哈爾濱工業大學

本科毕业论文（设计）中期报告

题　目：车辆驾驶安全场景下基于微服务的任务卸载优化方法

专　　业　软件工程

学　　生　张儒

学　　号　2021112678

指导教师　贺祥

日　　期　2025 年 3 月 3 日

哈尔滨工业大学教务处制

目　录

一、论文工作是否按开题报告预定的内容及进度安排进行

本课题源于车联网企业的实际业务需求，目标是为车联网企业设计一种基于微服务的任务卸载方案。车联网企业为货车配备了具有传感器的车载设备，能够实时采集 4-6 路监控视频数据、遵循车辆发动机等标准协议的车辆状态传感器数据，以及驾驶员视频数据等。依据这些数据，来判断当前驾驶环境是否存在安全隐患，从而警醒驾驶员规避风险，为行车提供安全保障。

然而，上述任务大多属于计算密集型任务，如实时图像识别、目标检测等，且涉及的数据量庞大。同时，由于车辆行驶速度较快，驾驶环境瞬息万变，对任务结果的实时性有较高的要求。计算延迟会导致驾驶员无法及时获得警示，可能造成严重的财产损失甚至人员伤亡。但是受限于车载环境中的成本、空间、能耗等因素，车载设备的算力往往有限，难以支撑大规模的实时数据处理与分析，无法确保任务的准确性以及实时性。

为满足任务需求，任务卸载成为必然选择。云计算的出现满足了部分对实时性要求较为宽松的任务需求。但对于实时性要求较高的任务，受限于通信开销以及端到端延迟，传统的云计算方案难以满足此类应用需求。为应对上述挑战，移动边缘计算（MEC）应运而生，将部分计算任务卸载到靠近用户的边缘服务器上，能够显著降低通信开销以及端到端延迟，有效满足任务的实时性要求。

同时，微服务架构凭借其模块独立性、可扩展性、松耦合等特性，为车载系统的任务卸载需求提供了支持。传统的单体架构，服务之间紧密耦合，若要在服务器之间迁移特定模块，需要迁移整个单体。而微服务架构能够实时调整任务卸载策略，灵活调度计算资源。以微服务作为任务卸载方案的底层支撑，可以提供灵活的资源调度能力，使得系统在复杂的驾驶场景下始终保持高效性能和低延迟响应。

本课题旨在以微服务作为底层支撑，借助边缘计算技术，为车联网企业提供高效且有效的任务卸载方案，在保证服务质量的同时，最大化利用系统资源，降低企业成本。开题报告中工作进度安排如下：

工作安排	周数	起止时间
相关文献的研究和调研	2	2024.11.20-2024.12.04
系统模型建立	3	2024.12.05-2025.01.01
任务卸载算法设计	4	2025.01.01-2025.01.30
数据采集以及仿真实验	4	2025.02.01-2025.02.28
性能分析与优化验证	4	2025.03.01-2025.03.28
结题，论文编写	2	2025.03.29-2025.04.15

阶段目标安排如下：

（1）中期目标。完成系统模型（网络模型、任务模型、多计算层级协同的计算模型）的建立以及优化问题的形式化表达。

（2）结题目标。实现具体的任务卸载模型和算法，并进行仿真实验验证所设计的卸载策略性能以及经济效益。

目前仍在进行任务卸载算法设计阶段，但已完成中期目标。

二、已完成的研究工作及成果

2.1 已完成的研究工作

目前，已成功将现实问题进行了深入的数学建模以及精准的问题形式化。采用遗传算法 (GA)、Gubori 求解器对问题进行了尝试求解，并在不同的场景下对建模进行了验证。在实验支撑方面，开发了基于 TC 命令的网络带宽控制工具以及基于 JMeter 的微服务测试工具，为后续的实验验证奠定了坚实基础。

2.1.1 系统建模

为了模拟系统运行时状态的变化，将系统的运行时间划分为 h 个时隙 $\mathcal{T} = \{t_1, t_2, ..., t_h\}$，每个时隙的长度为 t，在同一个时隙内，系统的状态可以看作是不变的（比如汽车的位置、车辆与边缘服务器之间的传输速率），但多个时隙间是变化的。

系统中共有 i 辆车，记作 $L = \{L_1, L_2, , L_i\}$，L_l 表示为一个元组：(V_l, P_l, c_l)，汽车 l 速度 $V_l = (V_x, V_y)$，汽车在时隙 t 的位置表示为：$P_l(t) = (X_l(t), Y_l(t))$。汽车车载设备 l 的可用计算能力:c_l。边缘节点 RSU 配备有边缘计算服务器，表示为 $E = \{P_e, \gamma_e, c_e, \beta_e\}$，其中 RSU 的坐标为:$P_e = (X_e, Y_e)$，覆盖范围为:$\gamma_e$，所拥有的计算资源为：$c_e$，带宽资源：$\beta_e$。云服务器 $C = \{P_c, c_c, \rho_c\}$，位置 $P_c = (X_c, Y_c)$，云服务器计算资源丰富，分配给每个任务的计算能力相同，均为 c_c，采用按需付费每 CPU 周期的价格 ρ_c。

