工业物联网中基于 Sarsa 算法的节能计算卸载方案

孙君，赵尚维康
（南京邮电大学江苏省无线通信重点实验室，江苏 南京 210003）

摘 要：针对任务有完成截止时间要求的工业物联网系统，为了降低该类系统的总能源损耗，提出一种基于 Sarsa 算法的节能计算卸载方案。基于 Sarsa 算法的特点对能耗优化问题进行了解耦，由 Sarsa 算法进行外部状态值函数的迭代，选择合适的边缘计算服务器进行计算任务卸载，再由粒子群优化算法解决其中的资源分配问题，最后更新环境信息进行下一轮算法迭代，直至算法满足结束条件。仿真结果表明，本方案具有更快的收敛速度并且有效降低了系统能耗。

关键词：工业物联网；边缘计算；Sarsa；节能计算卸载；资源分配
中图分类号：TN929.5
文献标志码：A
doi: 10.11959/j.issn.2096−3750.2022.00280

Energy-saving computation offloading scheme based on Sarsa algorithm in industrial internet of things

SUN Jun, ZHAO Shangweikang
Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

Abstract: In order to reduce the total energy consumption of industrial internet of things systems with deadline requirements, a computing offloading scheme based on Sarsa algorithm was proposed. Based on the characteristics of Sarsa algorithm, the energy consumption optimization problem was coupled. The Sarsa algorithm iterated the external state value function, selected the appropriate edge computing server to offload the computing task, and then the particle swarm optimization algorithm solved the resource allocation problem. Finally, the environmental information was updated for the next round of algorithm iteration until the algorithm met the end conditions. The simulation results show that this scheme has faster convergence speed and effectively reduces the system energy consumption.

Key words: industrial internet of things, edge computing, Sarsa, energy-saving computation offloading, resource allocation

0 引言

工业物联网（IIoT, industrial internet of things）连接了大量移动数字设备、制造机器、工业设备，这些设备不断产生大量数据和信号，用于传感、控制、系统维护和数据分析，其中涉及的通信和计算任务消耗大量的能量[1-2]。考虑 IIoT 设备上的资源有限，在智能电网场景以及生产制造业、采矿业这类恶劣的工业生产环境中，任务具有高可靠性和高能源效率要求，在这类 IIoT 系统中 IIoT 应用需要在电池上运行多年，节约能源消耗是延长这类 IIoT 系统寿命的关键因素[3-4]。根据 IIoT 的工作流程，在这类 IIoT 系统中，设备的计算任务有额定的完成截止时间要求，为了保证这类系统的高可靠性和高能源效率要求，需要考虑如何在高度保证任务在截止时间之内完成的前提下，最小化工业设备的能耗[5-6]。通

收稿日期：2021−12−14；**修回日期**：2022−06−15
通信作者：孙君，sunjun@njupt.edu.cn
基金项目：国家自然科学基金资助项目（No.61771255）；省部级重点实验室开放课题（No.20190904）
Foundation Items: The National Natural Science Foundation of China (No.61771255), The Open Project of Provincial and Ministerial Key Laboratories (No.20190904)

常做法是将一部分工业设备的任务卸载到具有足够计算资源的计算系统上执行[7-8]。因此，本文在 IIoT 系统中引入边缘计算，利用多接入边缘计算（MEC, multi-access edge computing）服务器强大的中央处理器（CPU, central processing unit）为密集的计算任务提供算力支持，由此减少系统的能耗。但边缘计算也有其自身的局限性[9-10]，例如，MEC 服务器具有有限的计算资源以及这些服务器之间存在不平衡负载[11-15]。针对这些问题，需要考虑合适的任务卸载决策和资源分配方案[16]。

现有的卸载决策和资源分配方案分为以下 3 类。第一类，采用常规凸优化算法[17]。如文献[18]将资源的最优分配问题转化为最小化能耗加权和的凸优化问题，利用拉格朗日算法迭代求取最优解，但此凸优化问题是 NP-hard 问题，通过拉格朗日算法很难求取最优解，因此现有这类工作求得的解一般为次优解[18-20]。第二类，采用进化学习算法。如文献[5]构建了一个最小化任务完成时间的混合整数非线性优化问题，通过迭代交替求解，采用基于模拟退火算法的启发式近似算法，得到资源分配最优解与任务卸载策略。文献[21]以最小化能耗为目标在多个设备之间构建博弈模型，利用博弈理论进行卸载决策分析。文献[22]提出了一种基于遗传算法的多站点协同卸载算法，将多站点协同卸载的问题建模为代价模型，并利用遗传算法寻找最小代价。但这类算法求得的解有可能不会收敛到全局最优解，且算法复杂度较高。第三类，采用强化学习（RL, reinforcement learning）算法[23]。如文献[23-25]定义各 MEC 服务器的 CPU 占用状态为状态空间，定义采取行为后所减少的能耗作为奖励值，采用 RL 中的 *Q*-learning 算法分析环境信息并通过状态值迭代的方式选择最合适的 MEC 服务器进行卸载，这一方案的问题在于只能为请求服务的设备分配指定的 MEC 进行卸载，而不能有效地分配 MEC 服务器的计算资源。若同时考虑计算资源的分配问题，需要把资源分配问题离散化处理并额外扩充系统的状态空间和行为空间[26-28]，这会导致极其庞大的算法复杂度并且不容易获得准确的最优分配方案[29]。

为解决以上问题，本文研究了 IIoT 中同时进行卸载决策和资源分配优化的高效方案。本文所提方案采用 RL 中的 Sarsa（state-action-reward-state-action）算法分析系统的环境信息，通过状态值函数的迭代更新选择合适的 MEC 服务器进行计算任务卸载，通过粒子群优化（PSO, particle swarm optimization）算法获得可使系统能耗降低的最优资源分配方案，之后，更新环境信息，进行下一轮 Sarsa 算法的迭代，直至 Sarsa 算法满足结束条件，以较低的算法复杂度同时实现卸载策略的制定和资源的分配方案，有效降低了系统能耗。本文的主要贡献如下：

