一种针对能耗优化的车联网计算卸载方案

高文轩[1]，杨新杰[1,2]
（1. 宁波大学信息科学与工程学院，浙江 宁波 315211;
2. 网络与交换技术全国重点实验室（北京邮电大学），北京 100876）

摘　要：车联网环境下面向车辆的应用普遍具有计算密集和时延敏感等特性，引入移动车辆闲置计算资源作为网络算力的补充，可有效缓解边缘服务器的计算负载压力。针对车联网环境中边缘计算卸载的任务分配问题，充分利用 RSU、用户车辆和 RSU 服务范围内移动车辆的计算资源组合，提出一种基于麻雀搜索算法的计算卸载方案（sparrow search based computation offloading scheme，S^2COS），用以优化整体系统能耗。此外，该方案充分考虑了车辆移动性带来的服务时间限制以及计算节点出现故障的可能性等现实问题。仿真结果表明，S^2COS 在处理计算密集和时延敏感型任务时可以满足任务时延要求，并且能够显著降低系统能耗。

关键词：车联网；计算卸载；节点故障；麻雀搜索算法
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.2023189

A computation offloading scheme for energy consumption optimization in Internet of vehicles

GAO Wenxuan[1], YANG Xinjie[1,2]
1. Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China
2. State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications), Beijing 100876, China

Abstract: In Internet of vehicles (IoV), vehicle-oriented applications are generally computation-intensive and latency-sensitive. Introducing idle computing resources from mobile vehicles as a supplement to network computing power can effectively alleviate the load pressure on edge servers. The problem of task allocation for edge computation offloading in the context of IoV environment were researched. By fully leveraging the combined computing resources of roadside units (RSU), user vehicles, and mobile vehicles within the RSU service range, a computation offloading strategy based on the sparrow search algorithm was proposed and referred to as sparrow search based computation offloading scheme (S^2COS), aiming to optimize the overall system energy consumption. In addition, this strategy fully taked into account practical network issues such as service time constraints caused by vehicle mobility and the

收稿日期：2023−07−19；**修回日期：**2023−09−28
基金项目：宁波市自然科学基金资助项目（No.2019A610073）；网络与交换技术全国重点实验室（北京邮电大学）开放课题资助项目（No.SKLNST-2021-1-12）
Foundation Items: The Ningbo Municipal Natural Science Foundation (No.2019A610073), Open Foundation of State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) (No.SKLNST-2021-1-12)

potential occurrence of computation node failures. The simulation results demonstrate that S^{2}COS can meet the latency requirements for computation-intensive and latency-sensitive tasks, while significantly reducing system energy consumption.

Key words: Internet of vehicles, computation offloading, node failure, sparrow search algorithm

0 引言

车联网（Internet of vehicles，IoV）是移动物联网技术的衍生形式[1-2]。IoV 将车辆和其他网络节点引入移动通信网络中，通过信息收集、处理和实时反馈为车辆提供服务，实现智能交通。随着车辆逐步智能化，智能交通快速发展，计算密集型和时延敏感型任务的需求显著增加[3]。这些任务，如路径规划、车辆感知、虚拟现实等，对计算资源的高效利用和实时响应提出了更高要求。鉴于车辆的有限计算能力，许多任务难以有效地在本地进行处理。同时，传统的云处理方式车辆与云服务器之间距离较远，导致较大的传输时延，降低了服务质量（quality of service，QoS）。

对于蜂窝网络中的卸载问题，为了优化系统的能量效率或任务时延，一些学者研究了二进制卸载策略，即每个用户可以选择在本地执行任务或者将任务卸载到附近的边缘服务器[4-5]。类似地，对于车联网环境中的卸载问题，用户车辆的任务可以卸载到附近配备边缘服务器的路边单元（road side unit，RSU）上进行处理。例如，文献[6]考虑在一个多用户多边缘服务器场景中，用户车辆采用二进制卸载方式且对服务器的负载进行均衡优化，在降低任务时延的同时提高了资源利用率。与二进制卸载相比，部分卸载方式在有限的车辆、服务器计算资源和无线信道资源的情况下显得更为合理。文献[7]在单用户单边缘服务器场景中采用部分卸载方式，提出一种协同优化卸载数据比例和传输功率的策略，既满足业务时延要求又可最小化用户能耗。文献[8]在多车辆、多 RSU 组成的车辆边缘计算（vehicular edge computing，VEC）网络中考虑部分卸载策略，对卸载比和 RSU 选择进行协同优化，同时考虑了车辆移动性造成的任务时延，有效提升了系统的能效。

前述研究提出了二进制卸载或部分卸载策略，将任务卸载到边缘服务器上。然而，这些策略未能有效利用 IoV 中的其他空闲车辆的计算资源。此外，过多地将任务卸载到边缘服务器可能会导致服务器过载，从而降低系统性能。因此，有必要考虑合理利用空闲车辆的计算资源，以实现边缘服务器负载的均衡分配。文献[9]提出了一种联合卸载方案，将用户任务划分为 3 个部分：本地车辆计算部分，利用 V2I（vehicle to infrastructure）通信模式将任务卸载至边缘服务器的部分，以及通过 V2V（vehicle to vehicle）通信模式将任务卸载至空闲车辆的部分，该方案旨在优化服务时延。为了减轻边缘服务器的负载，在多用户设备、多空闲车辆组成的移动边缘计算（mobile edge computing，MEC）场景中，文献[10]提出一种空闲车辆辅助的二进制 MEC 任务卸载和资源分配方案，旨在优化基于任务优先级加权的总处理时延。文献[11]采用二进制卸载方式，利用空闲车辆和边缘服务器共同为多个用户车辆提供计算服务，并提出任务卸载和资源分配联合设计的策略，旨在优化任务时延和系统能耗的加权和。由于所形成的问题具有 NP 难属性，作者采用了博弈论和 Q 学习方法分别求解卸载决策和资源分配问题。

可以看出，上述研究主要以时延、能耗、自定义效用或成本函数为优化目标，有的研究采用二进制或部分卸载将任务卸载到边缘服务器，却忽略了空闲车辆的资源利用。另一些研究虽然同时利用了边缘服务器和空闲车辆资源，但却选择了二进制卸载方式，这在一定程度上导致了用户

车辆计算资源的浪费。此外，大多数研究假设节点工作时不会发生故障，这与实际情况存在偏差，导致结果的准确性受到影响。

在当前绿色网络的快速发展的背景下，在满足任务基本性能指标如时延的同时，有效降低整体系统能耗是一个值得关注的重要目标。本文的工作将用户任务进行部分本地计算、部分卸载给配备边缘服务器的 RSU 和 RSU 范围内的空闲车辆进行计算以充分利用资源，用以降低系统总能耗。同时，本文考虑了各个计算节点的故障情况和车辆移动性对任务计算时长的影响，并针对节点故障提出任务再分配和重计算机制，以确保任务能够完成。因为该优化问题的特殊属性，可以使用群智能优化算法来求解，相较于文献[12-13]中所使用的遗传算法和模拟退火等群智能方法，麻雀搜索算法具有更好的收敛速度、收敛稳定性和全局寻优能力[14]。因此，本文采用更契合本文优化问题的麻雀搜索算法来优化任务分配以最小化系统总能量消耗。