车载设备持续采集视频数据，被划分为固定时长为 κ 的视频块，每个视频块对应一个处理任务。也就是每隔 κ 的时间，车辆 l 就会产生一个任务 T_l，由多个含有依赖关系的子任务构成，表示为:$Q_l = \{T_l, \mathcal{E}_l\}$。其中任务 $T_l = \{A_l^1, A_l^2 .... A_l^k\}$，表示汽车 l 的 k 个子任务，每个子任务定义为：$A_l^m(x_l^m, d_l^m, w_l^m, c_l^m, t_{A_l^m}^{start}, t_{A_l^m}^{end})$，$x_l^m$ 表示任务的输入数据大小 (bit),d_l^m 表示任务的输出数据大小，w_l^m 表示任务的工作量，c_l^m 表示分配给任务 A_l^m 的计算资源，$t_{A_l^m}^{start}$ 表示任务 A_l^m 的开始时隙，$t_{A_l^m}^{end}$ 表示任务 A_l^m 的结束时隙。任务 T_l 的最后截止时间为 t_l，为防止任务积压，$t_l = \kappa$，在下一个视频块任务到来之前，当前任务要处理完成。S_l 表示任务 T_l 的卸载策略:$S_l = \{s_l^1, s_l^2, s_l^3, ..., s_l^k\}$ 其中 $s_l^m \in \{0, 1, 2\}$，分别表示本地、边缘服务器、云服务器

处理任务。$\mathcal{E}_l = \{e_{A_l^m, A_l^n} | (m,n) \in \{1,2,3,...,k\} \times \{1,2,3,...,k\}\}$ 表示汽车 l 的子任务之间的依赖关系。若 $\mathcal{E}_l$ 包含依赖关系 $e_{A_l^m, A_l^n}$，那么表示只有 A_l^m 完成之后，任务 A_l^n 才能开始执行。

如果子任务在本地进行处理，子任务的时延只包括处理时延：$t_{A_l^i}^{loc} = \frac{w_l^m}{c_l^m \cdot t}$。如果被卸载到边缘服务器，那么子任务卸载时延包括：任务上传时延、计算时延。上传数据延迟：$t_{up}^{edg} = \min\{t : \sum_{=t_{A_{l,t}^m}^{start}}^{t} r_{l,,e}^m t x_{l,t}^m\}$，计算延迟：$t_{A_l^m}^{edg} = \frac{w_l^m}{c_l^m \cdot t}$。卸载到云服务器时，时延包括任务上传时延、计算时延以及传播时延。上传数据延迟:$t_{up}^{clo} = \min\{t : \sum_{=t_{A_l^m}^{start}}^{t} r_{l,,e}^m t x_l^m\} + \frac{x_l^m}{r_{e,c}t}$。计算延迟：$t_{A_l^m}^{clo} = \frac{w_l^m}{c_l}$，往返的传播时延：$t_{e,c}^{tra} = 2 \times \frac{d_{e,c}}{\frac{2}{3}c} = \frac{3d_{e,c}}{c}$（其中 c 为光速）。任务 T_l 所需时间：最后一个子任务完成的时间：$T_l^{\text{delay}} = \max_m \left(t_{A_l^m}^{\text{end}}\right)$

企业按期租赁边缘服务器，价格记作 ξ。云服务器采用按需计费，计算成本：$\rho_c \times w_l^i$，边缘服务器到云服务器的通信成本: $\epsilon \times x_l^i$。总云服务器处理成本：$f_{l,i}^{clo} = \epsilon \times x_l^i + \rho_c \times w_l^i$。优化问题的目的就是在任务按时完成的前提下，最小化企业成本，问题形式化表达为：

$$
\begin{aligned}
& \min T_l^{cost} \\
\text{s.t.} \quad & t_{A_l^m}^{end} + 1 \le t_{A_l^n}^{start}, \quad \forall e_{A_l^m, A_l^n} \in \mathcal{E}_{\updownarrow} \\
& \max_{m \in \{1,2,...,k\}} \left(t_{A_l^m}^{\text{end}}\right) \le \kappa \\
& s_l^m \in \{0,1,2\}, \quad \forall m \in \{1,2,...,k\} \\
& \sum_{l=1}^{i} \sum_{m=1}^{k} I(s_l^m = 1) \cdot c_l^m \le c_e, \\
& \sum_{l=1}^{i} \sum_{m=1}^{k} [I(s_l^m = 1) + I(s_l^m = 2)] \cdot \beta_{l,e} \le \beta_e
\end{aligned}
$$

