PDP: Parallel Dynamic Programming

Fei-Yue Wang, *Fellow, IEEE*, Jie Zhang, *Member, IEEE*, Qinglai Wei, *Member, IEEE*, Xinhu Zheng, *Student Member, IEEE*, and Li Li, *Fellow, IEEE*

***Abstract*—Deep reinforcement learning is a focus research area in artificial intelligence. The principle of optimality in dynamic programming is a key to the success of reinforcement learning methods. The principle of adaptive dynamic programming (ADP) is first presented instead of direct dynamic programming (DP), and the inherent relationship between ADP and deep reinforcement learning is developed. Next, analytics intelligence, as the necessary requirement, for the real reinforcement learning, is discussed. Finally, the principle of the parallel dynamic programming, which integrates dynamic programming and analytics intelligence, is presented as the future computational intelligence.**

***Index Terms*—Parallel dynamic programming, Dynamic programming, Adaptive dynamic programming, Reinforcement learning, Deep learning, Neural networks, Artificial intelligence.**

I. Introduction

Google DeepMind's deep reinforcement learning based *AlphaGo* computer program [1] won the historic Go match against world champion Lee Sedol in March 2016. The combination of Monte-Carlo tree search and deep reinforcement learning makes a breakthrough at Go playing which is believed impossible with brute-force search, and brings artificial intelligence a focus for the year. Most people pay more attention to the intuitive highly brain-like deep learning technologies. However, the other key to AlphaGo's success, the *Principle of Optimality* for dynamic programming, has been taken for reinforcement learning (RL). As a matter of fact, dynamic programming plays a very important role in modern reinforcement learning. The victory of AlphaGo is actually also the victory of dynamic programming.

Dynamic programming [2] has become well-known since 1950s in many fields. In 1977, Werbos combined DP, neural networks and reinforcement learning, and introduced approximate/adaptive dynamic programming (ADP) [3], [4] to solve the "curse of dimensionality" [2]. However, trial-and-error based reinforcement learning and deep learning both focus on engineering complexity and ignore the social complexity. In this article, we suggest another extension of dynamic programming considering both engineering and social complexities, aiming the "paradox of scientific methods" in complex system's "scientific solutions" [5]. We utilizing big data analytics and the ACP approach [6], [7]: *arttificial societies* for descriptive anlytics, *computational experiments* for predictive analytics, and *parallel execution* for prescriptive analytics. We name our approach *Parallel Dynamic Programming*.

This article is organized as follows. The next section reviews dynamic programming and adaptive dynamic programming. Then, we briefly discuss the neural network structure of ADP and AlphaGo. We present the ACP approach of analytics intelligence in Section IV. In Section V, we introduce the basic structure of parallel dynamic programming. The last section concludes the article.

II. From Dynamic Programming to Adaptive Dynamic Programming

Dynamic programming (DP) is a very useful tool in solving optimization and optimal control problems [8]–[10]. The dynamic programming technique rests on a very simple idea, the Bellman's principle of optimality [2]: "An optimal policy has the property that no matter what the previous decision (i.e., controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions."

DP can easily be applied to the optimal control of discrete-time nonlinear systems. Let the system be

$$x_{k+1} = F_k(x_k, u_k), \tag{1}$$

where x_k, u_k are the state and control, respectively, and $F_k(\cdot)$ is the system function at time k. Suppose we associate with this plant the performance index function

$$J_i(x_i) = \phi(N, x_N) + \sum_{k=i}^{N-1} U_k(x_k, u_k), \tag{2}$$

Manuscript received November 11, 2015; accepted December 21, 2016. This work was supported by National Natural Science Foundation of China (61533019, 61374105, 71232006, 61233001, 71402178).

Citation: F.-Y. Wang, J. Zhang, Q. L. Wei, X. H. Zheng, and L. Li, "PDP: parallel dynamic programming," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 1-5, Jan. 2017.

Fei-Yue Wang is with The State Key Laboratory of Management and Control for Complex Systems (SKL-MCCS), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing 100190, China, and School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the Research Center for Military Computational Experiments and Parallel Systems Technology, National University of Defense Technology, Changsha 410073, China (e-mail: feiyue.wang@ia.ac.cn).

