{j=1}^{d+1} c_j Q(s^{(j)}, a)$, and use the control law $\mu^*(x) = \arg\min_a \hat{Q}(x,a)$. Given that the barycentric coordinates c can be interpreted as individual probabilities that the MDP is in one of its discrete states, the function $\hat{Q}(x,a)$ is indeed the exact expected merit of taking action a in the continuous state x.

4. Application of barycentric quantization to problems

4.1. Contingent unit commitment

The proposed method for barycentric quantization of a continuous state space problem that results in a

discrete MDP has been applied to several practical problems. In [27] and [39], we describe the application of the proposed method to contingent unit commitment, and two possible methods for solving the resulting large factored Markov decision process.

In this domain, the application of the quantization method results in a factored Markov decision process models represented in the form of dynamic Bayesian networks (DBN). This representation is relatively compact and also easy to specify, maintain, and extend with new power sources. This approach is similar to that taken in the planning language RDDL, which also uses DBNs augmented with influence diagrams to represent the planning problem [32].

We have also proposed one concrete algorithm for finding such conditional operational schedules for power generation that depend on a single random variable – the net demand that aggregates in itself all sources of randomness. The algorithm focuses on small subsets of all possible configurations of generators in order to compute the schedule efficiently. Experimental results suggest that the resulting conditional plans are close to the truly optimal ones, and provide a much better trade-off between generation cost and risk of failure to meet demand than two known non-stochastic unit commitment algorithms that compute fixed schedules.

In the proposed solution algorithm, we use AND/OR trees to represent, find, and evaluate the optimal conditional plan. However, this algorithm is by no means the only possible way to solve stochastic generation problems represented by means of fMDPs and DBNs. We have also investigated another solution method based on approximate dynamic programming that could address much larger unit commitment problems [39]. The method works with a reduced state space whose size is polynomial in the number of generators and the number of selected target demand levels. We also proposed a functional metric to measure the similarity of the states in the reduced space. We employed an approximate dynamic programming method in which when a state's value is not updated, its most similar state can be found, and its value can be used instead. The proposed method yields lower costs and lower operational risk than the deterministic methods, and can solve larger problems than the previously developed decision space approximate method that relies on a tree structure and suffers from an exponential growth problem.

Multiple improvements to both solution algorithms are possible. The current solution algorithms aggregate the variability of all stochastic variables into the net demand to the controllable power generators, for the sake of computational efficiency. This simplifies the planning problem, because the branching in the AND/OR tree is based only on that single variable. However, even higher efficiency might be possible if the conditional schedule is conditioned on the values of each individual stochastic component. This would significantly increase the complexity of the planning process, and would depend critically on finding more computationally efficient solution methods for the underlying fMDP models.

For example, the method proposed in [16] represents the value function of the dynamic programming problem over continuous domains by adaptively discretizing such continuous variables. This approach might result in more accurate and compact representations than are possible with our method, where the tessellation of the continuous domains is performed a priori, before value functions are evaluated. Adaptive discretization is indeed compatible with our discretization scheme, too, for example by sub-dividing a simplex where the value function varies a lot (measured on its vertices), into multiple smaller simplices. The application of symbolic dynamic programming (SDP, [33]) to the factored MDP-based formulation of the operational planning problem might be possible, too.

Furthermore, the formulations of the fMDP that we have used so far assume that all generators will take on their intended configuration a_i^t without fail. This allows us to use the decision variables a_i^t as components of the state of the system, thus simplifying the planning process. If the possibility of equipment failure must be taken into account, the actual configuration A_i^t of the generators should be included as a random state variable in the DBN, and its probabilistic dependence on the intended configuration a_i^t can be modeled according to the failure probabilities of individual generators. Such an extension is completely compatible with the proposed modeling formalism of factored Markov decision processes.

4.2. Run-curve optimization

In [25], we discuss the application of the method to the problem of computing optimal run-curves for trains, for several instances of the problem. In that domain, if time is explicitly represented in the value function, as in Eq. (2), the number of states of the resulting MDP is too large for practical computation. However, several improvements are possible, resulting in three different methods for converting train dynamics

and run-curve optimization problems into MDPs. Of these, a method that creates MDPs with equal-distance steps, as opposed to the more customary equal-time steps, gave the best results in terms of computational time and accuracy, because such MDPs have no recurrent transitions, and can be solved very quickly by means of dynamic programming. Experimental results suggest that discretization steps of 20 m in distance and 2 km/h in velocity are sufficient for computation of accurate optimal run curves. Computational times shorter than 4 s for railroad track segments of length 2 km were achieved, allowing re-computation of optimal run curves to happen at each train station, while the train stops there and takes on new passengers [21].

