边云协同场景下基于强化学习的精英分层任务卸载策略研究

方娟，叶志远，张梦媛，史佳眉，滕自怡
（北京工业大学信息学部，北京 100124）

摘　要：随着 5G 的发展以及应用程序功能的丰富，应用程序对终端设备的计算能力提出了更高的要求，为了提高终端设备对应用程序的计算能力，降低任务的处理时间，针对移动边缘计算环境，提出了一种边云协同的任务卸载方式，并设计了基于强化学习的精英分层进化算法（RL-EHEA, elite hierarchical evolutionary algorithm combined with reinforcement learning）进行卸载决策，使多个具有依赖关系与截止时间的任务对计算资源竞争。结果表明，与遗传算法（GA, genetic algorithm）和精英遗传算法（EGA, elite genetic algorithm）相比，RL-EHEA 能缩短任务的处理时间，得到更优的资源分配策略。

关键词：移动边缘计算；任务卸载；边云协同；进化算法；串行任务
中图分类号：TP391
文献标志码：A
doi: 10.11959/j.issn.2096−3750.2022.00258

Research on elite hierarchical task offloading strategy based on reinforcement learning in edge-cloud collaboration scenario

FANG Juan, YE Zhiyuan, ZHANG Mengyuan, SHI Jiamei, TENG Ziyi
Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

Abstract: With the development of 5G and the enrichment of application functions, applications have put forward higher requirements on the computing capabilities of terminal devices. In order to improve the computing capabilities of terminal devices on applications and reduce the processing time of tasks, it is aimed at mobile edge computing environments, a task offloading method for edge-cloud collaboration was proposed，and an elite hierarchical evolutionary algorithm combined with reinforcement learning (RL-EHEA) was designed to perform offloading decisions, so that multiple tasks with dependencies and deadlines compete for computing resources. The simulation experiment results show that, compared with genetic algorithm (GA) and elite genetic algorithm (EGA), RL-EHEA can shorten task processing time and obtain better resource allocation strategy.

Key words: mobile edge computing, task offloading, edge-cloud collaboration, evolutionary algorithm, serial task

0　引言

随着移动互联网、物联网和 5G 的飞速发展，新型移动应用迅速受到了越来越多用户的青睐[1]。移动终端作为移动应用的主要载体，其功能虽比以往更加丰富，但有限的计算资源已经成为其运行复杂应用的障碍[2-3]。为降低应用程序的处理时间，任务卸载作为一种高效可行的解决方案被提出。

移动云计算（MCC, mobile cloud computing）作为一种有效的任务卸载方式，终端设备可通过将任务卸载至云端以降低任务的处理时间。云端虽然能提供充足的计算资源，但是任务卸载产生的通信成本随着终端设备和云端之间的物理距离增加而增加，产生的时延使其无法为强实时性应用提供计

收稿日期：2021−09−01；**修回日期**：2021−12−29
通信作者：方娟，fangjuan@bjut.edu.cn
基金项目：国家自然科学基金资助项目（No.61202076）；北京市自然科学基金资助项目（No.4192007）
Foundation Items: The National Natural Science Foundation of China (No.61202076), The Natural Science Foundation of Beijing (No.4192007)

算服务。因此，移动边缘计算（MEC, mobile edge computing）作为一种 5G 网络的关键技术应运而生[4-5]。MEC 服务器能够比云端更快地进行计算处理，但因不同区域内的终端设备数量不同，不同 MEC 服务器处理的任务数量存在差异，存在资源利用率低的问题。

综上，本文结合边缘层和云层的层间协同、边缘层内 MEC 服务器的层内协同两种方式，提出了一种边云协同的任务卸载方式，并设计了一种基于强化学习的精英分层进化算法（RL-EHEA, elite hierarchical evolutionary algorithm combined with reinforcement learning），得到任务处理时间最小的卸载策略。

本文的主要贡献包括以下 3 个方面。

1) 本文研究了具有依赖关系和截止时间的串行任务在边云协同场景下的任务卸载问题，并将问题归结为多背包问题。

2) 本文提出了一种基于强化学习的精英分层进化算法（RL-EHEA）。通过设计种群多样性判断方法，概率性地引入新种群，在进化过程中利用精英个体引导种群进化，同时，设计多种交叉算子和突变算子，通过设计基于强化学习的参数调整方法，实现进化参数的自适应调整。

3) 本文将设计的RL-EHEA在边云协同场景下进行实验。实验结果表明，相比于遗传算法（GA, genetic algorithm）和精英遗传算法（EGA, elite genetic algorithm），RL-EHEA 能够充分搜索与利用场景中的计算资源，得到任务处理时间最小的卸载策略。

1 相关工作

在移动边缘计算的任务卸载场景中，许多研究考虑单 MEC 服务器即可满足多个用户的任务卸载请求。

文献[6]提出了一种基于遗传算法的 MEC 任务缓存和迁移策略，以满足完成时间的限制和目标。文献[7]提出了一种基于遗传算法的联合资源优化方法，以优化 MEC 服务器中的卸载比例、信道带宽和 MEC 服务器的计算资源分配。文献[8]提出了一种基于深度 Q 网络的计算任务分发卸载算法，以提高任务执行效率。

随着任务量激增，单 MEC 服务器的计算资源逐渐无法满足用户需求。因此，许多学者开始考虑协同卸载，具体可分为层内协同和层间协同两种方式。

对于层内协同方式，它将边缘层内的 MEC 服务器连接，使得 MEC 服务器间可以协同计算。文献[9]提出了一个并行优化框架，以在设备电池电量有限的情况下最小化平均任务执行时间。文献[10]考虑了基站的协作缓存关系，通过提出的启发式分层协作缓存策略，最小化用户资源访问代价。文献[11]提出 MEC 服务器间的协作不仅可以减少用户任务的处理时延，而且可以降低能耗。文献[12]考虑多个 MEC 服务器合作，通过任务分流增强计算能力。