2.1.2 问题求解

针对上述优化问题，尝试采用 GA 算法以及 Gubori 求解器对问题进行求解。GA 算法在求解 NP-hard 混合整数非线性规划（MINLP）问题中具有显著优势，其核心在于其全局优化机制与适应性处理能力。MINLP 问题通常兼具整数变量、连续变量、非线性目标函数及复杂约束条件，传统梯度类算法易受限于局部最优解且难以处理离散变量，而遗传算法通过基于种群的并行搜索策略规避了这一问题：其通过模拟生物进化中的选择、交叉和变异操作，在解空间中多方向探索，避免对初始解或梯度信息的依赖；同时支持混合变量类型（如整数与连续变量共存），并能通过自适应罚函数、约束松弛等方法有效处理非光滑、非凸及离散约束。对于

NP-hard 问题中指数级增长的复杂度，遗传算法以启发式近似优化为核心，在有限时间内提供高质量可行解，兼具计算效率与工程实用性。

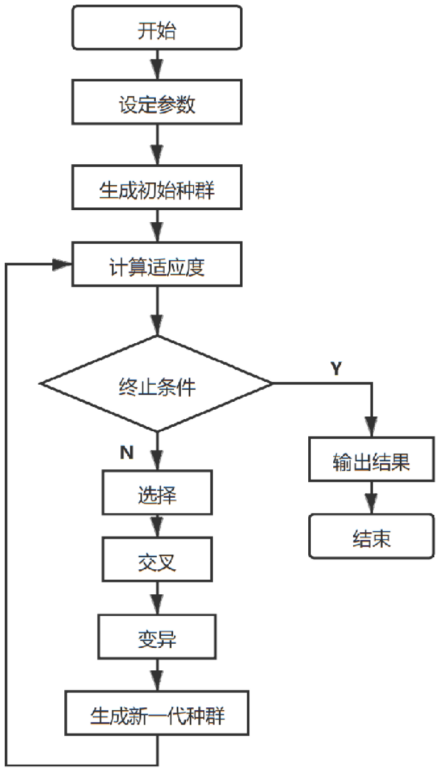

图 1: 遗传算法基本流程

在编码阶段，每个个体的编码包含了所有车辆的所有子任务的决策信息，其中每个子任务由三个连续的基因位表示：决策位置、计算资源分配量和带宽分配量。这种编码方式直观地反映了任务的调度方案。在生成初始种群时，采用智能资源分配策略，算法会根据 RSU 的资源状态动态调整决策概率。当检测到 RSU 的计算资源相对紧张时，会通过一定的权重来控制任务分配到本地、RSU 和云端的概率，有助于在初始阶段就避免 RSU 资源的过度使用。在 RSU 处理任务的资源分配过程中，需要同时兼顾计算资源与带宽的分配，一方面，通过设定资源的最小使用阈值，保障任务的基本运行需求；另一方面，确保资源使用总量不超过 RSU 的特定容量，避免出现资源的过多抢占。

遗传阶段的设计充分考虑了问题的特殊性。选择操作采用锦标赛选择方法，每次从种群中随机选择 3 个个体进行比较，选择适应度最好的个体，这种方式既保持了一定的选择压力，又避免了过早收敛；交叉操作以车辆为单位进行，对选中的父代个体，随机选择车辆的所有任务信息进行整体交换，这种设计保持了每辆

车任务决策的完整性；变异操作则分为决策变异和资源变异两个层面，当决策位置发生变异时（比如从 RSU 处理变为云处理），相应的资源分配也会随之调整，例如当任务从 RSU 迁移到本地处理时，计算资源和带宽都会被设置为 0，而迁移到云端时则只保留带宽分配，这种联动的变异机制确保了解的可行性。

另外，还采用了 Gurobi 求解器来尝试此问题。首先在决策变量设计上，通过三维决策矩阵来表示任务的卸载位置选择（本地、RSU 或云端），并配合资源分配矩阵来记录计算和带宽资源的具体分配方案。同时引入时间变量来追踪任务执行进度，这种设计既保证了模型的完整性，也便于后续约束条件的构建。在处理非线性约束方面，采用了辅助变量法将非线性约束转化为线性约束。在处理资源分配时，引入了时间槽离散化方法，将连续的时间域划分为离散的时间点，在每个时间点上分别考察资源使用情况，这种方法既保证了模型的可求解性，又能较好地反映实际系统的动态特性。对于任务执行顺序的约束处理，基于 DAG（有向无环图）表示任务的依赖关系，通过前驱任务完成时间和后继任务开始时间的关联来确保任务按照正确的顺序执行。同时，通过引入大 M 方法来处理二元变量与连续变量的耦合约束，有效解决了决策变量之间的复杂关联问题。求解完成后，通过专门的结果解析模块，将优化结果转化为具体的调度方案，包括每个任务的卸载位置、资源分配方案和执行时间安排等详细信息。

