车联网云边协同计算场景下的多目标优化卸载决策

朱思峰[1]，蔡江昊[1]，柴争义[2]，孙恩林[1]
（1. 天津城建大学计算机与信息工程学院，天津 300384；2. 天津工业大学计算机科学与技术学院，天津 300387）

摘 要：车联网场景下的计算任务对时延非常敏感，需要云边协同计算来满足这类需求。针对车联网云边协同计算场景下如何高效地进行服务卸载并同时考虑服务的卸载决策以及边缘服务器和云服务器的协同资源分配问题，设计了基于云边协同的车辆计算网络架构，在该架构下，车载终端、云服务器和边缘服务器都可以提供计算服务；通过对缓存任务进行分类并将缓存策略引入车联网场景，依次设计了缓存模型、时延模型、能耗模型、服务质量模型以及多目标优化问题模型；给出了一种基于改进的多目标优化免疫算法的卸载决策方案。最后，通过对比实验验证了所提卸载决策方案的有效性。

关键词：车联网；云边协同；卸载决策；边缘缓存；多目标优化免疫算法
中图分类号：TP393.1; TN929.5
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2022114

Multi-objective optimal offloading decision for cloud-edge collaborative computing scenario in Internet of vehicles

ZHU Sifeng[1], CAI Jianghao[1], CHAI Zhengyi[2], SUN Enlin[1]
1. School of Computer and Information Engineering, Tianjin Chengjian University, Tianjin 300384, China
2. School of Computer Science & Technology, Tiangong University, Tianjin 300387, China

Abstract: Computing tasks in Internet of vehicles are very sensitive to offloading delay, cloud-edge collaborative computing is required to meet such requirements. Aiming at the problem that in the cloud-edge collaborative computing scenario of the Internet of vehicles, it is a challenging problem that how to efficiently offload services, and simultaneously consider the offloading decisions of services with the collaborative resource allocation of edge servers and cloud servers, a vehicle computing network architecture based on cloud-edge collaboration was designed. In this architecture, vehicle terminals, cloud servers and edge servers could provide computing services. The cache strategy was introduced into the scenario of Internet of vehicles by classifying cache tasks. The cache model, delay model, energy consumption model, quality of service model and multi-objective optimization model were designed successively. An improved multi-objective optimization immune algorithm was proposed for offloading decision making. Finally, the effectiveness of the proposed offloading decision scheme was verified by comparative experiments.

Keywords: Internet of vehicles, cloud-edge collaboration, offloading decision, edge cache, multi-objective optimization immune algorithm

收稿日期：2022−01−24；**修回日期：**2022−05−09
通信作者：蔡江昊，jianghao_cai@163.com
基金项目：国家自然科学基金资助项目（No.61972456）；天津市自然科学基金资助项目（No.20JCYBJC00140）；泛网无线通信教育部重点实验室（BUPT）开放课题基金资助项目（No.KFKT-2020101）
Foundation Items: The National Natural Science Foundation of China (No.61972456), The Natural Science Foundation of Tianjin (No.20JCYBJC00140), The Open Project Fund of the Key Laboratory of Universal Wireless Communications (BUPT) of the Ministry of Education (No.KFKT-2020101)

0 引言

随着万物互联时代的到来，各类能够提升人们生活质量的物联网服务不断涌现，增强现实、自动驾驶和多维数字媒体等应用开始全面向移动终端迁移[1-3]。这些具有高资源需求性的任务对移动终端有限的计算和存储资源带来了极大的挑战。云计算允许用户将任务卸载到云端以使用其丰富的计算和存储资源。但车联网场景下的任务常常具有高时延敏感性的特点，上传到云端会带来不可避免的高额传输时延，这使常规的云计算模型在车联网场景下无法适用。边缘计算通过将任务卸载到网络边缘的服务设备上以提供就近的服务，实现了云计算资源的下沉，为云计算提供了良好的补充。虽然边缘计算可以显著降低任务的传输时延和传输过程中受到的干扰，但是与云计算相比，边缘计算所能提供的计算和存储资源是有限的，因此协同云计算和边缘计算可以提供更优质的服务[4]。此外，随着技术发展，车载终端所具有的资源也不应当被忽视，通过整合车辆空闲资源作为资源池可为车载任务提供服务。

目前，对于车联网场景下的云边协同计算卸载决策已经有了一定的研究基础。文献[5]在车联网场景下构建了单向公路模型并设计了边缘服务器和车载服务器协同工作的安全协商机制，提出了一种在车辆移动时的安全切换交互协议，完成了对卸载能耗和时延的优化；文献[6]在云边协同策略下，提出了一种协同优化计算卸载资源分配优化方案，在边缘服务器计算资源不足的情况下有效提高系统效用和计算时间。

此外，对于计算卸载策略的研究也已经较成熟，常用的方法主要有深度学习、强化学习和启发式算法等。文献[7]结合每个移动设备的系统效益和带宽分配建立了一个混合卸载模型，提出了一种分布式深度学习驱动任务卸载（DDTO, distributed deep learning-driven task offloading）算法，实现了对移动设备、边缘服务器和中心云服务器的共同优化；文献[8]设计了一种车辆辅助卸载的计算卸载网络架构，结合Q-learning方法和深度强化学习（DRL, deep reinforcement learning）方法获得计算卸载和资源分配的最佳策略；文献[9]结合云计算和雾计算设计了优化模型，采用基于非支配排序的遗传算法（NSGA-II, nondominated sorting genetic algorithm II）对多目标问题进行了优化求解，结果表明该模型可以有效地对系统的能量消耗和卸载时延进行优化；文献[10]在车联网场景下提出了一种基于时效性的数据传输模型，通过结合车对车（V2V, vehicle to vehicle）通信和车对基础设备（V2I, vehicle to infrastructure）通信，使用演化算法对数据质量和交付比2个相互冲突的目标同时优化。

在复杂的车载网络环境下，为了满足大量用户多样化的服务需求，用户可以将任务卸载到具有丰富资源的边缘服务器或云服务器执行，高效的计算卸载机制可以提供更优质的服务。然而服务端在处理任务时不仅需要具有充足的计算资源，也需要缓存相应的服务应用，现有工作通常只对服务端的计算资源进行管理，忽视了任务对服务缓存的需求。同时，边缘服务端有限的计算和存储资源也使计算卸载策略和边缘缓存策略呈相互耦合的关系。如何将边缘缓存策略引入车联网场景已经成为一个关键且具有挑战性的问题。文献[11]使用分层缓存策略对小基站（SBS, small base station）和宏基站（MBS, macro base station）进行了联合优化，在满足文件传输速率要求和缓存资源的前提下最大限度地提高网络容量；文献[12]通过异步分布式强化学习算法对车联网场景下的任务卸载和服务缓存问题进行了联合优化，对网络资源进行全面协同和管理。

上述文献分别从云边协同策略、计算卸载常用优化方法和边缘缓存策略多个角度对现有工作进行了介绍，可以看到，目前对于边缘计算卸载决策的研究较成熟，但对于车联网等特殊场景的计算卸载模式和边缘缓存策略在车联网领域的应用仍然存在盲点。针对该问题，本文将主要考虑在车联网场景下结合中心云、边缘云和车辆云三层架构模型，通过将卸载策略与信道状态结合，以突出计算卸载策略与边缘缓存策略之间的耦合关系，并采用多目标智能优化算法对模型进行求解。本文的主要贡献如下。

1) 设计了一种基于多目标免疫算法（MOIA, multi-objective immune algorithm）来实现对卸载时延、车载终端能耗和服务质量的多目标优化。