对于层间协同方式，其利用云端的计算资源提高边缘层的任务卸载能力。文献[13]针对计算任务卸载模式的选择及边云资源分配的问题，设计了一种基于次模理论的贪心算法。文献[14]研究了云端和 MEC 服务器在物联网中的协作，并通过分支定界算法解决了单用户计算的卸载问题。文献[15]通过云端和 MEC 服务器之间的协作，最小化所有移动设备的加权总时延。

在上述研究工作中，部分研究主要考虑单用户场景下的任务卸载问题，而在实际场景中需要考虑多个用户对有限计算资源的竞争。同时，上述研究或是只将 MEC 服务器作为独立的计算资源，或是只考虑了层内协同或层间协同两种方式之一，这可能导致出现资源利用率低的问题。此外，大部分研究没有考虑任务之间的依赖关系以及用户对任务完成时间的要求。针对上述不足，本文将应用设计成多个具有依赖关系和截止时间的串行任务，并构建终端层、边缘层和云层 3 层网络模型，提出一种边云协同卸载方式，并通过 RL-EHEA 算法得到任务处理时间最少的卸载策略。

2 模型构建和问题描述

2.1 场景模型

本文的场景模型由终端层、边缘层和云层组成，场景模型如图 1 所示。终端层共包含 n 个计算能力为 F_{U} 的设备，并设有等待队列，通过无线信道与所在区域的基站通信。边缘层由 k 个装有 MEC 服务器组的基站构成。其中，每个 MEC 服务器组中有多个 MEC 服务器，每个 MEC 服务器上配置了 l 个执行能力为 F_{M} 的虚拟机，当有多个计算任务时需要排队等待。相邻基站间通过高速链路链接，每个基站通过无线链路与计算能力为 F_{C} 的云端相连。

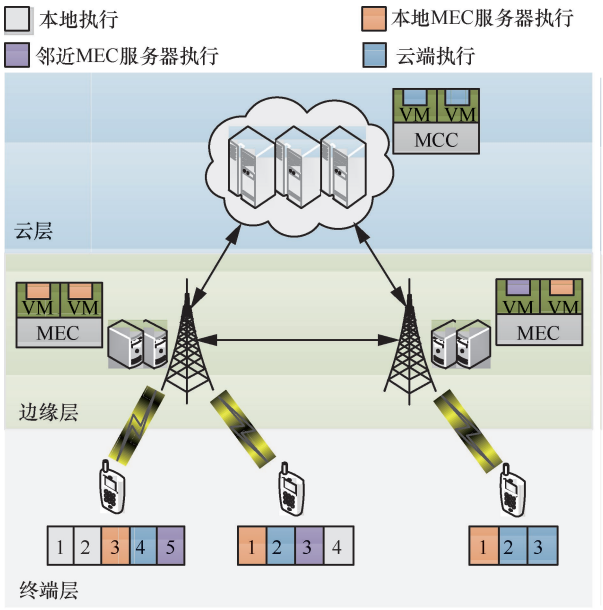

图 1　场景模型

2.2　任务模型

本文的应用程序由若干个串行任务构成，每个任务 Task 用六元组表示，采用线性链表 $L=\{V,E\}$ 表示任务间的依赖关系，后继任务 i 需要等待其前驱任务 j 执行完成后才可执行。其中，Task=$\{d_i,O_i,D_i,\mathrm{DDL}_i,\mathrm{pre},\mathrm{next}\}$，分别代表该任务的数据量、输出数据量、计算量、截止时间、前驱任务编号和后继任务编号。本文在链表的首部和尾部加入输入节点 In 和输出节点 Out，编号分别为−1 和 n。任务模型如图 2 所示。

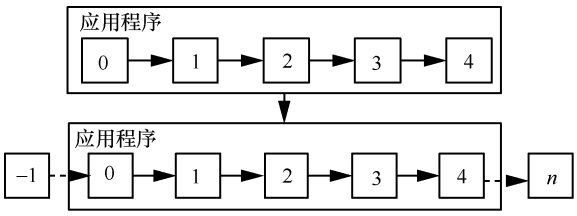

图 2　任务模型

2.3　卸载模型

场景中的任务可选择卸载到不同的位置执行。任务 i 需要等待前驱任务 j 执行完成并接收前驱任务的输出数据后才可参与资源竞争。当任务 j 与任务 i 位于同一执行位置时，其内部数据传输产生时间可忽略；反之，任务 j 与任务 i 间的数据传输时间将取决于两者所处位置。当任务 j 与任务 i 所处位置相邻时，传输时间由邻近位置间的传输能力决定；当任务 j 与任务 i 所处位置非相邻时，输出数据需要通过基站转发。因此，输出数据的传输时间 T_j^0 表示为

$$T_j^{\mathrm{O}}=\begin{cases}\dfrac{\mathrm{O}_j}{r_j^{P_j\to P_i}}, P_j\text{与}P_i\text{相邻}\\ \dfrac{\mathrm{O}_j}{r_j^{P_j\to P_x}}+\dfrac{\mathrm{O}_j}{r_j^{P_x\to P_i}}, P_j\text{与}P_i\text{非相邻}\\ 0,\text{其他情况}\end{cases}\tag{1}$$

其中，P_j、P_i、P_x 分别为任务 j、任务 i、中转基站 x 的位置，$r_j^{P_j\to P_i}$、$r_j^{P_j\to P_x}$、$r_j^{P_x\to P_i}$ 分别为 P_j 与 P_i、P_j 与 P_x、P_x 与 P_i 间的通信能力。

