EPIC-Q : Equivalent-Policy Invariant Comparison Enhanced Transfer Q-learning For Run-time SoC Performance-Power Optimization

Anmol Surhonne, Haitham S. Fawzi, Florian Maurer, Oliver Lenke, Michael Meidinger, Thomas Wild, and Andreas Herkersdorf

Technical University of Munich, Arcistrasse 21, 80333 Munich, Germany
`{anmol.surhonne}@tum.de`

Abstract. As power density becomes the main constraint of multicore systems, managing power consumption using DVFS while providing the desired performance becomes increasingly critical. Reinforcement learning (RL) performs significantly better than conventional methods in performance-power optimization under different hardware configurations and varying software applications. RL agents learn through trial-and-error by receiving rewards which is defined by an objective function (e.g. instructions-per-second (IPS)) within specified constraints (e.g. power budget). System and application requirements lead to changing objectives and constraints which in turn result in different reward functions. The RL agents adapt to these changing objectives and constraints (and hence reward functions). Equivalent-policy invariant comparison (EPIC) is a popular technique to evaluate different reward functions. EPIC provides a numerical score which quantifies the difference in two reward functions. In this work, we use this EPIC distance (score) to transfer knowledge and improve learning for changing reward functions. Experimental results using a DVFS enabled RISCV based system-on-chip implemented on an FPGA shows 16.2% lower power budget overshoots compared to a tabular Q-learning agent with direct transfer.

Keywords: EPIC, DVFS, reinforcement learning, reward functions, power management, transfer learning

1 Introduction

The power wall has become one of the primary challenges of managing modern computing systems. Meeting application objectives (e.g. IPS, FPS, response time etc.) within system constraints (e.g. power budget) is critical in managing these complex systems. The objectives and constraints are typically conflicting in nature (e.g. Instruction throughput vs power consumption) and hence call for performance-power optimization. Dynamic voltage and frequency scaling (DVFS) is one of the most widely used technique for performance-power optimization in CPU cores. It enables designers to design methods to set the

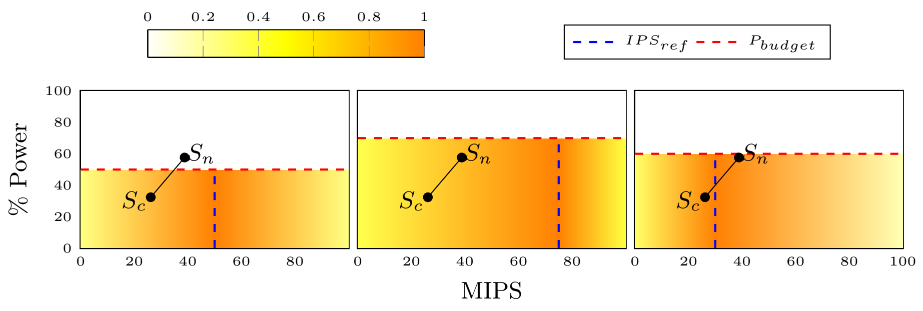

Fig. 1: Visual representation of 3 different reward functions showing an hypothetical state transition from S_c to S_n.

voltage/frequency (V/F) levels of a CPU core by monitoring several CPU performance metrics (e.g. utilization, memory accesses) to meet the application objectives and system constraints. Several heuristics have been proposed in literature to perform performance-power optimization. However these methods are inefficient in the face of stochastic workloads and environments, varying application requirements and system constraints [11].

Reinforcement learning (e.g. Q-learning) is a model-free technique which has been proven to be better than heuristics for run-time power management. RL agents monitor the current state of the system and learn the best action to be taken in the state for a given reward function via trial-and-error. Q-learning maintains a table of Q-values for all states and actions and typically learns by exploration/exploitation in the state-action space as defined by the action selection strategy. The time taken to learn an optimal policy increases exponentially with the size of the state and action spaces. This can be accelerated by using the Experience Replay technique, where the past experiences are stored in a memory buffer and used in batches to speed-up learning.

