IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 17, 2019, accepted August 26, 2019, date of publication September 3, 2019, date of current version September 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939161

An Approximate Quadratic Programming for Efficient Bellman Equation Solution

JIANMEI SU, (Student Member, IEEE), HONG CHENG, (Senior Member, IEEE), HONGLIANG GUO, RUI HUANG, (Member, IEEE), AND ZHINAN PENG

School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

Corresponding author: Hong Cheng (hcheng@uestc.edu.cn)

This work was supported by the National Key Research and Development Program of China for "New Energy Automobile", the Key Technology Research and Demonstration Operation of Electric Autonomous Vehicle Project under Grant 2017YFB0102603.

ABSTRACT This paper proposes an efficient algorithm which relies on quadratic programming for approximately solving the Bellman equation in reinforcement learning problem and guarantees to return optimal decision parameters. Through further applying universal approximation and fixed cardinality minimization techniques, the proposed algorithm in one hand expands the representation ability of basic linear value functions, on the other hand, it guarantees the convergence of the Bellman error. Experimental results on two canonical reinforcement learning scenarios demonstrate that the proposed algorithm achieves similar or better performance than the state-of-the-art algorithms, while reduces the computation time significantly and improves the robustness of the algorithm against state uncertainty.

INDEX TERMS Markov decision processes, approximate quadratic programming, Bellman equation solutions, universal approximation, fixed cardinality.

I. INTRODUCTION

Reinforcement learning(RL) has been a hot topic with the wide spread of the autonomous intelligent in the real world such as autonomous driving [1], play the game of go [2] and so on. The study of RL is how an agent interacts with its environment to learn optimal strategies for sequential decision problems by maximizing the expected cumulative rewards [3]. In most RL environments, the scenario can be modeled as the Markov Decision Processes (MDPs) framework. Solving large MDPs has attracted growing interests from researchers and enterprises, which is a very useful, but with computationally challenging problem. Fortunately, many advanced approximating approaches are proposed in the past few years, for example, the parameter identification of nonlinear equations with fully unknown parameters for generalized synchronization are well studied [4]–[6]. Additionally, With the widespread of the dynamic neural networks [7], topology identification and dynamic properties with time delay in neural networks [8] are widely investigated. Module-phase synchronization [9], projective synchronization [10] are derived in analysis which can be applied in the communication and many other fields. Furthermore, the optimality is also needed to be ensured in the control system [11]. Based on the Bellman optimality principle, adaptive dynamic programming for optimal control of nonlinear system is proposed by using a function approximation structure [12]. For most large MDPs, it can not obtain the optimal solution exactly. Therefore, a large number of researchers approximate the MDPs solution with a variety of mathematical programming approaches and mainly rely on Linear Programming(LP). Value function approximation with numbers of the linear basic function is a common approach for accessing approximate optimal solutions. For most LP methods for value function approximation of the Bellman equation solutions, the objective is the minimization of the value function.

Motivated by the above discussions, in this paper, we discuss an Approximate Quadratic Programming(AQP) algorithm, which is able to return the optimal decision parameters. We first transform the canonical Bellman equation into a set of linear constraints and then set up a Quadratic Programming (QP) problem to find the initial decision parameters. With the fixed cardinality minimization technique, we reformulate the previous QP problem into an LP problem, with this operator, the Bellman error is guaranteed to converge.

The algorithm transforms the Bellman equation solution process into standard mathematical programming

The associate editor coordinating the review of this article and approving it for publication was Yongming Li.

frameworks, i.e. QP and LP. While the actual solution process of QP and LP may still rely on iterated computation such as interior point method [13], we argue that representing the Bellman equation solution into compact mathematical programming problems, which improves the superiority of the solution process a lot, as there are many effective tools for improving the efficiency of the LP/QP solution [14].

Our contribution can be concluded as follows: (1) The AQP algorithm is proposed to solve the Bellman equation via an approximate manner, in which the Bellman equation solution process is modeled as classical mathematical programming forms and guarantees to return the optimal decision parameters; (2) Bellman error is directly treated in the objective of the formulation which makes the AQP algorithm converge to the minimal Bellman error, and we enhance the robustness property of AQP solution for state uncertainties; (3) relying on the Universal Approximation (UA) theory, the representation ability of the linear functions for optimal value function approximation is well expended, so as to reduce Bellman error, the fixed cardinality minimization technique guarantees the convergence of Bellman error. Extensive experimental results demonstrate the advantages of AQP over the state of the arts.

The remainder of this paper is organized as follows. In Section II, we introduce the literature review, Approximate Linear Programming (ALP) for solving MDPs, the introduction of UA and cardinality minimization algorithms. In Section III, we give the background of MDPs. Section IV presents the overview of the proposed algorithm. In Section V, we describe the practical implementation of the proposed algorithm, furthermore, we provide the computational complexity and ended with the core deployment. Section VI elaborates the potential improvement and generalizes the proposed algorithm to other applications. Experimental results and corresponding analysis are presented in Section VII. The conclusions and future work are given in VIII.

II. LITERATURE REVIEW

In this section, we will review canonical LP methods for Bellman equation solutions, introduce the universal approximation theory and cardinality minimization technique. As we are essentially proposing to use two canonical mathematical programming approaches (QP and LP) to solve the Bellman equation, we also give a brief review on LP approaches for MDP solutions in one of the subsections.

A. APPROXIMATE LINEAR PROGRAMMING APPROACHES FOR MDP SOLUTIONS

Date back to the research of Schweitzer and Seidmann [15], optimal value function can be obtained by considering linear approximations that combines a series of basic functions. Therefore, choosing basic functions for good approximation is one key factor for solving large MDPs. Following that, the original problem becomes to find the coefficients of these basic functions for giving a good approximation which is sufficient and efficiently without considering the large number of sates. Therefore, ALP approaches is well studied. By solving an LP with large number of linear constrains, the optimal solutions usually lie in the subspace spanned by the basic functions. Utilizing a linear combination of basic function architectures and combining with a mechanism to sample a small subset among the constraints, the parameters of the optimal solution are within a constant factor of the best approximation [16]–[18]. But, the approximate solution of ALP is limited by the large number of the constraints, therefore, the approximation of the constraints becomes a hot interests for many researchers [19], [20]. In fact, the constraints number is strong related with the state space, it is necessary to investigate the algorithm which can be done in time independent of the states [21]. When the state space is large enough, the best strategy with low excess loss can be found in a small family of the policies [22].

Selecting a tractable subset of the constraints while keeping the solution approximation well, as well as keeping computation tractable are worth to well investigate. In the LP formulation, linear objective is optimized by a point on the boundary of the feasible region, thus there are many constraints which is equal with the number of optimization variable. It is important to find a superset of the these linear constraints that can well approximate but without increasing much computation time. Guestrin *et al.* [23] designed factorized MDPs for constraints generation to generate violated constraints efficiently. de Farias and Van Roy [17], [18] give a more general approach by choosing states, and obtain a subset of the constraints for solving the LP problem. Further more, they impose an extra constraint on the optimization variables by obtaining the bound that scales with the worst approximation error. Desai *et al.* [24] and Bhat *et al.* [25] make some extensions with other algorithms. However, they are all based on LP with the values optimization, and do not aim at the Bellman errors convergence and the robustness of the coefficients.