2) 提出了一种基于云边协同的计算卸载网络模型，使用基于车对车通信的无线传输来缓解车辆在跨区域传输任务时所带来的带宽压力。

3) 通过对服务应用进行分类，设计了一种自适应的边缘缓存策略并应用于车联网场景。

1　系统模型

本文主要考虑云边协同三层架构下的双向直线型公路场景，在路边均匀分布多个路侧感知单元并分别配备边缘服务器，并设有一个中心服务器。车辆与路侧感知单元采用无线通信，相邻的边缘服务器和所辖的各路侧感知单元之间采用有线连接，本文设计的系统架构如图 1 所示。图 1 展示了车载任务处理的 4 种方式：本地执行、原属边缘服务器执行、中心云服务器执行和跨区域边缘服务器执行。系统架构主要分为三层：中心云、边缘云、车辆云，该架构的主要优势介绍如下[12]。

纵向协作。在该模型中，有多种可为车载终端提供服务的设备（远点的中心集群云服务器、近点的边缘服务器、车辆本身的服务设备），通过对任务的智能调度，可以为处在网络底层的车辆设备提供更高质量的服务。

横向协作。当大量任务从车辆终端卸载到边缘服务器时，会不可避免地使各个服务器的负载不均衡，轻负载的服务器具有大量的闲置资源，重负载的服务器将使服务质量下降甚至使部分任务无法完成。通过对边缘服务器的任务调度迁移，可以提高系统资源使用率。

跨区传输。为解决任务跨区域传输问题，在该架构中引入基于车对车通信的无线传输，为车载任务传输提供更多的选择，以缓解服务器之间进行大量任务调度所带来的带宽压力。

在该模型中，车载服务单元、边缘服务器和云服务器均可为任务提供服务，每个服务单元具有独立的存储资源 Z_j 和计算资源 C_j，将服务器 EC_j 的特征信息表示为 $\{Z_j,C_j\},1\leqslant j\leqslant n+1$，车辆在公路上匀速行驶且有一个特定类型的计算密集型任务可向服务器申请卸载，该任务需要存储资源 z_i 和计算资源 c_i。使用二进制变量 a_{ir} 表示所有任务的任务类型，当 $a_{ir}=1$ 时，表示任务 i 的任务类型为 r，将 task_i 的特征信息表示为 $\{z_i,c_i,r\},1\leqslant i\leqslant m,1\leqslant r\leqslant k$。每个时隙边缘服务器都可预先缓存多个不同类型的服务应用，每个类型应用都会占用额外的存储资源 z_r，使用 $b_{jr}(1\leqslant j\leqslant n+1)$ 表示服务器的缓存情况，当 $b_{jr}=1$ 时，表示 EC_j 缓存了类型为 r 的服务应用。

本文采用集中式决策。每个时隙开始时，车载终端和边缘服务器分别将任务特征信息 $\{z_i,c_i,r\}$ 和

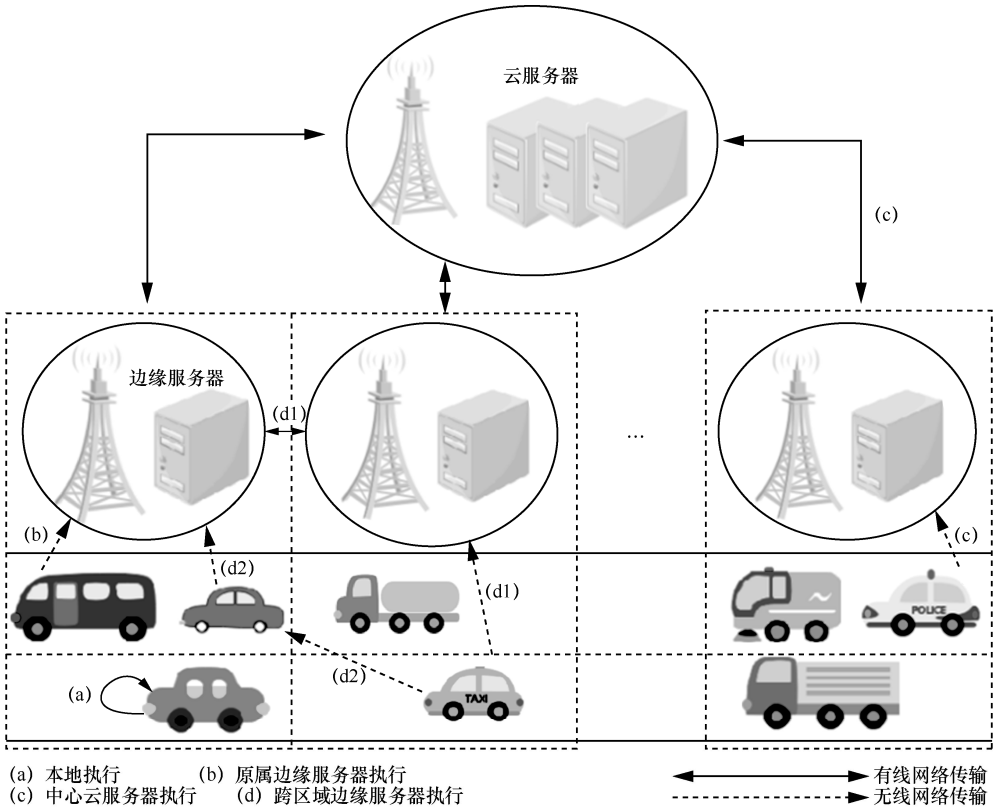

图 1　系统架构

服务器特征信息 $\{Z_j, C_j\}$ 上传至云服务器，云服务器将任务所需资源类型和各服务器所具有的资源进行汇总，结合卸载时延、车载能量消耗和服务质量成功率高效地为用户分配卸载目标服务器 x_i（云服务器、边缘服务器、车载终端）、对应的传输方式 h_i 和该时隙中边缘服务器的缓存策略 b_{jr}。特别地，当出现车辆任务卸载目标服务器不在该车辆所在区域时，车辆可以通过基于 V2V 技术的无线传输或通过服务器传输来完成车辆的跨区域传输。当使用 V2V 技术传输时，任务会先传给下一区域（为得出更有效的结论，本文将单个区域的长度定义为 V2V 技术的可靠传输范围）内的车辆，并由该车辆进行传递。

1.1 缓存模型

边缘缓存策略的优势在于通过提前在服务器端缓存相应的服务应用来更快地为指定类型任务提供更高效的服务，但边缘服务器有限的计算和存储资源无法缓存所有类型任务的服务应用，当边缘服务器使用未缓存类型服务应用时，需要从云服务器端下载该服务应用，该应用会在使用完成后被放弃，再次使用时需要重新下载，并带来额外的时延。而车载单元执行任务时，向边缘服务器请求服务应用，若边缘服务器未缓存该类型任务，则需要边缘服务器额外从云服务器下载并传递。设 g 为中心云服务器需要提供服务应用总次数，则云服务器吞吐率 R_{C2E}^{j} 为

$$R_{\mathrm{C2E}}^{j}=B\mathrm{lb}\left(1+\frac{SP}{\sigma^{2}}\right)\left(g+\sum_{i=1}^{m}u_{ij}\right)^{-1},j=n+1 \tag{1}$$