1) 本方案针对 IIoT 系统的特点构建了系统决策模型，基于 Sarsa 算法保守的特点对能耗优化问题进行了解耦，与现有方案相比，以较低的算法复杂度，同时完成了卸载决策和资源分配优化；

2) 本方案采用 RL 算法和 PSO 算法针对原始目标问题直接求解，通过 RL 算法保证了优化目标函数在不降维的情况下获得最优解，并通过 Sarsa 算法的 ε – greedy 策略和 PSO 算法的 rand(0,1) 参数避免得到局部最优解。

1　系统模型与问题建模

本文针对任务存在截止时间要求的具有高可靠性和高能源效率要求的 IIoT 系统，为了有效降低此类系统的能耗，提出了一种基于边缘计算和 Sarsa 算法的节能计算卸载方案。将部分设备的任务卸载到 MEC 服务器以减少工业设备的能耗，着重解决其中的任务卸载决策和资源分配问题。

本方案场景为短程 IIoT 系统，该类系统使用 RFID 和无线传感器网络技术进行通信，工作环境的最大覆盖距离在 1 km 以内，可控制在一个小区的覆盖范围内[1]。系统模型如图 1 所示。

系统模型如下所述，定义 N 为该 IIoT 系统中的工业设备数，在该 IIoT 系统所处的小区基站部署一个代理 MEC 服务器用于优化该 IIoT 系统内卸载决策和资源分配，由于 IIoT 的用户设备具有低移动性的特点，本文假设该 IIoT 系统内的设备处于固定位置。在靠近工业设备的位置部署 M 个 MEC 服务器，尽量使得系统内 N 个工业设备与距它们最近的 MEC 服务器距离相同，即尽量将 MEC 服务器部署在工业设备的中心位置，M 可随着系统内任务流强度的大小适当变化，对于包含大量节点的 IIoT 场景，可以部署额外的 MEC 来缓解系统的压力。M 个 MEC 与位于小区基站的代理 MEC 服务器通过有线链路连接。除基站部署的代理 MEC 外，其他 MEC 服务器不参与系统卸载决策的计算，但会实时地向位于基站的代理 MEC 发送和更新本地的 CPU 占用状态信息和周围设备的任务卸载请求信息。

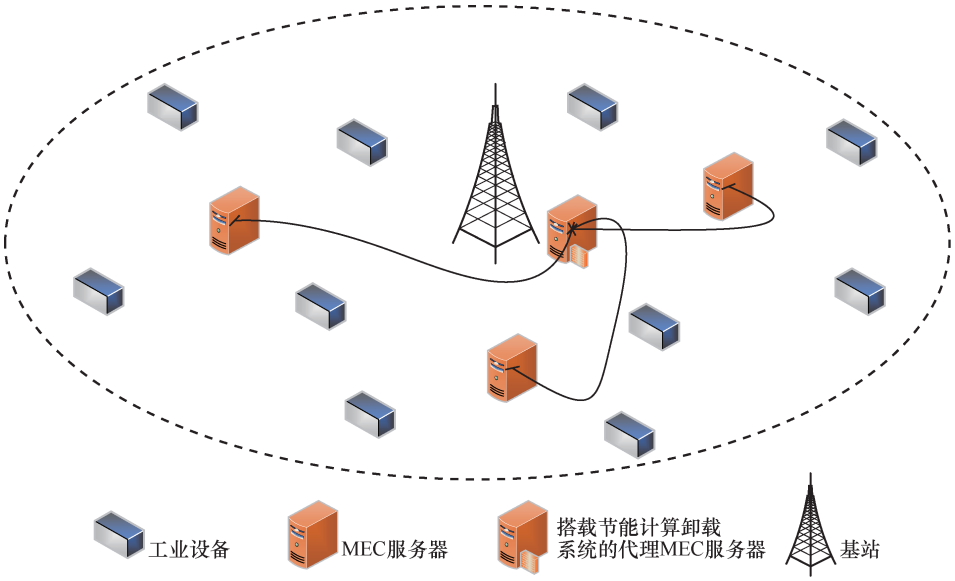

图 1 系统模型

系统可以通过分析通信环境信息确定数据的传输速率，并由此计算任务处理时间。

本方案为了分析通信环境信息，参考了文献[16, 25]构建了如下所述的通信模型。本方案定义 T_j 为设备 j 的计算任务，d_j 表示任务数据量的大小，c_j 表示完成计算任务所需的 CPU 周期总数，t_j^{dl} 表示计算任务完成的截止时间。定义 $S=\left\{s_1,s_2,\cdots,s_j,\cdots,s_N\right\}$ 表示系统的卸载策略，其中，$s_j\in[0,M],j\in[1,N]$ 表示第 j 个设备的卸载策略，若 $s_j=i,i\in[1,M]$，设备将本地任务卸载到第 i 个 MEC 服务器进行处理，若 $s_j=0$，设备则不进行计算任务的卸载，在本地执行计算任务。在选择执行卸载时，无论选择卸载到哪一个 MEC 服务器，都会首先将计算任务发送到距离此设备最近的 MEC 服务器，然后通过 MEC 服务器之间的有线链路转发送达目标 MEC 服务器。设备发送计算任务至 MEC 服务器，需要通过相对应基站的子信道进行数据传输，因此，可以计算得到设备通过子信道传输数据到 MEC 服务器的上行传输速率为

$$R_{j,i}=B_j\,\mathrm{lb}\left(1+\frac{p_jh_j}{\sigma^2}\right) \tag{1}$$

其中，B_j 表示设备 j 分配到的子信道带宽，p_j 表示设备 j 的传输功率，σ^2 表示高斯白噪声功率[29]。h_j 表示设备 j 与边缘接入点之间的信道增益，可表示为