当任务 i 收到前驱任务 j 的输出数据后，方可参与资源竞争。此时，其在等待队列中的时间主要取决于排列在其之前任务的处理时间。本文所设排队规则按照紧迫度排序，任务 i 的紧迫度 u_i 表示为

$$u_i=\frac{1}{\mathrm{DDL}_i-T_{\mathrm{now}}}\tag{2}$$

其中，DDL_i 为任务 i 的截止时间，T_{now} 为当前时间。因此，任务 i 处理前的等待时间 T_i^{w} 表示为

$$T_i^{\mathrm{w}}=\begin{cases}0, L_i^{\mathrm{Q}}=0\text{且}S_j=1\\ t_j^{\mathrm{D}}+T_j^{\mathrm{O}}, L_i^{\mathrm{Q}}=0\text{且}S_j\neq 1\\ t_i^{\mathrm{w}}, L_i^{\mathrm{Q}}\neq 0\text{且}S_j=1\\ \max\left(\left(t_j^{\mathrm{D}}+T_j^{\mathrm{O}}, t_i^{\mathrm{w}}\right), L_i^{\mathrm{Q}}\neq 0\text{且}S_j\neq 1\right.\end{cases}\tag{3}$$

其中，L_i^{Q} 表示等待队列中排在任务 i 前的任务数；S_j 表示任务 i 的前驱任务 j 的状态，当 S_j 为 1 时，表示任务 j 执行完成且输出数据已传输至后继任务 i；t_j^{D} 表示前驱任务 j 的完成时间；t_i^{w} 表示任务 i 在等待队列中的时间。不同卸载选择具体如下。

1) 在终端设备执行

当任务选择在终端设备执行时，该任务就加入到等待队列中，此时任务计算产生的时间 T_i^{Lexe} 表示为

$$T_i^{\mathrm{Lexe}}=\frac{D_i}{F_{\mathrm{U}}}\tag{4}$$

因此，任务在终端设备执行的时间 T_i^{Local} 表示为

$$T_i^{\mathrm{Local}}=T_i^{\mathrm{Lexe}}+T_i^{\mathrm{w}}\tag{5}$$

2) 在本地 MEC 服务器执行

当任务选择卸载至本地 MEC 服务器执行时，传输时间 $T_i^{\mathrm{U}\to\mathrm{M}}$ 表示为

$$T_i^{\mathrm{U}\to\mathrm{M}}=\frac{d_i}{r_i^{\mathrm{U}\to\mathrm{M}}} \tag{6}$$

其中，$r_i^{\mathrm{U}\to\mathrm{M}}$ 为终端设备与本地 MEC 服务器之间的传输速率。此时任务计算产生的时间 T_i^{Mexe} 表示为

$$T_i^{\mathrm{Mexe}}=\frac{D_i}{F_{\mathrm{M}}} \tag{7}$$

因此，任务卸载至本地 MEC 服务器的时间 T_i^{LMec} 表示为

$$T_i^{\mathrm{LMec}}=T_i^{\mathrm{Mexe}}+T_i^{\mathrm{U}\to\mathrm{M}}+T_i^{\mathrm{w}} \tag{8}$$

3) 在邻近 MEC 服务器执行

当用户所在区域的本地 MEC 服务器无法满足任务的计算需求时，本地 MEC 服务器则向邻近 MEC 服务器进行任务迁移，传输时间 $T_i^{\mathrm{M}\to\mathrm{N}}$ 表示为

$$T_i^{\mathrm{M}\to\mathrm{N}}=\frac{d_i}{r_i^{\mathrm{M}\to\mathrm{N}}} \tag{9}$$

其中，$r_i^{\mathrm{M}\to\mathrm{N}}$ 为本地 MEC 服务器与邻近 MEC 服务器之间的传输速率。因此，任务卸载至邻近 MEC 服务器的时间 T_i^{NMec} 表示为

$$T_i^{\mathrm{NMec}}=T_i^{\mathrm{Mexe}}+T_i^{\mathrm{M}\to\mathrm{N}}+T_i^{\mathrm{U}\to\mathrm{M}}+T_i^{\mathrm{w}} \tag{10}$$

4) 在云端执行

当边缘层中的 MEC 服务器都无法满足任务的计算需求时，本地 MEC 服务器则把任务传输至云端处理，传输时间 $T_i^{\mathrm{M}\to\mathrm{C}}$ 表示为

$$T_i^{\mathrm{M}\to\mathrm{C}}=\frac{d_i}{r_i^{\mathrm{M}\to\mathrm{C}}} \tag{11}$$

其中，$r_i^{\mathrm{M}\to\mathrm{C}}$ 为本地 MEC 服务器与云端之间的传输速率。此时任务计算产生的时间 T_i^{Cexe} 表示为

$$T_i^{\mathrm{Cexe}}=\frac{D_i}{F_C} \tag{12}$$

因此，任务卸载至云端的时间 T_i^{Cloud} 表示为

$$T_i^{\mathrm{Cloud}}=T_i^{\mathrm{Cexe}}+T_i^{\mathrm{M}\to\mathrm{C}}+T_i^{\mathrm{U}\to\mathrm{M}}+T_i^{\mathrm{w}} \tag{13}$$

2.4 问题描述

本文的目标是最小化场景中多个用户在任务卸载过程中产生的执行时间总和 T_{all}，具体公式为

$$T_{\mathrm{all}}=\sum_{k=1}^{N}\sum_{j=1}^{L}\left[\sum_{i=1}^{M}\left(aT_i^{\mathrm{Local}}+bT_i^{\mathrm{LMec}}+cT_i^{\mathrm{NMec}}+dT_i^{\mathrm{Cloud}}\right)\right]_{k,j} \tag{14}$$

其中，N、L、M 分别代表场景中的用户数、每个用户产生的应用程序数以及每个应用程序包含的子任务数。因此，本文将优化问题定义如下

$$\min_{a,b,c,d} T_{\mathrm{all}} \tag{15}$$

$$\text{s.t. C1：}\ t_i^{\mathrm{D}}\leqslant \mathrm{DDL}_i$$

$$\text{C2：}\ a+b+c+d=1,\ a、b、c、d\in\{0,1\}$$

其中，约束条件 C1 表示任务 i 的完成时间 t_i^{D} 不超过其截止时间；约束条件 C2 表示任务只能选择场景中的一个位置执行，a、b、c、d 分别表示任务的卸载选择，取值为 0 或 1。