A reward function is used to numerically quantify the effectiveness of an action taken in a particular state. The reward function is defined by the current objectives and constraints provided to the RL agent. The objective and constraint provided can change dynamically in run time resulting in different reward functions. A policy learnt by the RL agent for the source reward function is not always optimal for the target reward function i.e. the best action in a state for the source reward function might be sub-optimal for the target reward function and vice versa. The RL agent has to unlearn and relearn these sub-optimal state-action pairs which is inefficient and time-consuming. The rate at which the RL agent learns the new reward function can be improved by transfer learning. Equivalent Policy Invariant Comparison (EPIC) is an algorithm which allows us to evaluate the difference between two reward functions without learning the policy [4]. In this work, we propose to use the EPIC-distance as a metric to employ transfer learning in RL agents to improve its learning efficiency for a change in reward function.

Example 1. Consider the 3 different reward functions corresponding to 3 different combinations of IPS reference (IPS_{ref}) and Power budget (P_{budget}) as shown in Fig. 1. Consider the hypothetical state transition as shown from the state S_c to S_n by applying an action a. We can clearly see that the reward achieved by the RL agent for the above transition is different for the different reward functions. The agent has to learn by trial-and-error the Q-value for the state action pair (S_c, a). We can accelerate the learning efficiency by evaluating the EPIC distance between the reward functions and use it to update the Q-value.

The main contributions of this paper are as follows:

- EPIC-Q : A run time Q-learning agent performing performance (IPS) and power optimization in multi-core SoC. The agent is augmented by an experience replay (ER) technique.
- Determining the EPIC scores for different reward functions in run time using experiences stored in the ER buffer.
- Using the EPIC scores to modify the Q-values of states and actions to accelerate learning and improve performance.
- Empirical analysis of performance, number of power budget overshoots and resource utilization of EPIC-Q as a DVFS controller for a RISC based system-on-chip implemented on an FPGA.

2 Background and related work

2.1 Q-learning

Q-learning is a type of reinforcement learning where the agent learns the Q-value of an action in a particular state via trial-and-error. It uses the rewards given by the environment to learn hence model-free. Q-learning agents can handle problems with stochastic transitions and rewards. Tabular Q-learning (QT-DT) is one of the most popular forms of Q-learning where the agent keeps a Q-value for every state-action pair in a table. The Q-value depicts the learned expected reward when an action is applied in a particular state. At each learning epoch t, the Q-value is updated by the following equation:

$$Q(s,a) \leftarrow (1-\beta)Q(s,a) + \beta(R + \gamma max Q(s',a')) \tag{1}$$

where, $Q(s,a)$ is the Q-value, s is the current state, a is the action, s' is the next state, β is the learning rate, γ is the discount factor and R is the reward.

2.2 Experience replay

Experience replay (ER) is a memory replay technique used in reinforcement learning where we store the experiences of the agent at each time-step [14].

Experience replay speeds up the learning process of RL agents by reusing the stored experiences. An experience (e_t) added into the ER buffer at any epoch t is a tuple : $e_t = (s, a, R, s')$. The experiences are stored in a fixed-size memory buffer of size N_{ER}. The ER buffer is typically implemented as a FIFO with uniqueness property. FIFO implies that the most recent experiences are stored in the ER buffer and uniqueness implies that no two experiences in the buffer are identical. The experiences stored in the ER buffer are randomly sampled in each iteration and are used to update the Q-values of the experience's state and action using the experience's reward (using Equation 1).

2.3 EPIC: Equivalent-Policy Invariant Comparison [4]

Quantifying the dissimilarity between different reward functions in Reinforcement Learning can be very useful in gaining information about the learning process, analyzing the learned parameters, and optimizing the agent's behavior in the environment. Most work on quantifying the difference between reward functions evaluated the policies associated with the learned reward. This is time consuming and is sensitive to environment changes. Equivalent Policy Invariant Comparison (EPIC) distance is an elegant metric that can be used to directly quantify the difference between the reward functions without having to resort to their associated policies making it a more robust and invariant metric to the changes in the environment.
EPIC distance is calculated in two key steps. Firstly, the reward functions are canonically shaped (normalized) by evaluating the expectation of the rewards over some arbitrary distributions of the state-action spaces. Such canonicalization ensures that the reward function is independent of the initial state distribution and the transition dynamics within the environment. Secondly, the EPIC distance is calculated as the Pearson distance between the two canonically shaped rewards.

