无线移动边缘计算网络中在线计算卸载的深度强化学习

摘要

无线功率传输技术和移动边缘计算技术的结合有利于解决物联网设备的电池寿命和计算能力受到限制的问题，无线供能的移动边缘计算结合了这两种技术的优势最近成为增强低能耗网络数据处理能力的一种有前途的范例。复现论文考虑了一个采用二进制卸载决策的无线供能的移动边缘计算网络，该网络中无线设备的每个计算任务要么在本地执行，要么完全卸载至比无线设备计算能力要高的接入点上，并提出了一种基于深度强化学习的在线卸载框架 (DROO)，该框架实现了深度神经网络作为可扩展的解决方案，从经验中学习二进制卸载决策，并自动调整 DROO 算法的参数，降低了复杂性。它消除了求解组合优化问题的需要，从而大大降低了尤其在大规模网络中的计算复杂性。实验结果数据表明，所提出的算法能够实现近似最优性能。同时我也修改了部分实验参数，并分析修改参数造成实验结果变化的原因。

关键词： 移动边缘计算；无线电力传输；强化学习；资源分配

1 引言

随着计算技术的发展，现代物联网设备在我们的日常生活中扮演着越来越重要的角色，从智能家居到工业自动化。但是，这些设备通常在电池寿命和计算能力方面受到限制，这对其性能和使用体验产生了影响。然而，无线功率传输技术和移动边缘计算技术的发展为解决这些问题提供了新的可能性 [1] [2]。

无线功率传输主要通过电磁感应、电磁共振、射频、微波、激光等方式实现非接触式的电力传输，将制造出来的电力转换成为无线电波发送出去，再通过特定的接收装置将无线电波收集起来并转换为电力，供人们使用。无线功率传输的最新进展表明无线设备的电池可以在空中连续充电而无需更换电池 [3]。与此同时，利用移动边缘计算技术可以通过将无线设备的本地任务卸载至边缘服务器上从而增强设备的计算能力，以减少计算延迟和能耗 [4] [5]。

无线供能的移动边缘计算结合了这两种技术的优势，有利于解决物联网设备的两个基本性能限制，通过无线方式为设备供电，并利用边缘服务器增强计算能力，对于物联网设备的性能提升具有重要意义。本文复现的论文 [6] 考虑了如图 1所示的无线供能的移动边缘计算网络，该网络中有一个接入点 (AP) 和多个无线设备 (WD)，其中接入点负责向无线设备传输射频能量并从需要进行任务卸载的无线设备接收计算任务进行边缘计算处理。需要注意的是，无线设备遵循二进制任务卸载策略 [7]，该设备的任务要么在本地计算，要么卸载到 MEC 服

务器进行远程计算。在多用户场景中，主要挑战是单个计算模式 (卸载或本地计算) 和无线资源分配 (比如无限功率传输和卸载任务之间的时间分配) 的联合优化。在复现的论文中，目标是根据时变的无线信道共同优化单个无线设备的任务卸载决策、无线功率传输与任务卸载之间的传输时间分配以及多个无线设备之间的时间分配。为此，复现论文提出了一种基于深度强化学习的在线卸载 (DROO) 框架，以最大化所有无线设备的计算率的加权和，即单位时间内处理的比特数。

我通过复现原论文的代码得到结果表明 DROO 算法的计算率达到现有近最优基准方法的 99.5% 以上 [2]，接着修改了部分实验参数，并分析修改参数造成实验结果变化的原因。