其中，二进制变量 u_{ij} 表示车载任务的卸载目标，$u_{ij}=1$ 表示 task_i 在 EC_j 上执行，$u_{ij}=0$ 表示 task_i 在车载终端本地执行；B 表示服务器之间分配的带宽，S 表示信道损失，P 表示服务器的传输功率，σ^2 表示噪声功率[13-14]。则在一个时隙内，下载 task_i 所需的服务应用所带来的缓存时延 T_{cache}^{i} 为

$$T_{\mathrm{cache}}^{i}=\begin{cases}\dfrac{z_r}{\theta_{\mathrm{V2R}}}+\dfrac{(1-w_i)z_r}{R_{\mathrm{C2E}}^{j}},\sum\limits_{j=1}^{n+1}u_{ij}=0\\ \dfrac{(1-w_i)z_r}{R_{\mathrm{C2E}}^{j}},\sum\limits_{j=1}^{n}u_{ij}=1\\ 0,\text{其他}\end{cases} \tag{2}$$

其中，θ_{V2E} 为车辆到边缘服务器（V2E, vehicle to edge computing server）的字节传输速率。w_i 为二进制变量，$w_i=1$ 表示 task_i 的所属服务器已缓存所需的应用程序。

1.2 时延模型

任务的执行时延分为处理时延和排队时延两部分[15]，其中处理时延只与服务器性能和任务所需的计算资源相关，则 task_i 的执行时延 T_{exec}^{i} 为

$$T_{\mathrm{exec}}^{i}=\frac{c_i}{C_j} \tag{3}$$

任务的排队时延与该服务器当前状态的负载情况相关，设 K_j 为 EC_j 上已卸载未完成的任务集合，则 task_i 的排队时延 T_{queue}^{i} 为

$$T_{\mathrm{queue}}^{i}=\begin{cases}0,\sum\limits_{j=1}^{n+1}u_{ij}=0\\ \sum\limits_{i\in K_j}T_{\mathrm{exec}}^{i},\text{其他}\end{cases} \tag{4}$$

任务的传输时延分为车辆上传时延、边缘服务器之间的传输时延、车对车传输时延和边缘服务器上传到中心云服务器的上传时延（E2C, edge computing server to cloud computing server）4 个部分，由于本文考虑计算密集型任务，计算结果回传时延忽略不计。车辆上传时延与任务所占的存储资源和车载终端的发射功率相关，则 task_i 的上传时延 T_{upl}^{i} 为

$$T_{\mathrm{upl}}^{i}=\frac{z_i}{\theta_{\mathrm{V2E}}} \tag{5}$$

任务在边缘服务器之间的传输时延与任务所占的存储资源和分配到的带宽相关，在设计的公路模型中，只有相邻的服务器采用有线连接，当二进制变量 y_{ij}=1 时，task_i 将会使用 EC_j 和 EC_{j+1} 之间的信道。结合香农公式，得到协同服务器完成卸载决策后 EC_j 和 EC_{j+1} 之间通信的单位传输速率 R_{E2E}^{j} 为

$$R_{\mathrm{E2E}}^{j}=B1\mathrm{b}\left(1+\frac{SP_j}{\sigma^{2}}\right)\left(\sum_{i=1}^{m}y_{ij}\right)^{-1} \tag{6}$$

则 task_i 在边缘服务器之间的传输时延 T_{trans}^{i} 为

$$T_{\mathrm{trans}}^{i}=\sum_{j=1}^{n-1}\frac{z_i y_{ij}}{R_{\mathrm{E2E}}^{j}} \tag{7}$$

当任务需要跨区域传输时，除通过服务器之间传输外，还可以通过基于车对车通信技术的无线传输，则车对车传输时延除了与任务所需存储资源和车辆传输功率相关外，还与所跨区域数量相关[16]，

则基于车对车通信的传输时延 T^i_{V2V} 为

$$T^i_{\mathrm{V2V}}=\frac{z_i}{\theta_{\mathrm{V2V}}}(1-h_i)\rho \tag{8}$$

其中，二进制变量 h_i 表示需要跨区域传输时任务的传输方式，$h_i=0$ 表示 task_i 会通过车对车通信的无线传输，$h_i=1$ 表示 task_i 会由服务器进行转递；ρ 表示通过车辆跨区域传输次数，θ_{V2V} 表示车对车的字节传输速率。则 task_i 在通过边缘服务器上传到中心云服务器的传输时延 T^i_{E2C} 为

$$T^i_{\mathrm{E2C}}=\frac{z_i u_{ij}}{R^j_{\mathrm{C2E}}},j=n+1 \tag{9}$$

车载终端的可卸载服务单元有车载终端、边缘服务器和云服务器 3 种，则 task_i 的卸载总时延 T^i_{offl} 为

$$T^i_{\mathrm{offl}}=\begin{cases}T^i_{\mathrm{cache}}+T^i_{\mathrm{exec}},\ \sum\limits_{j=1}^{n+1}u_{ij}=0\\ T^i_{\mathrm{cache}}+T^i_{\mathrm{exec}}+T^i_{\mathrm{queue}}+T^i_{\mathrm{upl}}+(1-h_i)T^i_{\mathrm{trans}}+h_iT^i_{\mathrm{V2V}},\ \sum\limits_{j=1}^{n}u_{ij}=1\\ T^i_{\mathrm{cache}}+T^i_{\mathrm{exec}}+T^i_{\mathrm{queue}}+T^i_{\mathrm{upl}}+(1-h_i)T^i_{\mathrm{trans}}+h_iT^i_{\mathrm{V2V}}+T^i_{\mathrm{E2C}},\text{其他}\end{cases} \tag{10}$$

该系统的卸载总时延为

$$T=\sum_{i=1}^{m}T^i_{\mathrm{offl}} \tag{11}$$

1.3　能耗模型

本节设计的系统模型中的能耗主要考虑车载终端的使用能耗，分为本地执行能耗和任务卸载能耗两大部分。本地执行能耗与任务大小和车载服务器功率相关，则 task_i 在本地执行时的能耗 E^i_{exec} 为

$$E^i_{\mathrm{exec}}=\frac{c_i}{C_i}\alpha \tag{12}$$

其中，α 表示车载服务器的功率。则 task_i 由车载终端卸载到服务器的能耗 E^i_{trans} 为

$$E^i_{\mathrm{trans}}=\frac{z_i}{\theta_{\mathrm{V2V}}}(\rho+1)\beta \tag{13}$$

其中，θ_{V2V} 表示车载终端的传输速率，β 表示车载终端的传输功率，则 task_i 的总能耗 E_i 为

$$E_i=\begin{cases}E^i_{\mathrm{exec}},x_{ij}=0\\ E^i_{\mathrm{trans}},\text{其他}\end{cases} \tag{14}$$

则系统总能耗 E 为

$$E=\sum_{i=1}^{m}E_i \tag{15}$$

1.4　服务质量模型

随着计算卸载和边缘缓存模型的建立，将任务从有限的计算和存储资源的车载终端卸载到性能更高的边缘服务器来降低任务计算时延和终端设备能耗成为所有车载终端的倾向，但呈正相关的时延和能耗关系并不符合实际场景和设计需求。随着车载终端数量的增加，任务在服务器端的排队时延和各服务器之间的信道质量会有所下降。而一定程度上增加的排队时延和下降的信道质量是可以接受的，为了将卸载决策与任务排队时延和信道质量优化结合，本节假设车载终端的任务具有最大响应时延 $t_{\max}$ [15]，设二进制变量 q_i 表示 task_i 是否卸载成功，即

$$q_i=\begin{cases}1,T^i_{\mathrm{offl}}<t_{\max}\\ 0,\text{其他}\end{cases} \tag{16}$$