Definition 1. *Canonically shaped reward: Let $R : SxAxS \to \mathbb{R}$ be a reward function. Given distributions D_S and D_A over the states and actions, let X and X' be random variables independently sampled from D_S and A sampled from D_A. The canonically shaped reward is defined as:*

$$C_{D_S,D_A}(R)(s,a,s') = R(s,a,s') + \mathbb{E}[\gamma R(s',A,X') - R(s,A,X') - \gamma R(X,A,X')] \quad (2)$$

Definition 2. *Pearson distance: The Pearson distance between two random variables X and Y is defined as:*

$$D_p(X,Y) = \sqrt{1 - p(X,Y)}/\sqrt{2} \quad (3)$$

where, p(X,Y) is the Pearson correlation between X and Y defined as

$$p(X, Y) = \frac{\mathbb{E}[(X - \mu_x)(Y - \mu_y)]}{\sigma_x \sigma_y} \tag{4}$$

where, (μ_x, σ_x) and (μ_y, σ_x) are the mean and standard deviation of the variables X and Y respectively.

Definition 3. *EPIC Distance: Given two canonically shaped rewards $C_{D_S,D_A}(R_A)$ and $C_{D_S,D_A}(R_B)$ over some distributions D_S and D_A (which are distributions over states S and actions A respectively), the EPIC distance between two rewards R_A and R_B is:*

$$D - EPIC(R_A, R_B) = D_p(C_{D_S,D_A}(R_A), C_{D_S,D_A}(R_B)) \tag{5}$$

2.4 Related work on RL based performance-power optimization

Reinforcement learning based SoC performance-power optimization is a well studied topic in the community. Tabular Q-learning based approaches are proposed in [16, 7, 9]. Classifier system based approaches are proposed in [13, 3]. Neural network approaches using Deep Q-learning and Imitation learning are described in [11, 5, 10]. Many transfer learning techniques have been applied to RL based performance-power optimization. Chen et al. proposed a batch-update method to accelerate convergence of learning of the RL agent by using the reward received for a particular state to update all the other dominated states [2]. Jenkus et al. proposed intra-state (ISLT) and intra-task (ITLT) learning transfer which enhances the standard Q-learning to improve learning [8]. Chen et al. presented a Smart Knowledge Transfer Technique (STQL) which introduces a contradiction checking mechanism to speed up the learning process between two tasks by evicting inappropriate knowledge transfer [1]. All these methods transfer knowledge between states or tasks but for a given fixed reward function. This work proposes a transfer learning technique applied to changing reward functions. Transfer learning between states or tasks can be applied to this work to further improve learning which will be addressed in future work.

3 EPIC-Q based CPU performance-power optimization using DVFS

3.1 Working of Q-learning based DVFS

Fig. 2 shows the block diagram of the EPIC-Q based CPU performance-power optimization. The Q-learning agent operates periodically with a time period t_Q. At every iteration, the QL-agent reads the current state of the system (s) and recommends an action (a) to the CPU based on the action-selection strategy. In

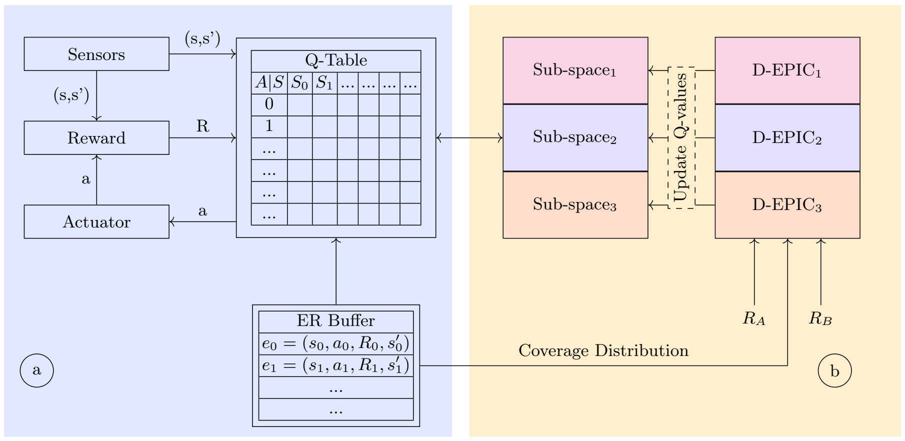

Fig. 2: Working of EPIC-Q: ⓐ Tabular Q-learning using an ER buffer ⓑ Dividing the state-action space into sub-spaces, evaluating EPIC distances for the sub-spaces and using them to update the Q-values.