2.1.3 实验工具

在车联网场景中，V2I 通信的带宽是动态变化的。为验证通信模型对动态带宽的适应性，本研究开发了基于 Linux tc 命令的带宽控制工具。tc 命令是一个强大的流量控制工具，还能实现对指定网络包的速度、顺序以及出入网卡的精细调控。我们可以轻松地完成诸如按流量类型限速、限制特定应用程序或网络接口的带宽使用等任务。此外，它还能模拟网络延迟和丢包等故障情况，为系统开发提供有效的测试手段。本研究中可以用于模拟车联网场景中动态变化的网络环境并验证任务卸载策略的适应性，工具通过设定分层队列规则（如 HTB）限制指定节点的上行带宽，模拟不同条件下的网络负载状态。该工具为通信模型的可靠性提供了底层支撑，确保理论分析与实际网络行为的一致性。

基于 JMeter 扩展的微服务测试工具，可以模拟系统中车辆对边缘服务器发起任务卸载请求，模拟车辆集群的并发任务流。同时，为线程组设置定时器可以很好的模拟出流式任务的请求方式。另外，还可以梯度增加并发任务数量，观测边缘服务器在 CPU 过载、内存瓶颈等极端条件下任务卸载的响应策略，例如任务排队机制或向云服务器的动态迁移。压测结果指导了任务卸载和资源分配算法的优化方向，确保模型在复杂场景下的稳定性与扩展性。

2.2 成果

2.2.1 算法求解

在研究场景中，考虑单个 RSU 覆盖范围内的 3 辆货车，每辆货车配备了具有计算能力的智能车载设备，用于采集和处理车辆的多路视频数据。系统将时间划分为 20 个时隙，每个时隙长度为 0.1 秒，用于刻画系统状态的动态变化。每辆货车会持续采集视频数据，并将视频数据划分为固定时长 2 秒的视频块进行处理。每个视频块对应一个处理任务，该任务可以进一步分解为 4 个相互依赖的子任务，形成一个有向无环图 (DAG)，不同车辆的任务具有不同的 DAG 结构，比如采用链式结构，车辆 1 采用分支合并结构，车辆 2 则采用较为复杂的混合结构：

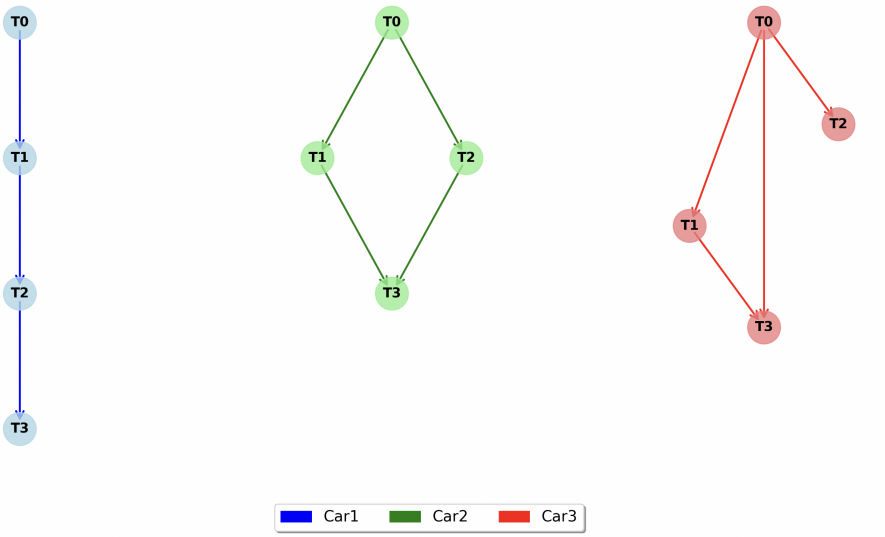

图 2: 子任务依赖关系图

每个子任务都有各自的的输入、输出数据大小和计算工作量。在计算资源方面，场景中部署了一个 RSU，位于道路两侧的位置，具有特定的计算能力和带宽资源。此外，还有一个位于远端的云服务器，提供丰富的计算能力。三辆车的初始位置和运动轨迹各不相同，但均处于 RSU 的覆盖范围内。每隔 2s，车辆生成一个视频块处理任务，向 RSU 的决策设备发送请求。决策设备需要在考虑任务依赖关系、资源容量约束、通信延迟等因素的基础上，为每个子任务确定最优的执行位置（本地、RSU 或云端）和资源分配方案，确保所有任务能在 2 秒的截止时间内完成（防止任务积压），同时最小化系统成本。采用 GA 算法对各个任务进行卸载决策，运行结果如下：

