端云协同的车载自组网分簇算法

马 玲,杨晓春*,王 斌,宋晓诗,李发明
(东北大学计算机科学与工程学院,辽宁沈阳 110169)

摘 要: 随着现代通信和信息技术的飞速发展,智能交通系统(Intelligent Transportation System,ITS)逐渐成为热门研究领域,车载自组网(Vehicular Ad Hoc Network,VANET)作为其关键技术,在实时道路信息共享和车辆间通信中起重要作用. 然而,现有VANET分簇算法仍存在簇稳定性低、分簇开销大等问题. 为解决这些问题,本文提出了一种端云协同的VANET分簇算法,在端云协同阶段,车辆通过路边单元(Road Side Unit,RSU)将自身特征数据上传至云,云侧根据特征变化,对车辆进行动态稳定性分类. 稳定的端节点具有更高的可靠性和更长的连接持续时间. 在端端协同阶段,考虑了稳定节点的相对移动性和覆盖节点数量等因素,进行簇头选举,简化簇头选举过程,提高了簇的稳定性. 此外,针对控制开销大的问题,本文提出了一种邻居发现和更新机制,限制HELLO消息的转发操作,降低开销并优化资源使用. 实验结果表明:本文提出的算法在簇稳定性、簇数量及分簇开销等关键性能指标上均优于基线算法,展示了其在实际交通场景中的应用潜力.

关键词: 车载自组网(VANET);多跳分簇;稳定节点;簇头选举;端云协同
基金项目: 国家自然科学基金(No.U22A2025,No.62232007,No.U23A20309);辽宁省自然科学基金(No.2023-BSBA-132)
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112(2025)02-0354-17
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240686

Clustering Algorithms for Vehicular Ad Hoc Networks Based on End-Cloud Collaboration

MA Ling, YANG Xiao-chun*, WANG Bin, SONG Xiao-shi, LI Fa-ming
(*School of Computer Science and Engineering*, *Northeastern University*, *Shenyang*, *Liaoning* 110169, *China*)

Abstract: With the advancement of modern communication and information technology, intelligent transportation systems(ITS) have emerged as a prominent area of research. The vehicular ad hoc network(VANET), serving as its pivotal technology, plays a crucial role in facilitating real-time road information sharing and inter-vehicle communication. However, existing clustering algorithms for VANET are plagued by issues such as low stability and high overhead. To address these challenges, this paper proposes a VANET clustering algorithm that leverages end-cloud collaboration. In the end-cloud collaboration phase, vehicles upload their feature data to the cloud via road side units(RSU), where the cloud performs dynamic stability classification based on changes in vehicle features. Nodes exhibiting stable behavior demonstrate higher reliability and longer connection durations. In the end-to-end coordination phase, factors including relative node mobility and cluster coverage are taken into account during cluster-head election to streamline the process while enhancing cluster stability. Furthermore, this paper introduces a neighbor discovery and update mechanism aimed at restricting HELLO message forwarding operations to reduce overhead and optimize resource utilization. Experimental results demonstrate that the proposed algorithm surpasses baseline algorithms across key performance metrics such as cluster stability, quantity of clusters formed, and clustering costs—highlighting its potential applicability in real-world traffic scenarios.

Key words: vehicular ad hoc network; multi-hop clustering; stable node; cluster head election; end-cloud collaboration

Foundation Item(s): National Natural Science Foundation of China (No.U22A2025, No.62232007, No.U23A20309); Joint Funds of Natural Science Foundation of Liaoning Province (No.2023-BSBA-132)

收稿日期:2024-07-22;修回日期:2025-03-06;责任编辑:孙瑶
*通讯作者:杨晓春

1　引言

近年来,随着无线通信技术、车载设备和道路基础设施的快速发展,智能交通系统(Intelligent Transportation System,ITS)[1]的研究受到了广泛关注. 作为ITS的重要组成部分,车载自组网(Vehicular Ad Hoc Network,VANET)在交通流监测与调度[2]、事故预警和处理[3]、车载导航以及车载娱乐等多个领域发挥着关键作用. VANET是移动自组织网络的一个特殊类型[4],节点为以不同速度移动的车辆. 车辆通过安装的车载单元(On Board Unit,OBU[5])和道路基础设施中的路边单元(Road Side Unit,RSU[5])进行数据的接收与发送,从而实现车辆之间及车辆与基础设施之间的通信.

由于VANET中车辆的高度移动性[6]以及频繁的网络拓扑变化[7],网络的稳定性和效率面临着巨大的挑战. 为了提高网络的性能,分簇算法被广泛应用,车辆被划分为若干个簇,由簇头管理簇内通信. 分簇机制能够有效减少通信开销[8]、提高网络的可扩展性[9],并简化路由管理. 在传统分簇算法中,车辆仅依赖邻近车辆的信息来选举簇头. 这种方法容易导致簇头频繁更换和簇结构不稳定[10],特别是在高动态环境下,车辆的快速移动和拓扑变化使得单靠局部信息[1]进行决策变得极为困难. 此外,过于依赖车辆端的信息传递会导致较大的控制开销,因为车辆必须频繁广播和转发HELLO消息[11],以更新邻居车辆信息,进而增加了网络的通信负担. 这些局限性使得传统分簇算法在应对复杂和动态的车载环境时显得不够可靠和高效.

针对上述问题,本文提出了一种端云协同的VANET分簇算法. 相较于仅依赖车辆端的分簇算法,本文通过引入云计算,实现了端云协同操作,充分利用了云侧的全局视角和强大计算能力. 具体而言,当车辆特征发生显著变化时,车辆会通过RSU将其特征数据上传至云,云侧通过全局分析识别出具有较长连接持续时间的稳定节点. 这些稳定节点在端端协同阶段被优先作为簇头候选,从而提高了簇稳定性. 本文主要贡献点总结如下:

(1)提出一种云侧稳定节点识别算法,在端云协同阶段,云利用从车辆上传的数据进行稳定性分析,识别出连接持续时间更长、可靠性更高的稳定节点. 在端端协同阶段,簇头节点被限制在稳定节点范围内,从而简化簇头选举过程并提高簇稳定性. 相比传统依赖局部信息的分簇算法,云侧的全局视角显著提高了簇的稳定性与选举效率.

(2)提出一种邻居发现和更新机制,非稳定节点无需转发HELLO消息,仅稳定节点需要转发最大跳数范围内其他稳定节点的消息. 该机制有效减少了网络中的控制消息数量,从而降低了分簇开销并优化了资源使用.

(3)在不同道路限速条件下的轨迹数据上的实验表明,本文提出的端云协同分簇算法在簇稳定性、簇数量及分簇开销等指标上均优于现有基线算法,验证了所提算法的有效性.

2　相关工作

本节总结了现有的VANET分簇算法,并将其分为五类:基于ID的分簇算法、基于节点度数的分簇算法、基于移动性的分簇算法、基于权值的分簇算法、基于智能优化算法的分簇算法.

2.1　基于ID的分簇算法

最小ID(Lowest-ID,LID[12])分簇算法于1995年提出,是VANET中的经典算法. 每个节点拥有一个唯一的ID,并周期性地广播包含其ID的HELLO消息. 节点生成一个包含其邻居ID的列表,如果节点仅接收到比自己ID高的节点信息,则它成为簇头;否则,最低ID的节点成为其簇头. 接收到多个簇头信息的节点将成为网关节点.

该算法简单易实现,但是存在显著局限. 车辆ID的地理分布通常是随机的,这可能导致簇头负载不均,在某些区域簇头密集,而在其他区域稀疏. 此外,由于ID是静态标识符,无法反映节点的实际移动性、度数、能量状态等重要信息. 因此,LID算法选出的簇头难以适应VANET的高度动态性环境. 车辆高度移动时,LID算法频繁进行簇头更换,导致簇稳定性较差.

2.2　基于节点度数的分簇算法

2002年提出的k-CONID算法[13]在LID算法的基础上加入了节点度数的考量. 节点度数被定义为节点k跳范围内的邻居数量. 每个节点与其k跳内的邻居进行竞争,其中,具有最高节点度数的节点被选为簇头. 若两个节点的度数相同,则ID较小的节点为簇头.