则系统的服务质量 Q 为

$$Q=1-\frac{1}{m}\sum_{i=1}^{m}q_i \tag{17}$$

1.5　问题模型

本文在车联网场景下设计的模型将最小化时延作为主要目标，并保证各个服务器负载均衡的同时降低服务器总能耗。因此将本文要解决的问题定义为

$$\min T,\ \min E,\ \min Q \tag{18}$$

$$\text{s.t.}\quad \sum_{i=1}^{m}u_{ij}z_j+\sum b_{jk}z_k\leqslant Z_j,1\leqslant j\leqslant n+1 \tag{19}$$

$$\sum_{i=1}^{m}u_{ij}c_j\leqslant C_j,1\leqslant j\leqslant n+1 \tag{20}$$

$$\sum_{\mathrm{r}=1}^{k}b_{jr}=k,j=n+1 \tag{21}$$

$$\sum_{j=1}^{n+1}u_{ij}<2 \tag{22}$$

其中，式(19)表示在卸载过程中服务器的存储资源不会过载（任务所需的存储资源、缓存资源和临时缓存资源）；式(20)表示在卸载过程中服务器的计算

资源不会过载；式(21)表示云服务器会缓存所有类型的任务；式(22)表示每个任务仅在一个服务器或者本地执行。

2 基于 MOIA 的卸载决策方案

传统的优化算法由于具有较高的时间复杂度导致其在车联网场景下的效果不太理想，而演化算法在近年来得到了极大的发展，并能够较好地满足车联网场景下高时延敏感性的特点。免疫算法是一种新兴的主流演化算法，近年来得到了极大的发展和完善。在免疫算法中，抗原代表待优化的问题，抗体代表问题的候选解，抗体中一个基因位代表一个任务的卸载决策，采用亲和度函数来评估抗体对抗原的适应能力。

2.1 编码

通过对问题固有特征分析并选择合适的编码方式是演化类算法的第一步，本文提出的问题主要是对 3 个相互耦合内容进行组合优化。第一部分是对任务的卸载目标服务器进行决策，跨区域数量会通过影响服务器直接的信道质量而对时延造成直接影响，二进制编码容易丢失问题的固有信息，因此本节将二进制变量 u_{ij} 转化为十进制变量 x_i，转化式为

$$x_i=\begin{cases}0, \sum\limits_{j=1}^{n+1}u_{ij}=0\\ j, u_{ij}=1\end{cases} \tag{23}$$

当 $x_i=j$ 时，表示 task_i 在 EC_j 上执行；$x_i=0$ 时，表示 task_i 在车辆本地执行。第二部分对需要跨区域任务的上传方式进行决策，使用二进制变量 h_i 表示，当 h_i=1 时，task_i 会通过基于车对车通信的无线链路进行传输；当 h_i=0 时，task_i 会通过服务器进行传递。第三部分对各个服务器的服务应用缓存情况进行决策，使用二进制变量 $b_{jr}(1\leqslant j\leqslant n+1)$ 表示，当 b_{jr}=1 时，表示 EC_j 缓存了类型为 r 的服务应用；当 b_{jr}=0 时，表示 EC_j 未缓存该类型服务应用。

2.2 抗体种群初始化

在本文设计的模型中，任务跨区域传输次数将直接导致任务传输时延的增加。因此在初始化过程中，x_{ij} 将根据每个任务所属服务器的初始状态生成而不是随机生成，将 h_i 和 b_{jr} 全部设置为 0，默认初始状态所有任务通过服务器传输且边缘服务器不会缓存服务应用。最后依概率进行变异生成初始种群，并采用约束式(19)~式(22)对抗体种群进行修正。

2.3 亲和度评价函数

在免疫算法中，亲和度函数用来评价抗体和抗原的结合情况，即该抗体对应解与问题结合强度，对应遗传算法中的适应度函数。本文从卸载时延、能量消耗和服务质量 3 个方面进行评价，则抗体的亲和度评价函数分别如式(11)、式(15)和式(17)所示。

2.4 变异算子

免疫算法通过变异算子对得到的抗体群进行局部搜索，算法中所有变异操作均采用单点变异，通过将抗体的某一位依概率进行随机变异或按位取反。由于 x_{ij} 表征的任务只能有一个卸载目标，而任务从初始服务器传递到其他服务器的跨度将降低服务器之间的信道使用程度并直接影响时延，因此本节设计了一种基于距离的自适应变异方法，如式(24)所示。

$$\begin{aligned}&x_i=\begin{cases}\text{round}\left(x_i^{\text{init}}-\dfrac{1}{\kappa^{\frac{1}{\lambda}}}\right), x_i^{\text{init}}+\dfrac{1}{\kappa^{\frac{1}{\lambda}}}>n\\ \text{round}\left(x_i^{\text{init}}+\dfrac{1}{\kappa^{\frac{1}{\lambda}}}\right), x_i^{\text{init}}-\dfrac{1}{\kappa^{\frac{1}{\lambda}}}<0\\ \text{round}\left(x_i^{\text{init}}\pm\dfrac{1}{\kappa^{\frac{1}{\lambda}}}\right), \text{其他}\end{cases}\\ &\text{s.t. }\kappa=\text{rand}\left(\frac{1}{\left(\max(x_i^{\text{init}}, n-x_i^{\text{init}})\right)^{\lambda}}, 2^{\lambda}\right)\end{aligned} \tag{24}$$

其中，x_i^{init} 为 task_i 的初始服务器，round 为四舍五入函数，rand 为随机生成某个范围内数值的函数，n 为服务器数量，λ 为自定义系数。该方法源自函数 $y=x^2$，随着 x 的增长，y 会呈指数增长，通过 y 随机取值后反推 x，使概率与距离成反比，通过加入 x_i^{init} 实现自适应效果。例如，当 $\lambda=2$、$x_i^{\text{init}}=0$、$n=10$ 时，对应函数为 $\kappa=\dfrac{1}{x^2}$，当 $\kappa\in\left(\dfrac{4}{9},4\right)$ 时，x_i=1；当 $\kappa\in\left(\dfrac{4}{25},\dfrac{4}{9}\right)$ 时，x_i=2。对 h_i 和 b_{jr} 的变异操作采用单点变异和按位取反操作，具体如图 2 所示。

$$\begin{array}{cccc|c|c}
x_i & x_i' & h_i & h_i' & \overbrace{b_{jr}} & \overbrace{b_{jr}'}\\
7 & 7 & 0 & 0 & 0\ 0\ \cdots\ 0 & 0\ 0\ \cdots\ 1\\
5 & 3 & 0 & 1 & 0\ 0\ \cdots\ 0 & 0\ 1\ \cdots\ 0\\
\vdots & \Rightarrow\vdots & \vdots & \Rightarrow\vdots & \vdots\ \vdots\ \ddots\ \vdots & \Rightarrow\vdots\ \vdots\ \ddots\ \vdots\\
4 & 4 & 0 & 0 & 1\ 1\ \cdots\ 1 & 1\ 1\ \cdots\ 1
\end{array}$$

图 2 变异操作

2.5 MOIA 框架

本节提出了一种改进的 MOIA。首先根据分配的初始状态就近随机生成初始抗体种群；在对当前种群进行标准化和关联之后，根据非支配层选择前 $\frac{\mathrm{NP}}{2}$ 个抗体进行免疫操作（选择、克隆、克隆抑制）；在进行免疫操作时，为了保证生成解的多样性，MOIA 借鉴了 NSGA-III 算法中参考线为核心的理念，使用支配原则和基于参考点的选择来对抗体种群进行更新；最后刷新种群大小至 NP。与常规的 NSGA-III 算法相比，改进的 MOIA 具有更好的多峰值搜索能力。算法 1 给出了 MOIA 的伪代码。MOIA 的计算复杂度为 $O((n_{\mathrm{D}}+n_{\mathrm{C}})^2)$，其中 n_{D} 为决策种群规模，n_{C} 为克隆种群规模[17]。