车辆	子任务	执行位置	计算资源	带宽	开始时间	结束时间	执行延迟
汽车1	T0	RSU	2154.77	392.97	0.00	0.67	0.67
汽车1	T1	云	-	1763.07	0.67	0.78	0.11
汽车1	T2	RSU	1877.03	1215.27	0.78	1.38	0.60
汽车1	T3	RSU	2851.56	1555.14	1.38	1.79	0.40
汽车2	T0	云	-	1361.96	0.00	0.15	0.15
汽车2	T1	RSU	2496.72	637.36	0.15	0.72	0.58
汽车2	T2	RSU	2698.31	1745.31	0.15	0.58	0.43
汽车2	T3	RSU	1799.96	1097.5	0.72	1.38	0.66
汽车3	T0	RSU	1756.88	589.88	0.00	0.76	0.76
汽车3	T1	RSU	2262.37	1777.29	0.76	1.26	0.51
汽车3	T2	RSU	1793.67	223.25	0.76	1.82	1.06
汽车3	T3	RSU	2690.57	570.96	1.26	1.83	0.57

总完成时间: 1.83 秒　截止时间: 2 秒　截止时间满足: 是　总成本: 418.05

图 3: GA 任务分配与详细执行时间

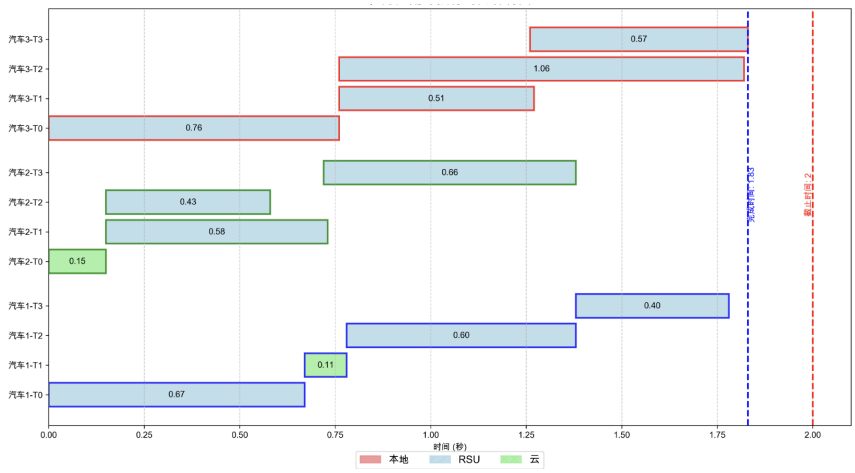

图 4: GA 任务分配与执行时间甘特图

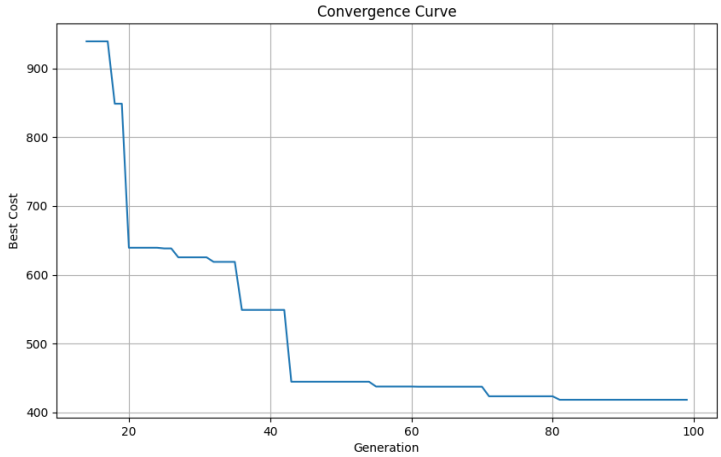

图 5: 适应度变化曲线

相同的场景，采用 Gubori 求解器进行卸载决策和资源分配，运行结果如下：

任务分配与执行时间详细表格

车辆	子任务	执行位置	计算资源	带宽	开始时间	结束时间	执行延迟
汽车1	T0	RSU	2501.2383	2000.0	0.00	0.53	0.53
汽车1	T1	RSU	10000.0	8000.0	0.95	1.26	0.32
汽车1	T2	RSU	5932.5711	8000.0	1.26	1.46	0.20
汽车1	T3	RSU	5938.2765	8000.0	1.58	1.78	0.20
汽车2	T0	RSU	3554.3544	3333.3333	0.00	0.32	0.32
汽车2	T1	RSU	5818.3843	5000.0	0.32	0.52	0.20
汽车2	T2	RSU	5884.1796	8000.0	0.43	0.63	0.20
汽车2	T3	RSU	5971.6783	8000.0	1.16	1.36	0.20
汽车3	T0	RSU	3201.2351	2500.0	0.00	0.40	0.40
汽车3	T1	RSU	5830.1813	8000.0	1.38	1.58	0.20
汽车3	T2	RSU	5232.4497	8000.0	0.53	0.74	0.21
汽车3	T3	RSU	5832.3873	8000.0	1.68	1.88	0.20

