Task Offloading Method for Industrial Internet of Things (IIoT) Targeting Computational Resource Management

Wenhui Wang[1(✉)], Xuanzhe Wang[1], Zhenjiang Zhang[1], and Zeng Jianjun[2]

[1] Beijing Jiaotong University, Shangyuan Village, Haidian District, No.3, Beijing, China
23111039@bjtu.edu.cn

[2] Beijing InchTek Technology, 1 Baiziwan South Road, Beijing, China

Abstract. In the context of industrial scenarios, devices exhibit specificity and task arrival rates vary over time. Considering real-world task queuing issues and incorporating edge computing offloading and D2D offloading techniques, this paper proposes TVTAO for computational resource management to meet latency requirements. First, three offloading decisions are introduced, then offloading policy constraints are proposed to restrict devices from selecting the same task for execution during task offloading. Simulation results demonstrate that the TVTAO algorithm can reasonably make task offloading decisions and allocate computational resources, effectively reducing the average processing latency of the overall system.

Keywords: edge computing offloading · D2D offloading · time-varying task offloading · deep reinforcement learning

1 Introduction

In an Industrial Internet of Things (IIoT) setting, intelligent devices often need to handle latency-sensitive and computationally intensive tasks, yet they have limited computational resources [1]. Task offloading strategies can be employed to transfer tasks to nearby edge computing servers or other intelligent devices for processing, thereby reducing the system's task latency [2]. However, real-world scenarios present challenges such as limited computational resources, varying task arrival rates [3], and device heterogeneity in the design of system policies. To address these issues, the paper proposes an offloading algorithm called TVTAO, utilizing deep reinforcement learning in conjunction with DDPG to find near-optimal solutions [4].

Given the challenges of dealing with numerous intelligent devices and various offloading constraints, this paper introduces the TVTAO offloading algorithm. It employs deep reinforcement learning to model the overall offloading strategy, aiming to find near-optimal solutions [5]. This algorithm effectively addresses the constraints posed by limited computational resources, fluctuating task arrival rates, and device heterogeneity in the context of the Industrial Internet of Things [6], resulting in a significant improvement in the overall service quality and reduction of task latency within the system [7].

D.-J. Deng and J.-C. Chen (Eds.): SGIoT 2023, LNICST 557, pp. 39–44, 2024.
https://doi.org/10.1007/978-3-031-55976-1_4

2 System Model

This paper considers an industrial Internet of Things (IIoT) scenario involving an Edge Computing Server (ESC) and multiple intelligent devices [8]. Within this scenario, intelligent devices can take on two roles: Computing Resource Consumer (CRC) and Computing Resource Provider (CRP) [9], as is shown in Fig. 1. CRC refers to an intelligent device that offloads some or all of its local tasks to other devices [10], requiring additional computing resources from these devices to complete its tasks efficiently, while Computing Resource Provider (CRP) is an intelligent device that executes its local tasks entirely within its own computing resources and may potentially offer computing resources to other devices [11].

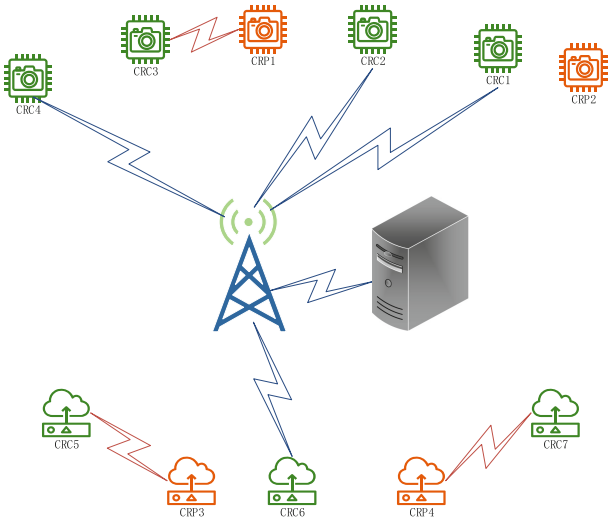

Fig. 1. Industrial Internet edge computing and D2D task offloading scenario

Assuming there are M intelligent devices in the scenario, a Base Station(BS) and an Edge Computing Server (ESC) [12]. The system proceeds through several time slots, with the intelligent devices making their offloading decisions within a single time slot. These offloading decisions primarily consist of three options: local computation, D2D offloading and edge computing offloading [13].