the next iteration, the reward (R) is used to update the Q-value of the state-action pair (s, a) using Equation 1. The Q-learning agent is augmented with an experience replay buffer. The ER buffer holds the previous experiences seen by the Q-learning agent. Each experience is tuple made up of the state, action, reward and next state (s, a, R, s'). The ER buffer implements the FIFO principle with the uniqueness property as described in Section 2.2. The states, actions, action selection strategy, objectives and rewards are defined as follows:

States and Actions The states and actions of our work are defined as follows:

– The current frequency (f), CPU utilization (u) and instructions-per-second (IPS) constitute the input state of the EPIC-Q agent i.e. $X = \{f, u, IPS\}$. The CPU utilization is calculated as :

$$u = 1 - \frac{\#cycles\_cpu\_stalled}{\#cycles\_total} \tag{6}$$

where, $\#cycles\_cpu\_stalled$ is the number of cycles the CPU was stalled due to cache or branch misses. $\#cycles\_total$ is the number of cycles in the time period t_{lct} at maximum CPU frequency.

– The frequency, utilization and IPS are binned into 8, 16 and 16 bins.

– The actions which can be applied by the EPIC-Q agent are increase (+2, +1) or decrease (-2, -1) or do nothing (±0) in frequency of the processor by a unit step. The voltage level is scaled proportionally w.r.t. the frequency.

Objective and Reward Function EPIC-Q agent is deployed as a low-level controller performing DVFS to optimize the performance (IPS) and power of a core. The objectives (goals) and constraints of the EPIC-Q agent are provided by a software supervisor. The objective of the EPIC-Q agent is to learn to efficiently control the frequency of the core to provide a reference IPS throughput (IPS_{ref}) with a power budget (P_{budget}). These two are conflicting objectives since increasing the frequency increases the IPS throughput and the power budget. The EPIC-Q agent has to provide the closest possible IPS to the IPS reference while minimizing the number of power budget overshoots. The objective ($\mathit{\Delta}$) and reward functions (R) are used to enforce this behavior in the EPIC-Q agent and hence are defined as:

$$\mathit{\Delta} = \frac{|IPS - IPS_{ref}|}{IPS_{max}} \tag{7}$$

$$R = \begin{cases} 1 - \mathit{\Delta}, & \text{if } power \leq power_{budget} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Any action that reduces the deviation to the IPS reference, reduces the objective function and hence receives a higher reward. Violating a power budget is critical since this may lead to thermal violations, and it is crucial to reduce the number of such violations. Therefore, a reward of 0 is assigned for every power budget violation. A visual representation of the defined reward function for different combinations of IPS_{ref} and P_{budget} is shown in Fig. 1.

3.2 Transfer learning using EPIC distance

A change in the objective (IPS_{ref}) or the constraints (P_{budget}) leads to a change in the reward function. The supervisor changes the objective and/or constraint adhering to application and system requirements. EPIC-Q is applied every time there is a change in the reward function. Let R_A be the current reward function and R_B be the new reward function. EPIC-Q works in two stages. First, the state-action space is divided into sub-spaces ($sub - space_n$). Secondly, the EPIC distances ($D - EPIC_n$) is calculated using R_A, R_B and coverage distribution supplied by the ER buffer.

State-action space division We first divide the state-action space into smaller sub-spaces. Fig. 1 shows the reward function for different combinations of IPS_{ref} and P_{budget}. We observe that the reward functions are similar in certain states and different in other states. Determining the EPIC distance for the complete state-action space and using this value for updating the Q-values is inefficient. Let (IPS_{refA},$P_{budgetA}$) and (IPS_{refB},$P_{budgetB}$) be the objectives and constraints for the reward functions R_A and R_B respectively. We can then divide the state-space using these metrics are boundaries to get the sub-spaces.

Updating Q-values using EPIC distances Once the sub-spaces are established, we find the EPIC distances for each sub-space using the reward functions

R_A, R_B and the ER buffer. The ER buffer holds a list of most recent and unique experiences faced by the Q-learning agent. We use this as coverage distribution for calculating the EPIC distances as defined in Section. The EPIC distance quantifies the difference in the two reward functions for the different sub-spaces based on the experiences stored in the ER buffer. We use the EPIC distance as a factor to forget some knowledge learned for the reward function R_A to accelerate the learning of the reward function R_B. An EPIC distance of 0 indicates no change in the sub-spaces and an EPIC distance of 1 indicates a complete contradiction in the reward functions. The EPIC distances of the different sub-spaces are used to then modify the Q-values of the state-action (s, a) pairs of the respective sub-spaces as:

$$Q(s,a) \leftarrow (1 - D - EPIC) * Q(s,a) + (D - EPIC * Q_0) \tag{9}$$

where, Q_0 is the value used for initialization of the Q-learning agent.