Jie Zhang is with The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences (SKL-MCCS, CASIA), Beijing 100190, China, and also with the Qingdao Academy of Intelligent Industries, Shandong 266000, China (e-mail: jie.zhang@ia.ac.cn).

Qinglai Wei is with The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences (SKL-MCCS, CASIA), Beijing 100190, China, and also with School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: qinglai.wei@ia.ac.cn).

Xinhu Zheng is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55414, USA (e-mail: zheng473@umn.edu).

Li Li is with the Department of Automation, Tsinghua University, Beijing 100084, China (email: li-li@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2017.7510310

where $[i, N]$ is the time interval of interest. According to the Bellman's principle of optimality, the optimal performance index function, which aims to minimize, satisfies the following equation

$$J_k^*(x_k) = \min_{u_k}\{U_k(x_k, u_k) + J_{k+1}^*(x_{k+1})\}. \tag{3}$$

Equation (3) is the Bellman's optimality equation. Its importance lies in the fact that it allow us to optimize over only one control vector at a time by working backward from N. It is called the functional equation of dynamic programming and is the basis for computer implementation of the Bellman's method. However, it is often computationally untenable to obtain the optimal control by directly solving the Bellman equation (3) due to the backward numerical process required for its solution, i.e., as a result of the well-known "curse of dimensionality" [2]. We have to find a series of optimal control actions that must be taken in sequence. This sequence will give the optimal performance index, but the total cost of these actions is unknown until the end of that sequence.

Approximate dynamic programming, proposed by Werbos [3], [4], builds a critic system to circumvent the "curse of dimensionality" by building a system, called "critic" to approximate the cost function in dynamic programming. The main idea of approximate dynamic programming as shown in Fig. 1. There are three parts in the the structure of approximate

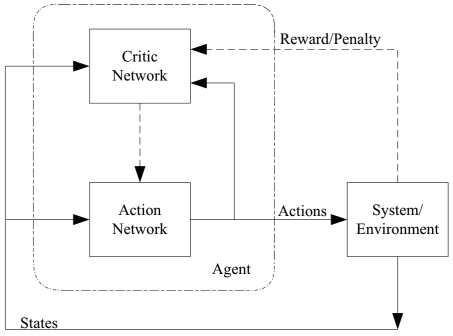

Fig. 1. ADP structure.

dynamic programming, which are dynamic system, the critic module, and the action module, respectively. First, the action module outputs a control policy according to the system state. Second, according to the system implementation, the critic module receives a evaluate signal. Third, a reinforcement signal is created by the critic network, which aims to indicate the action module to find a better control policy, at least not worse. The whole implementation is self-learning and the critic and action modules can be regarded as an agent. According to the principle in Fig. 1, the dynamic programming problem is desired to solve forward-time. In [11], [12], the approximate dynamic programming method was implemented, where each part in Fig. 1 was modeled by a neural network and hence is called "Neuro-Dynamic Programming". Its several synonyms are used, such as "Adaptive Critic Designs" [13], [14], "Asymptotic Dynamic Programming" [15], "Adaptive Dynamic Programming" [16]–[21], and "Neural Dynamic Programming" [22]. In 2006, the synonyms were unified as "Adaptive Dynamic Programming (ADP)" [23]–[36].

III. Neural Network Structure of ADP

HDP is the most basic and widely applied structure of ADP [11], [13]. The structure of HDP is shown in Fig. 2. HDP is a method for estimating the performance index function. Estimating the performance index function for a given policy only requires samples from the instantaneous utility function U, while models of the environment and the instantaneous reward are needed to find the performance index function corresponding to the optimal policy.

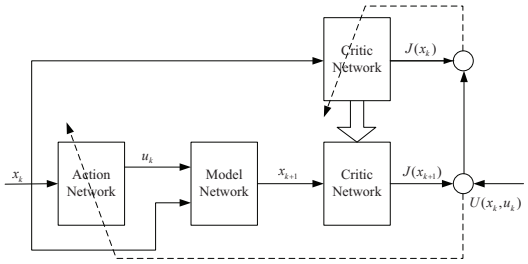

Fig. 2. The HDP structure diagram.