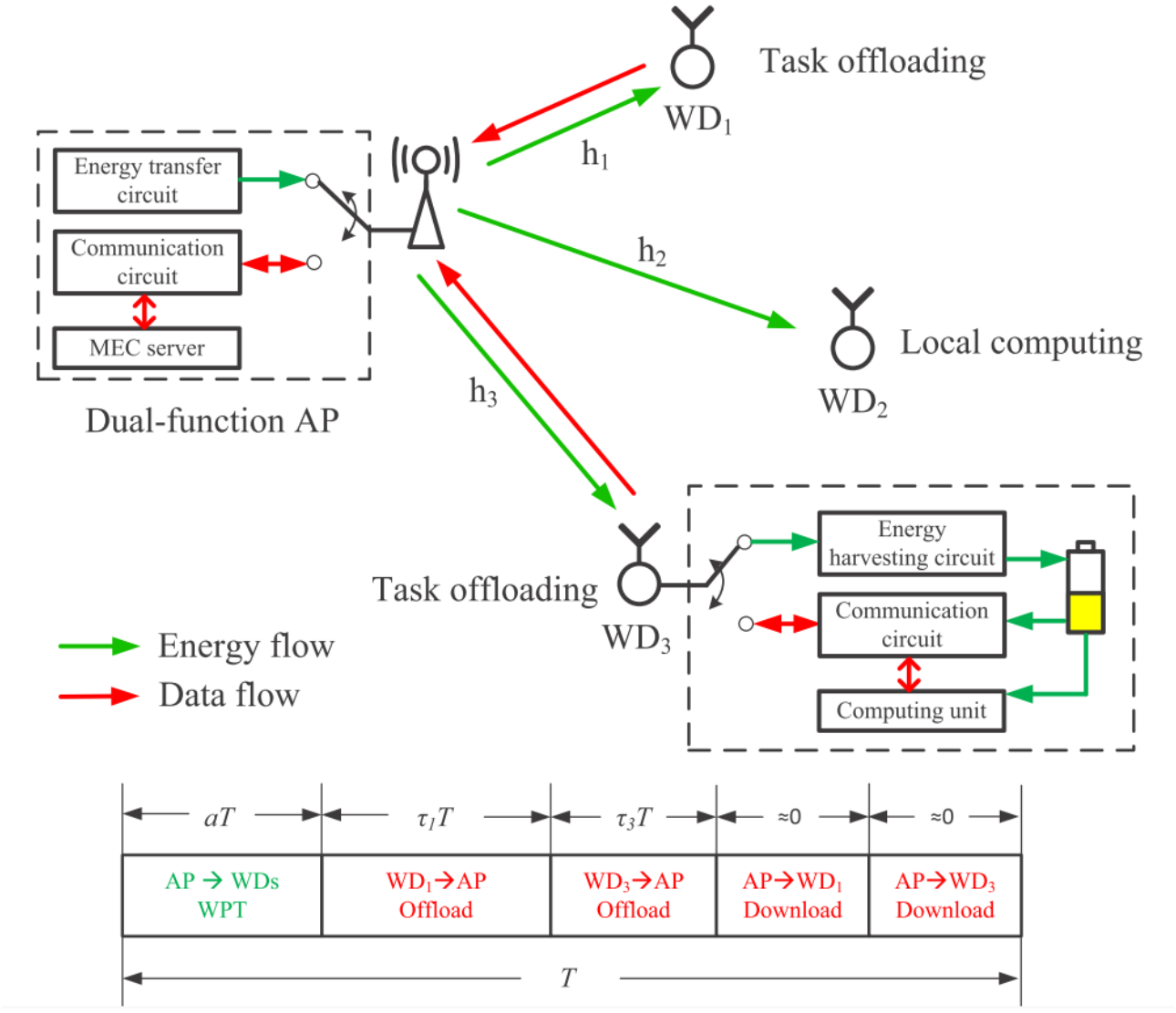

图 1. 考虑无线供电的 MEC 网络和系统时间分配的示例图

2 相关工作

目前已经有许多相关研究将移动边缘计算网络中的计算卸载问题和资源分配问题联合建模为混合整数规划问题。例如，[8] 针对多服务器移动边缘计算网络，研究出了一种类似的启发式搜索方法，该方法迭代地调整二进制卸载决策。另一个广泛采用的启发式方法是通过凸放松实现的，例如，将整数变量放松为在 0 和 1 之间的连续变量 [9]，或通过将二进制约束近

似为二次约束 [10]。但是简化复杂性的启发式方法的解质量并没有保证，并且基于搜索的方法和凸放松方法都需要相当多的迭代才能达到令人满意的局部最优解，这些方法都并不适用于快速衰落的信道。

3 本文方法

3.1 本文方法概述

复现论文旨在解决无线供能的移动边缘网络中根据时变的无线信道条件优化地调整任务卸载和无线资源分配，需要快速求解困难的组合优化问题，传统数值优化方法难以实现。文中提出基于深度强化学习的在线卸载框架（DROO），该方法采用二进制卸载策略，即无线设备的每个计算任务要么在本地执行，要么完全卸载到边缘服务器。将原始优化问题分解为卸载决策子问题和资源分配子问题，设计保序量化卸载动作生成，消除传统数值优化方法的复杂性 [11]，通过动态自适应程序进一步降低计算复杂性。