基于节点度数的分簇算法能够形成连通性高、规模较大的簇,有助于降低跨簇通信需求,提高簇内通信效率. 然而,由于缺乏簇成员数量上限的限制,在高密度区域易形成规模过大的簇,增加了簇内管理复杂性,对通信效率带来负面影响. 此外,该算法仅依据邻居数量选举簇头,忽略了节点的移动性特征,因此在网络拓扑结构频繁变化的环境下难以维持簇的稳定性.

2.3　基于移动性的分簇算法

MOBIC(Mobility Based Clustering)[14]是一种基于移动性的分簇算法,与基于ID或节点度数的分簇算法不同,MOBIC通过节点与其所有单跳邻居的相对移动性来选举簇头. 具体来说,节点通过比较来自同一节点的连续两条消息的信号强度变化,来衡量与单跳邻居的相对移动性,并计算出其相对所有单跳邻居的平均移动性. 在单跳邻居范围内,具有最小平均移动性的节点

被选为簇头. 如果多个节点的平均移动性相同,则选择ID较小的节点作为簇头.

文献[15]提出的Nhop算法引入了新的移动性度量,表示节点与多跳邻居之间的相对移动性. 节点利用连续收到的来自同一节点的消息延迟计算相对于多跳邻居的移动性. 在Nhop算法中,簇头的选举基于相对多跳邻居的平均移动性,与MOBIC算法相比,扩大了簇的规模. 然而,Nhop算法存在一定的缺点:当节点不属于任何簇时,它必须被动等待其他簇头的声明消息才能加入现有簇. 这会导致簇构建延迟,在车辆频繁改变位置的动态网络环境中,这种延迟可能会破坏通信效率和簇稳定性.

文献[16]提出的车辆多跳稳定聚类(Vehicular Multi-hop algorithm for Stable Clustering,VMaSC)算法利用节点相对多跳邻居的平均速度衡量其相对移动性,从而选出相对移动性最小的节点作为簇头,以延长簇头生命周期. 此外,节点可以主动从多跳邻居中选择一个相对移动性接近的簇头直接加入簇,而无需等待簇头声明. 这种主动的簇构建机制在VANET中具有更强的适应性,能够更好地应对网络拓扑的快速变化.

基于移动性的分簇算法考虑了车辆的移动性特征,更适合VANET的高移动性环境. 然而,此类算法仅以移动性为单一指标,未充分考虑簇数量和分簇开销的影响.

2.4 基于权值的分簇算法

在实际的VANET分簇中,簇头的选择往往受到多个因素的共同影响. 节点度数、移动性、剩余能量等单一指标难以全面反映节点适合作为簇头的潜力. 为此,基于权值的分簇算法应运而生,通过综合多个因素的影响,以加权方式计算每个节点的综合权值,权值最高或最低的节点被选为簇头,从而提升分簇的合理性和稳定性.

文献[17]提出的分布式多代理协调(Distributed Multi-Agent Coordination,DMAC)算法是一种基于单跳邻居权值的分簇算法. 在该算法中,每个节点会选择其单跳邻居中权值最大的节点作为簇头. 权值可以由节点度数、ID或能量值等指标决定. 此算法设计简单,由于仅依赖单跳信息,簇的规模和稳定性有限.

文献[18]对DMAC进行了改进,提出了多跳分簇算法(Multi-hop Distributed MAC,MDMAC). 在MDMAC中,节点发送的HELLO消息包含其ID、位置、速度、权值等信息. 节点在选择簇头时,不仅考虑簇头的权值,还会估计与簇头的可通信时长,这一改进有助于提高簇稳定性.

加权分簇算法(Weighted Clustering Algorithm,WCA)[19]引入了多因素加权的方法来选举簇头,综合考虑了节点的最佳度数、移动性、电池功率等因素,权值最小的节点被选为簇头. WCA通过预定义节点最佳度数实现负载均衡,但是每个节点需要知道网络中其他所有节点的权值,因此簇头选举过程消耗大量时间和通信开销.

基于权重的自适应聚类(Weighted-Based Adaptive Clustering,WBACA)[20]算法则改进了WCA,通过基于局部最小权值的方式选举簇头,避免了每个节点收集网络全局节点信息的需求. 节点权值的计算考虑了传输功率、速率、移动性、电池功率和节点度数多种因素.

WECA-MR[21]算法在传统加权度量(如节点度数和平均距离)的基础上,引入了结合相对速度和相对加速度的移动性度量,进一步提高了对动态环境的适应性. 此外,WECA-MR为每个簇配置了备份簇头,减少了簇头失效带来的影响,从而延长了簇头生命周期.

文献[22]提出了一种基于模糊C均值的车辆分簇算法,以最小化车辆的功耗为目标. 相较于现有的基于权值的分簇算法,该算法通过加权评分方法更全面地衡量簇头稳定性,综合考虑熵、加权移动性和相对位置. 熵衡量局部网络稳定性,熵较高表明邻居位置变化较小,有助于选择稳定簇头;加权移动性衡量邻居变动情况,值高表示邻居不稳定,不适合作为簇头;相对位置用于判断车辆靠近或远离,确保簇头与簇成员保持合理分布. 该方法相比于依赖单一移动性或连接度的加权策略,在提高簇稳定性方面更具有优势.

基于权值的算法综合考虑多个影响因素,灵活适应网络状态变化,提升网络稳定性并降低分簇重组频率. 然而,确保算法有效性和高效性的关键在于如何选择和平衡这些权值因素. 尤其在缺乏全局信息时,算法难以对不同因素的权重进行合理调控,从而影响分簇效果.

2.5 基于智能优化算法的分簇算法

基于智能优化算法的VANET分簇算法近年来受到广泛关注. 这类算法通过全局优化技术[如粒子群优化[23]、Harris Hawks优化(Harris Hawks Optimization,HHO)[24]、Gray Wolf优化[25]、遗传算法[26]等]来优化簇头的选择和簇结构.

文献[27]提出的综合学习粒子群优化(Comprehensive Learning Particle Swarm Optimizer,CLPSO)算法是一种基于粒子群优化的分簇算法. 问题空间由一群粒子组成,每个粒子被编码为一组簇头,代表一个候选解,然后使用适应度函数进行评估以确定最优解,即优化的簇头. CLPSO旨在找到最优的簇数量,以有效管理网络资源.

基于灰狼优化的聚类网络(Grey Wolf Optimization-based Clustering Network,GWOCNET)算法[28]和基于哈里斯鹰优化的聚类网络(Harris Hawks Optimization-

based Clustering Network,HHOCNET)算法[29]都是通过模拟自然界中动物狩猎行为来优化分簇.GWOCNET算法是一种基于灰狼优化的VANET分簇算法,该算法利用灰狼的社会行为和狩猎机制创建具有优化的簇数量的分簇结果.

HHOCNET算法利用HHO策略,通过模仿哈里斯鹰的群体狩猎行为,优化簇头的选择过程.在该算法中,每只鹰代表一组簇头,以实现两个优化目标:一是最小化簇头与其簇成员之间的总距离;二是使得每个簇头的簇成员数量接近预设的最佳值.HHOCNET算法通过不断执行HHO算法,来动态调整簇结构,从而获得每个时刻的分簇结果.

文献[30]提出的GACC(Genetic Algorithm-based Coalitional Clustering)算法是一种基于联盟博弈论的遗传算法,旨在在满足数据传输延迟约束的前提下最大化簇的规模.该算法通过计算簇内所有节点的效用和,引导节点根据合理的效用分配动态调整簇结构.联盟博弈论[31]用于建模节点和簇的利润与成本,使得簇头在权衡信道容量分配和网络覆盖范围扩展的情况下决策是否接受新成员.

这类全局优化算法基于网络中所有节点的信息,使其能够实现分簇优化目标的全局最优解.然而,为了达到此效果,所有车辆需要定期上传信息到云侧进行全局分簇优化.这种周期性上传增加了网络的通信负担,特别是在大规模网络中,数据上传频率过高会导致带宽占用和延迟问题.此外,这类基于智能优化算法的分簇算法主要关注的是簇数量优化,而未充分考虑簇的稳定性.