$$h_j=D_j^{-v}H^2 \tag{2}$$

其中，D_j 表示设备与其位置最近的 MEC 服务器之间的空间距离，v 表示路径损耗因子，H 表示信道衰落系数，是一个以瑞利分布为模型的随机变量[25]。

通过数据的传输速率，可以计算任务的处理时间，任务处理时间主要由两部分构成：计算时延和传输时延。所选定的卸载策略需要首先保证任务处理时间满足任务的截止时间要求。

下面基于本地处理与任务卸载两种情况分析任务的处理时间。

1) 本地处理

当 $s_j=0$ 时，设备选择本地处理，此时只存在计算时延，传输时延为 0，计算时延为

$$t_j^{\mathrm{l}}=\frac{c_j}{f_j^{l}} \tag{3}$$

其中，f_j^{l} 为设备 j 的本地计算能力（即每秒的 CPU 周期数）。

2) 任务卸载

当 $s_j=i$ 时，设备选择卸载处理，此时的计算时延为

$$t_{j,i}^{\mathrm{s}}=\frac{c_j}{f_{j,i}^{\mathrm{c}}} \tag{4}$$

其中，$f_{j,i}^{\mathrm{c}}$ 为第 i 个 MEC 分配给设备 j 的计算能力。

卸载的传输时延由两部分构成，首先需要将计算任务通过无线链路传输到相对应的边缘节点，然后由边缘节点通过有线链路将计算任务转发到目标 MEC 服务器，即传输时延为

$$t_{j,i}^{\mathrm{e}}=t_{j}^{\mathrm{wl}}+t_{j,i}^{\mathrm{w}}=\frac{d_j}{R_{j,i}}+\frac{D_{j,i}}{r} \tag{5}$$

其中，$D_{j,i}$为设备 j 对应的 MEC 服务器距离目标 MEC 服务器的空间距离，r 表示有线链路的信息传输速率，由于有线链路信息传输速率远远大于无线链路信息传输速率，式(5)中的有线链路传输时间 $t_{j,i}^{\mathrm{w}}$ 通常可以忽略不计。因此，卸载的处理时延为

$$t_{j,i}^{\mathrm{c}}=t_{j,i}^{\mathrm{s}}+t_{j,i}^{\mathrm{e}}=\frac{c_j}{f_{j,i}^{\mathrm{c}}}+\frac{d_j}{R_{j,i}} \tag{6}$$

本方案的目标是在保证任务处理时间满足截止时间要求的前提下减少工业设备的总能耗。系统内工业设备的总能耗由本地处理能耗和卸载能耗两部分构成。当设备 j 选择在本地处理计算任务时，设备会产生本地处理能耗即

$$E_j^{\mathrm{l}}=z_0 c_j \tag{7}$$

其中，z_0 为 CPU 每轮处理过程消耗的能量[27, 29]，且

$$z_0=10^{-27}\left(f_j^{\mathrm{l}}\right)^2 \tag{8}$$

当设备 j 选择将计算任务卸载到第 i 个 MEC 时，设备会产生卸载能耗，卸载能耗由传输能耗和待机能耗两部分组成。卸载时，设备首先将本地任务传输到对应的 MEC 服务器，这会产生传输能耗，之后，设备会保持待机状态等待回传结果，这会产生待机能耗。该卸载能耗可以表示为

$$E_{j,i}^{\mathrm{c}}=P_j\frac{d_j}{R_{j,i}}+P_s\frac{c_j}{f_{j,i}^{c}} \tag{9}$$

其中，P_j 为设备传输计算任务时的发射功率，P_s 为设备等待回传结果时的待机功率。

因此，本方案的优化目标可以表示为

$$\begin{aligned}&\min_{\left\{S,f_{j,i}^{\mathrm{c}}\right\}}\sum_{i=1}^{I}\sum_{j=1}^{J}w_j\left\{\alpha E_j^{\mathrm{l}}+\beta E_{j,i}^{\mathrm{c}}\right\}\\&\mathrm{C}_1:f_{j,i}^{\mathrm{c}}>f_j^{\mathrm{l}}\\&\mathrm{C}_2:0<\sum_{j=1}^{J}f_{j,i}^{\mathrm{c}}\leqslant f_{i,\max}^{\mathrm{c}}\\&\mathrm{C}_3:\sum_{i=1}^{I}\alpha t_j^{\mathrm{l}}+\beta t_{j,i}^{\mathrm{c}}<t_j^{\mathrm{dl}}\end{aligned} \tag{10}$$

其中，待优化变量为卸载决策变量 S 与资源分配变量 $f_{j,i}^{\mathrm{c}}$，w_j 表示设备 j 的权重，权重的高低视设备的重要程度而定，α 和 β 用于表示卸载决策

$$\alpha=\begin{cases}0, & s_j\neq 0\\1, & s_j=0\end{cases} \tag{11}$$

$$\beta=\begin{cases}0, & s_j\neq i\\1, & s_j=i\end{cases} \tag{12}$$

式(10)中，$\mathrm{C}_1\sim\mathrm{C}_3$ 为约束，C_1 用于保证设备从 MEC 服务器分配到的计算能力大于本地计算能力，否则卸载过程没有意义；C_2 用于保证 MEC 服务器分配出去的计算能力之和不大于自身拥有的计算能力；C_3 用于保证所选卸载决策和资源分配方案可以满足该任务的截止时间要求。

2　问题求解

2.1　基于 Sarsa 的卸载决策算法

这是一个 NP-hard 问题，问题中耦合着值为整数集合的卸载策略变量 S 和值为浮点数的资源分配变量 $f_{j,i}^{\mathrm{c}}$。通常情况下，该优化目标方程无法求解，为了求得原始目标问题的最优解，本方案考虑引入 RL 算法，通过分析环境信息来获得最优序列决策。

然而 RL 算法通常具有有限的状态空间和行为空间，这意味着通过 RL 算法一般只能解决解为整数类型的优化问题。如果使用 RL 算法解决解为浮点数类型的优化问题，需要对浮点数类型的解进行离散化，即用一个足够小的单位变量 κ 对解进行离散化[28]，这一过程会极大地扩充系统的状态空间和行为空间，会带来庞大的系统复杂度和训练决策时间，在 IIoT 的工作环境中一般不支持离散化带来的额外的时延和计算资源的负担。