3 算法描述

本文研究的任务卸载问题可以归结为多背包问题，是一种 NP-hard 问题。因此，本文设计 RL-EHEA 算法进行任务卸载决策。将任务卸载策略编码为进化过程中的个体；结合优化问题构建适应度函数；设计种群多样性判断方法，概率性地引入新种群；设计多种进化算子，并引入强化学习使算法具有参数自适应的能力。

3.1 基于边云协同的多参数初始化

本文的个体编码方式结合了整数编码和浮点数编码两种方式，并将每个任务的卸载选择由一个基因组表示。每个基因组由两个基因位构成，第一个为 A 基因位，它代表任务的执行区域，选择整数编码方式，其中，−1 表示云端，0 表示终端设备，每个正整数表示 MEC 服务器组；第二个为 P 基因位，它代表任务的执行位置，采用浮点数编码方式，整数部分表示任务选择的 MEC 服务器，小数部分表示 MEC 服务器内的虚拟机。当任务选择在本地执行或云端执行时，P 取值为 0.0。个体编码模型如图 3 所示。

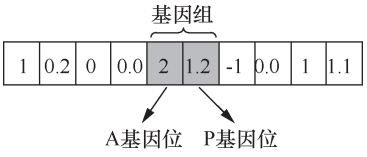

图 3 个体编码模型

本文研究的问题是获取任务处理时间最小化的任务卸载策略，因此，本文根据优胜劣汰的自然进化法则，将适应度函数 Fitness 表示为

$$\mathrm{Fitness}=\frac{1}{T_{\mathrm{all}}} \tag{16}$$

式(16)表明，任务处理时间越大的任务卸载策略，其 Fitness 值越小，则有更大的概率在进化中被淘汰。

3.2　基于多样性判断的种群调整方法

本文提出了个体多样性和基因多样性两种种群多样性判断方法，概率性地引入新种群，以扩大算法对卸载策略的搜索空间，避免因过早陷入局部最优卸载策略而停止搜索。下面是相关定义及具体介绍。

定义 1　个体多样性。个体多样性是指种群中个体 Fitness 值的频数分布情况。

定义 2　基因失衡。若个体中存在某类基因组的个数 m 超过个体总基因组个数 M 的一半时，本文则认为该个体存在基因失衡的现象。

定义 3　基因缺失。若个体中的基因组类型数 w 少于基因组类型总数 W 的一半时，本文认为该个体存在基因缺失的现象。

定义 4　基因多样性。基因多样性是指种群中个体出现基因失衡或基因缺失的频数分布情况。

3.2.1　个体多样性

若当前种群中存在超过半数个体的 Fitness 相同时，本文则认为当前种群的个体多样性不足。因此，种群的个体多样性判断函数 J_F 表示为

$$J_F=\begin{cases}0, \max(F)\leqslant\dfrac{h}{2}\\ 1, \max(F)>\dfrac{h}{2}\end{cases} \tag{17}$$

其中，h 为种群规模，F 为每个个体的 Fitness 值出现的频数。当 J_F 为 1 时，即种群中存在过多相似个体时，则引入新种群，以扩大对最优卸载策略的搜索空间。

3.2.2　基因多样性

若种群不存在个体多样性不足时，则考虑对种群做进一步的多样性分析，即基因多样性分析。

首先，本文根据 Fitness 值对种群中的个体进行排序。在数据分析过程中，箱线图法可得到数据的整体水平[16]，其中，组成箱线图的 5 个重要特征数分别是整体数据中的最小值 $d_{\min}$、上四分位数 d_{q1}、中位数 d_{mid}、下四分位数 d_{q3}、最大值 $d_{\max}$。因此，本文将种群中位于 5 个特征数位置的个体存入样本集 S 中，即 $S=\{d_{\min},d_{q1},d_{mid},d_{q3},d_{\max}\}$，以此评估种群的基因多样性水平。基因多样性判断函数 J_G 可表示为

$$J_G=\begin{cases}0, p\leqslant\dfrac{|s|}{2}\\ 1, p>\dfrac{|s|}{2}\end{cases} \tag{18}$$

其中，p 为 S 中基因失衡或基因缺失的个体数量，$|S|$表示样本集的容量，当 J_G 为 1 时，则引入新种群。

3.3　基于精英种群的策略搜索

为了发挥优秀个体在进化中的引导作用，本文引入精英种群，每次进化中 Fitness 值较高的个体将加入精英种群。在进化过程中，当前种群将依据适应度值分为较优种群与较差种群。较优种群可与精英种群中的最优个体进化以探索更优卸载策略。较差种群中的个体虽然适应度值较低，但存在优秀基因组合，其与精英种群中的其余个体进化可提升精英种群中个体适应度值，扩大对卸载策略的搜索空间。

同时，随着进化过程发展，精英个体将逐步加入当前种群，为了控制精英个体加入当前种群的规模，本文设置了加入比例系数 El，可表示为

$$\mathrm{El}=\frac{g}{G} \tag{19}$$

其中，g 为当前迭代次数，G 为总迭代次数。这使得精英个体将按比例增加加入当前种群的规模。

3.4　基于强化学习的进化设计

本文设计了减位交叉、合并交叉、减位突变、分裂突变 4 种算子，通过基于 Q-learning 的强化学习方法，自适应调整交叉率和突变率。

3.4.1　交叉阶段

在交叉阶段，单点交叉算子能够发挥优良基因的特性，提高算法收敛速度。但是当父代携带的卸载策略相近时，将进行许多无效交叉。并且单一的交叉算子不利于种群保持搜索能力。因此，本文设计了减位交叉和合并交叉两种交叉算子。