3　端云协同的VANET架构

本节首先提出端云协同的VANET架构,端侧提取车辆特征并上传至云,云通过这些特征将车辆识别为稳定或非稳定节点,并将识别结果返回给车辆,进而为后续分簇提供支持.接着介绍端侧车辆特征选择,最后提出云侧稳定节点识别算法.

3.1　系统建模

端云协同的VANET架构由车辆、RSU和云组成,如图1所示.当车辆进入道路后,会定期与周围车辆通信,获取邻居节点的信息并计算自身特征.车辆将当前特征与之前时间段的特征进行比较,若发生显著变化,车辆通过RSU将更新的特征上传至云.云侧在接收到车辆特征数据后,迅速进行处理,以识别该车辆是否为稳定节点,并将识别结果以低延迟方式广播给RSU,由其通知车辆.这种高效的端云协同机制确保了车辆能够在较短时间内获得分类结果.

相较于仅依赖车端的架构,端云协同的VANET架构具有独特的优势.首先,云侧能够基于大范围、长时间、多维度的特征数据进行全局化分析,使得节点稳定性的评估更全面和精准.其次,云侧计算资源丰富,能够确保稳定节点识别算法的快速执行,减轻车端设备的计算负担.云侧的全局视角优化了节点的稳定性识别,为车载网络的整体稳定性提供可靠支持.

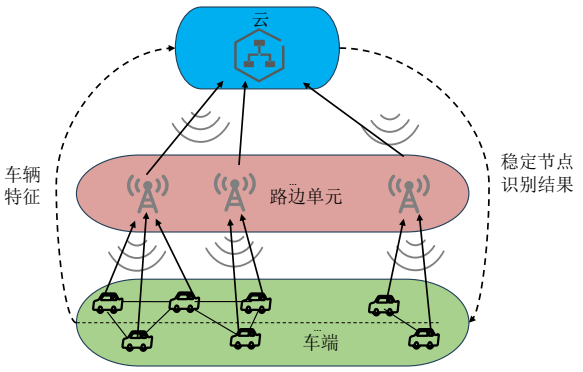


图1　端云协同的VANET架构示意图

稳定节点在VANET中具有更高的可靠性和更长的连接持续时间,因此更适合作为簇头.簇头是负责管理和协调簇内通信的关键节点,选择稳定节点作为簇头可以提高整个网络的性能和稳定性.本文将簇头限制在稳定节点的范围内,确保只有稳定节点才有资格成为簇头,从而增强了簇的可靠性和稳定性.

3.2　端侧车辆特征选择

车辆在道路上的运行稳定性决定了VANET的稳定性.本文考虑了多种因素作为刻画车辆运行模式的特征:相对平均速度(v_{rel})、车辆类型(type)、变道频率(f_{lane})、平均速度(v_{mean})、平均加速度(a_{mean}),进而将道路上行驶的车辆划分为稳定节点和非稳定节点.这种分类方法可以更精确地理解和预测车辆在网络中的行为表现,进而为后续的网络分簇和优化提供支持.

(1)相对平均速度.相对平均速度是指车辆相对于邻居节点的平均速度.研究表明,驾驶行为较为稳定的车辆通常会跟随周围车辆的行驶速度,保持较小的相对速度差[32].在VANET的分簇算法中,具有较小相对平均速度的车辆通常表现出更长的连接时间[33],并且不容易脱离簇,更适合作为簇头节点[34].

(2)车辆类型.不同类型的车辆(如轿车、货车、客车等)在道路上的运行特性有所不同.车辆类型对加速性能、惯性和制动能力有一定影响,它能在某种程度上反映车辆在道路上的行驶方式[35].

(3)变道频率.变道行为直接影响车辆的轨迹稳定性和邻居的连通性[36].频繁变道的车辆会不断进入和离开簇的通信范围,导致簇内车辆间的连接频繁断开和重建,这会降低簇的稳定性.因此,低变道频率的车辆更有可能是网络中的稳定节点[37].

(4)平均速度. 平均速度能够有效地反映车辆的驾驶习惯. 相对平稳的驾驶行为通常表现为较低且稳定的平均速度,而激进的驾驶行为则往往表现为较高且波动较大的平均速度[38]. 车辆的平均速度能够在一定程度上揭示其驾驶行为的激烈程度,进而影响其在VANET中的稳定性.

(5)平均加速度. 车辆的平均加速度可以反映其驾驶行为,较小的加速度变化通常意味着更平稳的驾驶方式[39].

在端侧车辆特征选择中,本文考虑了多种影响车辆稳定性和网络连接性的因素,选择了相对平均速度、车辆类型、变道频率、平均速度、平均加速度等特征. 这些特征能够较全面地反映车辆的行驶状态与行为模式,有助于提高稳定节点的识别准确性. 在数据上传时,根据特征变化的显著性来决定是否上传,以减少不必要的通信负担.

3.3 云侧稳定节点识别算法

在云侧稳定节点识别过程中,本文采用了随机森林(Random Forest,RF)模型[40]对稳定节点进行识别. RF是一种集成学习方法,具有较高的分类精度、良好的鲁棒性和较强的泛化能力. 与单一的决策树相比,RF通过集成多棵决策树来降低过拟合风险,并且能够在处理高维数据和复杂特征时保持稳定的性能[40]. 此外,RF对异常值和噪声具有较强的抗干扰能力,能够更准确地识别稳定节点. 同时,RF支持并行计算,适用于云侧高效处理大规模数据,从而满足VANET对低时延的需求.

RF通过多个决策树的集成进行分类,每棵树在训练时使用数据的不同子集,并在特征选择上引入随机性,从而增强模型的泛化能力并降低过拟合风险. 对于每一棵决策树,分类过程通常是通过一系列的特征划分来实现的. 对于给定的输入特征,每一棵决策树根据特征的不同阈值进行决策,最终输出一个类别标签,其中"1"表示稳定节点,"0"表示非稳定节点.

在RF中,最终的分类决策由所有决策树的投票结果决定. 若有T棵决策树,那么RF的输出类别$\hat{y}$可以通过式(1)进行计算:

$$\hat{y}=\text{majority\_vote}(\{y_1,y_2,\cdots,y_T\}) \tag{1}$$

其中,$\{y_1,y_2,\cdots,y_T\}$表示每一棵决策树的分类结果;majority_vote表示投票机制,即选择出现次数最多的类别作为最终预测结果.

本文输入向量$\boldsymbol{x}=(v_{\text{rel}},\ \text{type},\ f_{\text{lane}},\ v_{\text{mean}},\ a_{\text{mean}})$将作为决策树的输入,用于判断车辆是否为稳定节点. 基于这些特征,每一棵决策树会从根节点开始,通过一系列的判断条件(例如,平均速度大于某个阈值)逐层划分,最终到达叶节点,输出分类结果. 所有决策树的预测结果将通过投票机制得到最终的稳定节点分类.

4 端端协同的多跳簇构建和更新

在第3节提到的端云协同VANET架构中,云侧将稳定节点识别结果返回给车辆,车辆在得知自身是稳定或非稳定节点后,开始构建多跳簇. 针对稳定节点和非稳定节点不同的运行模式,本节设计了相应的邻居发现和更新机制以及簇构建和更新算法.

4.1 邻居发现和更新机制

邻居发现和更新的目标是使每辆车能够在任何时刻动态地感知周围车辆,节点根据掌握的邻居信息可以实现自适应的端到端分簇. 在单跳分簇算法中,节点仅需识别单跳范围内的邻居;而在多跳分簇算法中,需要获取更广泛的多跳范围内的邻居信息. 由于车辆具有高度移动性,邻居节点的组成可能会频繁变化,这要求每个节点能实时感知新邻居的加入与现有邻居的离开.