Zero Padding Experiences in EPIC calculations The performance of EPIC-Q is directly influenced by the size of the ER buffer and the number of experiences stored in it corresponding to the individual sub-spaces. In run time learning, the inputs to the learning agent are not uniform. The states visited by EPIC-Q is influenced by the objectives and constraints, the applications being run on the processing core and also the number of iterations the agent has been called. The number of experiences corresponding to a sub-space is also influenced by the size of the sub-space itself. In case the number of experiences for a sub-space is lower than a threshold $\theta_{\#exp}$, we perform zero padding while calculating the EPIC distance for the sub-space. This prevents a limited number of experiences to dominate the EPIC distance of the whole sub-space.

4 Experimental Setup and Results

4.1 Experimental Setup

We evaluate our methodology on an Xilinx Virtex 7 FPGA implementing the PULP RISC-V based system-on-chip [12]. The SoC implements a 3-core system and uses the general purpose configuration provided by the PULP platform. The SoC runs on the FPGA with a maximum frequency of 30MHz. Each core in the SoC is controlled by an EPIC-Q agent running as a software process on the same core. We use benchmarks from the MiBench benchmark suite [6] to constitute workloads for our experiment. The MiBench benchmarks have different compute and memory characteristics and we randomly schedule the benchmarks in sequential order to get workload with varying CPU intensiveness. We average each set of experiments over 50 randomly generated workloads. The reward function is changed every 25-30 seconds in our setup. The EPIC-Q agent has a time period t_Q of 5ms, ER buffer size of 100, zero padding threshold ($\theta_{\#exp}$) of 10 and the initial Q-value (Q_0) is 0.125. The learning rate (β) is 0.125 and the discount factor (γ) is 0.1.

4.2 Sweeping EPIC distances vs performance

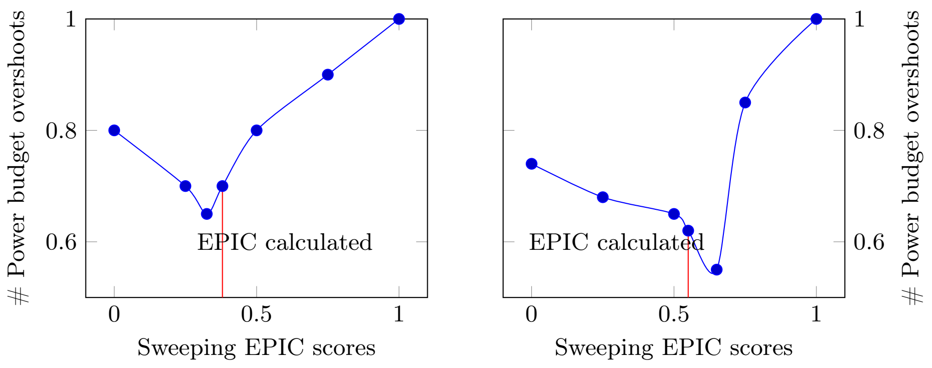

Fig. 3: Sweeping EPIC distance vs number of power budget overshoots where EPIC-Q is used to maximize the performance ($IPS_{ref} = IPS_{max}$) under changing power budgets. The Q-values are updated using Equation 9

Firstly, to see the benefits of transfer learning for a change in reward function using Equation 9, we first train the Q-learning agents for a source domain reward function. We then change the reward function and update the learnt Q-values with different EPIC distance values. The EPIC distances are obtained by linearly sweeping from the range [0,1]. Fig. 3 shows the number of power budget overshoots achieved by the different sweeping EPIC distances for two different scenarios of changes in reward functions. In both scenarios, we maximize the performance under different power budgets i.e. $IPS_{ref} = IPS_{max}$. We initally train the agents in the source domain with a power budget of $0.8P_{max}$. In the first scenario, we change the power budget from $0.8P_{max}$ to $0.6P_{max}$ whereas in the second scenario we change it from $0.8P_{max}$ to $0.4P_{max}$. The EPIC distances calculated in the two scenarios are 0.38 and 0.58 respectively. We can clearly observe that the performance of EPIC-Q by updating the Q-values using Equation 9 improves within the range [0,1]. We also observe that the performance improvement using the calculated EPIC distance is close to the best achieved improvement. This proves that the calculated EPIC distance is a good metric to update Q-values for a change in reward function.