1) 减位交叉算子

减位交叉算子是指两个个体间每次需要交叉的基因组个数随着种群进化过程的发展而减少的交叉算子。减位交叉算子进化前、后期分别如图 4 和图 5 所示。

该算子使得算法前期能有更大的邻域搜索空间，并在后期保持算法收敛速度的同时对全局卸载策略进行小范围搜索。

2) 合并交叉算子

合并交叉算子是指两个个体的基因组片段合并从而产生新个体的交叉算子，如图 6 所示。

该算子能够将父代个体携带的卸载策略合并，进一步扩大了对全局最优卸载策略的邻域搜索空间，得到更优的卸载策略。

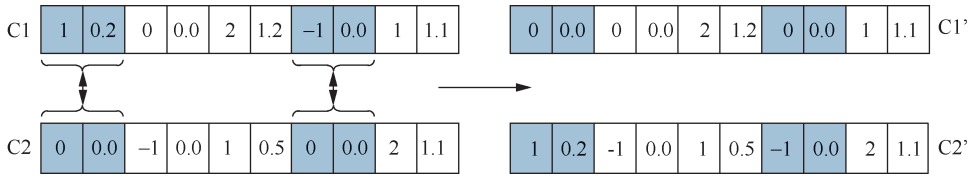

图 4　减位交叉算子（进化前期）

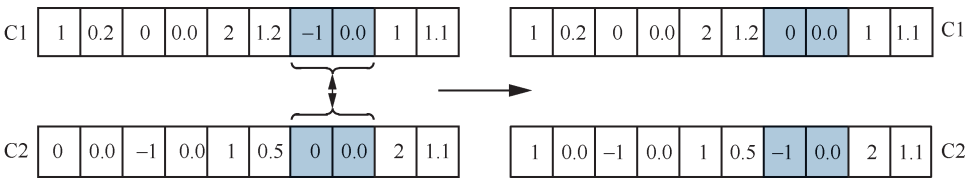

图 5　减位交叉算子（进化后期）

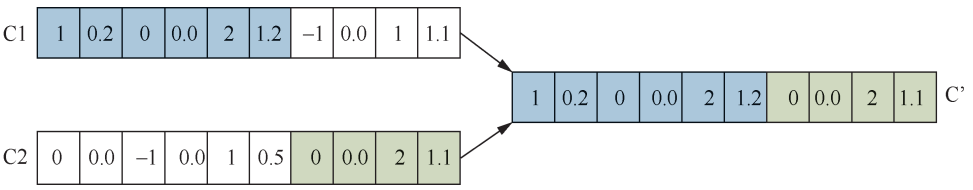

图 6　合并交叉算子

3.4.2　突变阶段

在突变阶段，单点突变算子能够在小范围改变个体的基础上对全局最优卸载策略进行逐步搜索。若在进化前中期采用这种算子，算法将较难跳出局部最优卸载策略。并且，单一的突变算子将使得算法的搜索空间过小。因此，本文在突变阶段设计了减位突变和分裂突变两种突变算子。

1) 减位突变算子

减位突变算子是指个体每次突变的基因组个数将随着种群进化过程的发展而减少的突变算子。减位突变算子进化前、后期分别如图 7 和图 8 所示。

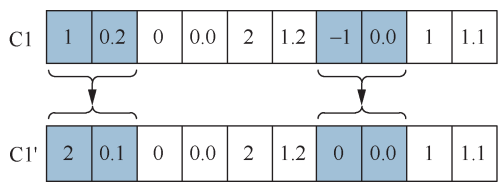

图 7　减位突变算子（进化前期）

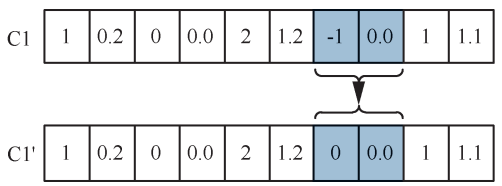

图 8　减位突变算子（进化后期）

该算子使个体在进化前期有更大的全局搜索范围，并在后期保留大部分原有卸载策略的情况下，逐步进行全局最优卸载策略的搜索。

2) 分裂突变算子

分裂突变算子是指个体以某对基因组为轴分裂成两个基因片段，每个基因片段与新随机产生的基因片段组合，直至片段长度达到与原个体的基因组合长度相同的突变算子，如图 9 所示。

该算子增加了种群的多样性，扩大了算法对全局最优卸载策略的搜索空间，降低了算法过早收敛于局部最优卸载策略而停滞搜索的可能。

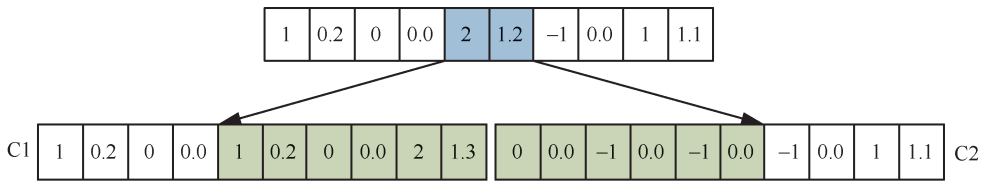

图 9　分裂突变算子

3.4.3　基于 *Q*-learning 的参数自适应方法

本文利用 *Q*-learning 的强化学习方式，使本文所提的进化算法能够实现参数自适应。具体描述如下。

1) 状态空间

本文将进化过程中的交叉率 e 和突变率 m 构成的二元组 State 作为状态空间，并根据卸载场景设定 e 和 m 的变动区间分别为[0.7,0.99]和[0.01,0.15]。

2) 动作空间

本文在综合考虑了状态空间以及进化算子对算法的影响后，将参数每次变化的步长设置为 0.05。因此，动作空间 Action 共有 9 种动作可供选择，即 Action={$(x,y)|x\in$ {−0.05,0,0.05}, $y\in$ {−0.05,0,0.05}}。

3) 奖励值

本文将 Agent 采取动作所获得的奖励值设置为由种群的最优适应度和平均适应度共同决定。因此，在学习过程中有 3 种奖励策略：平均适应度与最佳适应度都增加；只有平均适应度增加或最佳适应度增加；其他情况。不同的奖惩策略对应不同的奖惩值，这将有利于 Agent 朝着预期目标学习。

4) 参数自学习

Q-learning 强化学习方式是一种不断试错的学习过程，它通过贪婪策略不断更新状态—动作表，即 Q 表，其更新 Q 表的更新函数为

$$Q(s,a)\leftarrow(1-\alpha)Q(s,a)+\alpha\left[R(s,a)+\gamma\max Q(s',a')-Q(s,a)\right] \tag{20}$$