3.2 系统模型

论文考虑了一个如图 1由一个接入点和 N 个固定的无线设备组成的无线供能的移动边缘计算网络。其中，接入点具有稳定的电源，可以向每个无线设备广播射频能量供电，并且接入点拥有比无线设备更高的计算能力，需要任务卸载的无线设备可以将任务卸载至接入点进行任务处理。无线功率传输和计算卸载在同一频段，两者进行之间相互干扰。

将系统时间划分为等长 T 的连续时间帧，时间帧开始时，将有一部分时间 aT 用于无线功率传输，第 i 个无线设备获得的能量为 $E_i = \mu P h_i a T, a \in [0,1]$ ，其中 μ 表示能量收集效率，P 表示接入点的发射功率，h_i 表示在该时间帧内接入点和第 i 个无线设备之间的无线信道增益。获得能量后，在剩余时间内，每个无线设备需要在时间帧结束前完成一个优先的计算任务。第 i 个无线设备的权重 w_i 越大，分配给第 i 个无线设备的计算速率就越大。（本文忽略了接入点在任务计算和下载上花费的时间。）

3.3 问题公式化

假设在所考虑的周期内，只有无线信道增益 $h = \{h_i | i \in N\}$ 是时变的，其他的如 w_i 都是固定参数。因此，将无线供电 MEC 网络在时间帧中的加权和计算率记为：

$$Q(\mathbf{h}, \mathbf{x}, \boldsymbol{\tau}, a) \triangleq \sum_{i=1}^{N} w_i \left((1 - x_i) \, r_{L,i}^*(a) + x_i r_{O,i}^*(a, \tau_i) \right), \tag{1}$$

其中，x_i 为第 i 个无线设备的卸载决策（0 表示任务在本地计算，1 表示计算任务卸载至任务点），$\boldsymbol{\tau}$ 表示分配给无线设备用于任务卸载的时间的百分比，$r_{L,i}^*(a)$ 表示第 i 个无线设备的本地计算速率，$r_{O,i}^*(a, \tau_i)$ 表示第 i 个无线设备卸载任务计算速率。

论文目标是最大化加权和计算率：

$$\begin{aligned} Q^*(\mathbf{h}, \mathbf{x}) = \underset{\boldsymbol{\tau}, a}{\text{maximize}} \; & Q(\mathbf{h}, \mathbf{x}, \boldsymbol{\tau}, a) \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{N} \tau_i + a \le 1, \\ & a \ge 0, \tau_i \ge 0, \forall i \in \mathcal{N}, \end{aligned} \tag{2}$$