4.3 Results

EPIC-Q based transfer learning is applied when there is a change in reward function (i.e. change in objective IPS_{ref} and/or power budget). We evaluate the effectiveness of our approach against the performance of a standard Q-learning

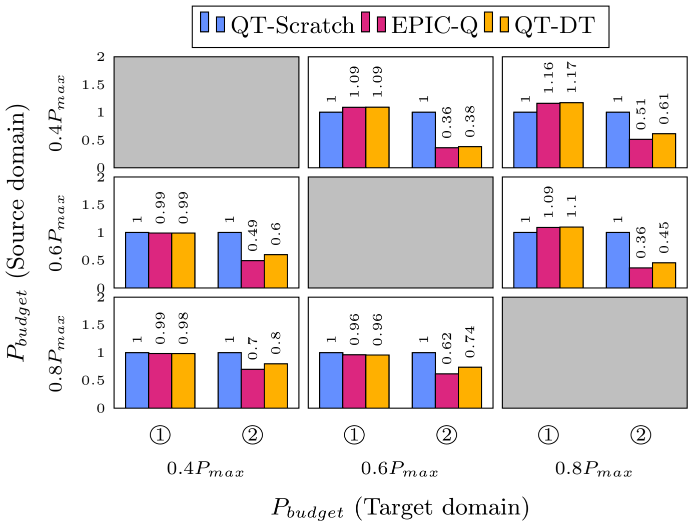

Fig. 4: Results comparing the performance of EPIC-Q, Q-table with direct transfer (QT-DT) and Q-table from scratch (QT-Scratch). ① IPS difference and ② Number of power budget overshoots as metrics.

agent with direct transfer (i.e. the Q-values are transferred from the source domain to target domain without modification) and a Q-learning agent from scratch (i.e. the q-values are reset for a change in reward function). Fig. 4 summarizes the results of our work. We use ① the IPS difference ($|IPS - IPS_{ref}|$) and ② number of power budget overshoots as metrics to evaluate our work. We set IPS_{ref} to IPS_{max} where the agents are required to maximize performance under a power budget. We use three different power budgets ($0.4P_{max}$, $0.6P_{max}$, $0.8P_{max}$) for the source and target domains which gives us 6 different transitions or reward function changes. The bottom left set of graphs in Fig. 4 depict a decrease in power budget whereas the top right set of graphs depict an increase in power budget.
For an increase in power budget, EPIC-Q achieves 59% lower and 17.2% lower power budget overshoots which achieving 11.35% higher and 1% higher IPS difference relative to the QT-Scratch and QT-DT respectively. For a decrease in power budget, EPIC-Q achieves 39.67% lower and 15.67% lower power budget overshoots which achieving 2% lower and 0% lower IPS difference relative to the QT-Scratch and QT-DT respectively. In both the scenarios, EPIC-Q reduces the number of power budget overshoots significantly while trying to maximize the IPS throughput of the CPU core.

Overhead EPIC-Q is used when there is a change in the objective or constraints i.e. the reward function. This in our experimental setup is done every 25-30 seconds. The calculation of the EPIC distances and updating the Q-values for the different sub-spaces requires less than 1ms in our experimental setup which is negligible. The time complexity increases exponentially w.r.t the size of the state-action space. It increases linearly w.r.t the number of sub-spaces.

5 Conclusion

In this work, we proposed EPIC-Q: A Q-learning agent enhanced with Equivalent Policy Invariant Comparison (EPIC) based transfer learning for run time SOC performance-power optimization. The Q-learning agent is augmented with an experience replay feature which accellerates the learning process by batch training. EPIC-Q based transfer learning is applied when there is a change in the objective or constraints (i.e. reward function) of the Q-learning agent. EPIC is a method which quantitatively evaluates the difference between two reward functions (EPIC-distance). We first divide the state-action space of the Q-learning agent into sub-spaces with the objectives (IPS reference) and constraints (power budget) as the margins. We evaluate the EPIC-score for the individual sub-spaces using the ER buffer for experiences and modify the Q-value of the state-action pairs in the sub-space. Experimental results on a DVFS enabled RISC-V based system-on-chip running on an FPGA shows 50% lower and 16.435% lower number of power budget overshoots compared to a Q-learning agent learning from scratch and with direct transfer respectively.
Intra-state and Intra-task learning transfer can further enhance the performance of EPIC-Q which will be explored as future work. The performance of EPIC-Q can be further improved by approximating the state-action space as done in LCS based systems [15] or neural networks [17]. Future work can also address the intersection of the state-action space approximation and EPIC based transfer learning to increase learning efficiency.