在当前的VANET多跳分簇算法中,邻居发现主要依赖广播和转发机制,节点需要维护一个包含邻居节点信息的多跳邻居列表. 为了维护这个列表,每个节点需要定期广播HELLO消息并转发其他节点的HELLO消息. 当节点收到含有发送节点信息的HELLO消息时,它会首先更新自己的邻居列表,并检查消息当前经过的跳数. 如果当前跳数小于最大设定值,节点会增加跳数并转发消息;否则,将直接丢弃消息,不再进行转发. 然而,频繁地广播和转发HELLO消息会导致网络中充斥着大量的HELLO消息,链路层存在争用和碰撞,引发广播风暴,从而影响邻居发现过程的效率.

为解决上述问题,本文引入了一种改进的VANET多跳邻居发现与更新机制,其核心在于通过减少转发HELLO消息的数量来降低通信开销. 在此机制下,每个节点周期性地广播HELLO消息. 非稳定节点发送的HELLO消息包含其ID、簇状态、簇头、位置、速度、消息经过的跳数、是否为稳定节点、跟随的稳定节点、消息发送时间. 稳定节点发送的HELLO消息包含其ID、簇状态、簇头、位置、速度、消息经过的跳数、是否为稳定节点、跟随者数量、覆盖节点数量、平均移动性、消息发送时间.

每个节点维护一个单跳邻居列表NBHD(Neighborhood),图2展示了单跳邻居列表的字段. id(发送节点ID)、state(簇状态)、CH_id(节点的簇头)、pos(节点位置)、vel(节点速度)、$\text{PktDelay}_{\text{old}}$和$\text{PktDelay}_{\text{new}}$(来自同一发送节点的两条连续的HELLO消息的延迟)、isStable(发送节点是否为稳定节点)、follower_number(跟随者数量,仅稳定节点)、myStable(跟随的稳定节点,仅非稳定节点)、timestamp(消息发送时间).

稳定节点额外维护一个多跳稳定邻居列表

id	state	CH_id	pos	vel	$\text{PktDelay}_{\text{old}}$	$\text{PktDelay}_{\text{new}}$	isStable	follower_number	myStable	timestamp

图2 单跳邻居列表NBHD字段示意图

STALINK,记录最大跳数范围内其他稳定节点的信息,如图3所示,STALINK包含的字段中.id、state、CH_id、pos、vel、hop2me、follower_number、cover_number、AVGREL、timestamp分别表示发送节点的ID、簇状态、簇头、位置、速度、多跳稳定邻居距离自身的跳数、跟随者数量、覆盖节点数量、平均移动性和消息发送时间.

id	state	CH_id	pos	vel	hop2me	follower_number	cover_number	AVGREL	timestamp

图3　多跳稳定邻居列表STALINK字段示意图

为了减少控制消息数量、减少分簇开销,只有稳定节点需要转发来自其他稳定节点的HELLO消息.当非稳定节点接收到HELLO消息,若当前跳数为0,表明发送节点与接收节点相距1跳,故需更新单跳邻居列表NBHD.若当前跳数大于0,表明发送节点与接收节点相距多跳,此时非稳定节点不做任何操作,无需转发消息.稳定节点在接收HELLO消息时,确认消息是否来自单跳范围内的非稳定节点,若是,则更新NBHD列表;若消息来自其他稳定节点,则将更新STALINK列表,记录多跳范围内的稳定节点.在更新STALINK列表后,稳定节点会根据HELLO消息中的当前跳数字段,决定是否转发该消息.预设的最大跳数是指稳定节点距离簇头的最大跳数,如果当前跳数字段值加1小于最大跳数,则转发该消息,否则将直接丢弃该消息,不再进行转发.

节点定期检查每个邻居节点的活跃性,对于每个邻居节点,在检查活跃性时,计算当前时间与最后收到HELLO消息的时间戳(timestamp)之间的时间差.对于单跳邻居列表NBHD,若时间差超过阈值$V_{\text{invalid}}^{\text{NBHD}}$,则判定该邻居节点已经失效,需要从单跳邻居列表中删除.对于多跳邻居列表STALINK,若时间差超过阈值$V_{\text{invalid}}^{\text{STALINK}}$,则从多跳稳定邻居列表中移除相关条目.

4.2　簇构建算法

簇状态定义了每辆车在网络中的功能角色,车辆通过改变这一状态来调整其在簇中的角色,且每辆车在任意时间点都会处于其中一种状态.

(1)初始化(Initialization,IN)状态

刚进入道路的车辆处于此状态,开始发送和接收HELLO消息以建立通信连接.在此状态下,节点维护单跳邻居列表NBHD,并通过RSU向云上传车辆特征,云侧将这些特征送入分类器,识别节点为稳定节点还是非稳定节点.预测结果通过RSU反馈给车辆.

(2)选举(Election,EL)状态

EL状态是一种过渡状态,节点在此状态中根据获取的邻居信息来决定其下一状态.

(3)簇头(Cluster Head,CH)状态

处于CH状态的车辆作为簇的领导节点,负责管理簇内成员.

(4)簇成员(Cluster Member,CM)状态

节点在此状态下加入并成为已存在簇的一部分.

4.2.1　稳定节点的簇构建算法

处于IN状态的车辆在收到RSU返回的稳定节点识别结果后转入EL状态.若该车辆的识别结果为稳定节点,则遵循算法1的步骤来构建簇.首先检查其多跳稳定邻居列表STALINK中是否存在簇头节点.如果存在多个簇头,将根据特定优先级规则进行选择,优先选择距离跳数最小的簇头,若跳数相同,选择平均移动性较低的簇头(2~16行).选定簇头后,节点加入该簇并转换为簇成员状态(31~32行).若STALINK中无簇头节点,稳定节点评估自身是否符合成为簇头的条件(18~25行).若符合,则转为CH状态(26~29行),否则重新执行算法1.

稳定节点在检查自己是否满足成为簇头的条件时,需要考虑三个指标.

(1)覆盖节点数量

稳定节点的覆盖节点数量可以描述为节点自身加上其多跳稳定邻居的数量、多跳稳定邻居的跟随者数量以及节点自身的跟随者数量之和.在图4中,黄色节点表示稳定节点,其中有四个稳定节点,分别为s_1~s_4.蓝色节点表示非稳定节点,总共有10个非稳定节点,编号为n_1~n_{10}.箭头表示跟随关系,例如稳定节点s_1的跟随者有n_1、n_6、n_9.表1展示了最大跳数为2时,图4中每个稳定节点的多跳邻居列表、跟随者数量以及覆盖节点数量.稳定节点的覆盖节点数量可以理解为其成为簇头后,预计簇中的节点个数,可以评估每个稳定节点所能够维护的簇的规模.选举覆盖节点数量较多的稳定节点为簇头能够扩大簇规模,减少簇头数量,从而提高VANET系统的效率.

(2)相对于跟随者的平均速度

稳定节点相对于跟随者的平均速度是衡量与其跟随者连接稳定性的重要指标,稳定节点i相对于跟随者的平均速度计算式为

$$V_i^f=\frac{\sum \Phi_i}{\text{fn}_i\cdot\max\left\{\Phi_i\right\}} \tag{2}$$

其中,fn_i表示稳定节点i的跟随者数量;Φ_i表示节点i与其所有跟随者的速度差的集合,见式(3);归一化因子为Φ_i的最大值.

$$\Phi_i=\left\{\left|\left|v_i-v_j\right|\right| j\in \text{NBHD}(i)\wedge j\to i\right\} \tag{3}$$

非稳定节点频繁更换跟随的稳定节点会导致簇内连接不稳定,增加通信开销并可能改变簇隶属关系,从而降低簇的稳定性.相对跟随者的平均速度较小意味着稳定节点与其跟随者之间的相对位置变化小,从而使得这种跟随关系更可能长期持续.选择相对跟随者的平均速度较小的稳定节点作为簇头,可以减少非稳

算法1 稳定节点的簇构建算法

输入:稳定节点集合S

输出:更新后的稳定节点集合S

```
1. for each s in S do
2.  min_hop2me = MAX_HOP + 1
3.  min_AVGREL = INFINITY
4.  tmp_ch = −1
5.  for each v in s.STALINK do
6.    if v.state == CH then
7.      if v.hop2me < min_hop2me then
8.        min _hop2me = v.hop2me
9.        tmp_ch = v.id
10.       min_AVGREL = v.AVGREL
11.     else if v.hop2me == min_hop2me and v.AVGREL <
    min_AVGREL then
12.       min_AVGREL = v.AVGREL
13.       tmp_ch = v.id
14.     end if
15.   end if
16. end for
17. if tmp_ch == −1 then
18.   beCH = true
19.   for each v in s.STALINK do
20.     if v.cover_number > s.cover_number then
21.       beCH = false
22.     else if v.cover_number == s.cover_number and
    v.AVGREL < s.AVGREL
23.       beCH = false
24.     end if
25.   end for
26.   if beCH then
27.     s.CH_id = id
28.     s.state = CH
29.   else
30.     Call Algorithm 1
31.   end if
32. else
33.   s.CH_id = tmp_ch
34.   s.state = CM
35. end if
36. end for
```

定节点更换跟随节点的次数,降低重新隶属的频率,从而增强簇的稳定性.