1　车辆边缘计算卸载系统模型

1.1　系统模型

本文考虑一个由多个移动车辆和单个RSU组成的移动边缘计算 IoV，车辆边缘计算卸载系统模型如图 1 所示。其中，RSU 配有边缘服务器，坐标为(0,0)，覆盖半径为 R，RSU 覆盖内的移动车辆配有车载处理单元（on-board unit，OBU），可提供计算服务。假设一个车辆发起任务，定义为用户车辆（user vehicle，UV），用 u 表示。另有 M 个空闲车辆，定义为服务车辆（service vehicle，SV），其集合用 $\mathcal{M}=\{1,2,\cdots,M\}$ 表示。

假设用户车辆 u 生成了一个计算密集且时延敏感的任务，用 $h_u=\{d,c,t^{\text{tole}}\}$ 表示，其中，d 为 h_u 的原始数据大小，c 为完成任务 h_u 所需的 CPU 周期数且满足 $c=d\alpha_u$，α_u 为复杂度，单位为 cycle/byte（周期/字节），t^{tole} 表示任务 h_u 的时延限制，即用户车辆 u 可接受的最大任务时延。由于自身算力有限，为满足任务时延要求，车辆 u 给 RSU 发送任务信息和卸载请求，RSU 随即在覆盖范围内广播请求信息，服务车辆收到广播后将车辆信息反馈给 RSU。服务车辆 i 的车辆信息用 $\{x_i,y_i,\vartheta_i,\nu_i,f_i\}$ 表示，其中，(x_i,y_i) 为坐标位置，ϑ_i 为运动方向，ν_i 为行驶速度，f_i 为可提供卸载计算的 CPU 算力，$i\in\mathcal{M}$。RSU 根据收到的车辆信息计算得到任务卸载比例 θ 并通知车辆 u 上传部分任务，RSU 进一步将收到的部分任务划分为更小的子任务，并通过 V2I 链路分发给服务车辆。每个服务车辆完成任务后将处理结果发回 RSU，由 RSU 合并后发回车辆 u。为方便分析，假设每个子任务是大小连续、复杂度相同且相互独立的。

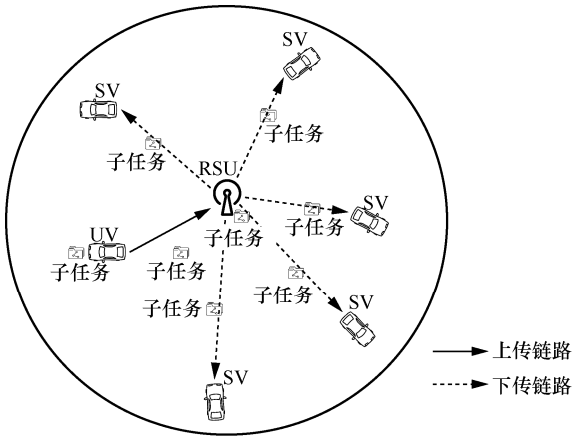


图 1　车辆边缘计算卸载系统模型

假定所有车辆在执行任务过程中行驶方向不变，定义 $t_{i,\text{R}}^{\text{hold}}$ 为服务车辆 i 驶出 RSU 覆盖前的时间，则：

$$\sqrt{\left(x_i+\nu_i\cos(\vartheta_i)t_{i,\text{R}}^{\text{hold}}\right)^2+\left(y_i+\nu_i\sin(\vartheta_i)t_{i,\text{R}}^{\text{hold}}\right)^2}=R \tag{1}$$

1.2　通信模型

在系统中，车辆 u 通过无线链路将部分任务卸载给 RSU，随后 RSU 将收到的任务分割成子任务并通过无线链路分发给服务车辆。本文考虑这些无线链路使用正交频分多址接入（orthogonal

frequency division multiple access，OFDMA）技术，系统给每个节点分配不同信道进行数据传输。这些信道由一定数量的子载波组成，由于子载波之间具有正交性，不同信道之间不会相互干扰。

用户车辆 u 到RSU的上行链路速率和接收信噪比（signal-to-noise ratio，SNR）可分别定义为[15]：

$$r_{u,\mathrm{R}} = B_{u,\mathrm{R}} \mathrm{lb}(1+\mathrm{SNR}_{u,\mathrm{R}}) \tag{2}$$

$$\mathrm{SNR}_{u,\mathrm{R}} = \frac{P_u \left|h_{u,R}\right|^2 \mathrm{di}_{u,\mathrm{R}}^{-\alpha}}{N_0} \tag{3}$$

$$\mathrm{di}_{u,\mathrm{R}} = \sqrt{x_u^{\,2} + y_u^{\,2}} \tag{4}$$

其中，(x_u, y_u) 为车辆 u 的坐标，P_u 为其发射功率，$\mathrm{di}_{u,\mathrm{R}}$ 为车辆 u 和RSU之间的距离，$B_{u,R}$ 为信道带宽，$h_{u,\mathrm{R}}$ 为瑞利信道小尺度衰落系数，$h_{u,\mathrm{R}}$ 服从复高斯分布CN(0,1)，α 为大尺度路径损耗因子，N_0 为加性高斯白噪声功率。

RSU 到服务车辆 i 的下行链路速率和接收信噪比具有类似定义，为避免冗余不再列出。

1.3 计算模型

车辆 u 将部分任务卸载至RSU并由RSU将划分后的子任务分发给服务车辆会产生相应的传输能耗、传输时延和计算时延、计算能耗。其中，任务的传输时延包含两部分：车辆 u 给RSU上传部分任务的传输时间；RSU给服务车辆 i 下发子任务的传输时间。因为任务处理后的数据量很小，返回结果的传输时间可忽略不计[16]。因此，车辆 u、RSU、车辆 i 的时延可以分别表示为：

$$t_u = \frac{d_u \alpha_u}{f_u} \tag{5}$$

$$t_{\mathrm{R}} = \frac{\theta d}{r_{u,\mathrm{R}}} + \frac{d_{\mathrm{R}} \alpha_u}{f_{\mathrm{R}}} \tag{6}$$

$$t_i = \frac{\theta d}{r_{u,\mathrm{R}}} + \frac{d_i}{r_{\mathrm{R},i}} + \frac{d_i \alpha_u}{f_i}, i \in \mathcal{M} \tag{7}$$

其中，f_u、f_{R}、f_i 分别是车辆 u、RSU、车辆 i 的计算资源，d_u、d_{R}、d_i 是它们的任务数据量，$r_{u,\mathrm{R}}$、$r_{\mathrm{R},i}$ 分别是车辆 u 上传给RSU、RSU下发给车辆 i 的链路速率。另外，RSU 还需要执行优化算法决定最优的任务分配，本文假设RSU配备边缘计算单元，具有强大的计算能力，其算法执行时间可以忽略不计。