本方案通过 Sarsa 算法来解决这一问题，对原始优化问题进行了拆分，将其分解为卸载决策和资源分配两部分。基于 Sarsa 算法的计算卸载方案流程如图 2 所示。

本方案中由 Sarsa 算法进行外部值函数的迭代，智能体（Agent）观测环境信息获得相应的回报，智能体根据回报更新状态值表并通过分析状态值表得出卸载决策，经过反复执行这一流程，系统不断优化卸载决策，使得优化目标即系统总能耗朝着最小化的方向收敛，Sarsa 迭代内部的基于 PSO 算法的资源分配模型通过优化目标函数不断评估粒子并更新优化参数，最终通过粒子群的迭代更新得到优化的资源分配决策。

之所以选择 Sarsa 算法来解决其中的卸载决策

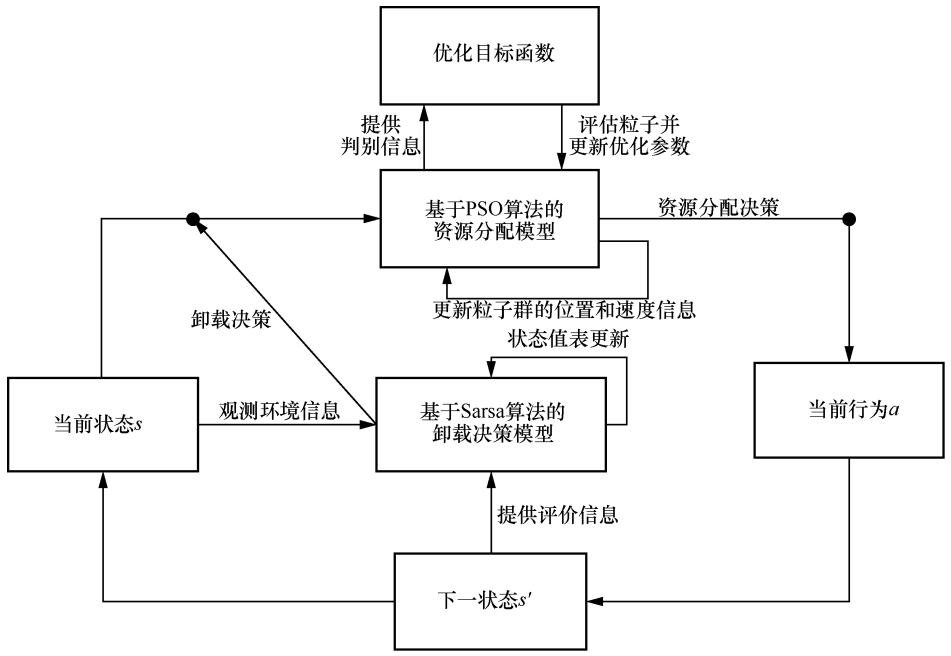

图 2 基于 Sarsa 算法的计算卸载方案流程

问题，是由于 Sarsa 算法具有保守的特性[29]。与其他 RL 算法相同，在 Sarsa 算法中，会训练一个智能体，智能体会通过与环境交互学习信息，寻找每个状态的最佳行动[30-31]。但 Sarsa 算法与其他 RL 算法的迭代更新过程不同，Sarsa 算法不是一个贪婪的 RL 算法，Sarsa 算法中的智能体在采取决策时会优先采取其观测的行为而不一定是 Q 值最大的行为，这使得决策会逐渐趋向于最优决策而不是直接得出最优决策，最后，Sarsa 算法会选取一种较为安全的策略[32-34]。因此，本方案采用 Sarsa 算法对原优化问题加以解耦，将原优化问题分解为卸载决策问题和资源分配问题两个部分。

Sarsa 算法中，必须明确 3 个重要要素，分别是优化问题中的环境状态、Agent 的可执行行为和环境对行为的评价奖励值[35]。结合 IIoT 系统任务卸载决策问题的特点，将设备卸载决策的集合设置为 Agent 的学习状态，即 $S=\{s_1,s_2,\cdots,s_j,\cdots,s_N\}$，$j\in[1,N]$。将 Agent 可执行的行为空间设置为 $A=\{a_1,a_2,\cdots,a_j,\cdots,a_N\},j\in[1,N]$，其中，$a_j=0$ 表示设备 j 的计算任务将在本地执行，$a_j=i$ 则表示设备 j 将计算任务卸载到第 i 个 MEC 服务器处理。系统通过检测行为所产生的能源损耗确定每一步行为的奖励值，用于评判 Agent 本次采取的行为，进而指导 Agent 的学习过程。据此，设置奖励函数为

$$r=\frac{E(S)-E(S')}{E(S)+E(S')} \tag{13}$$

其中，$E(S)$ 表示当前卸载决策所产生的能源消耗，$E(S')$ 则表示采取行为后下一状态所产生的能源消耗[23]。基于贝尔曼方程，通过奖励值和行为更新 Q 值表，即

$$Q(S,A)\leftarrow Q(S,A)+\alpha\left[r+\gamma Q(S',A')-Q(S,A)\right] \tag{14}$$

其中，$\gamma\in(0,1)$ 为折扣率，用于表示未来效益对当前状态值影响的衰减系数，用来限定未来状态对当前状态的影响比例。其中，$\alpha=1-\mathrm{e}^{-I}$ 为学习率，用于表示 Agent 在每一次行为中学习的比率，I 为迭代次数，随着迭代次数的增加，学习率逐步提升，可见 $\alpha\in(0,1)$。

为了避免系统陷入局部最优解，在每一轮 Sarsa 算法迭代过程中，本方案基于模拟退火算法的思想，引入了 $\varepsilon-$greedy 策略，在每一次根据回报值采取行为时，有一定概率跳出决策系统指定的决策来防止陷入局部最优解。基于 Sarsa 算法的卸载决策算法见算法 1。

算法 1 基于 Sarsa 算法的卸载决策算法

输入：设备信息 $\left(f_j^{\mathrm{l}},p_j,h_j\right)$、设备的计算任务信息 $T_j\left(d_j,c_j,t_j^{\mathrm{dl}}\right)$、MEC 的计算资源占用信息