算法 1　MOIA 的伪代码

输入　任务集合 $|M|$，服务器集合 $|N|$，种群规模 NP，最大迭代次数 $g_{\max}$

输出　种群 $P(g_{\max})$

开始

1) 初始化任务和服务器特征信息
2) 初始化种群 $P(1)$
3) for i=1:1:$g_{\max}$
4) 根据式(19)~式(22)对种群 $P(i)$ 进行约束
5) 根据式(11)、式(15)、式(17)计算种群 $P(i)$ 亲和度
6) 对种群 $P(i)$ 进行非支配排序，得到非支配层集合 $F(i)$ 和参考线集合 $R(i)$
7) 根据非支配层集合 $F(i)$ 对种群 $P(i)$ 进行免疫选择操作得到种群规模为 $\frac{\mathrm{NP}}{2}$ 的免疫种群 $Q(i)$
8) 根据参考线集合 $R(i)$ 对免疫种群 $Q(i)$ 进行免疫操作得到种群规模为 NP 的新种群 $P(i+1)$
9) end for

结束

2.6 标准化与子代种群

由于求解的问题需要同时对多个目标进行优化，而其数量级通常不在一个数量级上，因此在得到更新后的抗体种群后需要对其进行标准化以便进行下一步的操作，本节采用的标准化方式为

$$f_{\mathrm{np}}^{\ell}=\frac{\upsilon_{\mathrm{np}}^{\ell}-\upsilon_{\min}^{\ell}}{\upsilon_{\max}^{\ell}-\upsilon_{\min}^{\ell}},\ell=\{T,E,Q\},\mathrm{np}=\{1,2,\cdots,\mathrm{NP}\}\quad(25)$$

其中，υ_i^{ℓ} 为抗体的单个优化目标取值，NP 为种群规模。标准化完成后，将参考点与理想点相连得到参考线，并将每个个体和与其最近的参考线关联，根据关联后的非支配关系得到下一代种群。特别地，本文提出的 MOIA 的参考点设计使用 Das 和 Dennis[18]提出的方法，而理想点的取值为父代种群在每个目标上的最小值。

2.7 免疫操作

算法 2 给出了免疫操作的伪代码。在对选出的抗体进行克隆变异时，使用公式进行邻近变异以提升效率。同时，在改进的 MOIA 中深化了参考线的作用，在标准化完成后保留相关参数并在进行免疫操作时也会采用同样的参考标准，在克隆抑制操作时，当个体满足以下 3 个条件之一时，该个体就会被保留：1) 该个体是本次克隆种群内支配原个体程度最高的个体；2) 该个体使用新的参考线；3) 该个体的 3 个目标值小于原种群中的最小值，即不被理想点支配。特别地，条件 3)被用于处理优于理想点的个体，通过对条件 3)进行控制可以实现对优化目标的宏观调控（3.2 节）。在完成迭代后，将会返回一个略大于 $\frac{\mathrm{NP}}{2}$ 的种群，通过种群刷新操作保持种群规模为 NP。

算法 2　免疫操作的伪代码

输入　参考线集合 $R(i)$，免疫种群 $Q(i)$，种群规模 NP，克隆规模 NCL

输出　新种群 $P(i+1)$

开始

1) for np=1:1:NP
2) 对个体 $Q(i)$(np) 进行克隆操作生成种群规模为 NCL 的克隆种群 $L_{(\mathrm{np})}$
3) 根据式(24)对克隆种群 $L_{(\mathrm{np})}$ 进行变异
4) 根据式(19)~式(22)对种群 $L_{(\mathrm{np})}$ 进行约束
5) 根据式(11)、式(15)、式(17)计算种群 $L_{(\mathrm{np})}$ 亲和度
6) for nc=1:1:NCL
7) 个体 $L_{(\mathrm{np})}$ (nc) 支配了原个体 $Q(i)$(np)：使用 $L_{(\mathrm{np})}$ (nc) 替换原个体 $Q(i)$(np)
8) 个体 $L_{(\mathrm{np})}$ (nc) 使用了新的参考线：保存个体 $L_{(\mathrm{np})}$ (nc)
9) 个体 $L_{(\mathrm{np})}$ (nc) 不被理想点支配：保存个体 $L_{(\mathrm{np})}$ (nc)

10) 其他情况：删除个体 $L_{(np)}$ (nc)

11) end for

12) end for

13) 将免疫种群 $Q(i)$ 与克隆种群 L 结合生成新种群 $P(i+1)$

14) 通过种群刷新操作使新种群 $P(i+1)$ 的规模为 NP

结束

3　仿真实验及分析

为了验证本文提出的基于云边协同方案和 MOIA 的有效性，本节使用 MATLAB 软件进行仿真实验。在本节的仿真实验中，3.1 节将验证 MOIA 的有效性，并将其与传统的 NSGA-III[19-20]、NSGA-II[21]、MOEA-D[22] 和 Pareto 前沿进行对比；3.2 节对 MOIA 的定向优化性能进行实验验证；3.3 节对不同最大时延约束下 MOIA 方案的优化效果进行了对比；3.4 节对设计的通信策略和缓存策略进行了对比实验，并展示了 4 种卸载方案的各部分优化指标。本文实验的算法参数为迭代次数 Gen=100、种群规模 NP=100、变异概率 $\mathrm{Pc}=\frac{1}{m}$、克隆个数 NCL=10、任务数量 M=100、边缘服务器数量 N=5。本文实验中制定的详细仿真参数如表 1 所示。

表 1　　仿真参数

参数	数值
服务单元计算资源 C_j/MIPS	10，100，1 000
服务单元存储资源 Z_j/MB	100，1 000，10 000
任务请求计算资源 c_i/MIPS	1~10 上随机分布
任务请求存储资源 z_i/MB	10~100 上随机分布
服务应用存储资源 s_r/MB	10~100 上均匀分布
带宽 B/(Mbit·s^{-1})	200，2000
最大响应时延 $t_{\max}$/s	1，2，3
基于 V2V 技术的数据传输速率 θ_{V2V} /(Mbit·s^{-1})	400
基于 V2E 技术的数据传输速率 θ_{V2E} /(Mbit·s^{-1})	1 000
车载终端的计算功率 α /W	100
车载终端的传输功率 β /W	50

3.1　与不同卸载算法的优化效果对比

本节将本文提出的 MOIA 算法与传统的 MOEA-D、NSGA-II、NSGA-III 算法及 Pareto 前沿进行对比。其中 Pareto 是 4 种算法分别运行 100 次得到的最优前沿。图 3~图 5 分别展示了 4 种算法及其 Pareto 前沿的运行效果。可以看出，MOIA 算法具有更好的全局搜索能力，同时在单个目标的优化上具有较好的效果。特别地，由于在模型设计中卸载总时延和服务质量的评价没有呈现出对立性，因此在图 4 中 Pareto 前沿的设计中采用相同服务质量情况下的最小时延情况，即 Pareto 前沿为靠近服务质量一侧。