(3)相对于稳定邻居的平均速度

相对于稳定邻居的平均速度是一个关键指标,用于衡量稳定节点与其多跳稳定邻居之间的连接稳定性. 稳定节点i相对于其稳定邻居节点的平均速度计算式为

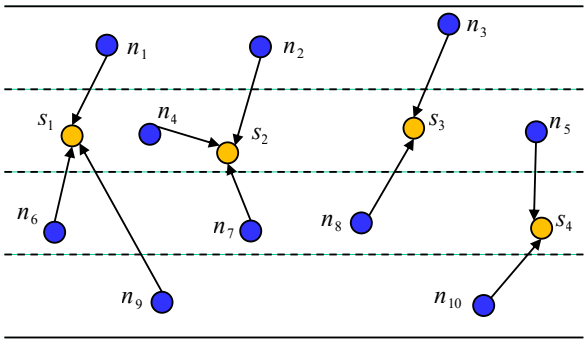


图4 覆盖节点数量计算示意图

表1 稳定节点的多跳邻居列表、跟随者数量、覆盖节点数量

稳定节点	多跳邻居列表	跟随者数量	覆盖节点数量
s_1	s_2, s_3	3	11
s_2	s_1, s_4	3	14
s_3	s_1, s_4	2	14
s_4	s_2, s_3	2	10

$$V_i^s = \frac{\sum \Psi_i}{\left|\mathrm{STALINK}_i\right| \cdot \max\left\{\Psi_i\right\}} \tag{4}$$

其中,$\left|\mathrm{STALINK}_i\right|$表示节点$i$的多跳稳定邻居列表的节点数量;归一化因子是$\Psi_i$的最大值. Ψ_i表示节点i与其所有多跳稳定邻居的速度差的集合,即

$$\Psi_i = \left\{\left|v_i - v_j\right| \;\middle|\; j \in \mathrm{STALINK}_i\right\} \tag{5}$$

当一个稳定节点成为簇头时,其多跳稳定邻居列表中的稳定节点以及这些稳定节点的跟随者都将被纳入该簇. 选取相对稳定邻居的平均速度较小的稳定节点作为簇头能够提升簇的稳定性. 因为较小的平均速度意味着稳定节点与邻居之间的相对位置变化小,这种关系更可能持续较长时间. 通过选取这样的稳定节点作为簇头,可以减少位置变化,降低簇内连接的不稳定性,提高整个簇的稳定性,使节点之间能够更可靠和稳定地通信和协作.

稳定节点广播的HELLO消息包括平均移动性(AVGREL)和覆盖节点数量(cover_number). 平均移动性是相对于跟随者和相对于稳定邻居的平均速度之和,取值范围为[0,2]. 稳定节点通过比较邻居节点的这两个指标来选择簇头,满足以下条件的稳定节点可以成为簇头:其覆盖节点数量应该不小于其他稳定邻居的覆盖节点数量,并且其平均移动性不大于具有相同覆盖节点数量的其他稳定邻居. 这种簇头选举策略有助于扩大簇的规模并增强其稳定性.

4.2.2 非稳定节点的簇构建算法

处于EL状态的非稳定节点尝试通过跟随其单跳邻居中的稳定节点加入现有簇,如算法2所示. 非稳定节点首先识别未处于EL状态的稳定节点作为候选跟

随节点. 为避免稳定节点跟随者过多,设定了最大跟随者数量MAX_FOLLOWER. 选择候选跟随者时,确保其跟随者数量加一不会超过此限制(2~7行). 接着,非稳定节点从候选集中选择所希望跟随的稳定节点(8~9行). 选择跟随的稳定节点主要基于以下三个因素.

算法2　非稳定节点的簇构建算法

输入:非稳定节点集合S'

输出:更新后的非稳定节点集合S'

1. for each $s' \in S'$ do
2. STA = { }
3. for each v in s'.NBHD do
4. if v.isStable and v.state != EL and v.follow_number + 1 ≤ MAX_FOLLOWER then
5. STA.add(v)
6. end if
7. end for
8. if len(STA) != 0 then
9. Choose $s \in$ STA according to stable node following strategy
10. else
11. Choose $v \in s'$.NBHD with the smallest $|\mathrm{RelM}_{s',v}|$
12. Set v.myStable as s to follow
13. end if
14. if $\mathrm{FOLLOW}_{\mathrm{TIMER}}$ ends then
15. Send FOLLOW_REQ
16. Start a $\mathrm{FOLLOW}_{\mathrm{TIMER}}$
17. Wait for FOLLOW_RESP
18. end if
19. if FOLLOW_RESP is received and $\mathrm{FOLLOW}_{\mathrm{TIMER}}$ does not expire then
20. s'.myStable = s.id
21. s'.CH_id = s.CH_id
22. s'.state = CM
23. End the $\mathrm{FOLLOW}_{\mathrm{TIMER}}$
24. end if
25. end for

(1)相对移动性

本文采用了Nhop算法中描述的方法,通过HELLO消息的传播延迟来计算相对移动性[15],即

$$\mathrm{RelM}_{x,y} = 10\lg\frac{\mathrm{PktDelay}_{x,y}^{\mathrm{new}}}{\mathrm{PktDelay}_{x,y}^{\mathrm{old}}} \tag{6}$$

其中,$\mathrm{PktDelay}_{x,y}^{\mathrm{new}}$和$\mathrm{PktDelay}_{x,y}^{\mathrm{old}}$分别表示节点$x$接收到来自节点$y$的两条连续HELLO消息的延迟. 相对移动性如果为正值,表示两个节点在相互远离,否则表示两个节点在相互靠近.

(2)当前跟随者数量

车辆倾向于选择拥有较多跟随者的稳定节点,式(7)表示节点x选择跟随节点y时,基于跟随者数量的收益.

$$\mathrm{benefit}_{\mathrm{fn}_{x,y}} = \frac{\mathrm{fn}_y}{\sum_{i \in \mathrm{NBHD}(x) \wedge \mathrm{isStable}} \mathrm{fn}_i} \tag{7}$$

其中,$i \in \mathrm{NBHD}(x) \wedge \mathrm{isStable}$表示节点$i$是节点$x$的单跳邻居中的稳定节点;$\mathrm{fn}_i$表示节点$i$的跟随者数量.

(3)历史归属信息

车辆倾向于跟随与其历史归属簇相同的稳定节点,这有助于提高簇的稳定性. 节点x跟随节点y获得的历史归属信息的益处,计算式为

$$\mathrm{benifit}_{\mathrm{CH}_{x,y}} = \begin{cases} 0.8, & \mathrm{CH}_x^{\mathrm{his}} = \mathrm{CH}_y^{\mathrm{cur}} \wedge \mathrm{CH}_x^{\mathrm{his}} \neq \mathrm{null} \\ 0.2, & \mathrm{CH}_x^{\mathrm{his}} \neq \mathrm{CH}_y^{\mathrm{cur}} \wedge \mathrm{CH}_x^{\mathrm{his}} \neq \mathrm{null} \\ 0, & \mathrm{CH}_x^{\mathrm{his}} = \mathrm{null} \end{cases} \tag{8}$$

其中,$\mathrm{CH}_x^{\mathrm{his}}$表示节点$x$上一时刻的簇头;$\mathrm{CH}_y^{\mathrm{cur}}$表示节点$y$当前的簇头. 稳定节点$y$目前所属的簇如果与节点$x$上一时刻所属簇相同,则节点$x$跟随节点$y$获得的基于历史信息的收益较大.