根据文献[17]，车辆 u、RSU、车辆 i 的计算能耗可分别表示为:

$$E_u^{\mathrm{exe}} = k f_u^{\,2} d_u \alpha_u \tag{8}$$

$$E_{\mathrm{R}}^{\mathrm{exe}} = k f_{\mathrm{R}}^{\,2} d_{\mathrm{R}} \alpha_u \tag{9}$$

$$E_i^{\mathrm{exe}} = k f_i^{\,2} d_i \alpha_u, i \in \mathcal{M} \tag{10}$$

其中，k 是处理器芯片能耗系数[18]。车辆 u 上传给RSU及RSU下发给车辆 i 的传输能耗可分别表示为：

$$E_{u,\mathrm{R}}^{\mathrm{off}} = P_u \frac{\theta d}{r_{u,\mathrm{R}}} \tag{11}$$

$$E_{\mathrm{R},i}^{\mathrm{off}} = P_{\mathrm{R}} \frac{d_i}{r_{\mathrm{R},i}}, i \in \mathcal{M} \tag{12}$$

其中，P_u、P_{R} 分别是车辆 u 和RSU的发射功率。

1.4 考虑工作节点故障的重计算机制

在实际系统中，节点处理任务时可能会突发故障，因此需要在系统设计中考虑这种情况，以提高整个系统的稳定性。在由多个计算节点连接组成的分布式计算系统（distributed computing system，DCS）中，可以采取一些策略来提高系统的可靠性，比如使用高可靠的处理单元硬件等。当DCS的硬件配置固定时，系统可靠性取决于如何将给定的计算任务分配到适当的节点上，以及每个节点成功执行分配任务的概率[19-20]。本节将分析各个工作节点发生故障的情况。

如果节点在处理任务时没有发生故障，即成功完成分配的任务，这种情况的概率定义为成功概率。反之，如果节点在处理任务时发生故障，将无法完成分配的任务，这种情况的概率定义为故障概率。节点的成功概率和故障概率可由如下计算式表示[21]。

$$\begin{cases}\mathrm{re}_i=\mathrm{e}^{-\lambda_i\left(\frac{d_i\alpha_u}{f_i}\right)}\\ \mathrm{fa}_i=1-\mathrm{e}^{-\lambda_i\left(\frac{d_i\alpha_u}{f_i}\right)}\end{cases},i\in\mathcal{M} \tag{13}$$

$$\begin{cases}\mathrm{re}_u=\mathrm{e}^{-\lambda_u\left(\frac{d_u\alpha_u}{f_u}\right)}\\ \mathrm{fa}_u=1-\mathrm{e}^{-\lambda_u\left(\frac{d_u\alpha_u}{f_u}\right)}\end{cases} \tag{14}$$

$$\begin{cases}\mathrm{re}_\mathrm{R}=\mathrm{e}^{-\lambda_\mathrm{R}\left(\frac{d_\mathrm{R}\alpha_u}{f_\mathrm{R}}\right)}\\ \mathrm{fa}_\mathrm{R}=1-\mathrm{e}^{-\lambda_\mathrm{R}\left(\frac{d_\mathrm{R}\alpha_u}{f_\mathrm{R}}\right)}\end{cases} \tag{15}$$

其中，re_i、re_u、re_R 分别为服务车辆 i、用户车辆 u 和 RSU 的成功概率，fa_i、fa_u、fa_R 分别为故障概率，各节点在单位时间内发生故障的次数遵循泊松分布，平均故障率分别为 λ_i、λ_u、λ_R [19]。

鉴于无法提前知道各个节点在执行任务时是否发生故障，因此本文需要考虑所有节点（RSU、用户车辆和 M 个服务车辆）故障情况的组合。用 1 代表节点成功执行任务，0 代表节点执行任务过程中发生故障，因此所有节点的故障情况有 2^{M+2} 种组合，每一种组合代表着一种可能发生的故障情况。假设 RSU 和用户车辆 u 发生故障后可自行进行本地重计算，并假设 RSU 具有热备份冗余硬件，从发生故障到重新启动计算的时间可以忽略。用户车辆作为任务发起者，会对任务执行过程进行管理和监测，当它本身出现故障时，可以迅速重启，保证任务的持续执行。因此，在分析中可以忽略用户车辆的重启时间。

用集合 $\mathbb{S}$ 代表成功执行任务的服务车辆 s 的集合，即 $s\in\mathbb{S}$；用集合 $\mathbb{F}$ 代表执行任务中发生故障的服务车辆 $\mathcal{F}$ 的集合，即 $\mathcal{F}\in\mathbb{F}$。重计算开始后，原分配给故障车辆 $\mathcal{F}$ 的任务将由 RSU 进行重新划分并分发给服务车辆 $\mathbb{S}$ 进行重计算。由于节点发生故障的时间无法确定，在分析中可以保守地假设故障发生在节点完成任务的时刻，以考虑最差的情况。

因此，在进行重计算前，服务车辆计算任务经历的最大时延为：

$$t_\mathrm{total}^\mathrm{bef}=\max_{i\in\mathcal{M}=\mathbb{S}\notin\bigcup\mathbb{F}}\{t_i\} \tag{16}$$

此时整个计算卸载任务的系统总能耗为：

$$E_\mathrm{total}^\mathrm{bef}=\sum_{i\in\mathcal{M}}\left(E_i^\mathrm{exe}+E_{\mathrm{R},i}^\mathrm{off}\right)+E_u^\mathrm{exe}+E_{u,\mathrm{R}}^\mathrm{off}+E_\mathrm{R}^\mathrm{exe} \tag{17}$$

在 2^{M+2} 种故障情况组合中，假设发生了第 β 种故障情况，发生的概率如下，其中，Υ_u、Υ_RSU 分别表示用户车辆 u 和 RSU 在执行任务的完成情况（1 为成功执行，0 为发生故障）。

$$\mathrm{Pr}^\beta=\begin{cases}\mathrm{re}_u\mathrm{re}_\mathrm{R}\prod_{s\in\mathbb{S}}\mathrm{re}_s\prod_{\mathcal{F}\in\mathbb{F}}\mathrm{fa}_\mathcal{F},\ \Upsilon_u=1,\Upsilon_\mathrm{RSU}=1\\ \mathrm{re}_u\mathrm{fa}_\mathrm{R}\prod_{s\in\mathbb{S}}\mathrm{re}_s\prod_{\mathcal{F}\in\mathbb{F}}\mathrm{fa}_\mathcal{F},\ \Upsilon_u=1,\Upsilon_\mathrm{RSU}=0\\ \mathrm{fa}_u\mathrm{re}_\mathrm{R}\prod_{s\in\mathbb{S}}\mathrm{re}_s\prod_{\mathcal{F}\in\mathbb{F}}\mathrm{fa}_\mathcal{F},\ \Upsilon_u=0,\Upsilon_\mathrm{RSU}=1\\ \mathrm{fa}_u\mathrm{fa}_\mathrm{R}\prod_{s\in\mathbb{S}}\mathrm{re}_s\prod_{\mathcal{F}\in\mathbb{F}}\mathrm{fa}_\mathcal{F},\ \Upsilon_u=0,\Upsilon_\mathrm{RSU}=0\end{cases} \tag{18}$$