The status of whether device m offloads a task to the ESC in time slot t is represented by $\beta_t^{m,n} \in \{0, 1\}$, $\beta_t^{m,n} = 1$ represents that device m offloads tasks to device n, $\beta_t^{m,m} = 1$ represents that device m is CRP [14]. The offloading decision of device m with respect to D2D can be represented as formula 1:

$$\beta_t^m = \{\beta_t^{m,1}, \beta_t^{m,2}, \ldots, \beta_t^{m,M}\}^T, m \in \mathcal{M} \tag{1}$$

3 Methods

To address the optimization problem in the context of task offloading for IIoT devices, it is essential to consider mathematical model information such as the system's channel conditions and statistical distributions. However, in practice, obtaining this information is often unfeasible. Therefore, we have adopted an online solution based on deep reinforcement learning. This approach enables real-time resource allocation through interaction with the system, effectively making decisions regarding task offloading [15]. We have designed state spaces, action spaces, and reward functions, and proposed a time-varying task offloading algorithm based on edge computing and D2D, allowing the system to reduce overall system latency.

3.1 Notation

The system state is regarded as a set of parameters that can be used to describe the system. Based on the system model proposed in this paper, the system state at any time slot t is defined as formula 2:

$$s_t = \{A_t, T^r_{SD,t}, T^r_{ESC,t}\} \tag{2}$$

while A_t represents the number of tasks, $T^r_{SD,t}$ represents the remaining task execution latency of intelligent devices, $T^r_{ESC,t}$ represents that of edge server.

Based on the observed system state s_t, deep reinforcement learning will select an action based on decision variables, which can be represented as formula 3:

$$act_t = \{\alpha_t, \beta_t, o_t, f_{ESC,t}\} \tag{3}$$

while α_t represents the device makes offload decision or not, β_t represents the device performs full local computation or not, o_t represents the number of tasks.

The reward function is designed as the negative of the system latency, as is shown in formula 4:

$$r_t = \mathcal{R}(s_t, a_t) = -T_t \tag{4}$$

3.2 Task Offload Algorithm

This article combines the DDPG algorithm with real-world Industrial Internet of Things scenarios and proposes the TVTAO algorithm, as illustrated in the algorithm process shown in Table 1.

Table 1. The description of TVTAO training algorithm

Input: $K_{max}, T_{max}, |\mathcal{R}_B|, B, \lambda^Q, \lambda^\mu, C^Q, C^\mu$

Randomly initialize $Q(s,a|\theta^Q)$, $\mu(s|\theta^\mu)$, $Q'(s,a|\theta^{Q'})$ and $\mu'(s|\theta^{\mu'})$, $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$。

Reset $\mathcal{R}_B$.

For each eposide $k = 1,2,\ldots,K_{max}$:

  Randomly set an initial state s_1。

  For each step $t = 1,2,\ldots,T_{max}$:

    $a_t = \mu(s|\theta^\mu) + \Delta\mu$。

    Do action a_t and observe reward $r_t = \mathcal{R}(s_t, a_t) = -T_t$ and new state s_{t+1}.

    Put transition(s_t, a_t, r_t, s_{t+1}) in $\mathcal{R}_B$. Drop the oldest sample if $\mathcal{R}_B$ is full.

    Randomly select B transitions(s_t, a_t, r_t, s_{t+1}) as mini-batch。

    For each transition $b = 1,2,\ldots,B$:

      Calculate $y^{(b)} = r_i^{(b)} + \gamma Q'(s_{i+1}^{(b)}, \mu'(s_{i+1}^{(b)}|\theta^{\mu'})|\theta^{Q'})$。

    End For

    Update $Q(s,a|\theta^Q)$ by $\theta^Q \leftarrow \theta^Q - \lambda^Q \nabla_{\theta^Q} L^Q$, while loss function is defined as $L^Q = \frac{1}{B}\sum_{b=1}^{B}[y^{(b)} - Q(s_i^{(b)}, a_i^{(b)}|\theta^Q)]^2$。

    Update $\mu(s|\theta^\mu)$ by $\theta^\mu \leftarrow \theta^\mu - \lambda^\mu \nabla_{\theta^\mu} J^\mu$, while loss function is defined as $J^\mu = \mathrm{E}_{s,a}[Q^\mu(s,a)]$, and:

$$\nabla_{\theta^\mu} J^\mu \approx \frac{1}{B}\sum_{b=1}^{B} \nabla_a Q\left(s_i^{(b)}, a\middle|\theta^Q\right)|_{a=a_i^{(b)}} \times \nabla_{\theta^\mu}\mu\left(s_i^{(b)}\middle|\theta^\mu\right)$$

    Update parameters $\theta^{Q'} = \theta^Q$, $\theta^{\mu'} = \theta^\mu$ every C^Q and C^μ steps, respectively.