输出：优化卸载决策 S、状态值表 $Q(S,A)$、资源分配策略 $f_{j,i}^{\mathrm{c}}$

初始化：选择一随机网络状态作为待优化的卸

载决策 S，状态值表置 0，折扣率 $\gamma=0.75$、学习率 $\alpha=1-\mathrm{e}^{-1}$、$\varepsilon=0.9$

while episode $\leqslant$ 10 000

产生 0～1 中的随机数 x

更新学习率 $\alpha=1-\mathrm{e}^{-I}$

if $x>\varepsilon$

从 $\{a_1,a_2,\cdots,a_j,\cdots,a_N\}, j\in[1,N]$ 中随机选取卸载行为 a_j

if $x\leqslant\varepsilon$

选取该状态下 Q 值最大的行为 a_j 作为卸载行为，若有 Q 值相同的多个行为，则从中随机选取一个卸载行为

end if

通过基于粒子群算法的资源分配算法确定最优资源分配策略 $f_{j,i}^{\mathrm{c}}$

根据当前环境得到行为 a_j 的奖励值

$$r=\frac{E(S)-E(S')}{E(S)+E(S')}$$

更新 Q 值

$$Q(S,A)\leftarrow Q(S,A)+\alpha\left[r+\gamma Q(S',A')-Q(S,A)\right]$$

end while

2.2　基于粒子群优化算法的资源分配算法

Sarsa 算法中包含着优化问题中的另一部分——资源分配问题。基于 Sarsa 算法保守的特点，在决策过程的每一轮迭代中，卸载决策 S 会向优化目标的方向逐步靠近，但相对保守，这使得系统有机会在决策过程中同时确定最优的资源分配策略，并且资源分配问题的求解过程不会受到决策过程的约束。

由于在每一轮的决策过程中，卸载决策 S 已通过 Sarsa 算法以 $\varepsilon-$greedy 方式确定，此时的资源分配问题变为

$$\begin{aligned}&\min_{\{f_{j,i}^{\mathrm{c}}\}}\sum_{i=1}^{I}\sum_{j=1}^{J}w_j\left\{\alpha E_j^{\mathrm{l}}+\beta E_{j,i}^{\mathrm{c}}\right\}=\\&\sum_{i=1}^{I}\sum_{j=1}^{J}w_j\left\{\alpha z_0c_j+\beta P_j\frac{d_j}{R_{j,i}}+\beta P_s\frac{c_j}{f_{j,i}^{\mathrm{c}}}\right\}\\&\mathrm{C}_1: f_{j,i}^{\mathrm{c}}>f_j^{\mathrm{l}}\\&\mathrm{C}_2: 0<\sum_{j=1}^{J}f_{j,i}^{\mathrm{c}}\leqslant f_{i,\max}^{\mathrm{c}}\\&\mathrm{C}_3: \sum_{i=1}^{I}\alpha t_j^{\mathrm{l}}+\beta t_{j,i}^{\mathrm{c}}<t_j^{\mathrm{dl}}\end{aligned}\tag{15}$$

此时，优化问题中只剩下 $f_{j,i}^{\mathrm{c}}$ 变量，这是一个有约束非凸的最优化问题，这类问题使用拉格朗日方程一般不易求解，因此本方案使用 PSO 算法来解决这一问题。

PSO 算法初始化一个随机粒子群，然后通过迭代的方式找到最优解，在每一次的迭代中，粒子通过式(16)、式(17)跟踪个体极值 pbest_i 和整个粒子群的全局最优解 gbest 来更新自己的速度 v_i 和位置 x_i [36]。

$$x_i=x_i+v_i\tag{16}$$

$$\begin{aligned}v_i=&\omega\times v_i+\\&c_1\times\mathrm{rand}(0,1)\times(\mathrm{pbest}_i-x_i)+\\&c_2\times\mathrm{rand}(0,1)\times(\mathrm{gbest}-x_i)\end{aligned}\tag{17}$$

其中，rand(0,1) 是 (0,1) 的随机数；c_1 和 c_2 是学习因子，通常 $c_1=c_2=2$；ω 是惯性因子，其值非负，该值会影响算法的全局寻优能力和局部寻优能力，其值较大，全局寻优能力强，其值较小，局部寻优能力强。

为了获得更好的寻优能力，加快决策系统的迭代速度，本方案采用动态的线性递减权值，即

$$\omega=\frac{(\omega_{\mathrm{ini}}-\omega_{\mathrm{end}})(G_k-g)}{G_k}+\omega_{\mathrm{end}}\tag{18}$$

其中，G_k 为最大迭代次数，g 为当前迭代次数，$\omega_{\mathrm{ini}}=0.9$ 为初始惯性权值，$\omega_{\mathrm{end}}=0.4$ 为迭代至最大进化代数时的惯性权值。

基于 PSO 算法的资源分配算法见算法 2。

算法 2　于 PSO 算法的资源分配算法

输入：设备信息 $(f_j^{\mathrm{l}},p_j,h_j)$、设备的计算任务信息 $T_j(d_j,c_j,t_j^{\mathrm{dl}})$、卸载决策 S、MEC 的计算资源信息 $f_{i,\max}^{\mathrm{c}}$

输出：资源分配策略 $f_{j,i}^{\mathrm{c}}$

初始化：数量为 N 的随机粒子，每个粒子具有随机的速度和位置，迭代结束阈值 $\varepsilon=0.1$，最大迭代次数 $G_k=100$，学习因子 $c_1=c_2=2$

while episode $\leqslant G_k$ and $|\mathrm{gbest}'-\mathrm{gbest}|>\varepsilon$ do

更新惯性权值

$$\omega=\frac{(\omega_{\mathrm{ini}}-\omega_{\mathrm{end}})(G_k-g)}{G_k}+\omega_{\mathrm{end}}$$