根据上述三个因素,稳定节点跟随策略可以有效选择最合适的稳定节点作为跟随目标,计算式为

$$\begin{aligned} F_x = & \mathop{\mathrm{argmin}}_{y \in \mathrm{STA}(x)} \mathrm{RelM}_{x,y} \\ & \times \left(1 - \mathrm{sgn}\left(\mathrm{RelM}_{x,y}\right) \cdot \mathrm{benefit}_{\mathrm{fn}_{x,y}}\right) \\ & \times \left(1 - \mathrm{sgn}\left(\mathrm{RelM}_{x,y}\right) \cdot \mathrm{benefit}_{\mathrm{CH}_{x,y}}\right) \end{aligned} \tag{9}$$

其中,F_x表示节点x选择的稳定节点;$\mathrm{sgn}\left(\mathrm{RelM}_{x,y}\right)$表示相对移动性的符号,两个相互靠近的节点更有可能保持在一个簇中. 综合这些因素,非稳定节点能够更智能地选择合适的的稳定节点跟随.

对于非稳定节点的单跳邻居列表中缺少稳定节点的情况,节点可以选择单跳邻居中与自己相对移动性较小的非稳定节点,与该非稳定节点跟随相同的稳定节点(11~12行),从而保持簇结构的稳定性.

选择完希望跟随的稳定节点后,若非稳定节点此时没有在等待其他稳定节点的回复,则会发送跟随请求消息FOLLOW_REQ,该消息包含发送节点和其想要跟随的节点(即接收节点)的ID,并启动一个计时器$\mathrm{FOLLOW}_{\mathrm{TIMER}}$来等待回复(9~13行). 若在$\mathrm{FOLLOW}_{\mathrm{TIMER}}$结束前收到回复,则非稳定节点会将自身的簇ID更新为稳定节点的簇头ID,转变为簇成员状态,并终止$\mathrm{FOLLOW}_{\mathrm{TIMER}}$(14~18行). 若$\mathrm{FOLLOW}_{\mathrm{TIMER}}$结束时未收到回复,则节点将继续保持EL状态. FOLLOW_RESP消息包含发送节点及其要回复的节点(即接收节点)的ID.

$\mathrm{FOLLOW}_{\mathrm{TIMER}}$的计算式为

$$\mathrm{FOLLOW}_{\mathrm{TIMER}} = \frac{2R}{C} + \frac{\mathrm{Size}_{\mathrm{req}} + \mathrm{Size}_{\mathrm{resp}}}{C_r} \tag{10}$$

其中，R表示消息最大传输范围；C表示无线信号的传输速度；Size_{req}和Size_{resp}分别表示FOLLOW_REQ和FOLLOW_RESP消息的大小；C_r表示DSRC节点的数据传输速率.

4.3 簇更新算法

在VANET中，由于车辆移动的动态特性，已经形成的簇可能会频繁发生变化. 为了保证网络的稳定性和通信的高效性，簇成员和簇头都需要不断适应网络结构的变动，因此需要设计专门的更新算法. 本节分别介绍簇成员和簇头节点的簇更新算法.

4.3.1 簇成员更新算法

簇成员更新算法主要用于确定簇成员是否需要脱离当前簇或加入新的簇，如算法3所示. 对于处于簇成员状态的稳定节点，检查其是否存在电量低于阈值或硬件故障的情况. 若存在，则该节点将立即切换为EL状态，并在失效前广播一次HELLO消息，通知其跟随者及时重新选择新的稳定节点(4~6行).

在未检测到失效的情况下，该稳定节点将进一步检查其多跳稳定邻居列表STALINK中是否包含当前簇头的条目(8~12行). 若没有与簇头相关的条目，则说明该节点已经与簇头断开连接，因此需要重新进行簇选择. 同样地，若发现簇头的状态已变为EL状态，则表明当前簇头无法继续担任簇头角色，无论是由于节点故障，还是因拓扑变化，均使其不再符合簇头的要求. 此时，该稳定节点也应该脱离当前簇，切换回EL状态，并按照算法1重新构建簇(13~16行).

对于处于簇成员状态的非稳定节点，需要进行以下检查. 首先，检查单跳邻居列表NBHD，确认是否存在所跟随的稳定节点(19~23行). 若不存在，则说明与其稳定节点之间的连接已经中断，因此需要回到EL状态. 若存在，但该稳定节点的状态是EL状态，则表示其与簇头节点断开连接. 因此，非稳定点节点也与簇头节点断开连接，需要切换回EL状态，遵循算法2重新加入某个簇(24~27行).

4.3.2 簇头更新算法

簇头更新算法确保在必要时能及时选举新的簇头，如算法4所示. 当簇头节点检查到电量低于阈值或硬件故障预警时，簇头将状态切换为EL状态(1~3行). 此时，它会在失效前广播一次HELLO消息，通知簇成员该状态变化(第4行). 这样簇成员可以立即启动选举流程，无需等待簇头节点完全失效和超时检测，减少簇头失效对网络稳定性的影响.

若簇头没有检测到失效预警，则会进一步检查其覆盖节点数量. 当覆盖节点数量降至1时，意味着簇头已与其他簇成员断开连接，仅剩簇头本身，簇头会切换

算法3 簇成员更新算法

输入:簇成员集合M
输出:更新后的簇成员集合M

```
1. for each m ∈ M do
2.    found = false
3.    if m.isStable then
4.       if hardware abnormality detected or battery power below the
threshold then
5.          m.state = EL
6.          Broadcast HELLO Message
7.       else
8.          for each v in m.STALINK do
9.             if v.id == m.CH_id and v.state! = EL then
10.               found = true
11.            end if
12.         end for
13.         if not found then
14.            m.state = EL
15.            Call Algorithm 1
16.         end if
17.      end if
18.   else
19.      for each v in m.NBHD do
20.         if v.id == m.myStable and v.state! = EL then
21.            found = true
22.         end if
23.      end for
24.      if not found then
25.         m.state = EL
26.         Call Algorithm 2
27.      end if
28.   end if
29. end for
```

算法4 簇头更新算法

输入:簇头集合C
输出:更新后的簇头集合C

```
1. for each c ∈ C do
2.    if hardware abnormality detected or battery power below the
threshold then
3.       c.state = EL
4.       Broadcast HELLO Message
5.    else
6.       if c.cover_number == 1 then
7.          c.state = EL
8.       end if
9.    end if
10. end for
```

至EL状态以确保网络的连通性(6~8行).

5　实验结果与分析

本节对本文提出的VANET多跳分簇算法进行实验测试与分析,设计的实验包括:稳定节点和非稳定节点分类测试、簇稳定性测试、簇数量测试、分簇开销分析.

5.1　实验设置

本文选择了在VANET领域应用广泛的SUMO(Simulation of Urban MObility)[41]软件生成的轨迹数据来验证本文提出的方案,本小节介绍数据集的具体信息和实验环境配置方案.

5.1.1　数据集

本文设计的VANET多跳分簇算法运行在NS3(ns-3.38)平台上,SUMO用于生成车辆轨迹数据,详细的仿真参数如表2所示. 在仿真实验环境中,设置了5组不同的限速(15 m/s、20 m/s、25 m/s、30 m/s和35 m/s),以评估本文提出的分簇算法在不同交通环境动态性下的适应性. 车辆总数设定为300辆,并将模拟时间设定为400 s,以确保实验结果的统计显著性.