系统执行重计算机制时，因为服务车辆是移动的，需要对各服务车辆的当前位置进行重新计算。此时，服务车辆 i 和 RSU 之间的距离为：

$$\mathrm{di}_{\mathrm{R},i}^*=\sqrt{\left(x_i+v_i\cos(\vartheta_i)t_\mathrm{total}^\mathrm{bef}\right)^2+\left(y_i+v_i\sin(\vartheta_i)t_\mathrm{total}^\mathrm{bef}\right)^2} \tag{19}$$

此时，车辆 i 和 RSU 的链路传输速率为：

$$r_{\mathrm{R},i}^*=B_{\mathrm{R},i}\mathrm{lb}\left(1+\frac{P_\mathrm{R}\left|h_{\mathrm{R},i}\right|^2\mathrm{di}_{\mathrm{R},i}^{*\,-\alpha}}{N_0}\right) \tag{20}$$

定义 d_s^* 为重计算时分配给无故障服务车辆 s 的任务量，d_R^* 和 d_u^* 分别为 RSU 和车辆 u 需要重计算的任务量（为 0 或原任务量）。考虑再次传输所需时间和再次处理任务的时间，则重计算后第 β 种情况下所需总时延为：

$$T_\mathrm{total}^\beta=\max_{s\in\mathbb{S}}\left\{t_u+\frac{d_u^*\alpha_u}{f_u},t_\mathrm{R}+\frac{d_R^*\alpha_u}{f_\mathrm{R}},t_\mathrm{total}^\mathrm{bef}+\frac{d_s^*}{r_{\mathrm{R},s}^*}+\frac{d_s^*\alpha_u}{f_s}\right\} \tag{21}$$

因此，车辆 s 和车辆 $\mathcal{F}$ 在第 β 种情况下所产生的时延分别为：

$$T_s^{\beta}=\frac{\theta d}{r_{u,\mathrm{R}}}+\frac{d_s}{r_{\mathrm{R},s}}+\frac{d_s\alpha_u}{f_s}+\frac{d_s^*}{r_{\mathrm{R},s}^*}+\frac{d_s^*\alpha_u}{f_s} \tag{22}$$

$$T_{\mathcal{F}}^{\beta}=\frac{\theta d}{r_{u,\mathrm{R}}}+\frac{d_{\mathcal{F}}}{r_{\mathrm{R},\mathcal{F}}}+\frac{d_{\mathcal{F}}\alpha_u}{f_{\mathcal{F}}} \tag{23}$$

整个系统在第 β 种故障情况下完成重计算后的总能耗为：

$$E_{\mathrm{total}}^{\beta}=E_{\mathrm{total}}^{\mathrm{bef}}+\sum_{s\in\mathcal{S}}kf_s{}^2d_s^*\alpha_u+kf_u{}^2d_u^*\alpha_u+kf_{\mathrm{R}}{}^2d_{\mathrm{R}}^*\alpha_u \tag{24}$$

用 $\left\{p_1,\cdots,p_s,\cdots,p_{|\mathcal{S}|}\right\}$ 表示服务车辆 s 在执行重计算任务过程中产生的时延，包括传输和计算时延。在计算再次转发给车辆 s 的任务量时，为满足总任务的时延需求，则重发任务量应保证 $p_1=\cdots=p_s=\cdots=p_{|\mathcal{S}|}$。以下做简单证明。

假设给车辆 s 分配的任务可实现 $p_1=\cdots=p_s=\cdots=p_{|\mathcal{S}|}$，因为重计算过程中总时延为 $T^{\mathrm{retran}}=\max\left\{p_1,\cdots,p_s,\cdots,p_{|\mathcal{S}|}\right\}$，此时 $T^{\mathrm{retran}}=p_1$。如果分配给任意两个服务车辆的任务量发生相互变化（例如车辆 1 的一部分任务量 ξ 被分配到车辆 s），它们各自的时延也会改变，则重计算过程总时延变为：

$$T^{\mathrm{retran}}=\max\left\{p_1-O_1(\xi),\cdots,p_s+O_s(\xi),p_{|\mathcal{S}|}\right\}=p_s+O_s(\xi)>p_1,\forall\xi>0 \tag{25}$$

其中，$O_1(\xi)$、$O_s(\xi)$ 分别为车辆 1 和车辆 s 在计算任务量 ξ 时的时延函数。

1.5 问题描述

在考虑节点故障和满足任务时延要求的情况下，本文的优化目标是获得最优的任务卸载分配和卸载比例 $\mathcal{D}=(d_1,\cdots,d_M,d_{\mathrm{R}},\theta)$，以最小化系统能耗。因此，优化问题可描述为：

$$\min_{\mathcal{D}}E_{\mathrm{total}}=\sum_{\beta=1}^{2^{M+2}}E_{\mathrm{total}}^{\beta}\mathrm{Pr}^{\beta}$$

$$\begin{aligned}\text{s.t.}\quad &\mathrm{C1}:\sum_{\beta=1}^{2^{M+2}}T_{\mathrm{total}}^{\beta}\mathrm{Pr}^{\beta}\leqslant t^{\mathrm{tole}}\\ &\mathrm{C2}:\sum_{\beta=1}^{2^{M+2}}T_i^{\beta}\mathrm{Pr}^{\beta}\leqslant t_{i,\mathrm{R}}^{\mathrm{hold}},i\in\mathcal{M}\\ &\mathrm{C3}:d_{\mathrm{R}}+\sum_{i\in\mathcal{M}}d_i=\theta d\\ &\mathrm{C4}:d_{\mathrm{R}}^*+\sum_{s\in\mathcal{S}}d_s^*=d^*\\ &\mathrm{C5}:\theta\in[0,1]\end{aligned} \tag{26}$$

其中，约束 C1 表示在考虑了节点故障情况下的任务时延不能超过时延约束。约束 C2 表示任务车辆 i 在任务过程中不能超出 RSU 的覆盖范围。C3 表示卸载的总任务量应该等于分配给各节点任务量之和。C4 表示重计算时分配的任务量总和等于需要重计算的总任务量。C5 表示卸载比例的取值范围。

分布式程序可靠性（distributed program reliability，DPR）是指分布式计算系统（DCS）中含有分布式文件的程序成功运行的概率。通过尽可能多地分配任务给节点来提高 DCS 可靠性是一个 NP 难问题[20,22]。式（26）是以 DCS 可靠性为基础的一个问题，因此也是一个 NP 难问题，且具有非凸性。对于此类问题，利用群智能算法求解可在多项式时间内获得近似最优解，相比一般的搜索算法更高效。因此，本文采用麻雀搜索算法对以上优化问题求解，并将其命名为基于麻雀搜索算法的计算卸载方案（sparrow search based computation offloading scheme，S^2COS）。