总完成时间: 1.8842 秒　截止时间: 2 秒　截止时间满足: 是　总成本: 100.0

图 6: 求解器任务分配与详细执行时间

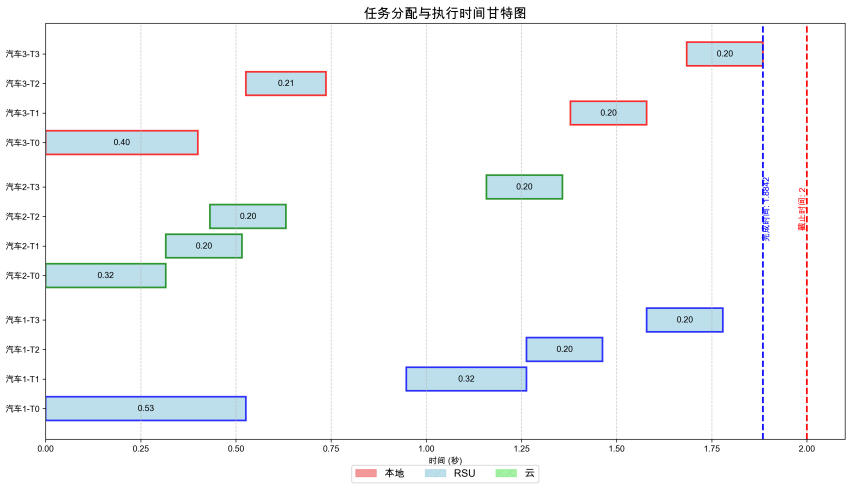

图 7: 求解器任务分配与执行时间甘特

可以看到 Gurobi 求解器和 GA 算法两者各有特点。Gurobi 求解器作为精确求解方法，能够找到全局最优解，从结果来看，其得到的总成本为 100，通过合理分配资源，将所有任务都分配到了边缘服务器处理。然而，求解器的计算时间会随着问题规模的增长呈指数级增长，在实际应用中无法满足实时性需求。相比之下，基于 GA 算法的启发式方法虽然得到的解（总成本 418）不如求解器，但其具有明显的计算效率优势，可以在较短的时间内求解。特别是在实际应用场景中，当车辆数量增加、任务依赖关系复杂化时，GA 算法仍能在可接受的时间内给出可行解，

这种快速响应能力更符合车联网环境下对实时性的要求。总的来说，求解器适合离线规划或小规模问题的精确求解，而遗传算法则更适合在线实时调度场景，特别是在大规模任务调度问题中，其计算效率的优势将更加明显。