表2　仿真参数

参数	值
仿真时间	400 s
道路长度	10 082.25 m
道路限速	15~35 m/s
车辆数量	300
传输范围	100~300 m
传输速率	27 Mbit/s
传播模型	Two-ray ground model
信道协议	MAC/802.11p
HELLO消息发送周期	250 ms
数据包发送周期	1 s

车辆的最大传输范围设置为100 m、200 m和300 m,旨在考察算法在不同通信条件下的表现. 选择在VANET中广泛应用的MAC/802.11p作为信道协议,该协议支持的数据传输速率范围为3~27 Mbit/s. 本文将数据传输速率设定为27 Mbit/s,以支持高速率数据传输. 同时,Two-ray ground model传播模型的选择考虑到其能更准确地反映VANET中车辆通信信号的传播特性.

HELLO消息发送周期被设定为250 ms,这一设定参考了相关文献中通常建议的范围(200~300 ms),以确保节点能够及时获取邻居信息. 数据包发送周期为1 s,该参数的设定借鉴了VANET消息传播相关论文中的建议,1 s的周期既能满足车辆之间的数据传输需求[42],又能避免频繁的数据传输对网络资源造成过大的压力.

5.1.2　评价指标

对比实验使用簇头生命周期、簇成员生命周期、簇数量、分簇开销、处于EL的节点数量这五个指标. 这些指标可以帮助评估分簇算法的性能,尤其是在动态环境下的网络稳定性和效率. 下面具体介绍所采用的评价指标.

(1)簇头生命周期

簇头生命周期是指节点从开始CH状态到转换为非CH状态的时长[1],衡量簇头的稳定性. 在本文提出的分簇算法中,簇头生命周期是指车辆从CH状态转换为EL状态或CM状态的时长. 平均簇头生命周期计算式为

$$\mathrm{AvgLTime_{CH}}=\frac{\sum_{i=1}^{n}\mathrm{AvgLTime_{CH}}(i)}{n} \tag{11}$$

其中,n表示在分簇阶段中出现的簇头节点数量;$\mathrm{AvgLTime_{CH}}$表示簇头节点的生命周期.

(2)簇成员生命周期

簇成员生命周期是指节点从加入某一个簇到离开该簇的时长[1],反映其在簇内的稳定性. 当簇成员所在簇的簇头发生改变时,或者当簇成员转换为簇头时,视为该簇成员生命周期结束. 平均簇成员生命周期计算方式类似于平均簇头生命周期的计算方式.

(3)簇数量

较少的簇意味着簇头节点与RSU之间、簇头节点之间的控制信息交换和管理开销较低[43]. 簇头节点通常负责协调簇内通信和路由等任务,因此减少簇的数量可以降低控制信息交换的频率和负担,从而减少网络的通信开销.

(4)分簇开销

分簇开销指为了维护和更新簇结构,节点之间通信和信息交换所产生的资源消耗. 本文将分簇开销定义为在分簇过程中生成的控制包数量与网络中所有数据包数量的比值[1],计算方法为

$$\mathrm{overhead}=\frac{\sum\mathrm{PacketSize_{ctl}}}{\sum\mathrm{PacketSize_{all}}}\times 100\% \tag{12}$$

本文提出的分簇算法中,控制包包括HELLO消息、跟随请求消息和跟随回复消息.

(5)处于EL状态的节点数量

EL状态是一个过渡状态,当簇成员与其所属的簇断开连接或簇头与其所有簇成员断开连接,它们将转换到EL状态. 处于EL状态的节点不属于任何簇,需要重新寻求新簇加入或自立为簇头. 若网络中存在大量处于EL状态的节点,表示分簇算法的簇稳定性较低[42].

5.1.3 对比算法

本文提出的VANET多跳分簇算法与以下几种基线算法进行对比.

(1)Nhop算法[15]. Nhop算法是一种基于移动性的多跳分簇算法,提出了用两个连续数据包的传递延迟之比度量多跳节点的相对移动性.

(2)VMaSC算法[16]. VMaSC算法选择相对多跳邻居节点移动性最小的节点作为簇头节点,并将具有相似移动性的车辆划分到同一个簇中.

(3) FCM-PMC (Fuzzy C-Means-Power Minimization Clustering)算法[22]. FCM-PMC算法是一种基于模糊C均值的分簇算法,以最小化车辆功耗为目标. FCM-PMC算法综合考虑熵、加权移动性和相对位置选举簇头.

(4)HHOCNET算法[29]. HHOCNET算法利用HHO策略,通过模仿哈里斯鹰的群体狩猎行为,优化分簇结果.

(5)GACC算法[30]. GACC算法结合联盟博弈论,利用遗传算法优化簇结构,以在满足传输延迟约束的前提下最大化簇规模.

5.1.4 算法参数设置

本文提出的VANET分簇算法中的参数值设置如表3所示. 本文提出的分簇算法将最大跳数设为3,在3跳范围内,能够保证簇的规模足够大,从而减少簇的数量,降低簇头之间的频繁交互,提升网络的管理效率. 同时,超过3跳会显著降低簇的稳定性,因为HELLO消息在更大跳数范围内传播时,争用增加,导致丢包率上升和节点之间的连接中断.

表3 算法参数设置

参数	值
MAX_HOP	3
$Size_{req}$	8 Bytes
$Size_{resp}$	8 Bytes
HELLO包大小	64 Bytes
数据包大小	1 024 Bytes

FOLLOW_REQ消息和FOLLOW_RESP消息包大小皆为8 Bytes,包含发送节点和接收节点ID. HELLO消息包大小设定为64 Bytes,该大小足以包含本文所提分簇算法中HELLO消息所需的字段. 同样,数据包大小设定为1 024 Bytes,确保数据负载容量足够.

基线算法的最大跳数、HELLO消息包大小、数据包大小与本文设定一致,以便进行公平的实验对比. 基线算法FCM-PMC的熵、加权移动性和相对位置的权重因子分别设为0.4、0.4、0.2. 在传输范围为100 m、200 m和300 m时,基线算法HHOCNET的最佳簇成员数量参数分别设为30、35、40,以确保簇头在较大的覆盖范围内容纳更多成员,从而获得合理的分簇结果. 在基线算法GACC中,为避免遗传算法陷入局部最优解,染色体种群的大小设置为1 500. 同时,节点效用函数的系数比按照原文中的实验设置为1∶1,以获得较优的实验结果.

5.2 实验测试与评估

5.2.1 节点分类测试与评估

为了评估所提出的稳定节点识别算法的有效性,实验采用动态分类测试方法,分别对RF、朴素贝叶斯(Naive Bayes,NB)和支持向量机(Support Vector Machine,SVM)在不同道路限速下的稳定节点分类性能进行对比. 在仿真过程中,每辆车在特征变化显著时向云侧上传数据,云侧基于当前特征进行稳定节点预测,并记录每次分类的预测类别与真实标签. 实验统计分类器的准确率、精确度、召回率、F_1-score以及平均推理时间,实验结果如表4所示.

表4 分类结果对比

限速/(m·s^{-1})	分类器	准确率/%	精确度/%	召回率/%	F_1-score/%	推理时间/ms
35	RF	88.33	90.28	88.71	88.25	1.605 0
	NB	56.67	62.89	57.73	52.38	0.222 9
	SVM	70.00	72.53	70.52	69.46	0.345 7
30	RF	83.33	83.33	83.94	83.26	1.560 4
	NB	73.33	78.75	76.02	73.06	0.212 5
	SVM	70.00	70.63	67.65	67.70	0.337 7
25	RF	88.33	88.33	88.38	88.33	1.589 6
	NB	73.33	77.66	73.97	72.57	0.251 7
	SVM	76.67	81.51	77.31	76.00	0.267 5
20	RF	88.33	88.38	88.33	88.33	1.618 0
	NB	76.67	77.14	76.42	76.43	0.218 7
	SVM	80.00	82.30	80.00	79.64	0.235 9
15	RF	91.67	91.74	91.60	91.65	1.578 0
	NB	88.33	88.39	88.26	88.30	0.223 7
	SVM	90.00	90.24	89.88	89.86	0.247 0

实验结果表明,RF在所有道路限速下均表现最佳,在低速(15 m/s)环境下,其准确率达到91.67%,在高速(35 m/s)环境下仍保持88.33%的高准确率,展现了良好的鲁棒性. 相比其他分类器,RF能够更有效地捕捉车辆特征之间的复杂关系,适应不同速度环境下的分类需求.