更新全局最优解

$\mathrm{gbest}=\min\{\mathrm{pbest}_i\}$

for i=1 to N

更新粒子 i 的位置 x_i 和速度 v_i

评估粒子，更新局部最优解 pbest_i 和全局最优解 gbest

if $E(x_i)<E(\mathrm{pbest}_i)$

$$\text{pbest}_i = x_i$$
$$\text{if}\ \ E(\text{pbest}_i) < E(\text{gbest})$$
$$\text{gbest} = \text{pbest}_i$$
$$\text{end for}$$
$$\text{episode} = \text{episode} + 1$$

end while

通过算法 1 和算法 2，系统可以得到优化的卸载决策和资源分配策略。

3 仿真分析

本文基于 Python 平台对所提的基于 Sarsa 的协同卸载算法进行了仿真实验，在仿真过程中使用了 Python 中的 math、random 等数学工具库以及 numpy、pandas、matplotlib 等数据分析工具库。仿真实验中，该工业物联网系统覆盖半径为 300 m，小区内随机分布着 N 个工业设备，工业设备的计算任务信息为服从均匀分布的随机变量，部署 M 个 MEC 服务器在工业设备的中心位置。仿真实验将本文所提算法与常规凸优化算法[16]、模拟退火算法[5]、Q-learning[23]算法进行了比较。仿真对比了这些卸载决策算法下的系统能耗。仿真相关参数依据文献[26-28]设定，覆盖了常用的 IIoT 应用场景，仿真主要参数见表 1。

表 1 仿真主要参数

参数项	参数值
工业设备数 N	N=20
小区内部署的 MEC 服务器数量 M	M=5
工业设备的计算能力 f_j^{l}	$f_j^{\mathrm{l}} \sim U(0.5,1.5)\times10^{10}$ cycle/s
MEC 服务器的计算能力 $f_{i,\max}^{\mathrm{c}}$	$f_{i,\max}^{\mathrm{c}} = 5\times10^{10}$ cycle/s
任务数据量的大小 d_j	$d_j \sim U(1,10)$ Mbit
计算任务被完成所需的计算周期数 c_j	$c_j \sim U(500,1500)$ cycle/bit
计算任务完成的截止时间 t_j^{dl}	$t_j^{\mathrm{dl}} \sim U(10,50)$ ms
带宽 B	B=10 MHz
设备的传输功率 p_j	$p_j = 500$ mW
设备等待回传结果时的待机功率 p_s	$p_s = 100$ mW
路径损耗因子 v	V=4
高斯白噪声功率 σ^2	$\sigma^2 = 2\times10^{-13}$ W

部署不同 MEC 服务器数量下系统的总能耗如图 3 所示。通过横向比较可以发现，随着部署 MEC 服务器数量的增多，系统总能耗会逐渐降低，但降低趋势会逐步放缓。这一方面是由于系统内计算资源增多，设备更倾向于将自身的计算任务卸载到 MEC 服务器进行处理，从而降低了能耗。另一方面，随着部署 MEC 服务器数量的增多，小区内设备与 MEC 服务器的距离也会降低，这使得设备卸载计算任务的能量消耗降低。而当系统内的计算资源可以满足当前的任务强度时，计算资源的增加不会带来额外的收益，此外，部署 MEC 服务器数量的增多也不会使设备与 MEC 服务器的距离无限接近，因此，能耗的降低趋势会逐步放缓。因此，实际场景中，当已知系统环境时，可以通过仿真实验确定部署的 MEC 服务器数量。将 4 种算法纵向对比可以发现，MEC 服务器数量为 8 个时，本文所提算法的系统总能耗相较于常规凸优化算法降低了 2.5%、相较于模拟退火算法降低了 1.5%、相较于 Q-learning 算法降低了 1%。

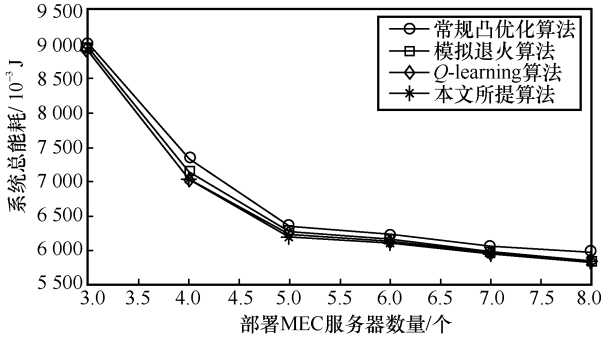

图 3 不同 MEC 服务器数量下的系统总能耗

不同迭代次数下的系统能耗如图 4 所示，展示了小区内存在 20 个设备、部署 5 个 MEC 服务器时不同迭代次数下的系统能耗，其他相关参数保持不变。RL 算法的复杂度与很多因素有关，主要与决策系统迭代次数相关，为此本文分析了不同决策算法的迭代速度。通过横向比较可以发现，4 种决策算法随着迭代次数的增多，系统总能耗都会逐渐降低并收敛。将 4 种算法纵向对比可以发现，本文所提算法具有最快的收敛速度，在迭代次数为 3 000 时就已经收敛。此外在决策系统收敛后，本文所提算法的系统总能耗低于其他 3 种算法。

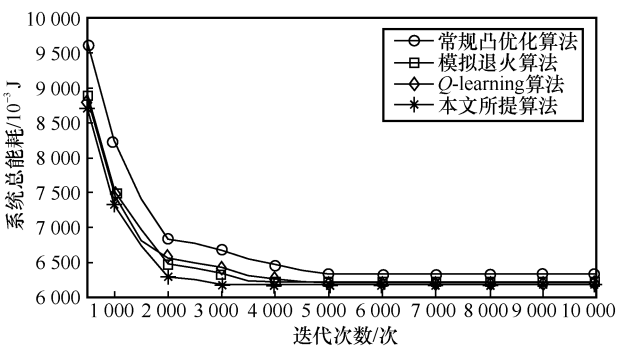

图 4 迭代次数下的系统能耗

不同任务数据量大小下使用不同卸载决策算法的系统能耗如图 5 所示，其他相关参数保持不变。通过横向比较可以发现，随着任务数据量的增多，系统总能耗逐渐增大，并且能耗增加的趋势逐步增大，这是由于随着系统内任务强度的增加，MEC 服务器的计算能力逐渐不能满足设备的卸载需求，因此实际场景中，应综合评估系统的平均任务强度，部署拥有合适计算能力的 MEC 服务器。将 4 种算法纵向对比可以发现，本文所提算法的系统总能耗始终低于其他 3 种算法。任务数据量为 30 MB 时，本文所提算法的系统总能耗相较于常规凸优化算法降低了 16%、相较于模拟退火算法降低了 1.5%、相较于 *Q*-learning 算法降低了 1%。