2.2.2 工具演示

使用带宽控制工具控制车辆与 RSU 之间的带宽、延迟等参数：

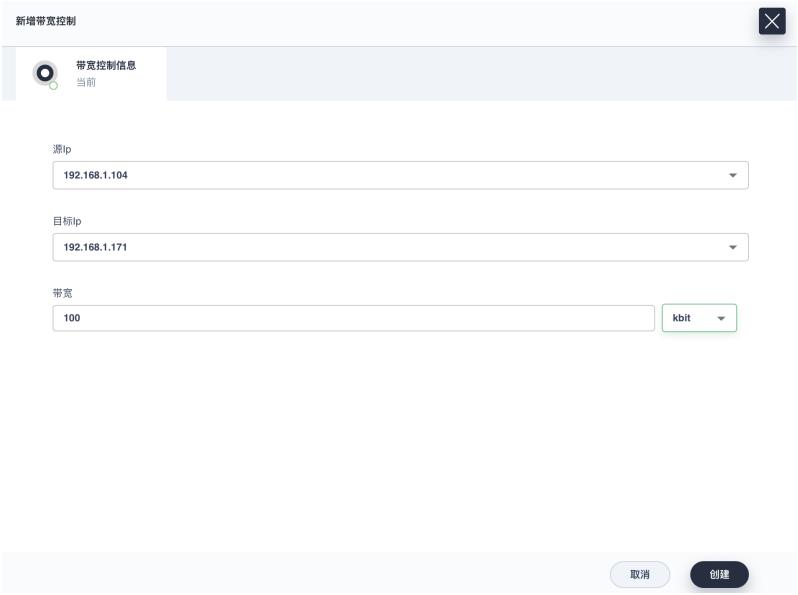

图 8: 控制两个设备之间的带宽

使用基于 JMeter 的测试工具来模拟车辆向 RSU 发出任务卸载请求：

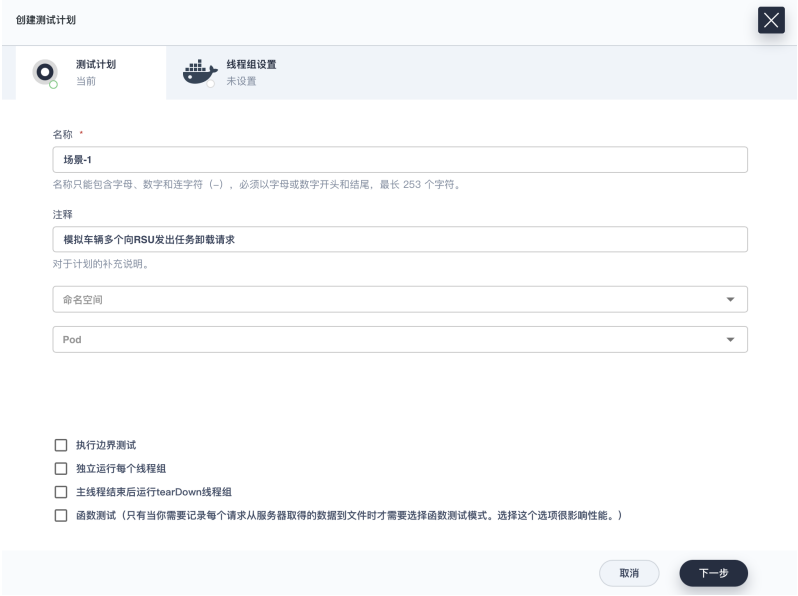

图 9: 添加测试计划，模拟任务卸载请求

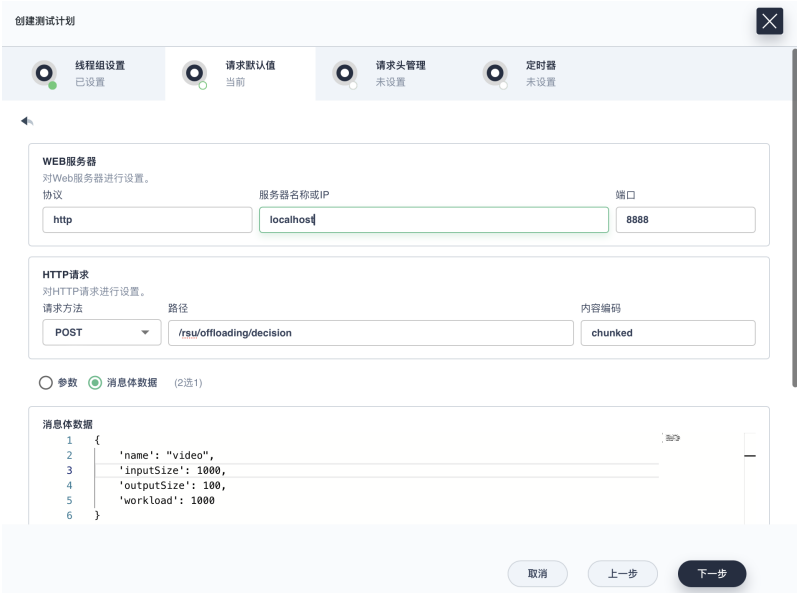

图 10: 设置请求参数

三、后期拟完成的研究工作及进度安排

3.1 后期工作

目前，在处理子任务之间的依赖关系时，我们采用了一种较为简单的拓扑排序方法，以此来确定任务的调度、处理顺序。但这种方法存在明显的局限性，它忽视了关键路径上可能存在的工作量繁重的任务。要是这些关键路径上的任务得不到优先处理，整个任务的完成时间就会受到影响。此外，未充分利用相邻子任务卸载位置的相关性，若存在前后具有依赖关系的两个子任务被卸载到同一侧的情况，那么后继任务便能省去传输时间与成本。基于上述两点，未来将研究一种依赖感知与关键路径协同优化的动态任务卸载机制。对于存在前后依赖关系的子任务，拟设计一种动态绑定策略，让卸载至同一节点的连续任务实现中间数据的本地化传递。同时，综合考量任务输出数据量、计算负载以及路径依赖深度等因素，构建多维度优化模型，优先调度、处理关键路径上的任务。

若时间充足，将在上述研究的基础上，进一步探索云边端分层混合式动态调度框架，通过引入任务优先级分类机制与多级资源协同分配策略，优化异构计算环境下的任务执行效率。具体而言，首先基于任务特性将任务划分为不同类型（比如实时型、带宽敏感型 和弹性计算型 ），并为每类任务赋予不同的优先级权重。其次，结合云边端三层的资源状态（如边缘节点算力、云端资源负载、终端设备能耗），设计动态分层调度引擎，优先将高优先级任务调度至满足其服务质量（QoS）的最优层级——例如，将关键路径上的实时任务绑定至低延迟边缘节点，而将非关键的大规模计算任务迁移至云端。

数据采集以及仿真实验环节，继续完善仿真实验平台，模拟场景中的复杂任务链和动态网络环境。在数据采集过程中，广泛收集不同场景下的车辆运行数据、网络状态数据以及任务执行数据，确保数据的多样性和完整性。利用数据挖掘技术，对采集到的数据进行深度分析，提取关键特征和规律，为后续的算法验证和性能评估提供坚实的数据基础。在仿真实验中，严格控制实验变量，全面模拟车联网实际运行中的各种复杂情况，包括车辆的高速移动、信号的干扰、任务的突发变化等，对设计的任务卸载算法进行全方位、多角度的验证，确保算法在实际应用中的可靠性和有效性。