In this paper, the proposed approach aims at minimizing the Bellman error directly in the objective function. Further more, by incorporating the minimization of the solution parameters, the robustness of the strategy for the agent in its scenarios is well improved. With these properties, the proposed approach shows good performance on efficiency and practicability in real world.

B. UNIVERSAL APPROXIMATION THEORY

Based on conventional neural network theories, universal approximation theorem states that a single-hidden-layer feed forward networks with numbers of hidden nodes [26], [27] which holds strong approximation ability that can approximate any continuous functions with numbers of neurons in the hidden layer under mild assumptions on the activation function [28], [29].

The incremental constructive method can randomly choose hidden nodes and then only need to adjust the output weights linking the hidden layer and the output layer [30]. The number of hidden nodes and the approximation error can be

examined [27] which ensures the modelling performance without over-fitting. In principle, the desired level of accuracy can be reached by increasing the number of hidden nodes [31], [32]. As deep learning is derived, deep stochastic configuration networks is widely used with the extension of traditional universal approximation with super representation [33]–[35].

C. CARDINALITY MINIMIZATION TECHNIQUE

The cardinality is represented by the number of elements which are not 0 in the matrix, such as the cardinality of $\vec{y}$ is 1, when the vector is $\vec{y} = [y_1, y_2, y_3, y_4] = [0, 0, 0, 2]$. Solving the minimization problem for a cardinality mechanism can be regarded as a series of optimization issues [36]. These problems talked before are usually deployed in graph-related problems with randomized approaches, as random separation [37], [38], which considers the solutions with exactly a fixed number k of elements that optimizes the solution values. Matrix rank minimization problem can be viewed as a special cardinality minimization problem and can be solved through iterated reweighed ℓ_1 algorithm [39]. Classical compressed sensing problem is to find sparsest solution to an under-determined system of linear equations. A good convex approximation is to minimize the ℓ_1 norm subject to the affine constraints [40].

III. BACKGROUND

We consider Markov decision processes (MDPs) model defined formally as a tuple: $\langle S, A, P^{s'}_{sa}, R(s,a), \gamma \rangle$, where S is the set of the states s, A is the set of the actions a, P is the transition from the current state s to the next state s', $R(s,a) \in \mathbb{R}$ is the reward function and $\gamma \in (0,1)$ is the discount factor [3]. The problem in the MDPs is that an agent reaches a given destination with an optimal policy which can be found by maximizing discounted future rewards $E(\sum_{k=0}^{\infty} \gamma^k R_k)$.

In order to obtain the optimal policy for the agent, the max value of each state can be calculated by approximately solving the Bellman equation under the optimal action chosen in the state i.e., $V^*(s) = \max_a \left(\gamma \sum_{s'} P^{s'}_{sa} V^*(s') + R(s,a)\right)$.

LP is a classical method for obtaining approximate optimal $V^*(s)$, the objective is to minimize the sum of the states' values under the proper constraints,

$$\begin{aligned} &\text{minimize} \sum_s V^*(s) \\ &\text{subject to } (\forall s, a) V^*(s) \geq R(s,a) + \gamma \sum_{s'} P^{s'}_{sa} V^*(s') \end{aligned} \quad (1)$$

ALP mainly aim at calculating the optimal approximate value function for large MDP which can be formulated as:

$$\begin{aligned} &\text{minimize} \sum_{s \in S} \rho(s) V(s) \\ &\text{subject to } V(s) \geq R(s,a) + \gamma \sum_{s' \in S} P^{s'}_{sa} V(s') \end{aligned} \quad (2)$$

where ρ is the distribution for the states, thus we have a set of state-relevance weight associate with every state in the optimization criterion. The constraints are for all $(s,a) \in (S,A)$ and $\sum_{s \in S} \rho(s) = 1$.

In many real world applications, the continuous state space is too large even infinite. In those cases we approximate the value function via a linear combination of the feature functions for each state. The variables in the ALP are the weights assigned to each basic function and the value of each state is computed as $\phi(s)^T \omega$, where $\phi(s)$ is the feature vector for the corresponding state, which is assumed to be constant for feasibility implementation. ω is the weight vector. The feature based linear program becomes:

$$\begin{aligned} &\text{minimize} \sum_s \rho(s) \phi^T(s) \omega \\ &\text{subject to } (\forall s, a) \phi(s)^T \omega \geq R(s,a) + \gamma \sum_{s'} P^{s'}_{sa} \phi(s)^T \omega \end{aligned} \quad (3)$$

It reduces the number of variables in the linear program by using features, but it does not reduce the number of constraints. However, the constraints number is larger than the number of variables, we usually try to reduce the constraints number to save the computation cost. With a assumptions over the sampling distribution [18], we utilise regularization [41] to sample large number of the constraints. By this way, the probability and the performance degradation generated by the non-sampled constraint or the missing constraints are bounded. The weights of the state-relevance will influence the programming solutions, a trade-off in the quality of the approximation across different states [17] can be imposed in the formulation.

IV. OVERVIEW OF OUR METHODOLOGY

In this paper, we propose an AQP algorithm, which returns the optimal decision parameters for Bellman equation solutions. With a linear approximator for the optimal value function representation, we first transform the canonical Bellman equation into a set of linear constraints and then set up a QP problem to find the initial decision parameters and the affiliated approximation errors. With the fixed cardinality minimization technique, we reformulate the previous QP problem into an LP problem, and Bellman error is guaranteed to converge.

Furthermore, we enhance our algorithm over miscellaneous factors, such as robustness against observation uncertainty and runtime goal programming problem. As the linear value function representation ability is limited, we use universal approximation strategy to expand the linear function representation ability. Moreover, the fixed cardinality minimization technique guarantees the convergence of the Bellman error. With UA and cardinality minimization, the expanded linear value function approximation and the convergence of the Bellman error improve the representation ability and make the agent perform near optimally.

V. APPROXIMATE QUADRATIC PROGRAMMING METHODOLOGY

This section introduces the AQP methodology in details. We first give the traditional QP formulation, then lay out how

we have used a QP problem formulation to approximate the Bellman equation solution. Then we display the UA theory for expanding the linear value function representation ability, followed by the fixed cardinality minimization technique to guarantee the convergence of Bellman error. After computation complexity analysis, we end the section with a pseudo code depicting the flow of the AQP methodology.