其中，α 是学习率，表示之前信息的利用程度；$R(s,a)$ 是从环境获得的奖励值；γ 是折现系数，表示 Agent 若选择到该新状态后可得到的部分 Q 值。

由于该算法的参数优化过程是没有终态的，因此，本文在 Agent 学习一定步数后，获取当前 Q 表中最大 Q 值对应的状态，作为进化中的交叉率和突变率。

综上，本文设计了一种基于强化学习的精英分层进化算法（RL-EHEA）。该算法根据场景中的处理器信息和任务信息构建初始种群。在进化过程中，利用多样性判断方法进行种群调整，并结合强化学习的进化方式实现基于精英种群的策略搜索。在多次进化后，种群中适应度最高个体携带的卸载策略即为最优卸载策略。RL-EHEA 的伪代码如算法 1 所示。

算法 1　RL-EHEA

输入　初始化种群 P

输出　最优任务卸载策略

for i←1 **to** 最大进化次数 **do**

　if 种群多样性低 **then**

　　引入新种群 N，并与种群 P 进化(P, N)

　end if

　//依据 Fitness 将 P 划分为 A_c 和 B

　(A_c, B)←(P [0:half], P [half:])

　// A_c、B 分别与 E_c 中最优个体、其余个体进化

　Evolution(A_c, E_c [0]);Evolution(B, E_c [1:])

　将进化得到的最优个体加入至 E_c，其余个体加入至 R

　E_c 中个体按比例加入至 R　根据个体适应度对 R 进行排序与选择

　P←R

end for

4　仿真实验

本节通过仿真实验评估 RL-EHEA 在边云协同场景下进行任务卸载决策的表现。

4.1　仿真环境

本文采用 PyCharm 2019 搭建了边云协同任务卸载实验环境[13-15]，实验参数见表 1。

表 1　　实验参数

实验参数	取值
终端设备运行的应用程序数量 N_p	[2,6]
应用程序包含的任务数量 N_T	5
终端设备的 CPU 频率 F_U/GHz	0.5
MEC 服务器的 CPU 频率 F_M/GHz	5
云端的 CPU 频率 F_C/GHz	15
任务计算所需的 CPU 周期 D_i/GHz	[0.1,0.9]
任务大小 d_i/MB	[0.5,1]
任务输出数据量 O_i/MB	[0.01,0.03]
终端设备与 MEC 服务器的传输速率 $r_i^{U\to M}$/(MB·s^{-1})	[3.5,5]
MEC 服务器间的传输速率 $r_i^{M\to N}$/(MB·s^{-1})	[10,20]
MEC 服务器与云端的传输速率 $r_i^{M\to C}$/(MB·s^{-1})	[2,3]

4.2　结果分析

1) 不同任务计算量

本文在不同任务计算量下进行仿真实验，以评估 RL-EHEA 的表现，不同计算量下的任务完成时间如图 10 所示。

从图 10 可以看出，随着任务计算所需的 CPU 周期数增加，任务计算的时间越来越长。对于 GA

而言，其对于全局计算资源的探索范围不足，造成多个任务在等待队列中堆积等待。对于 EGA，虽然其对 GA 进行了优化，但依然存在探索不足、过早收敛的问题。而 RL-EHEA 与 GA 和 EGA 相比，其通过更大范围地搜索场景中的计算资源，得到任务完成时间更短的卸载策略。其中，与 GA 相比，RL-EHEA 任务完成时间最少降低了 25.9%，最多降低了 28.8%，平均降低了 27.3%；与 EGA 相比，RL-EHEA 任务完成时间最少降低了 18.9%，最多降低了 21.9%，平均降低了 19.7%。

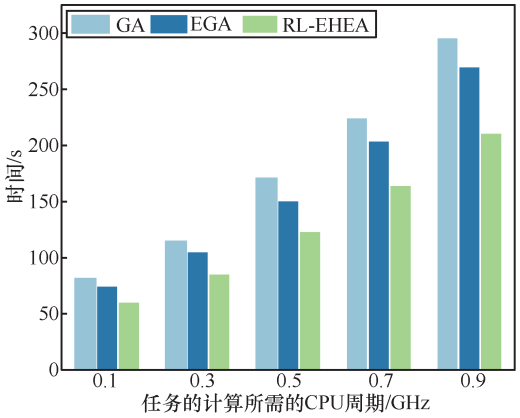

图 10　不同计算量下的任务完成时间

2) 不同终端设备数

本文通过在终端设备数量分别为 5、10、15、20 和 25 场景下的仿真实验，评估 RL-EHEA 资源分配表现，不同终端设备数下的任务完成时间如图 11 所示。

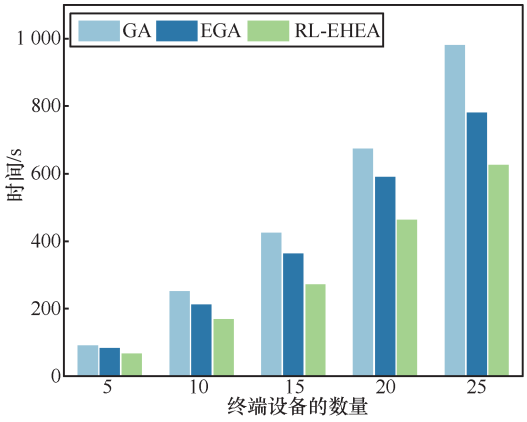

图 11　不同终端设备数下的任务完成时间

从图 11 可以看出，在资源受限场景中，当同一时隙内有多个终端设备参与任务卸载时，任务处理速度将与任务卸载速度不匹配，多个任务将在等待队列中等待执行，任务完成时间的增长幅度不断扩大。与 GA 和 EGA 相比，RL-EHEA 通过更大范围的资源探索能力，能更加全面地考虑场景中计算资源和等待队列情况，得到任务完成时间更短的任务卸载策略，与 GA 和 EGA 相比，RL-EHEA 平均任务完成时间降低了 21.6%和 15.3%。