Future work will focus on methods for further speeding up the computation of the optimal policy, and representing the control law compactly. Intuitively, the state space contains many states that the train simply cannot be in – either cannot get to them at that time, or if it is there, cannot get to the destination station on time. It might be possible to prune these states out of the state space of the MDP, speeding up computation considerably.

4.3. Scheduling of air conditioners

In [26], we describe one more application domain where this method can be applied: optimal scheduling of heating, ventilation, and air conditioning (HVAC) equipment. The domain of interest, building thermal dynamics, can be described sufficiently accurately by a low-order thermal circuit model [20]. The proposed method converts the continuous-state building dynamics into a discrete Markov decision problem. In that case, finding the optimal control policy for the converted MDP representation is computationally very efficient, and reduces to backward dynamic programming. This procedure is linear in the number of states of the MDP, and the overall computational time can be varied according to needs and the available computational resources by means of adjusting the number of states. The favorable computational complexity of the algorithm can be employed in a receding horizon controller for continuous re-optimization of the set-points of HVAC systems.

In a set of experiments with a single zone in cooling mode, the MDP-based scheduler significantly outperformed traditional scheduling strategies sometimes saving air conditioning costs in excess of 50%. Further research will extend this method to multi-zone buildings, and will address the influence of model inaccuracy and uncertainty in input data (outdoor temperature, etc.). Another possibility for improvement of the method is to also include humidity both in the building thermal model and the definition of the comfort zone for building occupants.

5. Conclusion

We have addressed the problem of solving sequential decision making in domains with discrete actions where at least some components are continuous, and have argued that a viable and generally applicable solution method for such problems can involve barycentric quantization of the continuous components of the state space into a discrete MDP, followed by exact MDP solution methods such as value and policy iteration, or approximate dynamic programming. Arguably, the method itself can be viewed as an approximate dynamic programming method over continuous domains.

Many numerical methods for solving optimal control problems have been proposed in the control systems, operations research, and reinforcement learning communities. The specific instance of the problem that we have considered here, namely continuous state space and discrete time and actions, is a special case of the more general optimal control problem, where all variables (state, actions and time) are continuous. As such, it can be solved by means of general optimal control methods, including both indirect and direct methods. In indirect methods, the problem is formulated as a two-point boundary-value problem, and its solutions are the desired control and state trajectories. A disadvantage of such indirect methods is that solving boundary-value problems numerically can be very difficult, error prone and computationally demanding. In direct methods, the state and control trajectories are approximated by means of appropriate parametric approximators, and the corresponding parameters are estimated by means of general-purpose nonlinear optimization methods. An example is the direct collocation method, currently implemented in powerful modeling and optimization tools and languages such as JModelica/Optimica [1]. Although such methods can be very fast and effective, the direct result of their computation is not a control law, but an optimal state and control trajectory, whereas the method proposed in this paper computes an entire control law over the entire state space. Furthermore, the MDP constructed by the proposed algorithm can be extended to handle uncertainty in system dynamics by modifying its transition

probabilities, whereas direct optimal control methods would have to solve a stochastic optimal control problem, which is much harder to solve than the deterministic case.

The reinforcement learning and operations research communities have also proposed multiple algorithms for solving sequential optimal decision and control problems, commonly incorporating the MDP framework. One of the dominant ideas has been to use universal value function approximators, such as feed-forward neural networks, radial-basis functions, k-nearest neighbor, etc., to represent the value function of a sequential decision problem over the entire state space of the problem, and then minimize the residual in the Bellman equation at select points, fitting the parameters of the function approximators in the process. To a certain degree, the method proposed in this paper is based on the same idea for using a universal function approximator to represent the value function, in this case piecewise linear approximation over a collection of simplices. However, two important differences exist. First, the proposed method approximates the dynamical system by an MDP, and then calculates the value function of the MDP *exactly*, as opposed to approximating the value function of the original system in the process of estimating it. This has important consequences as regards the convergence guarantees of the method. Since the method constructs a standard MDP, the convergence of the solution procedure used, such as value iteration, policy iteration, linear programming, etc., is guaranteed, and the rates of convergence can be estimated based on existing research [30]. In contrast, the convergence of value iteration when used with an arbitrary universal function approximator is not at all guaranteed, and research has shown that many popular function approximator schemes may in fact lead to divergence [18].