A. THE QUADRATIC PROGRAMMING FORMULATION WITH LINEAR CONSTRAINTS

QP is a special mathematical programming problem in nonlinear programming which has been widely applied in portfolio, constrained least squares problem solving, sequential quadratic programming in nonlinear optimization problems [42]. The nonlinear objective in the optimization problem always be approximated by the second-order system and be solved by mathematical methods. The QP formulation can be expressed as:

$$\begin{aligned} &\text{minimize } q(x) = \frac{1}{2}x^T Gx + cx^T \\ &\text{subject to } a_i^T x \geq b_i \, i \in \tau, \end{aligned} \tag{4}$$

where x is the vector with n-dimensional, $c \in R^n, A = a_i^T \in R^{p\times n}, b \in R^p$. τ is finite index set. If the G is positive definite, this kind of quadratic programming is strictly convex quadratic programming, then the global minimum is unique. Quadratic problem with equality constraints $a_i^T x = b_i$ plays special role in the mathematical programming which can be solved by effective set method and lagrange method. With standard solvers in mathematical programming, it can be solved quickly.

Following, we will introduce the QP formulation for solving the Bellman equation for MDPs.

B. QP FOR BELLMAN EQUATION SOLUTION

The formulation of Bellman equation is a basic process in most of the existing works of RL problems which can be expressed as:

$$V^*(s) = \max_a \left(\gamma \sum\nolimits_{s'} P_{sa}^{s'} V^*(s') + R(s,a)\right), \tag{5}$$

and it can be seen that this is a nonlinear equation and be solved just approximately. Furthermore, we omit the max operator and approximate the value function with the Bellman error as an linear equation:

$$V(s) = \gamma \sum\nolimits_{s'} P_{sa}^{s'} V(s') + R(s,a) + \varepsilon(s,a), \;\; \varepsilon(s,a) \geq 0, \tag{6}$$

where $\gamma \in (0, 1)$ is the discount factor. Choosing an action $a \in A$ in the current state from the state space $s \in S$, the agent will reach the next state s'. $V(s)$ is the value function in state s. $R(s, a)$ is the expected reward back from the corresponding action a in state s. Eq. (6) includes all the elements of (s, a, s', r).

In the proposed approach of the paper, we use a linear approximator with the weight vector θ for the value function approximation. For each state s, the related feature vector can be represented as $\boldsymbol{\phi}(s) = (\phi_1(s), \phi_2(s), \ldots, \phi_n(s))^\top$ with the same number of components as in $\boldsymbol{\theta}$. Therefore, the value function can be parameterized as: $V_\theta(s) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(s)$. Eq. (6) can be explicitly expressed as:

$$\boldsymbol{\theta}^\top \boldsymbol{\phi}(s) = \gamma \sum_{s'} P_{sa}^{s'} \boldsymbol{\theta}^\top \boldsymbol{\phi}(s') + R(s,a) + \varepsilon(s,a), \quad \varepsilon(s,a) \geq 0. \tag{7}$$

For each state, there will be different constraints for different actions. In AQP approach, the state-action pairs are sampled with linear value function approximation as the constraints in QP. From the machine learning theory on the sensitive property of the parameters [43], it shows that, in the objective of LP or QP problems, the robustness of the solution is highly related with the sensitivity of the parameter variables. Inspired by the discussion above, we design the objective of the QP with minimizing parameters as the form of $(\boldsymbol{\theta}^\top \boldsymbol{\theta})$ under linear constraints. By this way, the optimal solution of the value function holds little sensitivity against the state changing features. The detailed problem formulation is as follows:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\theta} \\ &\text{subject to } \boldsymbol{\theta}^\top \boldsymbol{\phi}(s) \geq \gamma \sum_{s'} P_{sa}^{s'} \boldsymbol{\theta}^\top \boldsymbol{\phi}(s') + R(s,a), \end{aligned} \tag{8}$$

where the meanings of the parameters are same as in Eq. (6). It is worth noting that, in order to obtain meaningful solutions of the QP problem, which means that, the parameters should be effective and not all zeros. Under this requirement, the rewards for the agent designed in the MDPs need to be guaranteed variety (it means that the reward value should include positive and negative). In the Eq. (8) problem, the number of the constraints is highly related with the dimension of the state space and action space. Therefore, the QP problem includes large number of inequality constraints. For large scale MDPs, the number of the constraints need keep in a solvable condition by sampling.

However, in the proposed QP formulation, the limited representation of the linear basic functions for approximating the value function reduces the quality of the optimal solution. Furthermore, the Bellman equation should be regarded as explicit objective to improve the quality of the policy. In the following two subsections, we will introduce UA and the fixed cardinality minimization technique to overcome these limitations.

C. UA FOR EXPANDING REPRESENTATION ABILITY

In AQP, we use a linear approximator for the value function approximation whose representation ability is limited. To overcome the limitation, we use the universal approximation strategy to expand the linear value function representation ability. The expanded features through UA are determined as $\boldsymbol{h_s} = \sigma(\boldsymbol{\phi}(s), r)$, where $\boldsymbol{h_s}$ is the extended feature vector of state s, $\sigma(\cdot)$ is the activation function with parameter r. The activation function is usually a nonconstant,

bounded, and monotonically-increasing continuous function, e.g. sigmoid function. Then the value function with the expanded features can be rewritten as $V_\theta(h) = \boldsymbol{\theta}^\top \boldsymbol{h}$, where the feature vector $\boldsymbol{h}$ is the representation of hidden node h_i which is defined as: $\boldsymbol{h} = (h_1, h_2, \ldots, h_m)$, where h_i covers the extension information of the state in the scene, and real vectors $\boldsymbol{\theta} \in R^m$.

Then we obtain the linear value function approximation with universal approximation of the expanded features as:

$$\boldsymbol{\theta}^\top \boldsymbol{h_s} = \gamma \sum_{s'} P_{sa}^{s'} \boldsymbol{\theta}^\top \boldsymbol{h_s'} + R(s,a) + \varepsilon(s,a),\ \ \varepsilon(s,a) \geq 0\,, \tag{9}$$

where the constraint with $\boldsymbol{h_s}$ is the expanded features of s replacing the constraint in Eq. (8).