为了进一步评估 RL-EHEA 的资源分配能力，本文对 GA、EGA、RL-EHEA 3 种算法得到的任务卸载策略中任务执行位置的分布情况进行了实验，GA、EGA、RL-EHA 不同终端设备数下的任务执行位置分布分别如图 12、图 13、图 14 所示。

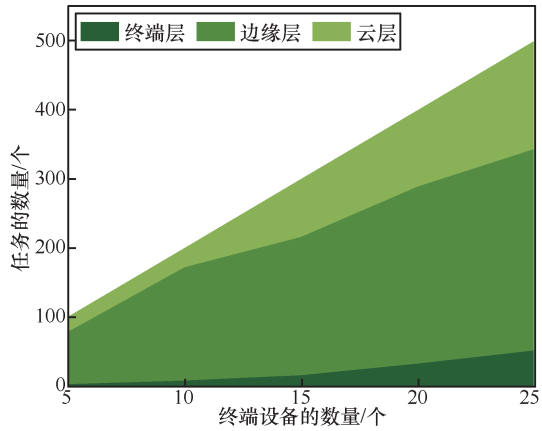

图 12　不同终端设备数下的任务执行位置分布（GA）

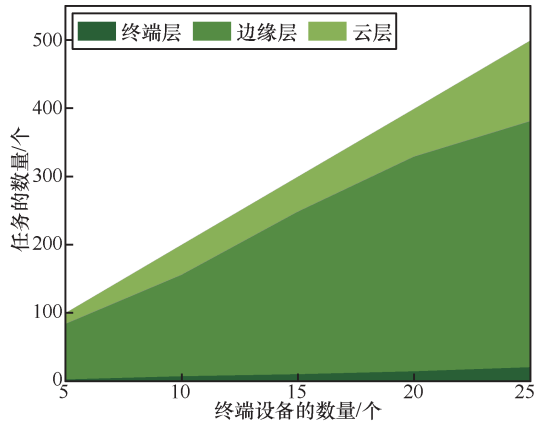

图 13　不同终端设备数下的任务执行位置分布（EGA）

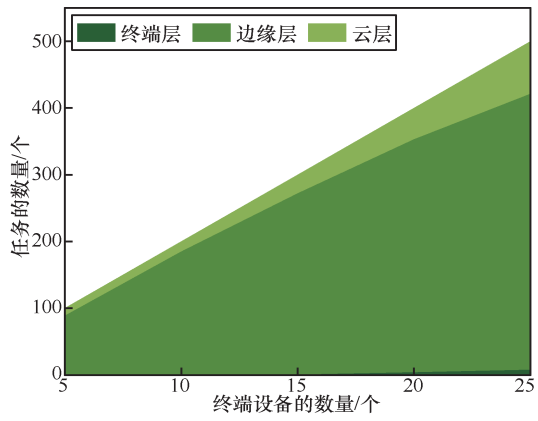

图 14　不同终端设备数下的任务执行位置分布（RL-EHEA）

从图 12、13、14 的实验结果可以看出，随着同时参与卸载的终端设备的数量不断增加，边缘层的计算资源逐渐无法满足任务的卸载需求，越来越多的任务在等待队列中堆积等待，GA、EGA、RL-EHEA 3 种算法在权衡了等待时间和传输时间两者的影响后，都选择将更多的任务卸载至云端或在终端设备执行，对于卸载到云端的比例，3 者分别从 14%、16%、7.5%增加到 31.4%、23.6%、15.8%，对于选择在终端设备执行的比例，3 者分别从 2%、2%、0%增加到 10.2%、4%、1.6%。此外，与 GA 和 EGA 相比，RL-EHEA 能够充分探索并利用边缘层中的计算资源，得到时间更短的卸载策略。相比于 GA 和 EGA，RL-EHEA 将卸载至边缘层的任务比例平均提高了 18.54%和 10.67%。

3) 不同 MEC 服务器数

为了能够更加全面地评估 RL-EHEA 对计算资源的搜索与利用能力，本文在不同 MEC 服务器数的场景下进行了仿真实验，不同 MEC 服务器数下的任务完成时间如图 15 所示。

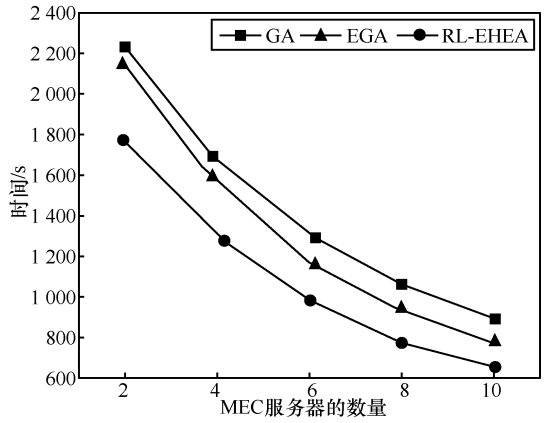

图 15　不同 MEC 服务器数下的任务完成时间

从图 15 的实验结果可以看出，当边缘层中的 MEC 服务器数量较少时，终端设备可选择的任务卸载位置十分有限。在此资源受限的情况下，RL-EHEA 通过对计算资源更多样地灵活分配，使其与 GA 和 EGA 相比，能更加充分使用有限的计算资源，得到时间更短的卸载策略，任务完成时间平均降低了 24.3%和 16.48%。随着 MEC 服务器个数增加，任务对计算资源的竞争程度减缓，当边缘层的计算资源超过了计算需求时，出现供过于求，使得算法间的差距缩小。