In the HDP structure, the model network aims to describe the dynamic of the system. The action network aims to approximate the control policy of the system, and the critic network aims to approximate the performance index function. If each neural network in HDP is chosen as three-layer back-propagation (BP) network, then the neural network structure of HDP can be expressed as in Fig. 3.

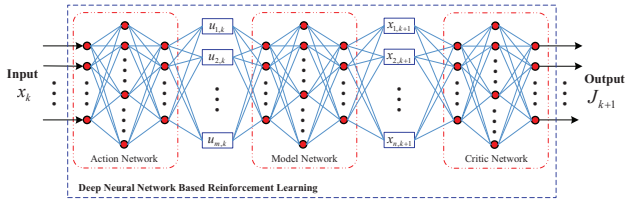

Fig. 3. Deep neural network structure of HDP.

In Fig. 3, we can say that we use three BP neural networks to implement the learning of the optimal control. However, if the three neural networks are regarded as one neural network, then we can say that we implement the learning of the optimal control by at least a nine-layer deep BP network [37]. In this point of view, the structure of HDP is a structure of a deep neural network. For all the other structures of ADP [13], such as dual heuristic programming (DHP), global dual heuristic programming (GDHP), and their action-depended versions, the structures can also be transformed into one deep neural network. Thus, the structure of ADP is naturally a deep neural network. The training target of the deep neural network is desired to force the following error

$$e_k = U_k(x_k, u_k) + J_{k+1}(x_{k+1}) - J_k(x_k) \tag{4}$$

to zero. Obviously, the training error e_k can be chosen as the reinforcement signal, which optimizes the control policy

to minimize the distance between e_k and the equilibrium point. Hence, the optimization process by ADP is actually a reinforcement learning process via a deep neural network. This is an amazing similar with the implementation of AlphaGo. Earlier works in [38], [39] also provided neural network based control method with knowledge architecture embed.

IV. Analytics Intelligence: From ACP to DPP

Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision-making. During the implementation process of ADP, it is emphasized that reinforcement learning is a key technique to find a better control policy of the system via trial-and-error. However, it should be pointed out that shortcomings inherently exist for the trial-and-error approach. Many real-world control systems cannot be "tried" sufficiently for the fact of security and cost. Particularly, for systems that involve human and societies (Cyber-Phisical-Social systems, CPSS) [40], sometimes the "error" is intolerable. The success of AlphaGo suggests the possibility of conducting reinforcement learning with a virtual Go-game played by two virtual players. However, we do not know the exact rules or dynamic systems in most of our real-world control and management problems as the Go-game.

Data driven parallel systems in cyberspace are the key to solve the trial-and-error challenge. Two founding pioneers of modern management sciences stated famous maxims for operations: W. Edwords Deming, "*In God we trust; all others must bring data*" and Peter F. Drucker: "*The best way to predict the future is to create it*". Our suggestion is to integrate artificial intelligence and analytics [6]: *artificial societies* (or systems) for descriptive analytics, *computational experiments* for predictive analytics, and *parallel execution* for prescriptive analytics. The DPP (descriptive, predictive and prescriptive) analytics intelligence can be built based on the ACP approach as is shown in Fig. 4.

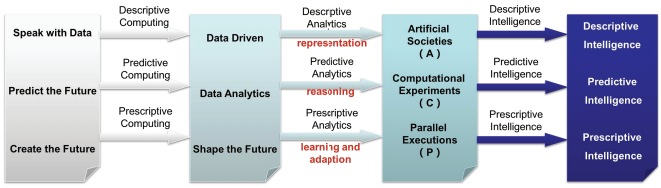

Fig. 4. The ACP approach of descriptive, predictive, and prescriptive analytics.

The representation procedure of descriptive models is to speak with data, which generally aims to tell us "what happened in history", "when did it happen", and "why did it happen". However, in real-world cases, people can only collect "small data" for decision making. Before we make a decision, a lot of possible "futures" ("big data" or artificial societies) are created during the analytics based on the collected "small data". Moreover, "futures" can be "created" by imaging and designing. Hence, in our ACP approach, the *descriptive analytics* by artificial societies are not only the model of real-world data, but also the model of virtual "predicted future" and "created future" in cyberspace.