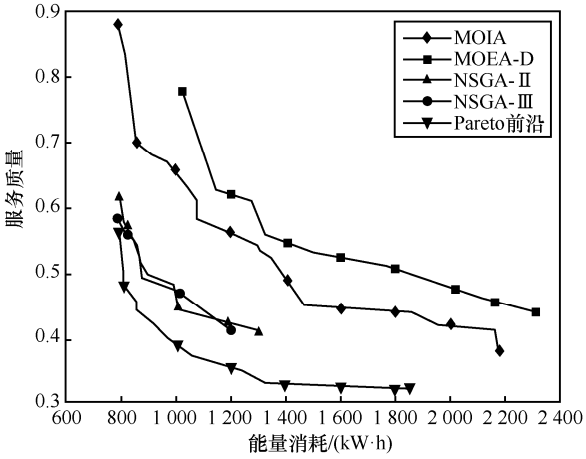

图 3　4 种算法及其 Pareto 前沿的能量消耗与服务质量对比

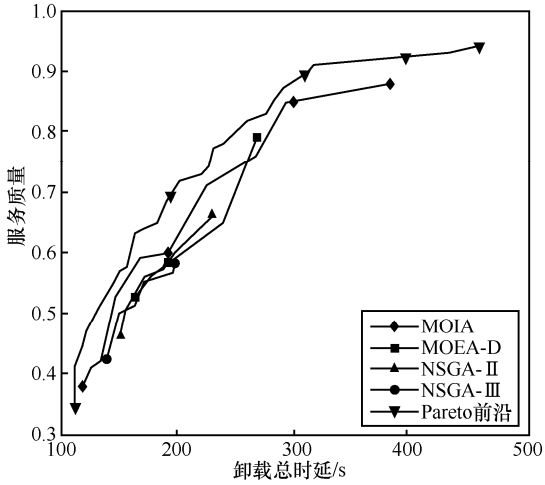

图 4　4 种算法及其 Pareto 前沿的卸载总时延与服务质量对比

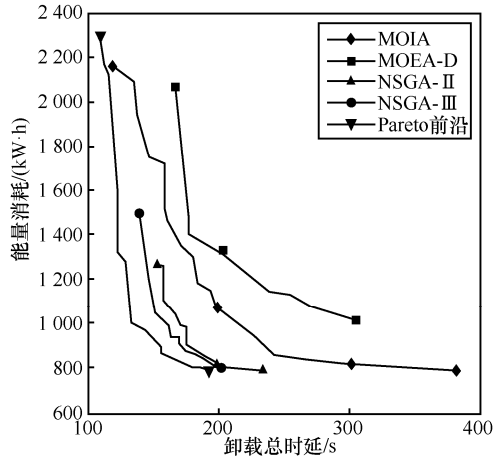

图 5　4 种算法及其 Pareto 前沿的卸载总时延与能量消耗对比

3.2　MOIA 定向优化能力测试

本节就MOIA在优化过程中对单个目标的调控能力进行实验，调控方法为在达到指定迭代次数前只对单个目标进行优化。图 6~图 8 为达到最大迭代次数后整个种群的分布情况，分别展示了对优化目标 T、E、Q 的对比优化效果，可以看出，在每个优化目标上的种群分布都更集中，且该目标上的最优值有所突破。其中，MOIA-T、MOIA-E、MOIA-Q 分别在前 50%的迭代次数中仅对该目标上满足条件 3)的个体（即在该目标上优于理想点的个体）进行保留。结合图 6~图 8 的种群分布情况，本文认为 MOIA 可以通过对迭代次数的控制来达到对特定目标的优化效果。特别地，结合图 6~图 8 的种群分布和基于参考点的算法理论，本文认为 MOIA 是通过增加种群在单个目标上的浓度以达到对更优解搜索的效果。

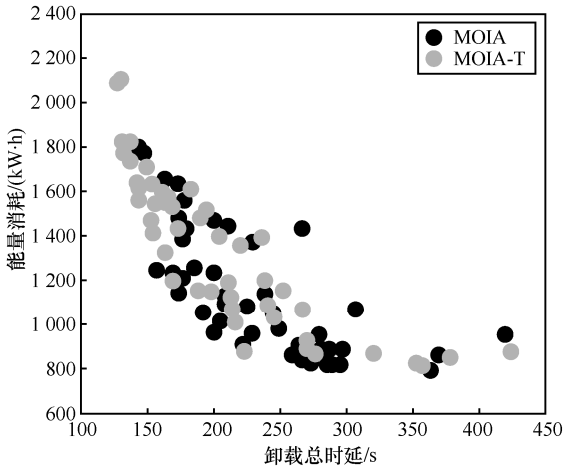

图 6　对卸载总时延优化时的种群分布

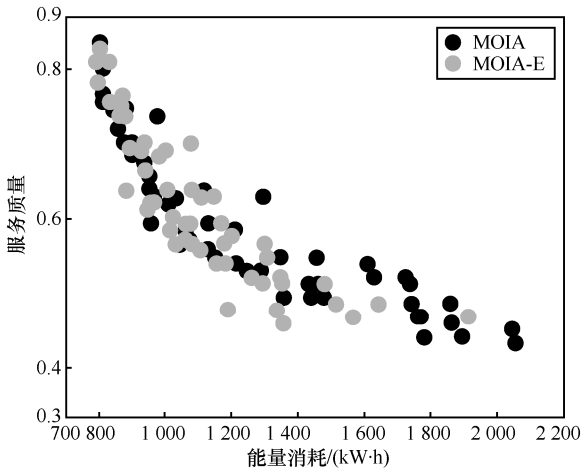

图 7　对能量消耗优化时的种群分布

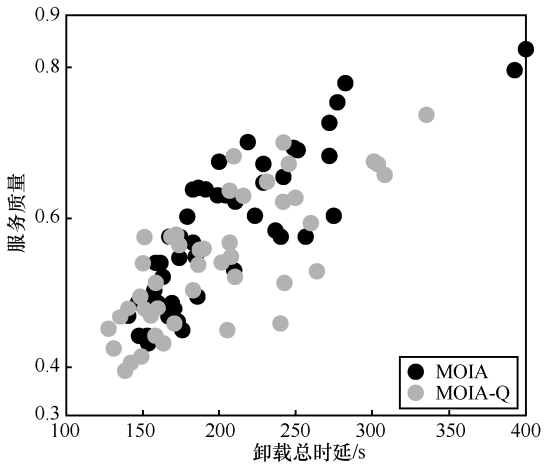

图 8　对服务质量优化时的种群分布

3.3　不同最大响应时延下的优化效果对比

图 9~图 11 分别展示了当最大响应时延 t_{max} 分别为 1 s、2 s、3 s 时 MOIA 的优化效果对比。图 9 和图 10 分别为服务质量与能量消耗和卸载总时延的关系，实验结果随最大响应时延变化呈梯次分布。图 11 为能量消耗和卸载总时延的关系，实验结果在同一梯度上，最大时延变化与能耗和任务卸载时延无直接关联，实验结果和理论依据相符。本文认为通过对最大时延的调整可以得到符合要求的实验数据。

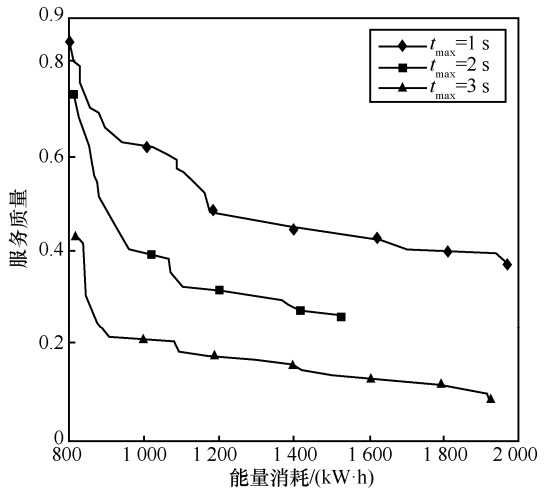

图 9　不同最大响应时延下能量消耗与服务质量对比

3.4　对缓存策略和通信策略的测试

为了对本文模型设计中的通信策略和缓存策略效果进行研究，本节选择 t_{max} =1 s 时 MOIA 方案在服务质量指标最低条件下能耗最小的个体进行展示。图 12 为各对比算法在服务质量上的取值；图 13 为各对比算法在能量消耗的取值；图 14 为各对比算法在各部分时延上的取值。其中，基准