D. FIXED CARDINALITY MINIMIZATION

To keep the convergence of the Bellman error, we use fixed cardinality minimization technique to minimize Bellman error which can be expressed as:

$$e = \left| V(s) - V^*(s) \right|\,. \tag{10}$$

For example, the number of inequality constraints are calculated by the product of the number of the state samples and the number of the actions. Such as in grid world scenario, a classical RL example, in each state, there are four actions available. In the proposed formulation, each linear constraint is obtained by the action acting at the corresponding state. If the state sample number is n_s, the action number on each state is n_a, thus the constraints number will be $n_s \cdot n_a$. In the MDPs, the optimal policy is obtained in these optimal actions chosen by the agent in the sates, and the error from the solution is the smallest one. Therefore, minimizing the sum of the Bellman errors in the objective will well improve the accuracy of the solution. We reformulate the original problem (as defined in Eq. (8)) as a fixed cardinality minimization problem, which is to minimize the summation of Bellman errors:

$$\begin{aligned} \text{minimize}\ \ & \sum_{i=1}^{M} \omega_i \varepsilon_i \\ \text{subject to}\ \ & \boldsymbol{\theta}^\top \boldsymbol{h_s} = \gamma \sum_{s'} P_{sa}^{s'} \boldsymbol{\theta}^\top \boldsymbol{h_s'} + R(s,a) + \varepsilon(s,a) \\ & \varepsilon(s,a) \geq 0\,, \end{aligned} \tag{11}$$

where M is equal to the total number of the equality constraints in the problem, ω_i is the coefficient of the Bellman errors and the value is the element from $\{0, 1\}$. The initial Bellman error is calculated by solving Eq. (8). For each given state, there are corresponding optimal action and approximate Bellman error. What means that, the optimal action $a \in A$ is corresponding to the smallest error on the state, and the accompanying coefficient is $\omega_i = 1$. Except the smallest one, other errors corresponding to the same state, the coefficients will be regarded as 0. Through this fixed cardinality minimization algorithm, for any given state s in the state samples, there is only one $\omega_i = 1$. Therefore, in the AQP problem, the number of the $\omega_i = 1$ is large enough to obtain a relative optimal solution.

E. COMPUTATION COST ANALYSIS

The number of the computation times for floating point is usually utilized to express the computation cost complexity for an algorithm. The operation for the floating point are four arithmetic operation [13]. When evaluate the computation complexity of an algorithm, we just need to count the total number of operations and express it as a function (usually a polynomial) of the dimension of the matrices and vectors involved, and simplify the expression by ignoring all the terms except for the leading one.

The classical algorithm for solving LP under numbers of the linear constraints is usually utilizing the Simplex method [44], [45] and similarly, and for solving QP, active set method is used [46]. In this paper, the proposed AQP algorithm is solved by QP and LP for the optimal final solution to find an optimal policy. Based on the theory of the floating point computation, the complexity of the computation cost will be performed by $O(mn^2)$ with matrix $\mathbb{Q} \in \boldsymbol{R}^{m \times n}$ [13]. m is corresponding to the number of the nonlinear or linear constraints in the QP and LP in our algorithm, which is the product of the state sample number and the number of the actions. n is the number of the decision variables (in our algorithm, it is the number of θ).

F. CORE DEPLOYMENT

Consider the linear value function approximation for solving the Bellman equation, we give the core part of the proposed approach and show it in Alg. 1. Following, we first give the input and output vectors. Lines $1-8$ give the process of QP, and the initial parameters are obtained. Then, Lines $9-16$ show the process of LP which guarantees the convergence of Bellman error. In particular, the errors received in the QP provide the information for the fixed cardinality minimization in the following LP. Especially, in QP, the linear constraints are a set of inequality equations which are used to obtain the initial Bellman error. Then, LP minimizes the initial Bellman error to achieve optimal solution.

VI. FURTHER ENHANCEMENT

The main methodology has been introduced in the previous sections. In this section, we will introduce how to enhance the core AQP methodology for two use important cases, one is to find the optimal values considering the state observation uncertainty, and the other is generalized goal during execution when the agent is given the start state and the goal state.

A. ROBUSTNESS AGAINST STATE UNCERTAINTY

In practical cases, the state of an agent from the scenario circumstance is obtained with various noise, even worse, they are not fixed value during the implement processing when the agent goes on. Therefore, it is not surprising that the actual performance of the optimal strategy degrades significantly from the model's prediction due to the state inaccuracy-the

Algorithm 1 AQP Methodology Flow Process

Input:
$h, \boldsymbol{\theta}, \gamma$.
Output: Optimal $V^*(s)$ for each state s
Set the objective of QP: minimize $\frac{1}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta}$
for *each state* s **do**
  Obtain the expanded features with UA
  if *state* $\neq$ *goal* **then**
    for *each action* $a \in \mathbb{A}$ **do**
      Get next state s' and calculate the return of state s,
      Construct inequality constraints.
Solve QP and calculate Bellman error $\boldsymbol{\varepsilon}$ from QP
Set the objective of LP: minimize $\sum \omega_i\varepsilon_i$.
for *each state* s **do**
  Obtain the expand features with UA
  if *state* $\neq$ *goal* **then**
    for *each action* $a \in \mathbb{A}$ **do**
      Get next state s' and calculate the return of state s,
      Construct equality constraints.
Solve LP and return optimal decision parameters ($\boldsymbol{\theta}$).
Calculate the reward on each state.
Final

deviation of the model values from the true ones. If the state observation holds some uncertainty, it is difficult to obtain the optimal solutions from the designed mechanism. Moreover, the inaccurate approximation of the value function will lead to failed decision for the task. In classical ML algorithms, there always is a penalty term in the optimization objective to alleviate the sensitivity and improve the robustness of the solutions. In the proposed approach, we need the parametrized value function representation of our algorithm to be robust against the state uncertainty, thus, besides Bellman error minimization, we also need to make the parameters (θ) as small as possible. This is a canonical QP problem which can be solved directly, and we can reformulate the fixed cardinality minimization problem into a QP problem expressed as:

$$\begin{aligned} \underset{\boldsymbol{\varepsilon}}{\text{minimize}} \ & \sum_{i=1}^{N} \omega_i\varepsilon_i + \frac{1}{2}C\boldsymbol{\theta}^\top\boldsymbol{\theta} \\ \text{subject to} \ & \boldsymbol{\theta}^\top\boldsymbol{h}_{\boldsymbol{s}} = \gamma \sum_{s'} P_{sa}^{s'}\boldsymbol{\theta}^\top\boldsymbol{h}'_{\boldsymbol{s}} + R(s,a) + \varepsilon(s,a) \\ & \varepsilon(s,a) \geq 0\,, \end{aligned} \tag{12}$$

where C is a balancing parameter which will adjust the relationship of the accuracy and robustness, what means that, the higher value of the balancing parameter and the more robustness of the solution. In this operation, the solution of the proposed formulation shows better performance on these properties.

B. GENERALIZED GOAL DURING EXECUTION

In a realistic scenario, the goal may not be static during execution. For example, when a cat is pursuing a mouse who is running all the time, it is impossible to predefine a fixed static goal for reinforcement learning algorithms to execute. Therefore, it is important to design a mechanism which can solve the dynamic changing goal problem.