2 基于麻雀搜索算法的计算卸载方案

在麻雀搜索算法中，麻雀种群分为发现者群体和加入者群体。选取群体中一定比例数量的位置较优的若干麻雀作为发现者群体，每一次的迭代寻优过程中都会更新发现者群体，种群中剩下的麻雀作为加入者群体，发现者负责全局搜索并为加入者提供觅食方向和区域，一旦发现者找到更好的位置，加入者便会向发现者方向飞去。在

每一轮迭代的后期，随机抽取一定数量的麻雀个体用作侦察预警，称作侦察者，对应麻雀在自然界中的反捕食行为，如果侦察者发现危险则放弃觅食飞向新的位置[23]。

2.1　种群的设定和更新

本文系统模型的优化目标是找到最优的计算卸载分配策略使得系统能耗最低。因此，本文用各节点所分配的任务量和卸载比例构成麻雀个体 g 在第 I 代的位置，即 $\mathcal{G}_g^I=\left(d_{1_g}^I,\cdots,d_{M_g}^I,d_{R_g}^I,\theta_g^I\right)$，则第 I 代由 n 只麻雀组成的麻雀种群可表示为：

$$\mathcal{G}(I)=\begin{bmatrix} d_{1_1}^I & \cdots & d_{M_1}^I & d_{R_1}^I & \theta_1^I \\ \vdots & & \vdots & \vdots & \vdots \\ d_{1_g}^I & \cdots & d_{M_g}^I & d_{R_g}^I & \theta_g^I \\ \vdots & & \vdots & \vdots & \vdots \\ d_{1_n}^I & \cdots & d_{M_n}^I & d_{R_n}^I & \theta_n^I \end{bmatrix} \tag{27}$$

发现者位置更新计算式如下：

$$\mathcal{G}_{g,j}^{I+1}=\begin{cases} \mathcal{G}_{g,j}^I\cdot\exp\left(-\dfrac{g}{a\cdot\mathrm{iter}_{\max}}\right), R_2<\mathrm{ST} \\ \mathcal{G}_{g,j}^I+Q\cdot\boldsymbol{L}, R_2\geqslant\mathrm{ST} \end{cases} \tag{28}$$

其中，$\mathcal{G}_{g,j}^I$ 表示种群中第 I 代第 g 只麻雀的 j 维位置，$j=1,\cdots,M+2$；$\mathrm{iter}_{\max}$ 为最大迭代次数；a 是 $(0,1)$ 内的一个均匀随机数；Q 为一个服从标准正态分布的随机数；$\boldsymbol{L}$ 表示每个元素都为 1 的 $1\times(M+2)$ 的矩阵；R_2 为预警值，范围为 $[0,1]$；ST 为安全阈值。

麻雀算法中一般将安全阈值设定为 $[0.5,1]$ 内的一个固定参数，当 $R_2<\mathrm{ST}$，发现者在区域内未发现捕食者，可继续向其他区域展开大范围搜索，反之，发现者意识危险存在随即发出警告，所有麻雀个体飞往其他安全区域。这种设定虽能发挥算法的全局搜索能力但无法发挥局部开发能力，可能会导致算法的过早收敛。为平衡全局与局部的能力，本文引入一种基于 Logistic 模型的自适应因子[24]，表达式为：

$$\psi(I)=\frac{1}{1+\mathrm{e}^{\frac{I}{\mathrm{iter}_{\max}\cdot b}+\mathrm{iter}_{\max}\cdot c}}+\omega \tag{29}$$

其中，$\left(\dfrac{I}{\mathrm{iter}_{\max}\cdot b}\right)$ 和 $(\mathrm{iter}_{\max}\cdot c)$ 分别为伸缩和平移因子，本文选取 $b=0.05$，$c=-0.14$。ω 用来设定自适应因子的上下界，范围是 $[0,1]$，本文取 0.05。

自适应因子乘上安全阈值 ST 用于调整 I 代的 ST，其非线性特征可使 ST 在迭代早期处于较大的取值范围，促使发现者尽可能进行大范围搜索，在迭代后期使 ST 处于较小的取值范围，发现者可在最优值附近进行局部开发，充分发挥算法的局部开发能力。

加入者位置更新计算式如下：

$$\mathcal{G}_{g,j}^{I+1}=\begin{cases} Q\cdot\exp\left(\dfrac{\mathcal{G}_{\mathrm{worst}}^I-\mathcal{G}_{g,j}^I}{g^2}\right), g>\dfrac{n}{2} \\ \mathcal{G}_P^{I+1}+\left|\mathcal{G}_{g,j}^I-\mathcal{G}_P^{I+1}\right|\cdot\boldsymbol{A}^+\cdot\boldsymbol{L}, g\leqslant\dfrac{n}{2} \end{cases} \tag{30}$$

其中，$\mathcal{G}_P^{I+1}$ 为发现者目前占据的最好位置，$\mathcal{G}_{\mathrm{worst}}^I$ 为种群中全局最差的位置，$\boldsymbol{A}$ 表示一个 $1\times(M+2)$ 的矩阵，矩阵元素随机赋值 1 或−1，且 $\boldsymbol{A}^+=\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}$。

侦察者的位置更新计算式如下：

$$\mathcal{G}_{g,j}^{I+1}=\begin{cases} \mathcal{G}_{\mathrm{best}}^I+\varpi\cdot\left|\mathcal{G}_{g,j}^I-\mathcal{G}_{\mathrm{best}}^I\right|, \mathcal{G}_{\mathrm{best}}^I\text{优于}\mathcal{G}_g^I \\ \mathcal{G}_{g,j}^I\ , \mathcal{G}_g^I\text{优于}\mathcal{G}_{\mathrm{best}}^I \\ \mathcal{G}_{g,j}^I+\left(\dfrac{\left|\mathcal{G}_{g,j}^I-\mathcal{G}_{\mathrm{worst}}^I\right|}{\left(F_g^I-F_w^I\right)+\epsilon}\right)\cdot K, \text{其他} \end{cases} \tag{31}$$

其中，$\mathcal{G}_{\mathrm{best}}^I$ 为全局最优位置，F_g^I、F_w^I 分别是当前个体的适应度值和当前全局最差的适应度值；ϖ 为一个服从标准正态分布的随机数，K 为 $[-1,1]$ 内的均匀随机数，ϵ 代表一个很小的常数，用来避免零除情况。

2.2　适应度函数的设定和约束处理

在本文中，适应度值定义为基于麻雀个体

所处位置所求得的系统能耗值，即 $F_g^I = F\left(\mathcal{G}_g^I\right)$。约束优化问题的求解存在等式约束、不等式约束、约束区域离散等挑战[25]。因此，本文引入约束违规值来处理约束 $V=(\upsilon_1,\cdots,\upsilon_g,\cdots,\upsilon_n)$，计算式如下：

$$\upsilon_g = \frac{\sum_{\Gamma=1}^{l} G_\Gamma(\mathcal{G}_g)}{l} \tag{32}$$

$$G_\Gamma(\mathcal{G}_g) = \max\left\{g_\Gamma(\mathcal{G}_g), 0\right\} \tag{33}$$