性能分析与优化验证阶段，对任务卸载算法在不同场景下的性能进行全面、深入的分析。不仅关注算法的执行效率和资源利用率，还将重点评估算法在保障任务实时性、稳定性和可靠性方面的表现。通过与现有主流算法进行对比分析，明确算法的优势和不足之处，为进一步的优化提供方向。针对性能分析中发现的问题，深入研究优化策略，从算法结构、参数设置、资源分配等多个方面进行优化，提高算法的整体性能。

结题，论文编写阶段，梳理整个研究过程，将研究成果进行全面、准确的总结和提炼。在完成初稿后，广泛征求导师和同学意见，对论文进行修改和完善，确保论文的质量达到较高水平，能够按时、高质量地完成结题和论文编写任务。

3.2 进度安排

工作安排	周数	起止时间
任务卸载算法设计	3	2025.03.01-2025.03.22
数据采集以及仿真实验	4	2025.03.23-2025.04.21
性能分析与优化验证	3	2025.04.22-2025.05.14
结题，论文编写	1	2025.05.14-2025.05.21

四、存在的问题与困难

此优化问题归属于 NP - hard 的混合整数非线性规划（MINLP），求解复杂度高。随着车辆规模的扩大、时隙的细化，会导致计算复杂度的大大增加。目前采用遗传算法进行求解，在仅 3 辆车，一辆车 3 个子任务的规模下，收敛时间约为 1s。然而，实际场景可能需要在极短时间内（如 500ms）做出响应，以关键任务的顺利执行。因此，如何在如此有限的时间窗口内获取较优的可行解，成为当前算法设计工作中有待研究和解决的问题。

在车联网的实际运行环境中，车辆处于高速移动状态，这使得信道状态处于持续且快速的改变之中。而现有的系统模型在构建时，虽考虑了部分静态网络因素，但对于这种动态变化的网络环境适应性不足。这就导致在实际应用中，模型所预测的任务卸载性能与实际情况存在较大偏差，无法准确指导任务卸载决策，进而影响整个车联网系统的稳定性和可靠性。如何使系统模型能够实时、准确地适配动态网络环境的变化，是亟待解决的重要问题。

在多车多任务的复杂场景下，面临着双重耦合约束的严峻挑战。一方面，任务内部存在严格的时序依赖关系。另一方面，边缘服务器的资源有限，众多车辆的任务同时竞争这些资源。当多个任务同时请求边缘服务器的计算资源时，会出现资源争抢导致部分任务等待时间过长，进而影响任务的整体执行效率。如何在这两种耦合约束条件下，实现 DAG 任务依赖与资源竞争的协同优化，确保任务按照正确的顺序高效执行，同时合理分配有限的资源，是当前研究工作中的又一难点问题。

五、论文按时完成的可能性

目前，研究工作已取得了一系列关键进展，为论文的按时完成奠定了坚实基础。在系统建模方面，成功建立了通信模型、任务 DAG 模型与多级计算资源协同模型的三层系统架构。该架构全面且细致地刻画了车联网系统中数据传输、任务结构以及计算资源分配等关键要素之间的关系，为后续的算法设计和性能分析提供了准确的模型支撑。

在算法验证方面，实现了基于遗传算法的启发式求解框架与 Gurobi 数学规划求解双引擎验证机制。通过这两种不同类型的求解引擎，对任务卸载问题进行了深入的探索和验证。遗传算法凭借其模拟生物进化的全局搜索特性，能够在复杂的解空间中寻找较优解；而 Gurobi 求解器则在处理线性规划等问题时展现出高效性和准确性。双引擎机制相互补充，从不同角度验证了算法的有效性和可行性，为算法的进一步优化提供了丰富的数据和经验。

在实验工具搭建方面，搭建了包含 TC 网络仿真与基于 JMeter 的测试工具。基于 Linux TC 的网络仿真工具能够精准模拟车联网场景中动态变化的网络环境，包括不同的带宽分配策略、信道争用情况以及光纤速率波动等，为验证任务卸载策略在各种复杂网络条件下的适应性提供了有力手段。而基于 JMeter 扩展的微服务压测工具，则可以模拟多车并发任务卸载请求，通过构造 DAG 任务链并逐步增加并发任务数，记录边缘服务器的响应时间，从而深入分析服务器在资源约束下的多任务处理能力与算法鲁棒性。

综合以上已完成的核心建模工作、算法验证以及实验工具搭建等方面的成果，

论文按时完成具有很大的可能性。后续研究工作将在现有基础上，针对存在的问题与困难，有针对性地开展算法优化和实验验证工作，确保能够按照预定计划完成论文的撰写和相关研究任务。