Published on IEEE UEMCON Conf

Deep Reinforcement Learning with IoT System Characterization and Knowledge Adaptation

Jiadao Zou
Department of Electrical and Computer Engineering
Purdue School of Engineering and Technology
Indianapolis, IN, USA

Qingxue Zhang, *Senior Member, IEEE*
Department of Electrical and Computer Engineering
Department of Biomedical Engineering
Purdue School of Engineering and Technology
Indianapolis, IN, USA

***Abstract*—With many devices in the IoT big data system, the challenge arises on how to configure the devices based on the system constraints like the energy budget. The typical scenarios include many wearable health sensors or inter-of-things. In this study, targeting this challenge, we propose a novel deep reinforcement learning framework for configuring the system for optimal operations, which automatically learns through actions and feedbacks. Our novel contributions are two-fold, which greatly boost the performance. First, we propose a system characterization approach that can extract the patterns of the system states of many devices to determine whether the system state has significant changes. Second, we propose a knowledge adaptation approach to determine when to update the learning buffer and how to select a learning batch. Further, we have investigated hyperparameters including learning rate, batch normalization and regularization methods, on deep reinforcement learning outcomes. Evaluated on a multi-device system setup, the proposed framework has demonstrated significant performance boosting with the novel designs. This study will greatly advance intelligent configuration for systems with many wearables or IoT devices, towards big data practices.**

Keywords—Deep Reinforcement Learning, Big Data, System Constraints, Optimization

I. INTRODUCTION

Smart devices like wearables and Internet of Things, are bringing emerging applications to many fields, such as smart health [1, 2], smart home and smart world [3, 4]. The wearable motion sensor can provide real-time physical activity type detection and energy expenditure tracking. And the Electrocardiogram monitor can provide real-time heart disease monitoring, to predict acute or manage chronic cardiac conditions.

However, along with so many devices, there is a significant requirement on the energy budget towards long-term applications. Nowadays, it is still challenging to support multiple devices for continuous and long-term monitoring applications, since these devices are usually energy hungry. In this study, targeting this challenge, we propose a novel deep reinforcement learning framework to optimally configure the devices for both energy optimization and performance maximization.

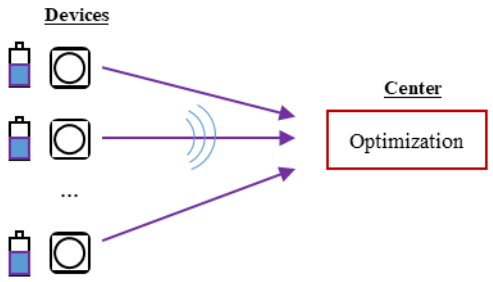

Fig. 1 Optimal multi-device system configuration with reinforcement learning on the center node.

The challenge, more specifically, arises from the complex relationships among the devices. With a number of devices and a center node that communicates with and controls the devices, the system dynamics are highly complex and manual configuration of the system is impractical. As shown in Fig. 1, multiple devices are wirelessly connected to the center node, and each device usually has its own energy budget and computing resources. The center is thus needed to effectively manage the tasks for the devices, while taking into account the crucial system constraints.

There are some previously reported studies on multi-device system optimization. To tackle such mixed integer programming problems [5, 6], traditional methods like dynamic programming [7, 8] and brand-and-bound algorithms [9] can be applied. Nevertheless, the computing complexity is high. The local search or other heuristic approaches [10-13] are then optional alternatives, but they still need a significant number of iterations for solution search. New approaches were also reported using deep reinforcement learning. Huang *et. al.,* [14] proposed the edge-mobile network optimization approach based on deep reinforcement learning of the actions and device states, which shows promising performance on the system configuration. However, for deep reinforcement learning, the learning effectiveness is critical. We aim to enhance the

This study is supported by NSF CAREER Award Grant, USA.
Jiadao Zou and Qingxue Zhang (qxzhang@purdue.edu) are with Purdue School of Engineering and Technology, Indianapolis, Indiana, USA.

intelligence of the deep reinforcement learning with more 'intelligence' so that it can characterize the system states more effectively and remember the knowledge more efficiently.