其中，$g_\Gamma(\mathcal{G}_g)$ 表示优化问题中 l 个不等式约束中第 Γ 个不等式约束关于解 $\mathcal{G}_g$ 的值，且 $g_\Gamma(\mathcal{G}_g) \leqslant 0$；$G_\Gamma(\mathcal{G}_g)$ 为第 g 个个体在 Γ 个不等式约束违规值；υ_g 表示第 g 个个体在 l 个不等式约束函数值总和的算术平均值，$\upsilon_g \neq 0$ 说明在 l 个不等式约束函数中至少有一个不满足。关于等式约束的处理，本文直接将每次更新的个体按照元素值的比例进行强约束，这样可以促使算法在可行域内搜索，加快算法的收敛。式（34）是评判个体优劣的式子，“$\gg$”代表优于。

$$\text{sort}(\mathcal{G}_g,\mathcal{G}_{g+1}) = \begin{cases} \mathcal{G}_g \gg \mathcal{G}_{g+1}, \upsilon_g = \upsilon_{g+1} = 0 \,\&\&\, F_g < F_{g+1} \\ \mathcal{G}_g \gg \mathcal{G}_{g+1}, \upsilon_g = 0 \,\&\&\, \upsilon_{g+1} \neq 0 \\ \mathcal{G}_g \gg \mathcal{G}_{g+1}, \upsilon_{g+1} > \upsilon_g > 0 \end{cases} \tag{34}$$

算法 1 中描述了 S^{2}COS 的具体流程。

算法 1 S^{2}COS 算法

输入 安全阈值 ST 的初始值，发现者群体占群体数量的比例 Θ，侦察者群体占群体数量的比例 Ω，n 只麻雀个体组成的群体

输出 $F(\mathcal{G}_{\text{best}})$，$\mathcal{G}_{\text{best}}$

for $g = 1:n$ do

随机生成一个符合要求的个体位置 $\mathcal{G}_g^0$，并计算 $F\left(\mathcal{G}_g^0\right)$

end for

选出群体中最优的个体作为当前全局最优 $\mathcal{G}_{\text{best}}^0$

while $I = 1:\text{iter}_{\max}$ do

$\text{ST} = \text{ST} \cdot \psi(I)$;

$R_2 = \text{rand}$;将群体中的个体由优到劣重新排序

for $g = 1:\Theta \cdot n$

使用式（28）对麻雀位置进行更新

end for

for $g = (\Theta \cdot n + 1):n$

使用式（30）对麻雀位置进行更新

end for

随机从群体中抽取 $(n \cdot \Omega)$ 个个体组成侦察者，并获得侦察者个体的索引集合 Ψ

for $g \in \Psi$

使用式（31）对麻雀位置进行更新

end for

获得当前群体的最优位置 $\mathcal{G}_{\text{best}}^I$，如果 $\mathcal{G}_{\text{best}}^I$ 优于 $\mathcal{G}_{\text{best}}^{I-1}$，则更新 $\mathcal{G}_{\text{best}}$

end while

return $F(\mathcal{G}_{\text{best}})$，$\mathcal{G}_{\text{best}}$

3 仿真与分析

系统仿真基于由单个用户车辆、单个 RSU 和多辆移动空闲车辆组成的如图 1 所示的 IoV 系统。参数设定参考文献[26-27]，并做了符合系统环境的参数调整，仿真参数及取值见表 1。除非有明确说明，否则仿真都使用参数的默认值。

为理解所提 S^{2}COS 的全面性能，考虑以下 S^{2}COS 简化应用方案并进行性能比较。

- 方案 1：用户车辆和服务车辆进行计算，RSU 只转发任务给服务车辆。
- 方案 2：RSU 和服务车辆进行计算，用户车辆卸载全部任务。
- 方案 3：用户车辆和 RSU 进行计算，不使用服务车辆的计算资源。

另外，将所提 S^{2}COS 和以下常用算法进行性能比较。

- 算法 1：文献[12]提出的优化能量效率的模拟退火算法（simulated annealing algorithm，SA）。
- 算法 2：文献[13]提出的基于遗传算法的卸载策略（genetic algorithm-based offloading strategy，GAO）。

表 1　仿真参数及取值

参数	取值
RSU 范围下服务车辆数目/辆	5
RSU 下发功率 P_{R} /W	1
任务车辆的上传发射功率 P_u /W	0.2
能耗系数 k	10^{-27}
上传带宽 $B_{u,\mathrm{R}}$ /MHz	3
下发带宽 $B_{\mathrm{R},i}$ /MHz	2
服务车辆的行驶方向 ϑ_i	[0°,360°]
服务车辆的车速 v_i /(m·s^{-1})	[15,20]
总任务时延限制 t^{tole} /s	0.6
总任务数据大小 d /KB	600
RSU 计算能力 f_{R} /GHz	6.2
任务发起车辆计算能力 f_u /GHz	2.3
服务车辆计算能力 f_i /GHz	[1.5,3]
复杂度 α_u /(cycle·byte^{-1})	7 000
RSU 通信半径 R /m	300
高斯白噪声 N_0 /dBm	−80

不同方案和不同任务量对能耗和时延的影响如图 2 所示。可以看出，所有方案的系统能耗都随着任务量的增加而增加，其中，方案 3 的能耗最大且无法满足时延要求。因为对于数据量大、时延要求高的任务而言，用户车辆和 RSU 的计算资源相对有限。S^{2}COS 和方案 2 均可满足任务时延要求，而 S^{2}COS 使用了用户车辆承担一部分计算任务，可以通过合理的任务分配进一步降低整个系统能耗。方案 1 的系统能耗虽然最小，但以牺牲时延要求为代价，不适合时延要求高的任务，如辅助驾驶、路径导航规划等应用。

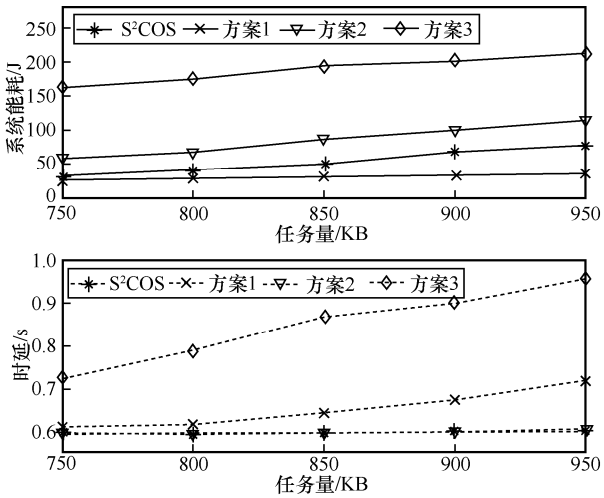


图 2　不同方案和不同任务量对能耗和时延的影响

不同的服务车辆数量和时延约束下，S^{2}COS 的不同服务车辆数和不同时延约束下能耗对比如图 3 所示，不同服务车辆数和不同时延约束下时延对比如图 4 所示。结果表明，随着服务车辆增多，系统能耗下降，且不同时延约束下的系统能耗性能差别逐步变小。这表明，随着服务车辆增多，系统会表现出更好的鲁棒性。这是因为更多的节点意味着更多的任务分配方式，因而可以进一步降低系统能耗。此外，从图 3 中可以看到，当服务车辆数增加到 6 辆以上时，对于时延约束相对宽松的 0.65 s 的情况，系统的能耗已经相差无几，这表明当服务车辆数达到一定数量后，系统能耗将难以进一步优化。因此，对于在不同时延约束、不同任务量的条件下，选择合适的服务车辆数可能是未来值得研究的。图 4 显示 S^{2}COS 在不同服务车辆数量时均可满足时延约束。