Predictive analysis [41]–[43] should be made according to the descriptive model and historical data to predict the future by reasoning. It tells us "what will happen", "when will it happen," and "why will it happen". In our ACP approach, *predictive analytics* are conducting computational experiments to predict the future for certain artificial society with certain control and management policy. "Big data" are created in the computational experiments in cyberspace.

Finally, no matter how many possible futures or policies, we can choose only one to implement in our real world. Hence, after the predictive analytics of different policies and different artificial societies, we reduce the "big data", extract rules, and create the real future through learning and adaption. In our ACP approach, *prescriptive analytics* are developed to find benefit policy from predictions based on the descriptive models and historical data through parallel execution. Data are collected for further descriptive analytics and predictive analytics during the parallel execution.

V. Parallel Dynamic Programming

In the practice of AlphaGo, one of the key ideas is to extract supervised learning policy, and to improve it to get a sub-optimal policy during the reinforcement learning procedure. In parallel dynamic programming (PDP), we suggest the ACP approach for decision making in CPSS with analytics intelligence.

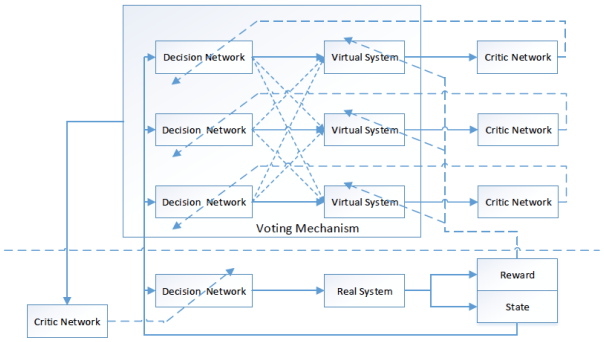

Fig. 5. Parallel dynamic programming with three parallel systems.

The descriptive analytics of parallel dynamic programming are data driven for systems with unknown or imperfect information. We collect state-action-reward-state data from real-world controls and observations, as is shown in Fig. 5. Since the highly complexity of real-world system, and the highly unpredictable of human behaviors, the data are not directly used for fitting the state equations of the dynamic system. On the contrary, in parallel dynamic programming, we focus on how to construct "possible" data consistent with the real-world observations. Combining these virtual data and the historical real-world data, we can build the artificial systems parallelized to the real-world system. The data driven virtual systems model the state equations of the artificial world, and the objectives of agents. An artificial system in PDP indicates feasibility and possibility, instead of similarity.

Within each artificial system, the predictive analytics can be viewed as optimal control problems with known dynamics in

PDP. Hence, assume we have n-artificial systems ($n = 3$ in Fig. 5) parallelized with the real-world system, we can employ dynamic programming or adaptive dynamic programming to solve the optimal control problems with known state equations in the trial-and-error approach in the virtual parallel system without any cost or risk, and get n optimal (or sub-optimal) decisions. Note that, this procedure is naturally distributed, and previous decisions in real-world can be used as an initial guess in the reinforcement learning iterations to reduce the computation. Then, the computational experiments can be conducted in a bionic manner: based on a voting mechanism n-virtual systems will vote for the n-decisions. Hence, we can get a winning decision and its corresponding critic network. Note that, the computational experiments search for an acceptable artificial system and an admissible decision, rather than the optimal control.

The parallel execution will based on the optimality principle of dynamic programming, and the critic network selected in the computational experiments. We adjust the decision according to the winning critic network and the observed real-world state. The virtual-real system interaction can be conducted by observing states and errors, updating the artificial systems and adjusting the voting mechanism.

The detailed implementation of the PDP algorithm for unknown discrete systems has been conducted and the result is very interesting and promising [44], more works are undertaking and will be reported.

VI. Remark and Conclusion

"Scientific solutions" need to satisfy two conditions: triable and repeatable [6]. In real-world systems that involves human and societies, the trial-and-error based reinforcement learning can not be conducted unless we already know the "error" will be harmless. On the other hand, the suggested parallel dynamic programming conduct computational experiments in virtual systems with the idea of optimality principle. Unlike the game of AlphaGo, PDP is based on parallel systems [45], [46] instead of the exact rules of real-world systems, and will be more flexible and feasible for complex problems.

References

[1] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature* 529.7587, pp. 484-489, 2016.