（Benchmark）方案为 MOIA 对通信策略和缓存策略均随机赋值优化后得到的方案[16,23]。

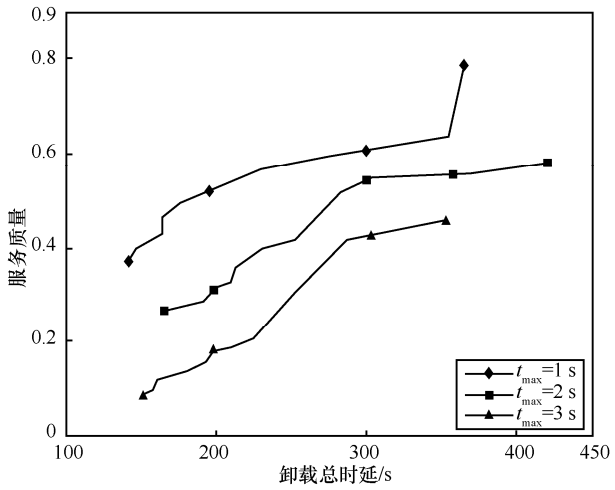

图 10 不同最大响应时延下卸载总时延与服务质量对比

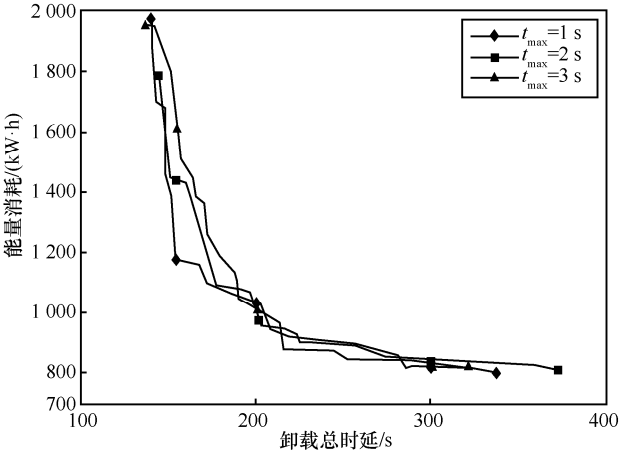

图 11 不同最大响应时延下卸载总时延与能量消耗对比

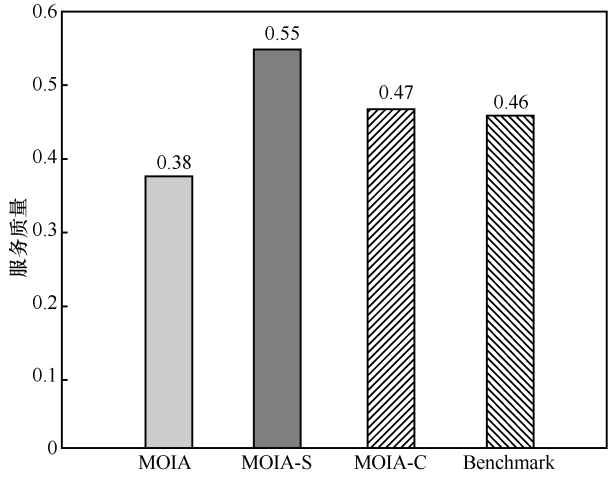

图 12　不同算法的服务质量对比

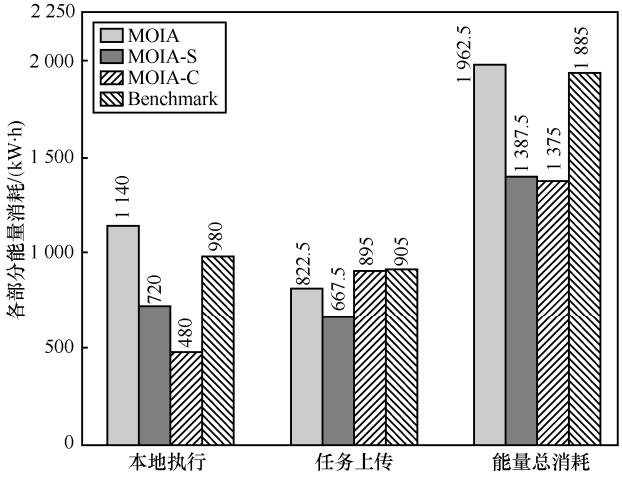

图 13　不同算法的各部分能量消耗对比

在本文设计的优化模型中，当任务需要卸载到其他边缘服务器时，可以通过基于 V2V 技术的无线传输或通过服务器进行传输；MOIA-S 方案对通信策略进行了测试，取消了基于 V2V 技术的无线传输，只能通过服务器进行传输。从图 14 中可以看出，MOIA-S 方案较 MOIA 方案和基准方案的卸载总时延分别增加了 58.45%和 28.67%；单独通过服务器进行任务传输会给带宽带来极大的压力，MOIA-S 方案的传输时延较 MOIA 方案和基准方案别分增加了 433.33%和 279.45%。同时，随着边缘服务器之间传输任务数量的增加，信道质量随之下降，MOIA 倾向于将任务卸载到云端服务器执行，因此图 14 中边缘服务器上传到云端的时延较 MOIA 方案增加了 62.58%，而缓存时延和执行时延分别下降了 35.29%和 29.13%，排队时延则上升了 7.25%。此外，由于取消了基于 V2V 技术的无线传输，车载任务只能通过服务器传输，因此图 13 中系统能耗大幅降低，且图 12 中服务质量指标也有所增加，表明多种传输方式会给用户更好的体验。

为对缓存策略进行测试，MOIA-C 方案取消了 50%的可缓存任务类型，即边缘服务器只会缓存一半的缓存任务类型。图 14 中 MOIA-C 方案卸载总时延较基准方案和 MOIA 方案分别增加了 10.35%和 35.88%。为了尽可能地减少缓存策略所带来的影响，算法倾向于更频繁地将任务卸载到具有该类型服务应用的边缘服务器上，图 14 中任务的传输时延和缓存时延较 MOIA 方案分别增加了 153.89%和 196.85%。同时，云服务器的高性能使算法倾向于将更多的任务卸载到云服务器，以降低额外下载服务应用所带来的时延，上传到云服务器的时延较