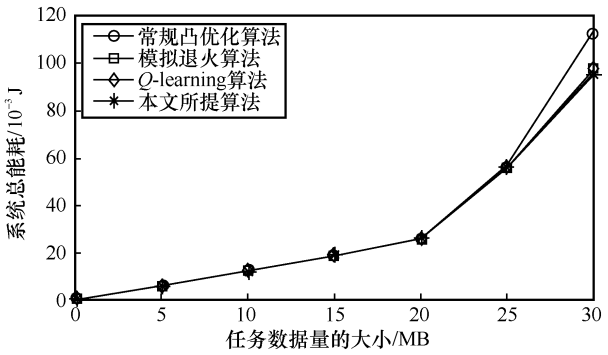

图 5　不同任务数据量大小下使用不同卸载决策算法的系统能耗

4　结束语

针对任务存在截止时间要求的具有高可靠性和高能源效率要求的 IIoT 系统，为了有效降低此类系统的能耗，本文在 IIoT 系统中引入边缘计算技术并采用 RL 中的 Sarsa 算法对能耗优化问题进行了拆分，将其分解为卸载决策和资源分配两部分进行求解，由 Sarsa 算法进行外部值函数的迭代，再由 PSO 算法解决其中涉及浮点数类型解的资源分配问题，之后更新环境信息，进行下一轮 Sarsa 算法的迭代，直至 Sarsa 算法满足迭代截止条件，以较低的算法复杂度实现了任务卸载决策和计算资源的高效分配，有效降低了系统能耗。

本文的研究工作还存在部分需要改进的地方：

1) 在本研究中，在考虑资源分配方案时只考虑了计算节点的选择和计算资源的分配，对于复杂的大型 IIoT 系统，在未来工作中可以进一步考虑设备发射功率的分配；

2) 在本研究中，为简化模型，在边缘计算服务器部署时，部署于工业设备的中心位置，从实际使用可行性和不同设备差异性角度分析，可能拥有更好的部署 MEC 服务器的方式，部署算法存在可优化的空间。

参考文献：

[1] MAO W L, ZHAO Z W, CHANG Z, et al. Energy-efficient industrial Internet of things: overview and open issues[J]. IEEE Transactions on Industrial Informatics, 2021, 17(11): 7225-7237.

[2] 姚鑫. 软件定义工业物联网计算卸载技术研究[D]. 成都: 电子科技大学, 2019.
YAOX. A research on software-defined industrial Internet of things computing offloading technology[D]. Chengdu: University of Electronic Science and Technology of China, 2019.

[3] CHEN B T, WAN J F, SHU L, et al. Smart factory of industry 4.0: key technologies, application case, and challenges[J]. IEEE Access, 2018, 6: 6505-6519.

[4] YOUNAN M N, HOUSSEIN E H, ELHOSENY M, et al. Challenges and recommended technologies for the industrial Internet of things: a comprehensive review[J]. Measurement, 2020, 151: 107198.

[5] 刘斐, 曹钰杰, 章国安. 车联网场景下移动边缘计算协作式资源分配策略[J]. 电讯技术, 2021, 61(7): 858-864.
LIU F, CAO Y J, ZHANG G A. Collaborative resource allocation strategy for mobile edge computing in vehicular networks[J]. Telecommunication Engineering, 2021, 61(7): 858-864.

[6] 代振楠. 工业互联网中基于边缘计算的任务卸载策略研究[D].长沙: 湖南大学, 2019.
DAI Z N. Research on task offloading strategy based on edge computing for industrial Internet[D]. Changsha: Hunan University, 2019.

[7] DUAN X T, XU F,SUNYY. Research on offloading strategy in edge computing of Internet of things[C]//Proceedings of 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA).Piscataway: IEEE Press, 2020: 206-210.

[8] HOU X W, REN Z Y, YANG K, et al.IIoT-MEC: a novel mobile edge computing framework for 5G-enabled IIoT[C]//Proceedings of 2019 IEEE Wireless Communications and Networking Conference. Piscataway: IEEE Press, 2019: 1-7.

[9] MENG J Y, TAN H S, XU C, et al. Dedas: online task dispatching and scheduling with bandwidth constraint in edge computing[C]//Proceedings of IEEE INFOCOM 2019 - IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2019: 2287-2295.

[10] LI M, YU F R, SI P B, et al. Energy-efficient machine-to-machine (M2M) communications in virtualized cellular networks with mobile edge computing (MEC)[J]. IEEE Transactions on Mobile Computing, 2019, 18(7): 1541-1555.

[11] JIANG C F, CHENG X L, GAO H H, et al. Toward computation offloading in edge computing: asurvey[J]. IEEE Access, 7: 131543-131558.

[12] HE X, DOU W C. Offloading deadline-aware task in edge computing[C]//Proceedings of 2020 IEEE 13th International Conference on Cloud Computing. Piscataway: IEEE Press, 2020: 28-30.

[13] 黄永明, 郑冲, 张征明, 等. 大规模无线通信网络移动边缘计算和缓存研究[J]. 通信学报, 2021, 42(4): 44-61.
HUANG Y M, ZHENG C, ZHANG Z M, et al. Research on mobile edge computing and caching in massive wireless communication net-

work[J]. Journal on Communications, 2021, 42(4): 44-61.