Acknowledgements

We would like to thank our colleagues from the Chair of Integrated Systems at TU Munich, our partners in the IPF project: Rolf Ernst (TU Braunschweig, Germany), Nikil Dutt (UC Irvine, USA) and Fadi Kurdahi (UC Irvine, USA) and acknowledge the financial support from the DFG under Grant HE4584/7-2.

References

1. Chen, L., Li, X., Jiang, F., Li, C., Xu, J.: Smart knowledge transfer-based runtime power management. In: 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE) (2023)
2. Chen, Z., Marculescu, D.: Distributed reinforcement learning for power limited many-core system performance optimization. In: 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE) (2015)

3. Donyanavard, B., Mück, T., Rahmani, A.M., Dutt, N., Sadighi, A., Maurer, F., Herkersdorf, A.: Sosa: Self-optimizing learning with self-adaptive control for hierarchical system-on-chip management. In: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (2019)
4. Gleave, A., Dennis, M., Legg, S., Russell, S., Leike, J.: Quantifying differences in reward functions. arXiv preprint arXiv:2006.13900 (2020)
5. Gupta, U., Mandal, S.K., Mao, M., Chakrabarti, C., Ogras, U.Y.: A deep q-learning approach for dynamic management of heterogeneous processors. IEEE Computer Architecture Letters (2019)
6. Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T., Brown, R.B.: Mibench: A free, commercially representative embedded benchmark suite. In: Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538) (2001)
7. Iranfar, A., Shahsavani, S.N., Kamal, M., Afzali-Kusha, A.: A heuristic machine learning-based algorithm for power and thermal management of heterogeneous mpsocs. In: 2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED) (2015)
8. Jenkus, D., Xia, F., Shafik, R., Yakovlev, A.: Runtime energy minimization of distributed many-core systems using transfer learning. In: 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE) (2022)
9. Liu, W., Tan, Y., Qiu, Q.: Enhanced q-learning algorithm for dynamic power management with performance constraint. In: 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010) (2010)
10. Mandal, S.K., Bhat, G., Patil, C.A., Doppa, J.R., Pande, P.P., Ogras, U.Y.: Dynamic resource management of heterogeneous mobile platforms via imitation learning. IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2019)
11. Rapp, M., Amrouch, H., Lin, Y., Yu, B., Pan, D.Z., Wolf, M., Henkel, J.: Mlcad: A survey of research in machine learning for cad keynote paper. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2021)
12. Rossi, D., Conti, F., Marongiu, A., Pullini, A., Loi, I., Gautschi, M., Tagliavini, G., Capotondi, A., Flatresse, P., Benini, L.: Pulp: A parallel ultra low power platform for next generation iot applications. In: 2015 IEEE Hot Chips 27 Symposium (HCS) (2015)
13. Surhonne, A., Doan, N.A.V., Maurer, F., Wild, T., Herkersdorf, A.: Gae-lct: A run-time ga-based classifier evolution method for hardware lct controlled soc performance-power optimization. In: International Conference on Architecture of Computing Systems (2022)
14. Surhonne, A., Maurer, F., Wild, T., Herkersdorf, A.: Lct-der: Lct with dynamic sized experience replay for runtime soc performance-power optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion (2023)
15. Surhonne, A., Maurer, F., Wild, T., Herkersdorf, A.: Lct-tl: Learning classifier table (lct) with transfer learning for runtime soc performance-power optimization. In: 2023 IEEE 16th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC) (2023)
16. Wang, Z., Tian, Z., Xu, J., Maeda, R.K., Li, H., Yang, P., Wang, Z., Duong, L.H., Wang, Z., Chen, X.: Modular reinforcement learning for self-adaptive energy efficiency optimization in multicore system. In: 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC) (2017)
17. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proceedings of the IEEE (2020)