[2] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[3] P. J. Werbos, "Advanced forecasting methods for global crisis warning and models of intelligence," *General Syst. Yearbook*, vol. 22, 1977.

[4] P. J. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control*, W. T. Miller, R. S. Sutton and P. J. Werbos (Eds.), Cambridge: MIT Press, 1991, pp. 67−95.

[5] F.-Y. Wang, et al., "Where does AlphaGo go: from church-turing thesis to AlphaGo thesis and beyond", *IEEE/CAA J. Autom. Sinica*, vol. 3, no. 2, pp. 113−120, April 2016.

[6] F.-Y. Wang, "A big-data perspective on AI: Newton, Merton, and analytics intelligence", *IEEE Intell. Syst.*, vol. 27, no. 5, pp. 2−4, 2012.

[7] L. Li, Y.-L. Lin, D.-P. Cao, N.-N. Zheng, and F.-Y. Wang, "Parallel learning-a new framework for machine learning," *Acta Autom. Sinica*, vol. 43, no. 1, pp. 1−8, 2017 (in Chinese).

[8] J. Li, W. Xu, J. Zhang, M. Zhang, Z. Wang, and X. Li, "Efficient video stitching based on fast structure deformation," *IEEE Trans. Cybern.*, article in press, 2015. DOI: 10.1109/TCYB.2014.2381774.

[9] C. Vagg, S. Akehurst, C. J. Brace, and L. Ash, "Stochastic dynamic programming in the real-world control of hybrid electric vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 3, pp. 853−866, Mar. 2016.

[10] P. M. Esfahani, D. Chatterjee, and J. Lygeros, "Motion planning for continuous-time stochastic processes: A dynamic programming approach," *IEEE Trans. Autom. Control*, vol. 61, pp. 2155−2170, 2016.

[11] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D.A. White and D.A. Sofge (Eds.), New York: Van Nostrand Reinhold, 1992, ch. 13.

[12] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[13] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997−1007, Sep. 1997.

[14] J. Han, S. Khushalani-Solanki, J. Solanki, and J. Liang, "Adaptive critic design-based dynamic stochastic optimal control design for a microgrid with multiple renewable resources," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 2694−2703, Jun. 2015.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[16] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, "Adaptive dynamic programming," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 2, pp. 140−153, May 2002.

[17] Q. Wei, F. L. Lewis, D. Liu, R. Song, and H. Lin, "Discrete-time local value Iteration adaptive dynamic programming: Convergence analysis," *IEEE Trans. Syst., Man, Cybern. A, Syst.*, article in press, 2016. DOI: 10.1109/TSMC.2016.2623766.

[18] Q. Wei, F. L. Lewis, Q. Sun, P. Yan, and R. Song, "Discrete-time deterministic Q-learning: A novel convergence analysis," *IEEE Trans. Cybern.*, article in press, 2016. DOI: 10.1109/TCYB.2016.2542923.

[19] Q. Wei, D. Liu, and G. Shi, "A novel dual iterative Q-learning method for optimal battery management in smart residential environments," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2509−2518, Apr. 2015.

[20] Q. Wei and D. Liu, "A novel iterative θ-Adaptive dynamic programming for discrete-time nonlinear systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 4, pp. 1176−1190, Oct. 2014.

[21] Q. Wei, D. Liu, Q. Lin, and R. Song, "Discrete-time optimal control via local policy iteration adaptive dynamic programming," *IEEE Trans. Cybern.*, article in press, 2016. DOI: 10.1109/TCYB.2016.2586082.

[22] R. Enns and J. Si, "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 929−939, Aug. 2003.

[23] R. Kamalapurkar, J. R. Klotz, and W. E. Dixon, "Concurrent learning-based approximate feedback-Nash equilibrium solution of N-player nonzero-sum differential games," *IEEE/CAA J. Autom. Sinica*, vol. 1, no. 3, pp. 239−247, Jul. 2014.

[24] Q. Wei, D. Liu, and Q. Lin, "Discrete-time local iterative adaptive dynamic programming: Terminations and admissibility analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, article in press, 2016. DOI: 10.1109/TNNLS.2016.2593743.

[25] Q. Wei, R. Song, and P. Yan, "Data-driven zero-sum neuro-optimal control for a class of continuous-time unknown nonlinear systems with disturbance using ADP," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 444−458, Feb. 2016.