[14] LI Z, ZHOU X, QIN Y F. A survey of mobile edge computing in the industrial Internet[C]//Proceedings of 2019 7th International Conference on Information, Communication and Networks (ICICN). Piscataway: IEEE Press, 2019: 94-98.

[15] 安宜豪. 移动边缘计算中面向低功耗的任务迁移问题研究[D]. 哈尔滨: 哈尔滨工业大学, 2019.
AN Y H. Research on task offloading for low-power in MEC[D]. Harbin: Harbin Institute of Technology, 2019.

[16] LYU X C, TIAN H, SENGUL C, et al. Multiuser joint task offloading and resource optimization in proximate clouds[J]. IEEE Transactions on Vehicular Technology, 2017, 66(4): 3435-3447.

[17] WANG C M, LIANG CC, YU F R, et al. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing[J]. IEEE Transactions on Wireless Communications, 2017, 16(8): 4924-4938.

[18] BOYD S. Distributed optimization and statistical learning via the alternating direction method of multipliers[J]. Foundations and Trends® in Machine Learning, 2010, 3(1): 1-122.

[19] YOU C S, HUANG K B, CHAE H, et al. Energy-efficient resource allocation for mobile-edge computation offloading[J]. IEEE Transactions on Wireless Communications, 2017, 16(3): 1397-1411.

[20] EL HABER E, NGUYEN T M, ASSI C, et al. An energy-efficient task offloading solution for MEC-based IoT in ultra-dense networks[C]//Proceedings of 2019 IEEE Wireless Communications and Networking Conference. Piscataway: IEEE Press, 2019: 1-7.

[21] LI Q P, ZHAO J H, GONG Y, et al. Energy-efficient computation offloading and resource allocation in fog computing for Internet of Everything[J]. China Communications, 2019, 16(3): 32-41.

[22] 季子豪, 江凌云. 一种基于遗传算法的多站点协同计算卸载算法[J]. 计算机工程与科学, 2021, 43(3): 426-434.
JI Z H, JIANG L Y. A genetic-based multi-site collaborative computation offloading algorithm[J]. Computer Engineering &Science, 2021, 43(3): 426-434.

[23] KE H C, WANG J, WANG H, et al. Joint optimization of data offloading and resource allocation with renewable energy aware for IoT devices: a deep reinforcement learning approach[J]. IEEE Access, 7: 179349-179363.

[24] 张延华, 杨乐, 李萌, 等. 基于 Q-learning 的工业互联网资源优化调度[J]. 北京工业大学学报, 2020, 46(11): 1213-1221.
ZHANG Y H, YANG L, LI M, et al. Optimization of resource allocation for industrial Internet based on Q-learning[J]. Journal of Beijing University of Technology, 2020, 46(11): 1213-1221.

[25] 石雪琴. MEC 网络中联合优化计算卸载算法研究[D]. 重庆: 重庆邮电大学, 2020.
SHI X Q. Research on joint optimization of computation offloading in MEC network[D]. Chongqing: Chongqing University of Posts and Telecommunications, 2020.

[26] 程百川. 基于深度强化学习的 MEC 计算卸载和资源分配研究[D]. 北京: 北京邮电大学, 2019.
CHENG B C. Research on MEC computing offloading and resource allocation based on deep reinforcement learning[D]. Beijing: Beijing University of Posts and Telecommunications, 2019.

[27] 李季. 基于深度强化学习的移动边缘计算中的计算卸载与资源分配算法研究与实现[D]. 北京: 北京邮电大学, 2019.
LI J. Research and implementation of computation offloading and resource allocation algorithm in mobile edge computing based on deep reinforcement learning[D]. Beijing: Beijing University of Posts and Telecommunications, 2019.

[28] 李超. 基于深度强化学习的移动边缘计算卸载策略及其物理层安全研究[D]. 广州: 广州大学, 2020.
LI C. Research on mobile edge unloading strategy based on deep reinforcement learning and its physical layer security[D]. Guangzhou: Guangzhou University, 2020.

[29] KHALIL R A, SAEED N, MASOOD M, et al. Deep learning in the industrial Internet of Things: potentials, challenges, and emerging applications[J]. IEEE Internet of Things Journal, 2021, 8(14): 11016-11040.

[30] 黄冬晴. 移动边缘计算中联合计算卸载和资源分配策略研究[D].上海: 华东师范大学, 2021.
HUANG D Q. Research on joint computation offloading and resource allocation strategy in mobile edge computing[D]. Shanghai: East China Normal University, 2021.

[31] ALFAKIH T, HASSAN M M, GUMAEI A, et al. Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA[J]. IEEE Access, 8: 54074-54084.

[32] LI Y Y, FADDAE, MANERBA D, et al. Reinforcement learning algorithms for online single-machine scheduling[C]//Proceedings of the 2020 Federated Conference on Computer Science and Information Systems", "Annals of Computer Science and Information Systems. Piscataway: IEEE Press, 2020: 277-283.

[33] AHSAN W, YI W Q, QIN Z J, et al. Resource allocation in uplink NOMA-IoT networks: a reinforcement-learning approach[J]. IEEE Transactions on Wireless Communications, 2021, 20(8): 5083-5098.

[34] JIANG H, GUI R J, CHEN Z, et al. An improved sarsa(-) reinforcement learning algorithm for wireless communication systems[J]. IEEE Access, 7: 115418-115427.

[35] KUCHIBHOTLA V, HARSHITHA P, GOYAL S. An N-step look ahead algorithm using mixed (on and off) policy reinforcement learning[C]//Proceedings of 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS).Piscataway: IEEE Press, 2020: 677-681.

[36] KENNEDY J, EBERHART R. Particle swarm optimization[C]//Proceedings of ICNN'95 - International Conference on Neural Networks. Piscataway: IEEE Press, 1995: 1942-1948.

[作者简介]

孙君（1980− ），女，南京邮电大学副研究员、硕士生导师，主要研究方向为无线网络、无线资源管理和物联网。

赵尚维康（1997− ），男，南京邮电大学硕士生，主要研究方向为无线网络、无线通信中的资源分配及边缘计算。