3.4 DROO 算法

3.4.1 目标

设计一个卸载策略函数 π，一旦时间帧开始时给出信道增益 h，它可以快速生成最佳卸载操作 $x^* \in \{0,1\}^N$，策略表示为：

$$\pi : h \to x^* \tag{3}$$

所提出的 DROO 算法从经验中逐渐学习这样的策略函数 π。

3.4.2 算法概述

DROO 算法的结构图如下：

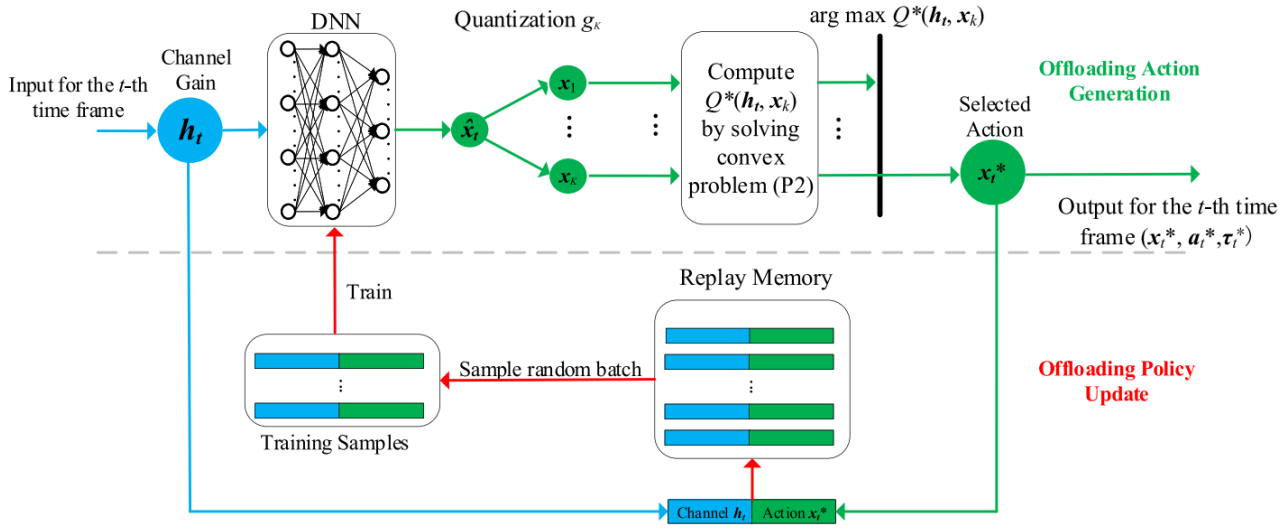

图 2. DROO 算法的原理图

DROO 算法：卸载动作生成和卸载策略更新。

卸载动作生成：使用 DNN 产生，在时间帧 t，DNN 输入该时间帧内的信道增益 h_i，根据当前的卸载策略 π_{θ_t} 输出一个值在 0 到 1 之间的放松卸载动作 $\hat{x}$。然后将放松卸载动作量化为 K 个二进制卸载动作，接着对每个量化后的卸载决策 x_i 算法通过解决一个凸优化问题来确定资源分配，计算完每个卸载决策对应的计算速率，选择一个最佳的卸载动作 x_t^*，输出对应的卸载策略 (x_t^*, a_t^*, τ_t^*)。执行卸载动作 x_t^* 获得一个奖励，并将其添加至 Replay Memory 中。

卸载策略更新：从 Memory 中提取一批训练样本来训练 DNN，从而更新 DNN 的参数，新的卸载策略用于下一个时间步的计算，通过不断迭代，DNN 的策略逐渐改进。

4 复现细节

4.1 与已有开源代码对比

本次复现使用了复现论文所公布的代码，通过对论文源代码的复现生成了与论文中实验效果一致的实验结果。因为卸载决策生成的二进制阈值和更新 K 的时间间隔是算法中较为关

键的参数，所以我提高了二进制的阈值，将其从 0.5 提高至了 0.7，让算法在决定是否卸载任务时变得更加严格，只有当信道条件较好时（DNN 输出值较高）才选择卸载。接着减少了 K 值的更新时间间隔，让算法更频繁地调整考虑的卸载策略数量。

4.2 实验环境搭建

python==3.7.9
Tensorflow==1.15.0
numpy==1.21.6

4.3 创新点

提高卸载决策生成的二进制阈值以及减少了 K 值的更新时间间隔。

5 实验结果分析

本实验考虑一个 $N = 10$ 的无线供电边缘计算网络，初始的二进制生成决策数量 K 初始化为 10，在提出的 DROO 算法中，考虑一个由一个输入层、两个隐藏层和一个输出层组成的全连接 DNN，其中第一和第二隐藏层分别有 120 和 80 个隐藏神经元，训练批大小为 128，内存大小为 1024，Adam 优化器的学习率为 0.01。

为了体现 DROO 算法的计算率的优劣判断，定义归一化计算速率 $\hat{Q}(\mathbf{h}, \mathbf{x}) \in [0, 1]$，计算公式如下：

$$\hat{Q}(\mathbf{h}, \mathbf{x}) = \frac{Q^*(\mathbf{h}, \mathbf{x})}{\max_{\mathbf{x}' \in \{0,1\}^N} Q^*(\mathbf{h}, \mathbf{x}')}, \tag{4}$$