In this study, we propose a novel deep reinforcement learning framework, which automatically learns through actions and feedbacks. Our novel contributions are two-fold, aiming to greatly boost the performance. First, we propose a system characterization approach that can extract the patterns of the system states to determine whether the system state has significant changes. Second, we propose a knowledge adaptation approach to determine when to update the learning buffer and how to select a learning batch. Further, we have investigated hyperparameters including learning rate, batch normalization and regularization methods, on deep reinforcement learning outcomes. Evaluated on a multi-device system setup, the proposed framework has demonstrated significant performance boosting with the novel designs. This study will greatly advance intelligent configuration for multi-device systems, towards big data practices.

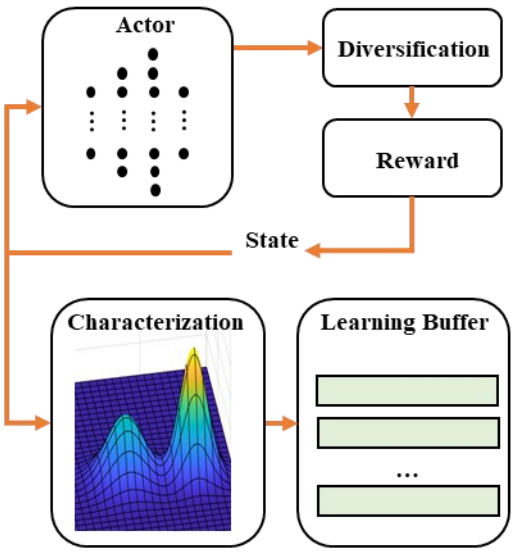

Fig. 2 The proposed novel deep reinforcement learning framework with system characterization and knowledge adaptation.

II. Methods

In this section, we detail the proposed novel deep reinforcement learning architecture, for multi-device system configuration.

A. System Diagram

A system diagram is given in Fig. 2, where the deep reinforcement learning framework reads in the system state, generates actions, diversifies, and selects the action.

More importantly, we propose a system characterizer to intelligently extract the patterns of the system and determine whether the system state has significant changes. We further propose a knowledge adaptation learning buffer which determines when to save and how to select a batch from the buffer.

B. System Characterization

The multi-device system state changes over time, and correspondingly, the deep reinforcement learning framework needs to learn from these state changes. However, usually the system states and actions are simply stored in the buffer for training the framework later. In such a case, there is no differentiation of contributions from the information saved in the buffer, meaning that some information stored may have limited contribution to the framework training due to redundancy.

Therefore, we propose to extract the patterns from the system states, and leverage these patterns to determine where new knowledge needs to be stored in the buffer. We have introduced the probability-based approach, Gaussian Mixture Model (GMM), to characterize the system states. GMM pre-assumes all the data points are from the mixture of a finite number of multivariate Gaussian distributions while the key parameters of mean and covariance are unknown. The GMM model is given in (1), where θ is the underlying distributions, h is the measurement such as the channel gain for each device, and π_i is the weighting factor if needed. GMM aims to estimate the underlying distributions of the current system information.

$$P(h|\theta) = \sum_{i=1}^{C} \pi_i P_i(h|\theta) \quad (1)$$

C. Intelligent Learning Buffer for Knowledge Adaptation

With the system characterization, the learning buffer can then be updated based on the detection of new system states, meaning that if new knowledge is detected, we then store it to the learning buffer. The new knowledge usually reflects significant system state changes and thus can contribute to the diversity of the training process, thereby improving the deep reinforcement learning outcomes. Three learning buffer updating strategies – loose similarity check, strict similarity check, and partial similarity check, are compared to comprehensively demonstrate the outcomes.

We propose to further investigate an effective strategy to leverage the new knowledge in the buffer when generating the gradient for the deep actor neural network during the training process. We here consider two strategies: random batch, and random batch plus new knowledge. The former one selects randomly samples from the buffer; the latter one also selects randomly samples from the buffer but at the same time selects the new knowledge just stored. In such a case, random batch plus knew knowledge is expected to enhance the learning process of the proposed framework.

D. Investigation on Hyperparameters

We have further investigated the critical design parameters for the deep reinforcement learning framework. The learning rate of deep reinforcement learning is essential, since it is based on the rewarding feedback to update the parameters. It is of great importance to select an appropriate speed for the parameter search process. We therefore have different learning rates.