[26] H. Zhang, C. Qin, B. Jiang, and Y. Luo, "Online adaptive policy learning algorithm for H_∞ state feedback control of unknown affine nonlinear discrete-time systems," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2706−2718, Dec. 2014.

[27] F.-Y. Wang and G. N. Saridis, "Suboptimal control for nonlinear stochastic systems," *Proc. 31st IEEE Conf. Decision Control*, 1992.

[28] G. N. Saridis and F.-Y. Wang, "Suboptimal control of nonlinear stochastic systems," *Control Theory and Advanced Technology*, vol. 10, no. 4, pp. 847−871, 1994.

[29] Q. Wei, D. Liu, and X. Yang, "Infinite horizon self-learning optimal control of nonaffine discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 4, pp. 866−879, Apr. 2015.

[30] Q. Wei, D. Liu, Y. Liu, and R. Song, "Optimal constrained self-learning battery sequential management in microgrid via adaptive dynamic programming," *IEEE/CAA J. Autom. Sinica*, article in press, 2016. DOI: 10.1109/JAS.2016.7510262.

[31] Q. Zhao, H. Xu, and S. Jagannathan, "Near optimal output feedback control of nonlinear discrete-time systems based on reinforcement neural network learning," *IEEE/CAA J. Autom. Sinica*, vol. 1, no. 4, pp. 372−384, Oct. 2014.

[32] Q. Wei, D. Liu, G. Shi, and Y. Liu, "Optimal multi-battery coordination control for home energy management systems via distributed iterative

adaptive dynamic programming," *IEEE Trans. Ind. Electron.*, vol. 42, no. 7, pp. 4203−4214, Jul. 2015.
[33] Q. Wei, D. Liu, and H. Lin, "Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 840−853, Mar. 2016.
[34] Q. Wei, F. Wang, D. Liu, and X. Yang, "Finite-approximation-error based discrete-time iterative adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2820−2833, Dec. 2014.
[35] H. Li and D. Liu, "Optimal control for discrete-time affine non-linear systems using general value iteration," *IET Control Theory Appl.*, vol. 6, no. 18, pp. 2725−2736, Dec. 2012.
[36] W. Gao and Z.-P. Jiang, "Adaptive dynamic programming and adaptive optimal output regulation of linear systems," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 4164−4169, Dec. 2016.
[37] Y. Duan, Y. Lv, J. Zhang, X. Zhao, and F.-Y. Wang, "Deep learning for control: The state of the art and prospects," *Acta Autom. Sinica*, vol 42, no. 5, pp. 643−654, 2016.
[38] F.-Y. Wang, "Building knowledge structure in neural nets using fuzzy logic," *Robotics and Manufacturing: Recent Trends in Research Education and Applications*, M. Jamshidi (Eds.), New York, NY, ASME (American Society of Mechanical Engineers) Press, 1992.
[39] F.-Y. Wang and H.-A. Kim, "Implementing adaptive fuzzy logic controllers with neural networks: a design paradigm," *J. Intell. Fuzzy Syst.*, vol. 3, no. 2, pp. 165-180, 1995.
[40] F.-Y. Wang, "The emergence of intelligent enterprises: From CPS to CPSS," *IEEE Intell. Syst.*, vol. 25, no. 4, pp. 85-88, 2010.
[41] C. Nyce, "Predictive analytics white paper," American Institute for Chartered Property Casualty Underwriters/Insurance Institute of America, 2007.
[42] W. Eckerson, "Extending the value of your data warehousing investment," The Data Warehouse Institute, USA, 2007.
[43] J. R. Evans and C. H. Lindner, "Business analytics: The next frontier for decision sciences," *Decision Line*, vol. 43, no. 2, pp. 1−4, Mar. 2012.
[44] J. Zhang, Q. Wei, and F.-Y. Wang, "Parallel dynammic programming with an average-greedy mechanism for discrete systems, " SKL-MCCS/QAII Tech Report 01-09-2016, ASIA, Beijing, China.
[45] F.-Y. Wang, "Parallel control: a method for data-driven and computational control," *Acta Autom.a Sinica*, vol.39, no. 2, pp. 293−302, 2013.
[46] F.-Y. Wang, "Control 5.0: From Newton to Merton in Popper's Cyber-Social-Physical Spaces," *IEEE/CAA J. Autom. Sinica*, vol. 3, no. 3, pp. 233−234, 2016.