复现原文代码实验结果如下：

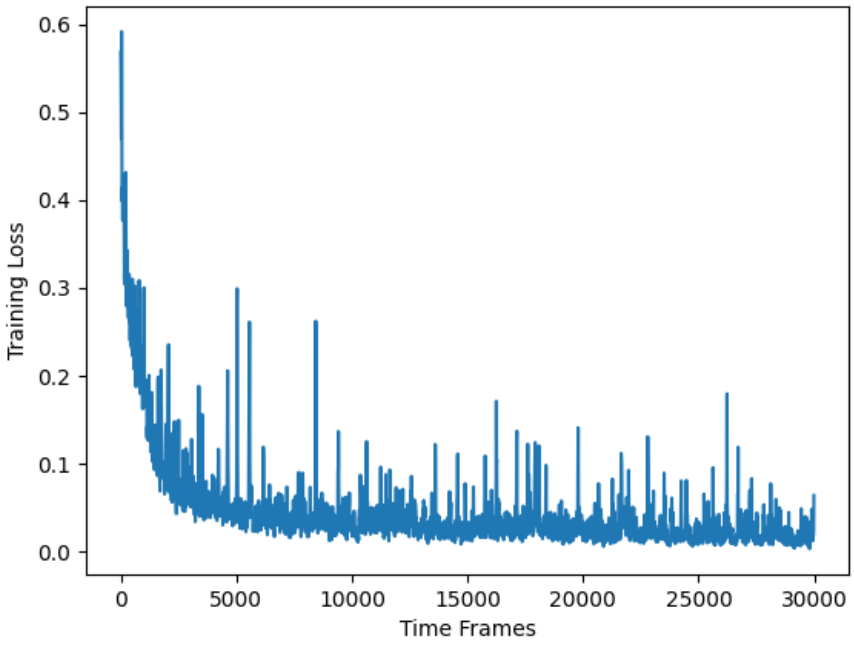

图 3. 复现实验衰落信道下 DROO 算法的训练损失

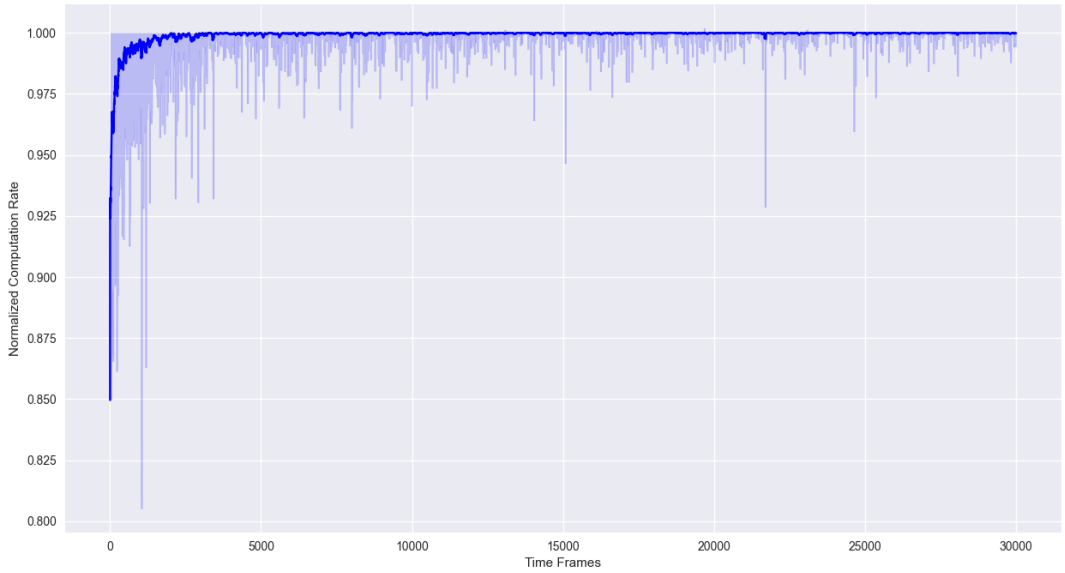

图 4. 复现实验衰落信道下 DROO 算法的归一化计算率

运行修改后的代码得出以下实验结果：

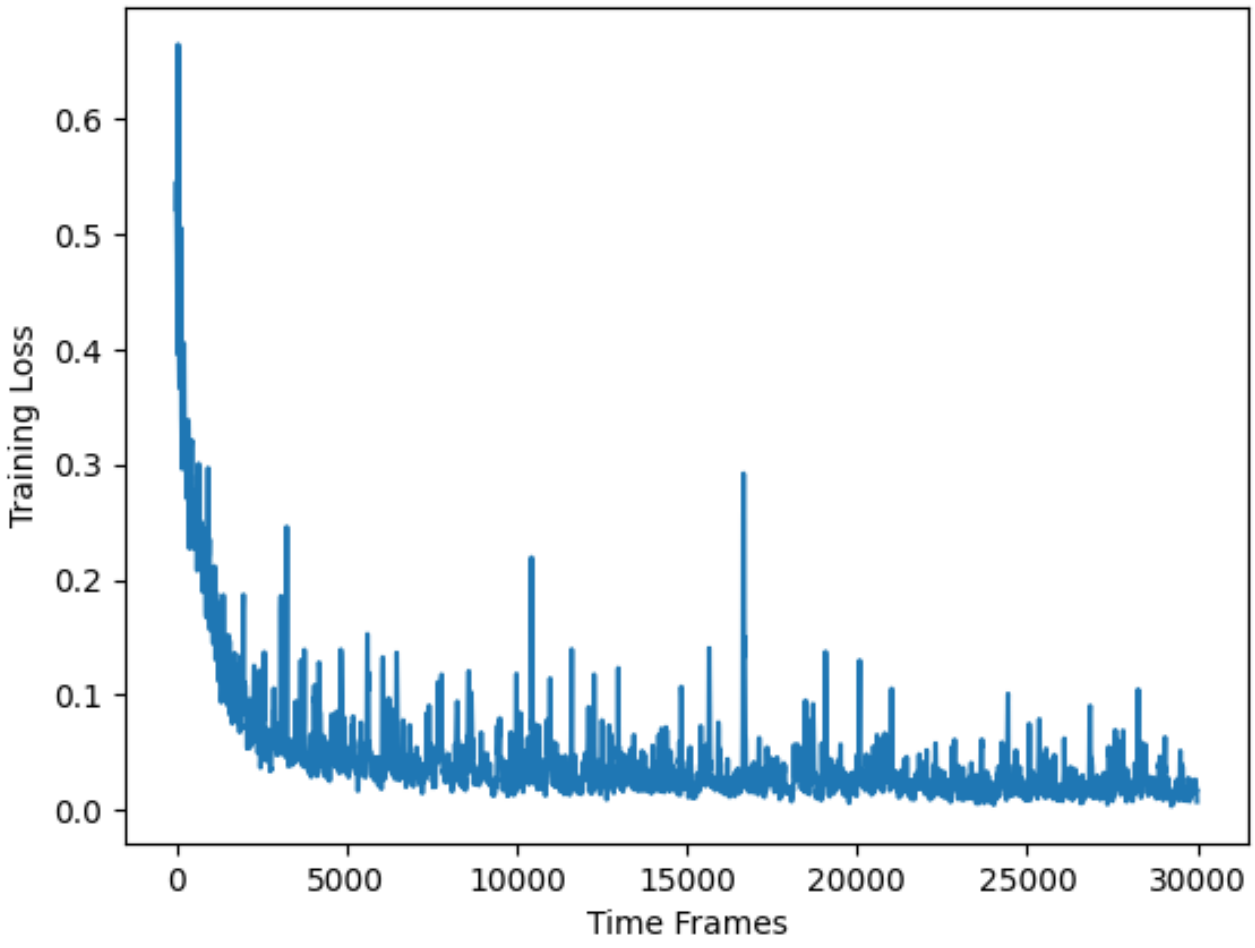

图 5. 修改参数后的训练损失

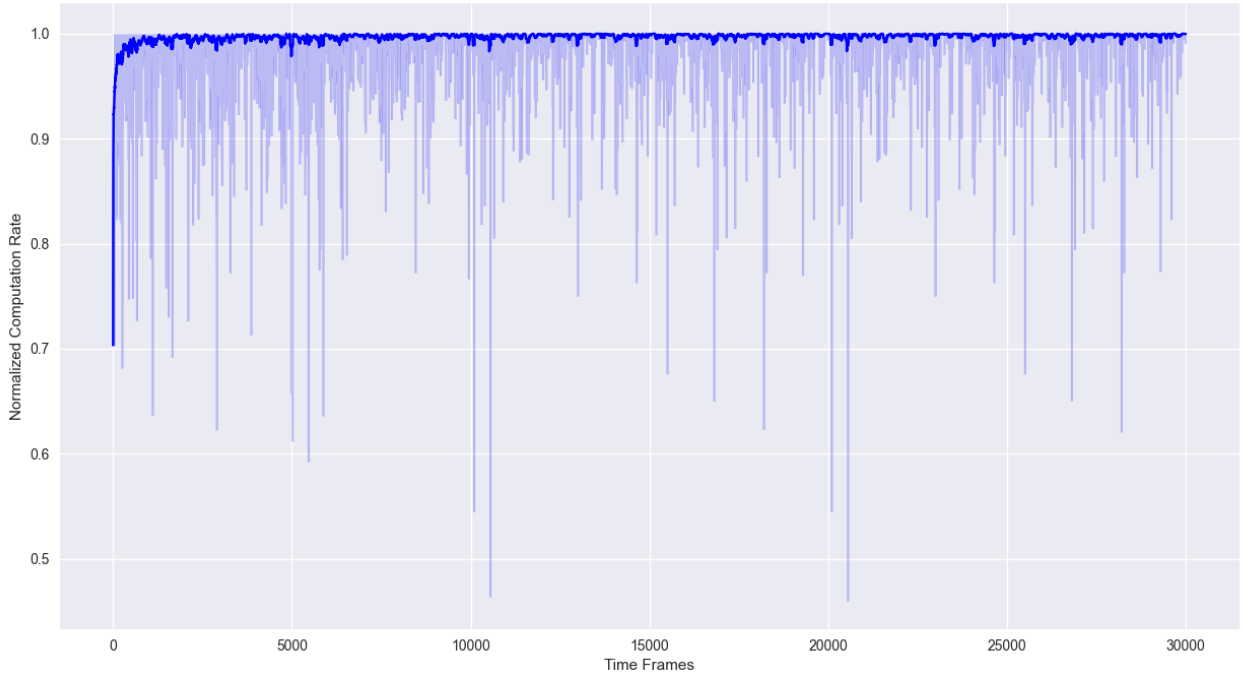

图 6. 修改参数后的归一化计算率

表 1. 实验结果对比

	Averaged normalized computation rate	Total time consumed	Average time per channel
复现	0.9997129952559707	263.85615158081055	0.008795205052693685
改进	0.997628976696836	176.17261457443237	0.005872420485814413

在图 3 和图 4 中，我们分别绘制了复现代码 DROO 算法的训练损失函数和归一化计算率 $\hat{Q}$ 关于时间帧的变化，在图 5 和图 6 绘制了修改参数后的 DROO 算法的训练损失函数和归一化计算率 $\hat{Q}$ 关于时间帧的变化，图 4 和图 6 中紫色曲线表示最近 50 帧的 $\hat{Q}$ 的移动平均值，浅紫色阴影表示最近 50 帧 $\hat{Q}$ 的最大值和最小值，在表 1 中我们进行了原实验代码和修改参数后的部分性能指标数值对比。