In our algorithm, we form the new features of state as $\boldsymbol{\phi}(s, g)$. The feature vector (s, g) is the representation of state s and the goal g which is defined as: $(s, g) = (s_1, s_2, \ldots, s_k, g_1, g_2, \ldots, g_m)$, where s_i covers the information of the *state* and g_i is the feature information of the *goal*. Therefore, the expanded feature of the value function approximation with UA can be written as $\boldsymbol{h}_{\boldsymbol{s},\boldsymbol{g}} = \sigma(\boldsymbol{\phi}(s, g), r)$, where $\boldsymbol{h}_{\boldsymbol{s}}, \boldsymbol{g}$ is the extension feature of the state and the goal, $\sigma(.)$ is the activation function usually chosen as the sigmoid function. The formulation with generalised goal is shown as:

$$\begin{aligned} \underset{\boldsymbol{\varepsilon}}{\text{minimize}} \ & \sum_{i=1}^{N} \omega_i\varepsilon_i + \frac{1}{2}C\boldsymbol{\theta}^\top\boldsymbol{\theta} \\ \text{subject to} \ & \boldsymbol{\theta}^\top\boldsymbol{h}_{\boldsymbol{s},\boldsymbol{g}} = \gamma \sum_{s'} P_{sa}^{s'}\boldsymbol{\theta}^\top\boldsymbol{h}'_{\boldsymbol{s},\boldsymbol{g}} + R(s,a) + \varepsilon(s,a) \\ & \varepsilon(s,a) \geq 0\,. \end{aligned} \tag{13}$$

Eq. (13) is similar to Eq. (12), however, the feature vector $\boldsymbol{h}_{\boldsymbol{s},\boldsymbol{g}} = \sigma(\boldsymbol{\phi}(s, g), r)$ contains both the state and goal information, hence, it can be applied to the runtime goal use case. Other parameters are the same as in Eq. (9).

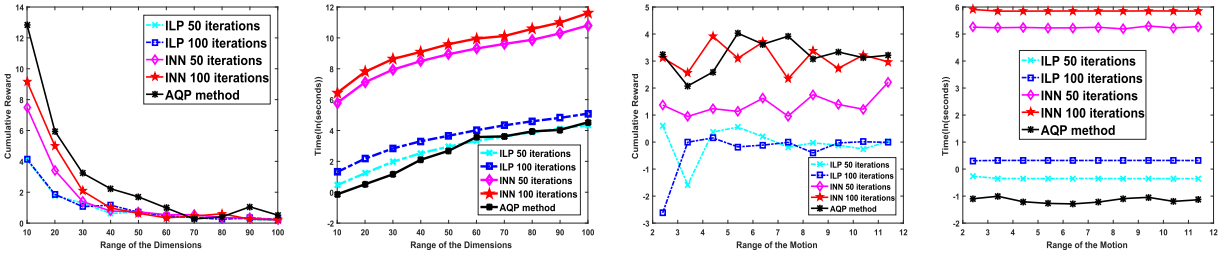
VII. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, in order to evaluate the performance of the proposed approach, we conduct numbers of experiments under the same computer devices with the system of ubuntu14.04, and the algorithms are implemented in MATLAB 2015a with standard QP and LP solvers.

To show the quantized quality and efficiency of the proposed algorithm, we compare the AQP algorithm with these two canonical algorithms: (1) ILP [47] algorithm computes the cumulative discounted future reward in 100 time steps; (2) INN [3] algorithm computes the cumulative discounted reward. These two algorithms are well representative, moreover, they have compared with the state-of-the-art algorithms.

We validate the proposed approach with two reinforcement learning benchmark problems including grid world, and cart pole [3]. The experimental results indicate that the AQP effectively outperform other state-of-the-art algorithms. We demonstrate the quality and efficiency, the property of universal approximation and fixed cardinality minimization. Furthermore, the proposed approach is enhanced to deal with MDPs with uncertainty state noise and the generalized goal during execution.

a. grid world In our experiments, we utilise the simple finite MDP scenario square grid world. Each cell of the grid corresponds to one state in the agent environments. The agent in each state holds four equal probability actions possible: north, south, east, and west, with which the agent



(a) Reward on grid world (b) Time on grid world (c) Reward on cart pole (d) Time on cart pole

FIGURE 1. Comparative results of AQP with ILP and INN for grid world and cart pole.

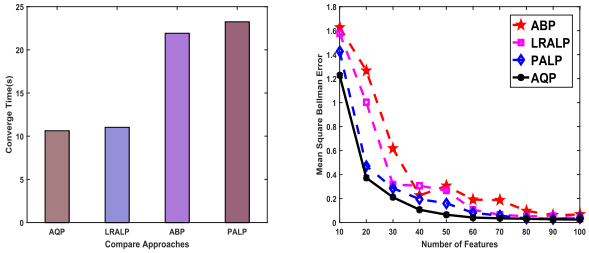
deterministically moves from the start state in the respective direction to the goal state without obstacles except the walls around the grid. The MDPs tuple is set: (1) State $s(s_1, s_2)$, s_1 is the horizontal coordinate, and s_2 is the vertical coordinate; (2) Action $a \in$ {north, east, south, west}; (3) Reward $R = 1/d$, d refers to the distance between the current state and the goal, and $R = -1$ is the penalty reward when the agent meets a wall which will hinder it to the right destination. In our experiments, the number of the cells in horizontal is equal to the vertical of the grid world scenario, and the dimension ranges from $10 \sim 100$. In the proposed algorithm, we set the hidden node number as 50 for universal approximation and set $\gamma = 0.9$,$P = 0.95$ by experience.

b. cart pole The goal in this scenario is forcing a cart, with two directions (left and right), to move along a track and the pole hinged to the cart should keep balance. In realistic, a failure occurs when the pole falls past a given angle from vertical, and then the pole will be reset again. The track length of the cart in our experiments ranges from $2.4 \sim 4.4$. The tuple of this case is set as: (1) State $s(x, \dot{x}, \theta, \dot{\theta})$ refers to the position of the cart, the speed of the cart, the position of the pendulum and the speed of the pendulum; (2) Action $a =$ {Left, Right}; (3) Reward $R = x/x_{max}$, where x is the current position of the cart and x_{max} is the range of the motion; if the pendulum falls, $R = -1$. We set $\gamma = 0.99$, $P = 0.95$.

A. THE PERFORMANCE OF AQP AGAINST ILP AND INN

1) THE REWARD

the cumulative discounted reward obtained by running the agents based on the greedy policy acquired from the approximated action values is plotted in Fig. 1(a) and Fig. 1(c)[1] (**The figures are better shown in color**). It can be seen that AQP algorithm always achieves the highest cumulative reward, showing that AQP algorithm performs the best in both grid world and cart pole scenarios. ILP obtains the lowest reward. Here, we show the ILP with 50 iterations and 100 iterations respectively. As the representation ability of the linear basic functions of ILP is limited, even for 100 iterations, the reward is still lower than our algorithm. INN based algorithm obtain almost the same cumulative discounted reward as our algorithm does, which owes to its strong representation ability for value function approximation.



(a) Efficiency Comparison (b) Precision Comparison

FIGURE 2. Comparative results of AQP for cart pole.