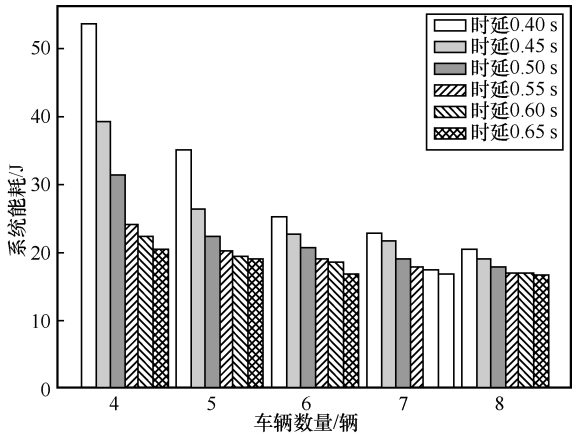


图 3　S^{2}COS 的不同服务车辆数和不同时延约束下能耗对比

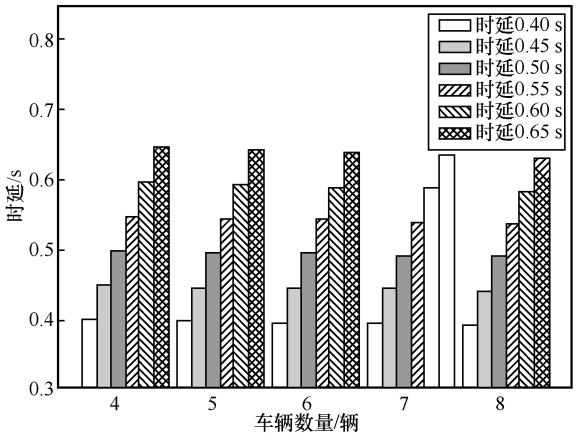


图 4　不同服务车辆数和不同时延约束下时延对比

为了评估重计算机制对系统性能的提升，本文对比了 S²COS 方案在重计算机制和无重计算机制时的性能。在无重计算机制中，所有故障节点的任务退还给用户车辆计算。重计算机制与无重计算机制的比较如图 5 所示，设定任务的时延约束为 0.55 s，随着总任务量增加，系统的能耗也随之增加，且考虑节点故障并进行重计算的系统的能耗要比没有重计算机制的系统显著降低。另外，无重计算的系统当任务量大于 800 KB 时无法满足时延约束，而有重计算的系统将此指标提升至 950 KB。这是因为，在无重计算的系统中，不成功的任务只能退还给用户车辆计算，而有重计算的系统会提前考虑节点故障情况并通过任务再分配尽可能降低系统能耗。

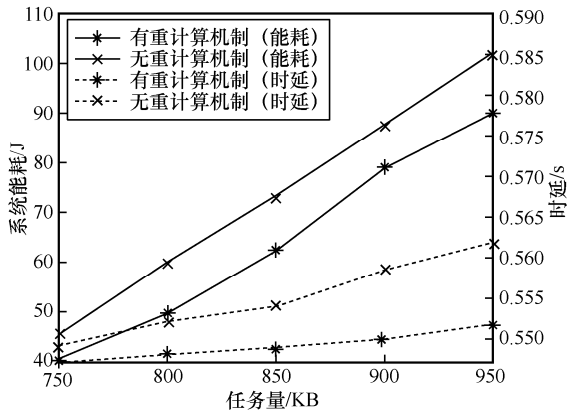


图 5　重计算机制与无重计算机制的比较

不同系统环境下有无重计算机制的对比如图 6 所示，进一步评估了不同节点故障率对于系统性能的影响，将任务量设定成 800 KB，复杂度设定为 7 500，时延约束设定 0.5 s。本文假设在每一个故障率区间内，故障率遵循均匀分布并将故障率大于 0.1 的区间定义为恶劣系统环境。由图 6 可知，系统的能耗会随着故障率的升高而升高。相比无重计算机制，重计算机制可显著降低系统能耗，并可满足时延要求，无重计算机制在平均故障率大于 0.2 s 时无法满足时延要求。这说明所提方案在恶劣系统环境下仍具有良好的鲁棒性。

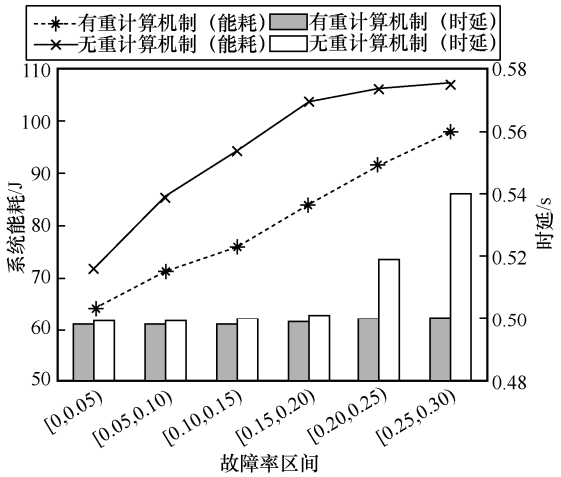


图 6　不同系统环境下有无重计算机制的对比

不同算法的系统能耗对比如图 7 所示，显示了 S²COS 和其他两个常用对比算法的系统能耗比较，任务复杂度设置为 6 500。可以看出，随着任务量的增加各算法所需系统能耗都会增加，所提方案相较于其他两种算法拥有更好的全局搜索最优解能力，在时延约束下，可找到更好的任务卸载比例和任务分配卸载决策以达到最低的系统能耗，避免了能量的浪费，有效降低系统能耗。

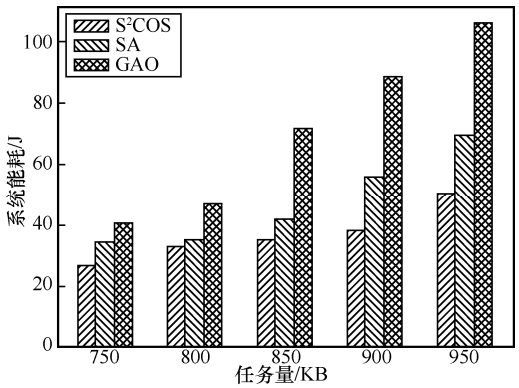


图 7　不同算法的系统能耗对比

4　结束语

本文针对一个由 RSU、用户车辆和多个服务车辆组成的车联网系统，提出了一种联合卸载方案 S^2COS。该方案引入 RSU 范围内服务车辆的计算资源，在满足任务时延要求的前提下，通过获得最优的任务分配方式最大限度地降低系统能耗。此外，所提出的方案考虑每种节点的故障情况，提出一种故障节点任务重计算机制。仿真结果表明，S^2COS 在满足时延要求和降低能耗方面相对于对比方案和其他算法具有明显优势，并且具备良好的优化效果。未来计划将工作扩展到多 RSU 小区场景，研究资源的分配和因车辆的移动性而产生的信道时变对系统的影响。

参考文献：

[1] ZHUANG W, YE Q, LYU F, et al. SDN/NFV-empowered future IoV with enhanced communication, computing, and caching[J]. Proceedings of the IEEE, 2019, 108(2): 274-291.