The batch normalization usually scales the input or learned features to standardize them before feeding them into the next layer. It can normalize the contribution of a mini batch and make the learning process smoother. The dropout technique is a popular regularization approach and is expected to improve the generalization ability of the trained model. Therefore, we have also thoroughly studied the contribution of batch normalization and dropout regularization in the deep reinforcement learning process.

E. Evaluation

To evaluate the proposed deep reinforcement learning framework, we have created a complex database with random channel gains that has 1000 time points, and the first 700 are used for training and remaining for testing. 15 devices are connected to the center node, making a complex multi-device system with different energy budget and computing resources. Within each time window, each device can harvest the energy from the center node, and choose to upload the computing task to the center node or choose to perform the task by itself. The center node, within each time window, will transfer energy wirelessly to the devices, and handle the tasks uploaded by devices if there are. The goal is to maximize the computation rate of the whole system including the center and the devices, while making sure the energy constraint on the devices is met. We have run 15 trials per model setting to report the averaged performance.

III. Results

We in this section detail the simulation results and demonstrate that the proposed deep reinforcement learning framework can effectively generate optimal configuration for the multi-device system.

A. Overview of the Performance

Fig. 3 shows the deep reinforcement learning outcomes for the training and testing phases, in time window 1-700, and 701-1000, respectively. The absolute computation rate and the rate normalized to the ground truth found by a traversal-search method are both given. We can observe that the absolute computation rate fluctuates over time due to both system state changes, and different actions generated. More informative illustration can be found from the normalized curve, in which the training part fluctuates between around 100% and 55%, indicating the learning process of the deep reinforcement learning framework. In the testing part, the normalized curve shows attractive performance – above 90%.

B. Intelligent Learning Buffer

TABLE I. Evaluation of different learning buffers.

Learning Buffer	Performance Improvement
Learning Buffer A	8.8
Learning Buffer B	-3.6
Learning Buffer C	0.4

Note. Three learning buffer updating strategies – loose similarity check (A), strict similarity check (B), and partial similarity check (C).

Three learning buffer updating strategies – loose similarity check (A), strict similarity check (B), and partial similarity check (C), are compared in Table I. For A, 90% of system characterization patterns need to fall within the range of the previously stored patterns in the learning buffer. For B, the similarity need is set as 100%. For C, at least one GMM distribution falls within the range specified by the current buffer. The performance is reported as performance improvement, by comparing to the previously reported state-of-the-art method in [14]. The comparison shows the loose similarity check (A) achieves the highest performance, due to its ability to check all the patterns and allow for some flexibility for each pattern.

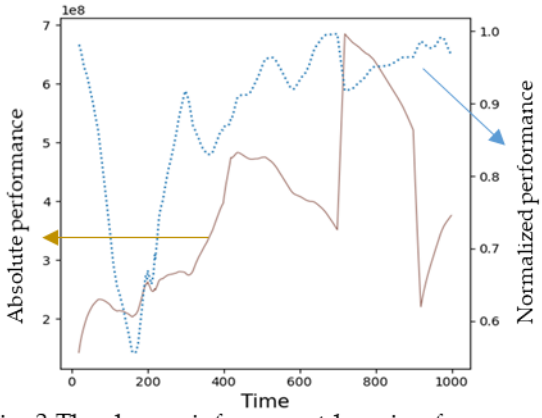

Fig. 3 The deep reinforcement learning framework learning and testing process, in time window 1-700, and 701-1000, respectively.

C. System Characterization

Multiple learning patterns for system characterization are evaluated here. The window sizes are set as 20 (A), 10 (B), and 50 (C) time points, respectively, as shown in Table II. The window size determines how much information is leveraged in the learning process. We found that 20-time points are the best case for scenario characterizing during deep reinforcement learning. It means that if the window size is too small, no enough information is evaluated, and if it is too big, the system status already changes significantly.

TABLE II. Evaluation of different learning patterns.

Chracterization	Performance Improvement
Learning Pattern A	8.8
Learning Pattern B	-1.8
Learning Pattern C	-0.9

Notes. The system characterization window sizes are set as 20 (A), 10 (B), and 50 (C) time points, respectively.

D. Knowledge Management

Further, when fetching a batch from the knowledge buffer for deep reinforcement learning model updating, we have designed two approaches – random fetch plus new knowledge (A) and random batch (B). As shown in Table III, the former achieves better performance because the new knowledge has been leveraged in batch learning, which is beneficial for calculating the model gradients for backward error propagation.