The second difference between the proposed method and most solution methods from the field of model-free reinforcement learning such as Q-learning and TD(λ) is that such methods use the system dynamics only as a source for sampling system transitions, whereas the proposed method uses the system dynamics directly for the exact calculation of the transition probabilities of the MDP. This has the practical consequence that once the MDP model is constructed, finding the optimal control law over the entire state space is very fast (O(Nd)), whereas estimating value functions and optimal control policies from sampled system transitions can be excruciatingly slow.

We have also described several hard industrial problems that are amenable to the proposed method. In all cases, superior performance in comparison to domain-specific solution methods has been demonstrated. Still, the best performance is achieved when the proposed method is customized to the respective domain, although this has more to do with the size of the state space and the nature of the dynamics of the domain, than with the general applicability of the method. This gives hope that the method can be applied to even more decision problems of this class.

References

[1] J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl and H. Tummescheit, Modeling and optimization with Optimica and JModelica.org – Languages and tools for solving large-scale dynamic optimization problems, *Computers & Chemical Engineering* **34**(11) (2010), 1737–1749.

[2] H.K. Alfares and M. Nazeeruddin, Electric load forecasting: Literature survey and classification of methods, *International Journal of Systems Science* **33**(1) (2002), 23–34.

[3] D.P. Bertsekas, *Dynamic Programming and Optimal Control, Volume 1*, Athena Scientific, Belmont, MA, 2000.

[4] D.P. Bertsekas, *Dynamic Programming and Optimal Control, Volume 2*, Athena Scientific, Belmont, MA, 2000.

[5] S. Bogomolov, D. Magazzeni, A. Podelski and M. Wehrle, Planning as model checking in hybrid domains, in: *AAAI Conference*, 2014.

[6] C. Boutilier, T. Dean and S. Hanks, Decision-theoretic planning: Structural assumptions and computational leverage, *Journal of Artificial Intelligence Research* **11** (1999), 1–94.

[7] J.R. Campion, Short term unit commitment as a planning problem, Durham thesis, Durham University, 2014, available at Durham e-theses: http://etheses.dur.ac.uk/10650.

[8] J.R. Campion, C. Dent, M. Fox, D. Long and D. Magazzeni, Challenge: Modelling unit commitment as a planning problem, in: *International Conference on Automated Planning and Scheduling ICAPS*, 2013.

[9] P. Carpentier, G. Gohen, J.-C. Culioli and A. Renaud, Stochastic optimization of unit commitment: A new decomposition framework, *IEEE Transactions on Power Systems* **11**(2) (1996), 1067–1073.

[10] A.R. Cassandra, L.P. Kaelbling and J.A. Kurien, Acting under uncertainty: Discrete Bayesian models for mobile robot navigation, in: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[11] A.J. Coles, A.I. Coles, M. Fox and D. Long, COLIN: Planning with continuous linear numeric change, *Journal of Artificial Intelligence Research* **44** (2012), 1–96.

[12] T.L. Dean and M.P. Wellman, *Planning and Control*, Morgan Kaufmann, San Mateo, CA, 1991.

[13] G. Della Penna, D. Magazzeni and F. Mercorio, A universal planning system for hybrid domains, *Applied Intelligence* **36**(4) (2012), 932–959.

[14] G. Della Penna, D. Magazzeni, F. Mercorio and B. Intrigila, UPMurphi: A tool for universal planning on PDDL+ problems, in: *International Conference on Automated Planning and Scheduling ICAPS*, AAAI Press, 2009, pp. 106–113.

[15] B. Ernst, F. Reyer and J. Vanzetta, Wind power and photovoltaic prediction tools for balancing and grid operation, in: *Integration of Wide-Scale Renewable Resources Into the Power Delivery System, 2009 CIGRE/IEEE PES Joint Symposium*, July 2009, 2009, pp. 1–9.