2) COMPUTING TIME

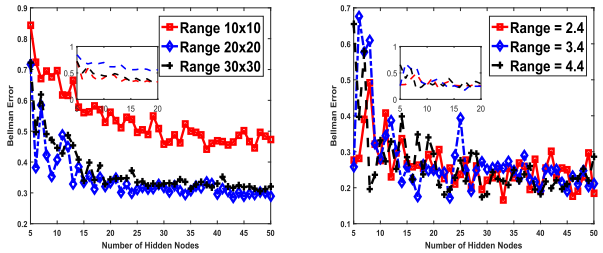
normally, the computation time in each iteration of ILP and INN is associated with the number of state samples. In our experiments, we use all the sampled states to approximate the value function. Thus, we evaluate the computation time of our algorithm and the average computation time of the ILP and INN algorithm with respect to different sizes of the scenarios, as plotted in Fig. 1(b) and Fig. 1(d).[2] Due to the large gap between the AQP algorithm with ILP and INN, we plot the log of the computation time in the figures. We also evaluate the ILP with 50 and 100 iterations, even for the case of 50 iterations, the needed computation time is still higher than our algorithm. The low computation time is the result of the core mathematical programming feature of the proposed algorithm, as stated in the 'Introduction Section', representing the solution process in the form of compact mathematical programming benefits the computing software to find efficient off-the-shell solvers.

B. THE PERFORMANCE OF AQP AGAINST APPROXIMATE LINEAR PROGRAMMING APPROACHES.

Fig. 2 shows the computation cost for a task in cart pole. The computation time is defined by the convergency efficiency and stop when the changes of the value function under a small constant δ. Mean Square Bellman Error(MSBE) of the four approaches for cart pole scenario under the scenario

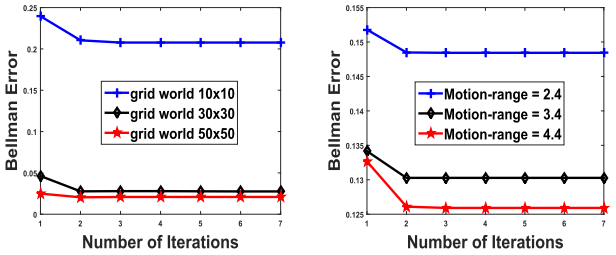
[1]For grid world, cumulative discounted reward decreases when we increase dimension, as it becomes more difficult to reach destination. For cart pole, cumulative discounted reward is achieved as long as pendulum does not fall, thus the number keeps stable.

[2]For grid world, we use all the state-action pairs to approximate the value function, thus the computation time increases with the dimension. For cart pole, we use a constant 100 samples, and the computation time keeps stable.



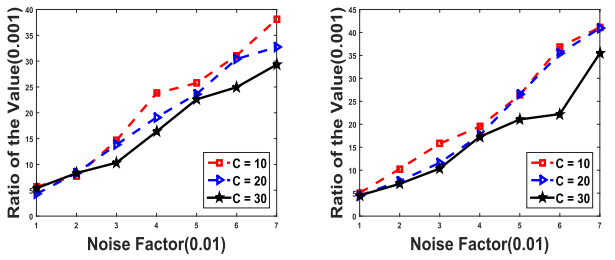
(a) Error with UA for gridworld (b) Error with UA for cartpole

FIGURE 3. Universal approximation for different scenarios.



(a) For grid world (b) For cart pole

FIGURE 4. Fixed cardinality minimization effects.



(a) Robustness for grid world (b) Robustness for cart pole

FIGURE 5. Robustness against the noise factor.

setting above. It can be seen that, the proposed AQP approach cost the least computation time than the other algorithms. From Fig. 2(b) we can see that the AQP holds the lowest MSBE. LRALP uses a smaller constraints system instead of standard constraints, and gives a performance loss bound for the strategy. The better results mainly rely on the choice of the constraints satisfying the theorem proposed in the paper. With little constraints, the computation cost is small which almost similar with the proposed AQP algorithm. By removing the worst approximation errors, it rank the second of the MSBE [19]. ABP generalizes the approximate policy iteration by employing global optimization to provide a priori knowledge which guarantees both robustness and expected policy loss [48], the advantage of ABP is the robustness for the scenario, it rank the highest MSBE and costs more time due to the unavoidable complexity of the Bellman residual minimization. PALP formulate the states into the constraints by using penalty functions which is constructed as a point-wise maximum with a family of low-order polynomials. It increases the computational burden and costs the most time and rank the third MSBE [49].

C. EVALUATION ON UNIVERSAL APPROXIMATION

We test the Bellman errors against the number of hidden nodes from UA for grid world and cart pole. Since the feature space of states is limited and less than 5 for both scenarios, we test the performance of UA for the number of hidden nodes ranging from 5 $\sim$ 50 and the experimental results are shown in Fig. 3. It can be seen that when we increase the number of hidden nodes, the errors decrease. When the number of hidden nodes is more than 30, the performance keeps stable. For grid world, we test three cases of the grid size, 10×10, 20×20, 30×30 respectively. Since the number of constraints in the LP increases with dimensions, the errors decrease as is shown in Fig. 3(a). Moreover, we test the performance in the cart pole. The position of the cart ranges from 2.4 $\sim$ 4.4. In each range, we take 100 samples for AQP. The error for each motion range decreases when we increase the number of hidden nodes up to 30 and will keep stable thereafter as shown in Fig. 3(b).

D. BELLMAN ERROR FOR FIXED CARDINALITY MINIMIZATION

Fig. 4 shows the evolution traces of Bellman error clearly. As shown in the figure, Bellman error is decreasing and converging as iteration goes on. Note that in the AQP methodology, we only use one time of LP to obtain the Bellman error, which is iteration number 2 in the figure, for most cases, it is enough to get the minimal error. In real use cases, if we want the Bellman error to be minimal, we can increase the iteration number to 3 or 4.

E. ROBUSTNESS AGAINST STATE UNCERTAINTY

We also test the robustness of AQP against state uncertainty. Eq. (12) gives the objective function in the proposed algorithm. In order to enhance the algorithm's robustness property, we can increase the balancing robustness parameter C. In our experiment, the observed state holds a noise factor from one percent to seven percent, which means that the state feature vector $\phi(s)$ has a one percent to seven percent deviation with respect to its real observation. We can define the ratio as $\mathrm{p} = \frac{|V_r - V_s|}{|V_s|}$ to characterize the deviation between calculated value function (V_r) and true value function (V_s). Fig. 5 shows that, the deviation value will increases with the increasing degree of the observation noise factor but also lie in a quite small region (between five over one thousand and forty-five parts per thousand). Furthermore, when the balancing parameter increases, the ratio of deviation decreases for the same noise factor as shown in the figure, and the robustness of the solution increases.

F. GENERALIZED GOAL DURING EXECUTION

We test the performance of AQP for the generalized goal scenario. For example, when given a pursuit problem (i.e., a cat catching a mouse), the target position is not static and previously unknown to the RL agent. In this case, we assume that at runtime, the position of the target is known, and we put the goal (current position of the target) and the current

TABLE 1. Cumulative discounted reward for grid world with generalised goals (The first row gives the three generalised goals; the first column gives the random start point. The numerical value is the cumulative discounted reward.).