TABLE III. Evaluation of different gradient generator methods.

Knowledge Management	Performance Improvement
Learning Batch A	8.8
Learning Batch B	-0.2

Notes. The deep learning gradient is generated based on random batch plus new knowledge (A) and only random batch (B), respectively.

E. Hyperparameters

We have evaluated the learning rate, data normalization, and model regularization, as shown in Table IV and V. The simulation indicates that a learning rate of 0.1 is better for the framework to find effective parameters and optimize the system performance. The batch normalization and dropout bring performance improvement for learning pattern A, i.e.,

when the window size for system characterization is 20-time points. These comprehensive studies have determined the optimal combination of crucial design hyperparameters.

TABLE IV. Evaluation of different learning rates.

Learning Rate	Performance Improvement
Learning Rate A	-7.4
Learning Rate B	8.8

Note. The learning rate is set as 0.005 (A) and 0.1 (B), respectively.

TABLE V. Evaluation of bath normalization and regularization.

BathNorm and Regularization	Performance Improvement
Learning Pattern A, BN, DO	8.8
Learning Pattern A	0.0
Learning Pattern B, BN, DO	-1.8
Learning Pattern B	-0.7

Notes. The batch normalization (BN) and dropout regularization (DO) have impact on the performance. Under learning pattern A, the BN and DO contribute positively to the performance.

IV. Conclusion

We in this study have proposed, developed, and evaluated a novel deep reinforcement learning framework. Compared to traditional deep reinforcement learning, our framework can boost the performance with two major contributions. First, the system characterization approach is proposed to analyze the system states. Second, the knowledge adaptation approach is proposed to determine when to update the learning buffer based on the system characterization results, and how to combine the historical knowledge and the new knowledge for effective learning. Further, we have studied the key design hyperparameters, and then determined the optimal combination. The framework has been evaluated on a complex multi-device configuration task and demonstrates promising effectiveness, compared to the state-of-the-art method. This study will greatly advance complex system configuration applications in the era of big data like wearable and IoT devices.

V. Acknowledgement

This material is based upon work supported by the National Science Foundation CAREER Award 2047849. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. Zou and Q. Zhang, "eyeSay: Eye Electrooculography Decoding with Deep Learning," in *2021 IEEE International Conference on Consumer Electronics (ICCE)*, 2021: IEEE, pp. 1-3.

[2] J. Stauffer, Q. Zhang, and A. Comer, "Deep Reconstruction Learning Towards Wearable Biomechanical Big Data," in *IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES 2021)*, 2020.

[3] J. C. Talwana and H. J. Hua, "Smart world of Internet of Things (IoT) and its security concerns," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2016: IEEE, pp. 240-245.

[4] H. Ning *et al.*, "From Internet to smart world," *IEEE Access,* vol. 3, pp. 1994-1999, 2015.

[5] A. Lodi and A. Tramontani, "Performance variability in mixed-integer programming," in *Theory driven by influential applications*: INFORMS, 2013, pp. 1-12.

[6] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Automated configuration of mixed integer programming solvers," in *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, 2010: Springer, pp. 186-202.

[7] E. V. Denardo, *Dynamic programming: models and applications*. Courier Corporation, 2012.

[8] S. M. Ross, *Introduction to stochastic dynamic programming*. Academic press, 2014.

[9] S. Wahyuningsih and D. R. Sari, "Study of the Brand and Bound Algorithm Performance on Traveling Salesman Problem Variants," in *1st International Conference on Mathematics and Mathematics Education (ICMMEd 2020)*, 2021: Atlantis Press, pp. 204-211.

[10] M. Fischetti and A. Lodi, "Heuristics in mixed integer programming," *Wiley Encyclopedia of Operations Research and Management Science,* 2010.

[11] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30, no. 1.

[12] D. Gulczynski, B. Golden, and E. Wasil, "The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results," *Computers & Industrial Engineering,* vol. 61, no. 3, pp. 794-804, 2011.

[13] A. M. Turhan and B. Bilgen, "Mixed integer programming based heuristics for the Patient Admission Scheduling problem," *Computers & Operations Research,* vol. 80, pp. 38-49, 2017.

[14] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing,* vol. 19, no. 11, pp. 2581-2593, 2019.