[16] Z. Feng, R. Dearden, N. Meuleau and R. Washington, Dynamic programming for structured continuous Markov decision problems, in: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, UAI'04*, AUAI Press, Arlington, VA, USA, 2004, pp. 154–161.

[17] M. Fox and D. Long, Modelling mixed discrete-continuous domains for planning, *Journal of Artificial Intelligence Research* **27**(1) (2006), 235–297.

[18] G.J. Gordon, Stable function approximation in dynamic programming, in: *Proceedings of the International Conference on Machine Learning*, 1995, pp. 261–268.

[19] C.A. Hall and T.A. Porsching, *Numerical Analysis of Partial Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[20] M.J. Jimenez and H. Madsen, Models for describing the thermal characteristics of building components, *Building and Environment* **43**(2) (2008), 152–162.

[21] S. Kimura, K. Yoshimoto, K. Ueda, S. Takahashi and D.N. Nikovski, Markov decision process-based run curve optimization for energy saving and ride comfort, *IEEJ Transactions on Electronics, Information and Systems* **134**(10) (2014), 1577–1583.

[22] C. Li, R.B. Johnson and A.J. Svoboda, A new unit commitment method, *IEEE Transactions on Power Systems* **12**(1) (1997), 113–119.

[23] H.X. Li and B.C. Williams, Generative planning for hybrid systems based on flow tubes, in: *Proceedings of the International Conference on Automated Planning and Scheduling ICAPS*, 2008, pp. 206–213.

[24] J. Löhr, P. Eyerich, T. Keller and B. Nebel, A planning based framework for controlling hybrid systems, in: *International Conference on Automated Planning and Scheduling ICAPS*, 2012.

[25] D. Nikovski, B. Lidicky, W. Zhang, K. Kataoka and K. Yoshimoto, Markov decision processes for train run curve optimization, in: *Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS)*, 2012, 2012, pp. 1–6.

[26] D. Nikovski, J. Xu and M. Nonaka, A method for computing optimal set-point schedules for HVAC systems, in: *Proceedings of the 11th REHVA World Congress, CLIMA 2013*, 2013.

[27] D. Nikovski and W. Zhang, Factored Markov decision process models for stochastic unit commitment, in: *2010 IEEE Conference on Innovative Technologies for an Efficient and Reliable Electricity Supply (CITRES)*, September 2010, 2010, pp. 28–35.

[28] A. Olsen, Pond-Hindsight: Applying hindsight optimization to partially-observable Markov decision processes, MS thesis, All Graduate Theses and Dissertations, Paper 1035, University of Utah, 2011.

[29] F.P. Preparata and M.I. Shamos, *Computational Geometry*, Springer Verlag, Heidelberg, 1990.

[30] M.L. Puterman, *Markov Decision Processes – Discrete Stochastic Dynamic Programming*, Wiley, New York, NY, 1994.

[31] S.J. Russell and P. Norvig, *Artificial Intelligence. A Modern Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1995.

[32] S. Sanner, Relational dynamic influence diagram language (RDDL): Language description, Australian National University, 2010, unpublished manuscript.

[33] S. Sanner, K.V. Delgado and L.N. de Barros, Symbolic dynamic programming for discrete and continuous state MDPs, in: *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, UAI'11*, AUAI Press, Arlington, VA, USA, 2011, pp. 643–652.

[34] J.-A. Shin and E. Davis, Processes and continuous change in a sat-based planner, *Artificial Intelligence* **166**(1) (2005), 194–253.

[35] R.F. Stengel, *Optimal Control and Estimation*, Dover, Mineola, NY, 1986.

[36] S. Takriti, J.R. Birge and E. Long, A stochastic model of the unit commitment problem, *IEEE Transactions on Power Systems* **11**(3) (1996), 1497–1508.

[37] A.J. Wood and B.F. Wollenberg, *Power Generation, Operation, and Control*, Wiley, New York, 1996.

[38] H. Zhang, D. Liu, Y. Luo and D. Wang, *Adaptive Dynamic Programming for Control: Algorithms and Stability*, Springer Science & Business Media, 2012.

[39] W. Zhang and D. Nikovski, State-space approximate dynamic programming for stochastic unit commitment, in: *North American Power Symposium (NAPS)*, August 2011, 2011, pp. 1–7.