Fei-Yue Wang (S'87-M'89-SM'94-F'03) received his Ph. D. in Computer and Systems Engineering from Rensselaer Polytechnic Institute, Troy, New York in 1990. He joined the University of Arizona in 1990 and became a Professor and Director of the Robotics and Automation Lab (RAL) and Program in Advanced Research for Complex Systems (PARC-S). In 1999, he founded the Intelligent Control and Systems Engineering Center at the Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, under the support of the Outstanding Oversea Chinese Talents Program from the State Planning Council and "100 Talent Program" from CAS, and in 2002, was appointed as the Director of the Key Lab of Complex Systems and Intelligence Science, CAS. In 2011, he became the State Specially Appointed Expert and the Director of The State Key Laboratory of Management and Control for Complex Systems. Dr. Wang's current research focuses on methods and applications for parallel systems, social computing, and knowledge automation. He was the Founding Editor-in-Chief of the International Journal of Intelligent Control and Systems (1995-2000), Founding EiC of IEEE ITS Magazine (2006-2007), EiC of IEEE Intelligent Systems (2009-2012), and EiC of IEEE Transactions on ITS (2009-2016). Currently he is EiC of China's Journal of Command and Control. Since 1997, he has served as General or Program Chair of more than 20 IEEE, INFORMS, ACM, ASME conferences. He was the President of IEEE ITS Society (2005-2007), Chinese Association for Science and Technology (CAST, USA) in 2005, the American Zhu Kezhen Education Foundation (2007-2008), and the Vice President of the ACM China Council (2010-2011). Since 2008, he is the Vice President and Secretary General of Chinese Association of Automation. Dr. Wang is elected Fellow of IEEE, INCOSE, IFAC, ASME, and AAAS. In 2007, he received the 2nd Class National Prize in Natural Sciences of China and awarded the Outstanding Scientist by ACM for his work in intelligent control and social computing. He received IEEE ITS Outstanding Application and Research Awards in 2009 and 2011, and IEEE SMC Norbert Wiener Award in 2014. Corresponding author of this paper.

Jie Zhang (M'16) is an associate professor with The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. His current research interests include mechanism design and optimal control in e-commerce and traffic systems. He received his Ph.D. degree in Technology of Computer Application from University of Chinese Academy of Sciences in 2015. He received his BSc. degree in Information and Computing Science from Tsinghua University in 2005, and received MSc. degree in Operations Research and Control Theory from Renmin University of China in 2009.

Qinglai Wei (M'11) received Ph.D. degree in control theory and control engineering, from the Northeastern University, Shenyang, China, in 2009. From 2009–2011, he was a postdoctoral fellow with The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. He is currently a Professor of the institute. He is also a Professor of the University of Chinese Academy of Sciences. He has authored two books, and published over 60 international journal papers. His research interests include adaptive dynamic programming, neural-networks-based control, optimal control, nonlinear systems and their industrial applications.

Dr. Wei is an Associate Editor of IEEE Transaction on Systems Man, and Cybernetics: Systems since 2016, Information Sciences since 2016, Neurocomputing since 2016, Optimal Control Applications and Methods since 2016, Acta Automatica Sinica since 2015, and has been holding the same position for IEEE Transactions on Neural Networks and Learning Systems during 2014–2015. He is the Secretary of IEEE Computational Intelligence Society (CIS) Beijing Chapter since 2015.

Xinxu Zheng received the B.S. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2011. He is currently working toward the Ph.D. degree in computer science and engineering at the University of Minnesota, Minneapolis, MN, USA. His research interests include social computing, machine learning, and data analytics.

Li Li (S'05-M'06-SM'10-F'17) is currently an associate professor with Department of Automation, Tsinghua University, China. His research interests include complex and networked systems, intelligent control and sensing, intelligent transportation systems and intelligent vehicles. Dr. Li had published over 50 SCI indexed international journal papers and over 50 international conference papers as a first/corresponding author. He serves as an Associate Editor for *IEEE Transactions on Intelligent Transportation Systems.*