Start \ Goal	goal(3,4)	goal(9,2)	goal(10,10)
start(1,1)	11.5632	6.8282	4.0322
start(10,13)	5.4423	6.6274	10.1034
start(19,12)	8.4463	9.1694	9.2346

position of the RL agent into the state feature encoding. Then we apply the AQP method, and the results are shown in Table 1. In Table 1, the expected reward is dependent on both the goal and the starting point, which is as expected. It is worth noting that, in the current AQP version, we assume that the runtime goal is known to the agent. If the goal is unknown, the problem becomes a searching problem, which needs a new problem formulation for the solution.

VIII. CONCLUSION AND FUTURE WORKS

This paper demonstrates the effective AQP approach for solving the Bellman equation to obtain optimal solutions with higher accuracy and robustness. From the experimental results, we can see that, the proposed algorithm converges much faster than other classical approaches and receives similar or better cumulative discounted reward. Additionally, we expand the linear value function representation ability through universal approximation, and reduce Bellman error with fixed cardinality minimization technique. We also enhance the performance of the AQP algorithm with respect to its robustness against state uncertainty and generalized goal during execution.

In the future, we would like to incorporate the proposed AQP algorithm with classical RL algorithms such as Q-learning. The AQP algorithm can quickly return a 'near-optimal' value function approximation, while Q-learning interacts with the environment and easily adapts its Q-values with the real time input. We can let AQP provide good initial Q-values for the Q-learning algorithm, and continue to use Q-learning to explore the unknown environment. Currently, we use UA to expand the value function representation ability, in the future, we will explore the deep learning representation theory, and try to automatically generate features for linear value function representation.

Due to the importance of the nonlinear complex systems in the natural world, we further would like incorporate the proposed AQP algorithm in the fuzzy neural network structure with additional uncertainties for prediction problems [50]. Based on machine learning for the linear region identification, the proposed AQP will help to obtain accurate largest Lyapunov exponent to reduce the errors by human [51].

ACKNOWLEDGMENT

The authors would like to thank their laboratory team members' assistance.

REFERENCES

[1] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proc. CVPR*, Jul. 2017, pp. 2174–2182.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: https://arxiv.org/abs/1312.5602

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[4] C. Luo and X. Wang, "Hybrid modified function projective synchronization of two different dimensional complex nonlinear systems with parameters identification," *J. Franklin Inst.*, vol. 350, no. 9, pp. 2646–2663, 2013.

[5] H. Wang, C. Luo, and X. Wang, "Synchronization and identification of nonlinear systems by using a novel self-evolving interval type-2 fuzzy LSTM-neural network," *Eng. Appl. Artif. Intell.*, vol. 81, pp. 79–93, May 2019.

[6] S. Wang, X. Wang, and B. Han, "Complex generalized synchronization and parameter identification of nonidentical nonlinear complex systems," *PLoS One*, vol. 11, no. 3, 2016, Art. no. e0152099.

[7] W. Li, D. Wang, and T. Chai, "Multisource data ensemble modeling for clinker free lime content estimate in rotary kiln sintering processes," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 303–314, Feb. 2015.

[8] Y.-J. Liu, C. L. P. Chen, G.-X. Wen, and S. Tong, "Adaptive neural output feedback tracking control for a class of uncertain discrete-time nonlinear systems," *IEEE Trans. Neural Netw.*, vol. 22, no. 7, pp. 1162–1167, Jul. 2011.

[9] H. Zhang, X.-Y. Wang, and X.-H. Lin, "Topology identification and module–phase synchronization of neural network with time delay," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 6, pp. 885–892, Jun. 2017.

[10] H. Zhang and X.-Y. Wang, "Complex projective synchronization of complex-valued neural network with structure identification," *J. Franklin Inst.*, vol. 354, no. 12, pp. 5011–5025, 2017.

[11] H.-G. Zhang, X. Zhang, Y.-H. Luo, and J. Yang, "An overview of research on adaptive dynamic programming," *Acta Automatica Sinica*, vol. 39, no. 4, pp. 303–311, 2013.

[12] D. Liu, X. Yang, D. Wang, and Q. Wei, "Reinforcement-learning-based robust controller design for continuous-time uncertain nonlinear systems subject to input constraints," *IEEE Trans. Cybern.*, vol. 45, no. 7, pp. 1372–1385, Jul. 2015.

[13] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[14] A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Philadelphia, PA, USA: SIAM, 1990.

[15] P. J. Schweitzer and A. Seidmann, "Generalized polynomial approximations in Markovian decision processes," *J. Math. Anal. Appl.*, vol. 110, no. 2, pp. 568–582, 1985.

[16] V. V. Desai, V. F. Farias, and C. C. Moallemi, "Approximate dynamic programming via a smoothed linear program," *Oper. Res.*, vol. 60, no. 3, pp. 655–674, 2012.

[17] D. P. de Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming: Theory and application," Ph.D. dissertation, Dept. Manage. Sci. Eng., Stanford Univ., Stanford, CA, USA, 2002.

[18] D. P. De Farias and B. Van Roy, "On constraint sampling in the linear programming approach to approximate dynamic programming," *Math. Oper. Res.*, vol. 29, no. 3, pp. 462–478, 2004.

[19] C. Lakshminarayanan, S. Bhatnagar, and C. Szepesvári, "A linearly relaxed approximate linear program for Markov decision processes," *IEEE Trans. Autom. Control*, vol. 63, no. 4, pp. 1185–1191, Apr. 2018.

[20] F. Chen, Q. Cheng, J. Dong, Z. Yu, G. Wang, and W. Xu, "Efficient approximate linear programming for factored MDPs," *Int. J. Approx. Reasoning*, vol. 63, pp. 101–121, Aug. 2015.

[21] M. Petrik and S. Zilberstein, "Constraint relaxation in approximate linear programs," in *Proc. 26th Annu. Int. Conf. Mach. Learn*, 2009, pp. 809–816.