[2] 王鲲, 董振江, 杨凡, 等. 基于 C-V2X 的车路协同自动驾驶关键技术与应用[J]. 电信科学, 2023, 39(3): 45-60.
WANG K, DONG Z J, YANG F, et al. Key technologies and applications of C-V2X based vehicle-infrastructure cooperative autonomous driving[J]. Telecommunications Science, 2023, 39(3): 45-60.

[3] WANG J, FENG D, ZHANG S, et al. Computation offloading for mobile edge computing enabled vehicular networks[J]. IEEE Access, 2019(7): 62624-62632.

[4] LIU C, LI K, LI K. A game approach to multi-servers load balancing with load-dependent server availability consideration[J]. IEEE Transactions on Cloud Computing, 2021, 9(1): 1-13.

[5] FAN W, HAN J, YAO L, et al. Latency-energy optimization for joint Wi-Fi and cellular offloading in mobile edge computing networks[J]. Computer Networks, 2020(181): 107570.

[6] XU X, XUE Y, LI X, et al. A computation offloading method for edge computing with vehicle-to-everything[J]. IEEE Access, 2019(7): 131068-131077.

[7] REN C, ZHANG G, GU X, et al. Computing offloading in vehicular edge computing networks: full or partial offloading?[C]//Proceedings of 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC). Piscataway: IEEE Press, 2022: 693-698.

[8] DAI Y, XU D, MAHARJAN S, et al. Joint load balancing and offloading in vehicular edge computing and networks[J]. IEEE Internet of Things Journal, 2019, 6(3): 4377-4387.

[9] WANG H, LI X, JI H, et al. Federated offloading scheme to minimize latency in MEC-enabled vehicular networks[C]//Proceedings of 2018 IEEE Globecom Workshops (GC Wkshps). Piscataway: IEEE Press, 2018: 1-6.

[10] FAN W, LIU J, HUA M, et al. Joint task offloading and resource allocation for multi-access edge computing assisted by parked and moving vehicles[J]. IEEE Transactions on Vehicular Technology, 2022, 71(5): 5314-5330.

[11] ZHANG H, WANG Z, LIU K, et al. V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks[J]. China Communications, 2020, 17(5): 266-283.

[12] 王艳阁, 奚建清. 具有隐私保护的车联网边缘计算任务卸载资源分配策略[J]. 计算机工程与设计, 2023, 44(2): 372-378.
WANG Y G, XI J Q. Privacy preserving resource allocation strategy for edge computing task offloading in internet of vehicles[J]. Computer Engineering and Design, 2023, 44(2): 372-378.

[13] 余翔, 刘一勋, 石雪琴, 等. 车联网场景下的移动边缘计算卸载策略[J]. 计算机工程, 2020, 46(11): 29-34.
YU X, LIU Y X, SHI X Q, et al. Mobile edge computing offloading strategy under Internet of vehicles scenario[J]. Computer Engineering, 2020, 46(11): 29-34.

[14] 李雅丽, 王淑琴, 陈倩茹, 等. 若干新型群智能优化算法的对比研究[J]. 计算机工程与应用, 2020, 56(22): 1-12.
LI Y L, WANG S Q, CHEN Q R, et al. Comparative study of several new swarm intelligence optimization algorithms[J]. Computer Engineering and Applications, 2020, 56(22): 1-12.

[15] LIU J, WANG Y, ZHANG W, et al. A novel offloading and resource allocation scheme for time-critical tasks in heterogeneous internet of vehicles[C]//Proceedings of 2023 2nd International Conference for Innovation in Technology (INOCON). Piscataway: IEEE Press, 2023: 1-7.

[16] RAZA S, LIU W, AHMED M, et al. An efficient task offloading scheme in vehicular edge computing[J]. Journal of Cloud Computing, 2020(9): 1-14.

[17] GUO S T, XIAO B, YANG Y, et al. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing[C]//Proceedings of IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. Piscataway: IEEE Press, 2016: 1-9.

[18] WANG Y, SHENG M, WANG X, et al. Mobile-edge computing: partial computation offloading using dynamic voltage scaling[J]. IEEE Transactions on Communications, 2016, 64(10): 4268-4282.

[19] PLANK J S, ELWASIF W R. Experimental assessment of workstation failures and their impact on checkpointing systems[C]//Proceedings of Digest of Papers, Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing

(Cat. No. 98CB36224). Piscataway: IEEE Press, 1998: 48-57.

[20] VIDYARTHI D P, TRIPATHI A K. Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm[J]. Journal of Systems Architecture, 2001, 47(6): 549-554.

[21] LAWLESS J F. Statistical models and methods for lifetime data[M]. Hoboken: John Wiley & Sons, 2011.

[22] LIN M S, CHEN D J. The computational complexity of the reliability problem on distributed systems[J]. Information Processing Letters, 1997, 64(3): 143-147.

[23] 薛建凯. 一种新型的群智能优化技术的研究与应用[D]. 上海: 东华大学, 2020.
XUE J K. Research and application of a new swarm intelligence optimization technology[D]. Shanghai: Donghua University, 2020.

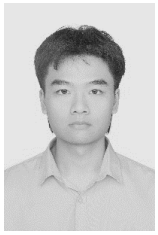
[24] 陈刚, 林东, 陈飞, 等. 基于 Logistic 回归麻雀算法的图像分割[J]. 北京航空航天大学学报, 2021, 49(3): 636-646.
CHEN G, LIN D, CHEN F, et al. Image segmentation based on logistic regression sparrow algorithm[J]. Journal of Beijing University of Aeronautics and Astronautics, 2021, 49(3): 636-646.

[25] WU G, MALLIPEDDI R, SUGANTHAN P N. Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization[R]. 2017.

[26] DINH T Q, TANG J, LA Q D, et al. Offloading in mobile edge computing: task allocation and computational frequency scaling[J]. IEEE Transactions on Communications, 2017, 65(8): 3571-3584.

[27] GUO H, LIU J, REN J, et al. Intelligent task offloading in vehicular edge computing networks[J]. IEEE Wireless Communications, 2020, 27(4): 126-132.

[作者简介]



高文轩（1998− ），男，宁波大学硕士生，主要研究方向为车联网系统的资源管理和性能优化。

杨新杰（1971− ），男，宁波大学信息科学与工程学院教授、硕士生导师，主要研究方向为下一代移动通信系统架构、移动物联网接入技术、协作中继网络性能等。