通过观察实验结果，表格中的三个指标均呈现出数值降低的情况，并且归一化计算速率不稳定。我们可以发现，增加二进制决策阈值意味着算法在决定是否卸载任务时变得更加严格，只有当信道条件较好时才选择卸载，这可能导致更多的任务在本地计算，造成归一化计算速率的降低以及不稳定。减少 K 值的时间间隔算法会更频繁地调整考虑的卸载策略数量，从而减少总体运行时间，而每个信道的平均时间减少则可能是因为更有效的卸载决策减少了处理每个信道所需的时间。

6 总结与展望

在复现的论文中，作者提出了一种基于深度强化学习的在线卸载算法 DROO 从而最大限度地提高具有二进制计算卸载的无线供能移动边缘计算网络的加权和计算率。通过对复现论文的理解以及代码复现，DROO 实现了与现有基准测试方法相近的近乎最优性能，使实时系统优化真正能适用于衰落环境下的无线功能边缘计算网络。我在修改了部分关键参数之后，

得出了低于原论文的效果，并作出了结果分析，该实验在未来中可以探索 DROO 算法在异构网络环境中的表现，例如，当网络中包含不同类型的设备和多种通信技术时的网络环境。

参考文献

[1] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE transactions on wireless communications*, vol. 17, no. 3, pp. 1784–1797, 2017.

[2] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.

[3] S. Bi, C. K. Ho, and R. Zhang, "Wireless powered communication: Opportunities and challenges," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 117–125, 2015.

[4] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[5] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM transactions on networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[6] 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.

[7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[8] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2018.

[9] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE, 2016.

[10] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[11] D. Bertsekas, *Dynamic programming and optimal control: Volume I*, vol. 4. Athena scientific, 2012.