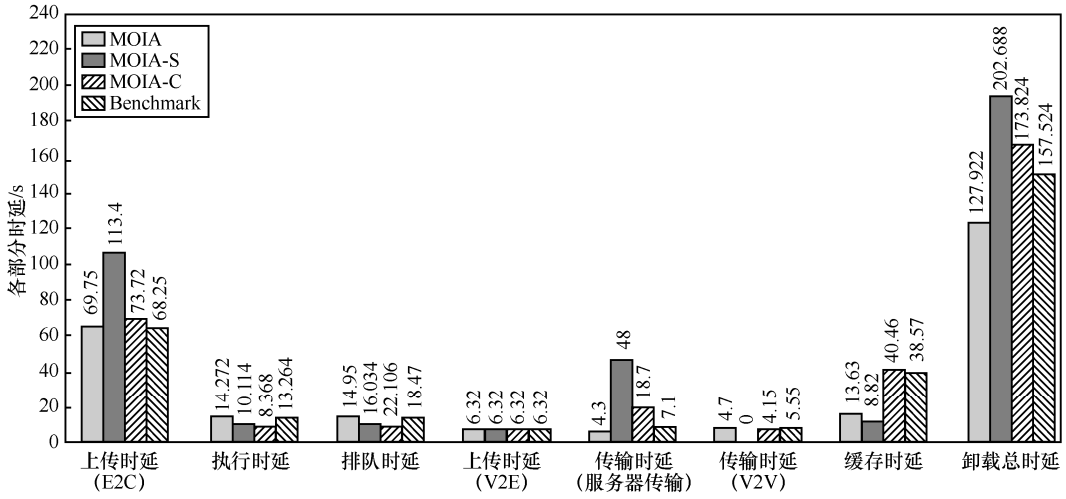

图 14　不同方案的各部分时延指标对比

MOIA 方案增加了 5.69%。由于上传到高性能云服务器的任务增加，任务的执行时延较 MOIA 方案降低了 41.37%，但排队时延增加了 47.87%。此外，部分边缘服务器没有相应的服务应用，MOIA-C 方案中任务会更多地向边缘服务器卸载而不是在车辆终端本地执行，而任务上传所带来的能量消耗远小于在本地执行的能量消耗，因此图 13 中车载终端的执行能耗大幅下降、上传能耗有所提升，总能耗较 MOIA 方案降低了 29.94%。图 12 中 MOIA-C 方案的服务质量指标较 MOIA 方案增加了 23.68%。本文认为，高效的缓存策略可以对边缘服务器的有限存储空间进行高效的利用，极大提升整体服务质量。此外，计算卸载策略和边缘缓存策略之间的耦合关系主要表现为边缘服务器有限的存储资源和车载终端多样的任务类型需求之间的矛盾，高效的边缘缓存决策可以极大降低从边缘服务器或云服务器下载服务应用的时间，实验结果与理论分析相吻合。

最后，结合图 12~图 14 中各优化目标的各部分指标和对 MOIA-S、MOIA-C 方案的对比实验可知，本文提出的缓存策略和通信策略是高效可行的。

4　结束语

本文基于车联网中的车对车无线传输技术和边缘缓存技术提出了一种自适应的服务缓存和任务卸载策略，主要在一条双向直线型公路上实现，在保证服务质量的基础上尽可能地降低车载任务的卸载总时延和车辆的能量消耗，通过建立卸载决策与传输时延的联系以加深对缓存策略与信道状态的联系。本文采用基于支配关系的免疫算法完成多目标优化，并给出了该算法的伪代码。仿真实验结果表明，本文提出的 MOIA 较传统的 NSGA-III 等多目标算法具有相当的全局搜索能力和更好的单目标上的搜索性能，且能够对设计的缓存策略和通信策略进行高效使用，达到了为车辆提供更优质服务的目的。

本文主要对服务器之间的信道质量和缓存策略进行使用，完成了对服务质量、车辆能量消耗和任务卸载时延的优化，但没有考虑车辆在任务传输时出现跨区域的情况。之后的工作将会在更复杂的交通场景下对车辆跨区域移动问题设计相关策略，以期更符合实际情况。

参考文献：

[1] 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.

[2] MAO S, WU J S, LIU L, et al. Energy-efficient cooperative communication and computation for wireless powered mobile-edge computing[J]. IEEE Systems Journal, 2022, 16(1): 287-298.

[3] CAO X W, WANG F, XU J, et al. Joint computation and communication cooperation for energy-efficient mobile edge computing[J]. IEEE Internet of Things Journal, 2019, 6(3): 4188-4200.

[4] MACH P, BECVAR Z. Mobile edge computing: a survey on architecture and computation offloading[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.

[5] 宋宇波, 金星妤, 燕锋, 等. 车联网中移动边缘计算的安全高效节能卸载策略[J]. 清华大学学报(自然科学版), 2021, 61(11): 1246-1253.
SONG Y B, JIN X Y, YAN F, et al. Secure and energy efficient offloading of mobile edge computing in the Internet of vehicles[J].

Journal of Tsinghua University (Science and Technology), 2021, 61(11): 1246-1253.

[6] ZHAO J H, LI Q P, GONG Y, et al. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(8): 7944-7956.

[7] WU H M, ZHANG Z R, GUAN C, et al. Collaborate edge and cloud computing with distributed deep learning for smart city Internet of things[J]. IEEE Internet of Things Journal, 2020, 7(9): 8099-8110.

[8] LIU Y, YU H M, XIE S L, et al. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(11): 11158-11168.

[9] ABBASI M, MOHAMMADI P E, KHOSRAVI M R. Workload allocation in IoT-fog-cloud architecture using a multi-objective genetic algorithm[J]. Journal of Grid Computing, 2020, 18(1): 43-56.

[10] DAI P L, LIU K, FENG L, et al. Temporal information services in large-scale vehicular networks through evolutionary multi-objective optimization[J]. IEEE Transactions on Intelligent Transportation Systems, 2019, 20(1): 218-231.

[11] ZHANG S, ZHANG N, YANG P, et al. Cost-effective cache deployment in mobile heterogeneous networks[J]. IEEE Transactions on Vehicular Technology, 2017, 66(12): 11264-11276.

[12] 刘雷, 陈晨, 冯杰, 等. 车载边缘计算中任务卸载和服务缓存的联合智能优化[J]. 通信学报, 2021, 42(1): 18-26.
LIU L, CHEN C, FENG J, et al. Joint intelligent optimization of task offloading and service caching for vehicular edge computing[J]. Journal on Communications, 2021, 42(1): 18-26.

[13] YANG B, CAO X L, BASSEY J, et al. Computation offloading in multi-access edge computing: a multi-task learning approach[J]. IEEE Transactions on Mobile Computing, 2021, 20(9): 2745-2762.

[14] WANG C M, YU F R, LIANG C C, et al. Joint computation offloading and interference management in wireless cellular networks with mobile edge computing[J]. IEEE Transactions on Vehicular Technology, 2017, 66(8): 7432-7445.

[15] HUSSAIN A, MANIKANTHAN S V, PADMAPRIYA T, et al. Genetic algorithm based adaptive offloading for improving IoT device communication efficiency[J]. Wireless Networks, 2020, 26(4): 2329-2338.

[16] XU X L, GU R H, DAI F, et al. Multi-objective computation offloading for Internet of vehicles in cloud-edge computing[J]. Wireless Networks, 2020, 26(3): 1611-1629.

[17] GONG M G, JIAO L C, DU H F, et al. Multiobjective immune algorithm with nondominated neighbor-based selection[J]. Evolutionary Computation, 2008, 16(2): 225-255.

[18] DAS I, DENNIS J E. Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems[J]. SIAM Journal on Optimization, 1998, 8(3): 631-657.

[19] DEB K, JAIN H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(4): 577-601.

[20] JAIN H, DEB K. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part Ⅱ: handling constraints and extending to an adaptive approach[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(4): 602-622.

[21] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[22] ZHANG Q F, LI H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731.

[23] LIU Q, MO R C, XU X L, et al. Multi-objective resource allocation in mobile edge computing using PAES for Internet of things[J]. Wireless Networks, 2020(3): 1-13.

[作者简介]

朱思峰（1975− ），男，河南周口人，博士，天津城建大学教授，主要研究方向为边缘计算、人工智能算法及应用等。

蔡江昊（1998− ），男，湖北武汉人，天津城建大学硕士生，主要研究方向为边缘计算、人工智能算法等。

柴争义（1976− ），男，陕西渭南人，博士，天津工业大学教授，主要研究方向为智能物联网、边缘计算等。

孙恩林（1996− ），男，河北邢台人，天津城建大学硕士生，主要研究方向为边缘计算、人工智能算法等。