  End For

End For

Output: approximate optimal policy μ^*

4 Simulation Results

Due to the inclusion of two types of intelligent tasks with different CPU cycles, the setting of CPU cycles is based on the original task CPU cycle setting, with a relative increase or decrease. The specific relationship is shown in Fig. 2.

Due to the inclusion of two types of intelligent tasks with different CPU cycles, the setting of CPU cycles is based on the original task CPU cycle setting, with a relative increase or decrease. The specific relationship is shown in Fig. 2.

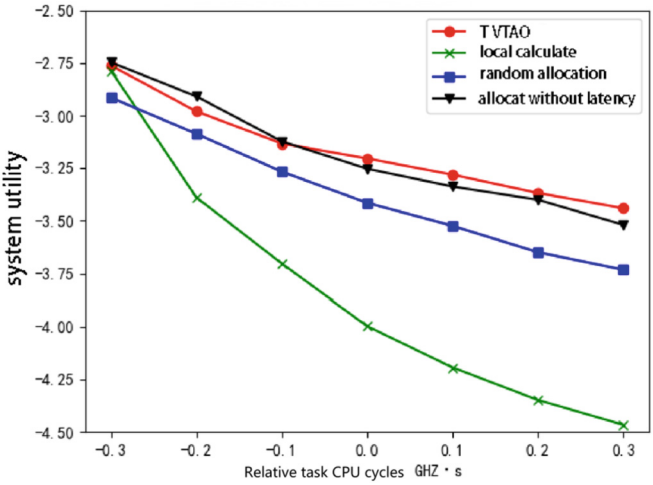

Fig. 2. The relationship between system utility and the number of CPU cycles

It can be seen that as the number of CPU cycles in a task increases, the system utility decreases. This is because as the number of CPU cycles increases, the task execution time becomes longer and the queuing delay also increases. Local computing significantly reduces system utility due to task accumulation on local devices; Random allocation can randomly use edge computing server and other device resources, so the system utility decline is less than that of local computing. The TVTAO algorithm can analyze the queuing delay and computing resource status of the current system environment. Compared to other strategies, it can choose the optimal offloading strategy to achieve higher system utility when the number of CPU cycles in the task changes.

5 Conclusion

This paper investigates the problem of managing computational resources for multiple tasks in the context of resource-constrained intelligent devices within an Industrial Internet of Things (IIoT) scenario. Firstly, considering the entire IIoT scenario, three offloading decisions are proposed: local computation, D2D offloading, and edge offloading. Secondly, to simulate the varying number of tasks due to factors like geographic location in industrial scenarios, a task offloading algorithm named TVTAO, based on deep deterministic policy gradients, is proposed to handle time-varying task arrivals. Finally, experiments and simulations are conducted using the PyTorch framework. The simulation results demonstrate that the algorithm performs exceptionally well in complex industrial scenarios.

References

1. Mohri, M, Rostamizadeh A, Talwalkar A. Foundations of machine learning. MIT press, 2018
2. Hassan, N., Gillani, S., Ahmed, E., et al.: The role of edge computing in internet of things. IEEE Commun. Mag. **56**(11), 110–115 (2018)
3. S. Caldas, J. Konečny, H.B. McMahan, et al.: Expanding the reach of federated learning by reducing client resource requirements. arXiv preprint arXiv:1812.07210 (2018)
4. Hsieh, K., Phanishayee, A., Mutlu, O., et al.: The non-iid data quagmire of decentralized machine learning. In: International Conference on Machine Learning, pp. 4387–4398. PMLR (2020)
5. Mining, W.: Data mining: concepts and techniques. Morgan Kaufinann **10**, 559–569 (2006)
6. Smith, V., Chiang, C.K., Sanjabi, M., et al.: Federated multi-task learning. Advances in neural information processing systems, p. 30 (2017)
7. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
8. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
9. Schulman, J., Wolski, F., Dhariwal, P., et al. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
10. Caruana, R.: Multitask learning. Mach. Learn. **28**, 41–75 (1997)
11. Hanzely, F., Richtárik, P.: Federated learning of a mixture of global and local models. arXiv preprint arXiv:2002.05516 (2020)
12. Ruder, S.: An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098 (2017)
13. Watkins, C.J.C.H., Dayan, P.: Q-learning. Mach. Learn. **8**, 279–292 (1992)
14. Ghosh, A., Hong, J., Yin, D., et al.: Robust federated learning in a heterogeneous environment. arXiv preprint arXiv:1906.06629 (2019)
15. Jie, Y., Guo, C., Choo, K.K.R., et al.: Game-theoretic resource allocation for fog-based industrial internet of things environment. IEEE Internet of Things J. **7**(4), 3041–3052 (2020)