[22] Y. Abbasi-Yadkori, P. L. Bartlett, X. Chen, and A. Malek, "Large-scale Markov decision problems via the linear programming dual," 2019, *arXiv:1901.01992*. [Online]. Available: https://arxiv.org/abs/1901.01992

[23] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored MDPs," *J. Artif. Intell. Res.*, vol. 19, pp. 399–468, Oct. 2003.
[24] V. Desai, V. Farias, and C. C. Moallemi, "A smoothed approximate linear program," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 459–467.
[25] N. Bhat, V. F. Farias, and C. C. Moallemi, "Non-parametric approximate dynamic programming via the kernel method," in *Proc. Adv. Neural Inf. Process. Syst. Conf.*, 2012, pp. 386–394.
[26] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.
[27] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 930–945, May 1993.
[28] G. Lewicki and G. Marino, "Approximation of functions of finite variation by superpositions of a sigmoidal function," *Appl. Math. Lett.*, vol. 17, no. 10, pp. 1147–1152, 2004.
[29] H. D. Nguyen, L. R. Lloyd-Jones, and G. J. McLachlan, "A universal approximation theorem for mixture-of-experts models," *Neural Comput.*, vol. 28, no. 12, pp. 2585–2593, 2016.
[30] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.
[31] S. Scardapane and D. Wang, "Randomness in neural networks: An overview," *Wiley Data Mining Knowl. Discovery*, vol. 7, no. 2, p. e1200, 2017.
[32] H. Dinh, S. Bhasin, R. Kamalapurkar, and W. E. Dixon, "Dynamic neural network-based output feedback tracking control for uncertain nonlinear systems," *J. Dyn. Syst. Meas. Control*, vol. 139, no. 7, 2017, Art. no. 074502.
[33] D. Wang and M. Li, "Stochastic configuration networks: Fundamentals and algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3466–3479, 2017.
[34] G. Montufar and N. Ay, "Refinements of universal approximation results for deep belief networks and restricted Boltzmann machines," *Neural Comput.*, vol. 23, no. 5, pp. 1306–1319, 2011.
[35] D. R. Muir, "Feedforward approximations to dynamic recurrent network architectures," *Neural Comput.*, vol. 30, no. 2, pp. 546–567, 2017.
[36] H. AbouEisha, M. Farhan, I. Chikalov, M. Moshkov, "An algorithm for reduct cardinality minimization," in *Proc. 5th Int. Conf. Granular Comput.*, 2013, pp. 1–3.
[37] L. Cai, S. M. Chan, and S. O. Chan, "Random separation: A new method for solving fixed-cardinality optimization problems," in *Proc. Int. Workshop Parameterized Exact Comput.*, 2006, pp. 239–250.
[38] M. Bruglieri, M. Ehrgott, H. W. Hamacher, and F. Maffioli, "An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints," *Discret Appl. Math.*, vol. 154, no. 9, pp. 1344–1357, 2006.
[39] M. Fazel, H. Hindi, and S. P. Boyd, "Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2003, pp. 2156–2162.
[40] K. Mohan and M. Fazel, "Iterative reweighted least squares for matrix rank minimization," in *Proc. 48th Annu. Allerton Conf. Commun., Control, Comput. (CCC)*, Sep./Oct. 2010, pp. 653–661.
[41] M. Petrik, "Optimization-based approximate dynamic programming," Ph.D. dissertation, Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Sep. 2010.
[42] M. Fu, Z.-Q. Luo, and Y. Ye, "Approximation algorithms for quadratic programming," *J. Combinat. Optim.*, vol. 2, no. 1, pp. 29–50, 1998.
[43] T. M. Mitchell, *Machine Learning*. Beijing, China: China Machine Press, 2003.
[44] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965.
[45] Y. Ye, "The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate," *Math. Oper. Res.*, vol. 36, no. 4, pp. 593–784, 2011.
[46] M. Hintermüller, K. Ito, and K. Kunisch, "The primal-dual active set strategy as a semismooth newton method," *SIAM J. Optim.*, vol. 13, no. 3, pp. 865–888, 2002.
[47] D. Bello and G. Riano, "Linear programming solvers for Markov decision processes," in *Proc. IEEE Syst. Inf. Eng. Design Symp.*, Apr. 2006, pp. 90–95.
[48] M. Petrik and S. Zilberstein, "Robust approximate bilinear programming for value function approximation," *J. Mach. Learn. Res.*, vol. 12, no. 12, pp. 3027–3063, 2011.
[49] P. N. Beuchat and J. Lygeros, "Approximate dynamic programming via penalty functions," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11814–11821, 2017.
[50] C. Luo, C. Tan, X. Wang, and Y. Zheng, "An evolving recurrent interval type-2 intuitionistic fuzzy neural network for online learning and time series prediction," *Appl. Soft Comput.*, vol. 78, pp. 150–163, May 2019.
[51] S. Zhou and X. Wang, "Identifying the linear region based on machine learning to calculate the largest Lyapunov exponent from chaotic time series," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 28, no. 12, 2018, Art. no. 123118.

JIANMEI SU received the M.Sc. degree in computational mathematics from the School of Management Science and Engineering, Chengdu University of Technology (CDUT). She is currently pursuing the Ph.D. degree in control science and engineering with the Machine Intelligence Institute, School of Automation, University of Electronics Science and Technology of China (UESTC). Her research interests include video processing, optimization, and machine learning.

HONG CHENG received the Ph.D. degree in pattern recognition and intelligent systems from Xi'an Jiaotong University, in 2003, where he has been an Associate Professor, since 2005. He held a postdoctoral position with the Computer Science School, Carnegie Mellon University, USA, from 2006 to 2009. Since July 2000, he has been with Xi'an Jiaotong University, where he was a Team Leader of the Intelligent Vehicle Group, Institute of Artificial Intelligence and Robotics, until 2006. He is a Full Professor with the School of Automation and the Vice Director of Center for Robotics, UESTC, where he is currently the Founding Director of the Machine Intelligence Institute. His team in XJTU had developed an intelligent driving platform-Springrobot, which has an important social effect in China. His current research interests include computer vision and machine learning, robotics, human computer interaction, and multimedia signal processing.

Dr. Cheng has been a Senior Member of ACM and an Associate Editor of the *IEEE Computational Intelligence Magazine*. He is a Reviewer for many important journals and conferences, including the IEEE TITS, MAV, CVPR, ICCV, ITSC, IVS, ACCV, etc. He serves as the Finance Chair of ICME 2014, the Local Arrangement Chair of VLPR 2012, and the Registration Chair of the 2005 IEEE ICVES.

HONGLIANG GUO received the B.E. degree in dynamic engineering and the M.E. degree in dynamic control from the Beijing Institute of Technology, China, and the Ph.D. degree in electrical and computer engineering from the Stevens Institute of Technology, USA. He joined Almende, Rotterdam, The Netherlands, as a Postdoctoral Researcher, in 2011. In 2013, he joined NTU, as a Research Fellow. His research interests include selforganizing systems and agent-based technologies.

RUI HUANG received the Ph.D. degree in control science and engineering from the University of Electronic Science and Technology of China (UESTC), in July 2018, where he is currently a Postdoctoral with the School of Automation Engineering, Center for Robotics. He was a Joint Training Doctoral Student with the TAMS, University of Hamburg, from 2016 to 2017. His current research interests include reinforcement learning, exoskeleton, human–robot interaction, and robot control.

ZHINAN PENG received the B.S. degree in information and computing science from Fuyang Normal University, Fuyang, China, in 2014, and the M.S. degree in computational mathematics from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2016, where he is currently pursuing the Ph.D. degree with the School of Automation Engineering. His current research interests include neural networks-based control, multi-agent systems, adaptive dynamic programming, and reinforcement learning.