4) 算法收敛情况

本文对 GA、EGA 和 RL-EHEA 3 种算法进行收敛性对比实验，算法收敛性对比如图 16 所示。

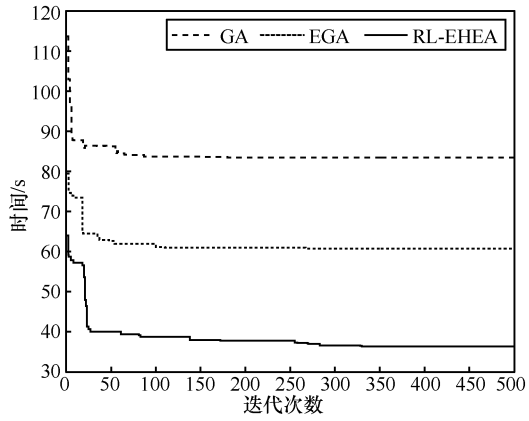

图 16　算法收敛性对比

从图 16 可以看出，RL-EHEA 算法在前 100 次迭代过程中进行大范围的搜索，使得其与 GA 和 EGA 相比，能更快搜索到任务完成时间更短的卸载策略。GA 和 EGA 分别在约为 200 次和 275 次迭代陷入局部最优卸载策略，而 RL-EHEA 通过不断增加种群的多样性，在保持收敛性的同时不断扩大对最优卸载策略的搜索空间，使其在约为 350 次迭代时收敛至稳定状态，得到任务完成时间更短的卸载策略。

5　结束语

本文将 MEC 服务器间的层内协同方式和 MEC 服务器与云端的层间协同方式相结合，提出了一种边云协同的任务卸载方式，并将应用构建为多个具有依赖关系和截止时间的串行任务。为了得到任务完成时间最小的卸载策略，本文提出了 RL-EHEA 算法进行卸载决策。实验结果表明，与 GA 和 EGA 相比，RL-EHEA 能得到任务完成时间更短的卸载策略。下一步，笔者考虑把任务处理时间和能耗作为联合优化目标。

参考文献：

[1] ZHANG K, LENG S P, HE Y J, et al. Mobile edge computing and networking for green and low-latency Internet of Things[J]. IEEE Communications Magazine, 2018, 56(5): 39-45.

[2] SABELLA D, VAILLANT A, KUURE P, et al. Mobile-edge computing architecture: the role of MEC in the Internet of Things[J]. IEEE Consumer Electronics Magazine, 2016, 5(4): 84-91.

[3] NING Z L, XIA F, ULLAH N, et al. Vehicular social networks: enabling smart mobility[J]. IEEE Communications Magazine, 2017, 55(5): 16-55.

[4] TALEB T, DUTTA S, KSENTINI A, et al. Mobile edge computing

potential in making cities smarter[J]. IEEE Communications Magazine, 2017, 55(3): 38-43.

[5] 施巍松, 张星洲, 王一帆, 等. 边缘计算: 现状与展望[J]. 计算机研究与发展, 2019, 56(1): 69-89.
SHI W S, ZHANG X Z, WANG Y F, et al. Edge computing: state-of-the-art and future directions[J]. Journal of Computer Research and Development, 2019, 56(1): 69-89.

[6] TANG L J, TANG B, KANG L Y, et al. A novel task caching and migration strategy in multi-access edge computing based on the genetic algorithm[J]. Future Internet, 2019, 11(8): 181.

[7] LI Z, ZHU Q. Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing[J]. Information, 2020, 11(2): 83.

[8] 赵海涛, 张唐伟, 陈跃, 等. 基于DQN的车载边缘网络任务分发卸载算法[J]. 通信学报, 2020, 41(10): 172-178.
ZHAO H T, ZHANG T W, CHEN Y, et al. Task distribution offloading algorithm of vehicle edge network based on DQN[J]. Journal on Communications, 2020, 41(10): 172-178.

[9] WANG Y, TAO X F, ZHANG X F, et al. Cooperative task offloading in three-tier mobile computing networks: an ADMM framework[J]. IEEE Transactions on Vehicular Technology, 2019, 68(3): 2763-2776.

[10] 葛志诚, 徐恪, 陈靓, 等. 一种移动内容分发网络的分层协同缓存机制[J]. 计算机学报, 2018, 41(12): 2769-2786.
GE Z C, XU K, CHEN L, et al. A hierarchical cooperative caching strategy for mobile content delivery network[J]. Chinese Journal of Computers, 2018, 41(12): 2769-2786.

[11] XIAO Y, KRUNZ M. QoE and power efficiency tradeoff for fog computing networks with fog node cooperation[C]//Proceedings of IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2017: 1-9.

[12] NGUYEN P D, HA V N, LE L B. Computation offloading and resource allocation for backhaul limited cooperative MEC systems[C]//Proceedings of 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall). Piscataway: IEEE Press, 2019: 1-6.

[13] 梁冰, 纪雯. 基于次模优化的边云协同多用户计算任务迁移方法[J]. 通信学报, 2020, 41(10): 25-36.
LIANG B, JI W. Multiuser computation offloading for edge-cloud collaboration using submodular optimization[J]. Journal on Communications, 2020, 41(10): 25-36.

[14] NING Z L, DONG P R, KONG X J, et al. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things[J]. IEEE Internet of Things Journal, 2019, 6(3): 4804-4814.

[15] REN J K, YU G D, HE Y H, et al. Collaborative cloud and edge computing for latency minimization[J]. IEEE Transactions on Vehicular Technology, 2019, 68(5): 5031-5044.

[16] LI Y, ZHAO Y L, SHAO J. Empirical study on CNG refilling behaviors of different timescale based on boxplot method[J]. IOP Conference Series: Earth and Environmental Science, 2018, 186: 012021.

[作者简介]

方娟（1973− ），女，博士，北京工业大学信息学部教授、博士生导师，中国计算机学会会员，主要研究方向为高性能计算、边缘计算、大数据技术等。

叶志远（1996− ），男，北京工业大学信息学部硕士生，主要研究方向为移动边缘计算。

张梦媛（1997− ），女，北京工业大学信息学部硕士生，主要研究方向为移动边缘计算。

史佳眉（1997− ），女，北京工业大学信息学部硕士生，主要研究方向为移动边缘计算。

滕自怡（1997− ），女，北京工业大学信息学部博士生，主要研究方向为移动边缘计算、高性能计算。