在低限速(15~30 m/s)环境中,NB的准确率可达73.33%~88.33%,但在高速(35 m/s)环境下,其准确率急剧下降至56.67%,表明该模型对高速场景适应性较差. 这可能是因为NB假设特征之间相互独立,而在高限速下,车辆的速度、加速度、变道频率等特征之间的相关性增强,从而违背了NB的假设,导致分类性能下降.

SVM在15~25 m/s的限速环境下的分类性能相对稳定,准确率在76.67%~90%. 然而,在30~35 m/s限速

下,准确率下降至70%. 这可能是因为在高速环境中,车辆的运动模式更加均一,稳定和不稳定节点的特征差距缩小,导致SVM难以找到合适的决策边界,从而降低了分类性能.

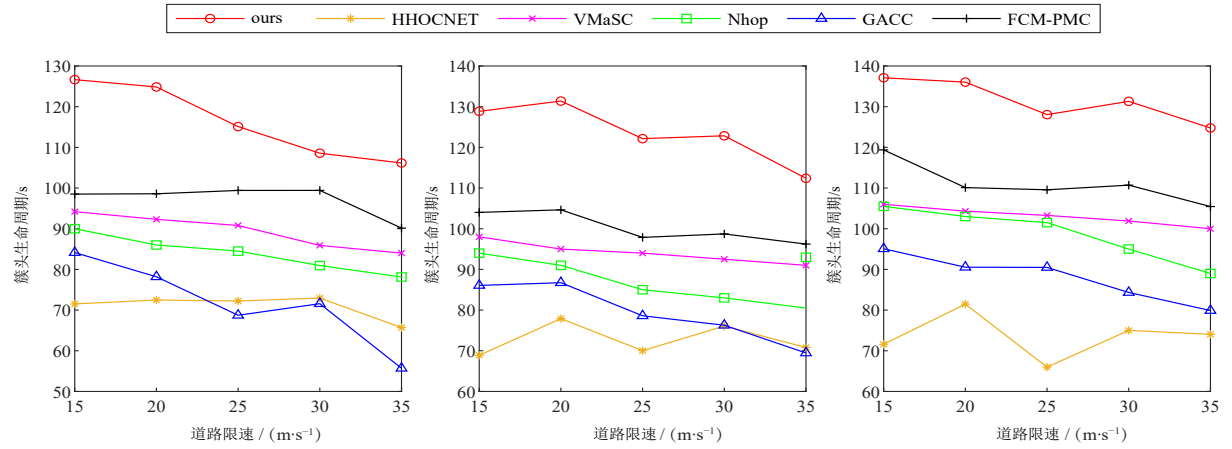
从平均推理时间来看,NB的推理速度最快,约为0.21~0.25 ms,SVM的推理时间介于0.23~0.34 ms,而RF的推理时间略高,为1.56~1.62 ms. 尽管NB计算开销最低,但其在高限速环境下的分类精度远低于RF,影响了实际应用的可靠性.

实验结果表明,RF在所有道路限速下均表现最佳,准确率保持在83.33%~91.67%,推理时间控制在2 ms以内,满足VANET的实时性要求. 本文提出的稳定节点识别算法通过合理选取车辆特征,充分考虑车辆运动稳定性,使得稳定节点在不同道路条件下均能被准确识别. 相比NB和SVM,该算法在高限速环境下仍保持较高精度,证明了其适用性. 准确的稳定节点识别为后续分簇奠定了基础,有效提高簇稳定性,减少簇内重组开销,优化簇头选择,最终提升VANET的通信稳定性.

5.2.2　簇稳定性测试与评估

(1)平均簇头生命周期测试与评估

图5展示了平均簇头生命周期和道路限速的关系,其中图5(a)~图5(c)分别表示传输范围为100 m、200 m和300 m的结果. 随着道路限速的增加,本文算法(ours)、VMaSC算法、Nhop算法、FMC-PMC算法、GACC算法的平均簇头生命周期呈下降趋势. 因为道路限速的增加会导致车辆速度的增加,车辆之间的连接断开和新建更为频繁,从而导致网络拓扑变化更快. 相比其他算法,HHOCNET算法的簇头生命周期随道路限速变化呈现波动,表明其基于HHO的簇头选择策略难以适应动态的车速变化.



(a) 传输范围为100 m的平均簇头生命周期　(b) 传输范围为200 m的平均簇头生命周期　(c) 传输范围为300 m的平均簇头生命周期

图5　平均簇头生命周期测试结果

对比图5(a)~图5(c)可以看出,本文算法、VMaSC、Nhop、FMC-PMC、GACC算法的簇头生命周期随着传输范围的增加而增加,说明较大的传输范围有助于这些算法维持簇头和簇成员之间的连接,从而提升簇头的稳定性. HHOCNET算法的簇头生命周期也随传输范围增加而提升,但波动依然明显,表明即使增大传输范围. 其优化策略在不同道路限速下的稳定性仍有不足.

由图5可知,本文算法具有更高的簇头生命周期. 主要原因有两点:其一,本文算法将车辆分类为稳定节点和非稳定节点,稳定节点具有更稳定的驾驶行为,只有稳定节点才具备成为簇头的资格;其二,选举策略考虑了簇头相对于其跟随者和稳定邻居的相对移动性,簇头与其跟随者和稳定邻居的位置变化较小,从而提高了簇头稳定性.

VMaSC算法和Nhop算法的平均簇头生命周期接近,因为它们都选择相对多跳邻居移动性最小的节点作为簇头. VMaSC算法和Nhop算法的平均簇头生命周期均低于本文算法的平均生命周期,原因在于VMaSC和Nhop算法中的簇头必须在多跳邻居中表现出最小的平均相对移动性,簇头在高速移动环境下长时间保持这个条件成立很困难. 在本文算法中,簇头的稳定性只需要考虑其与跟随者以及稳定邻居的相对移动性,从而提高了算法的鲁棒性.

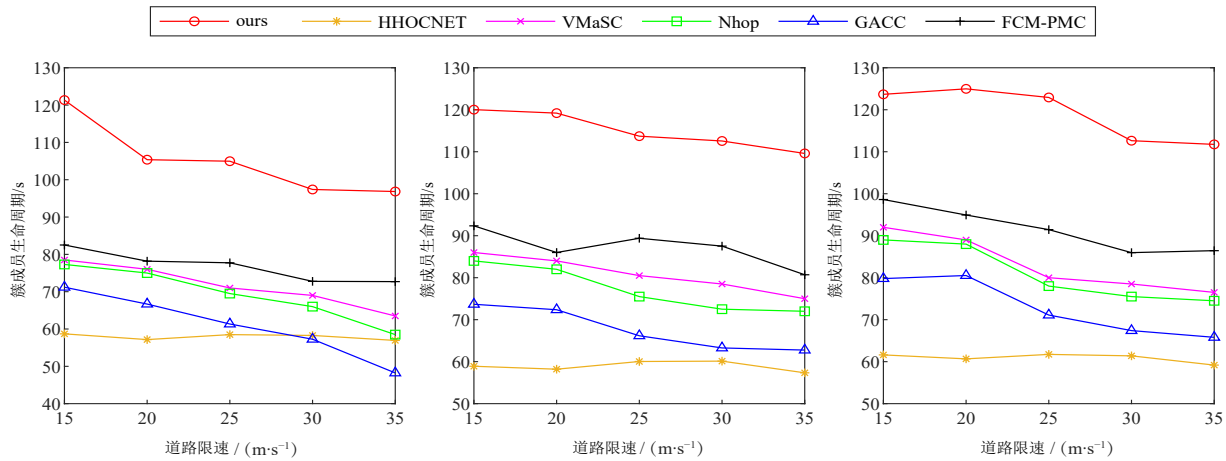
FMC-PMC算法的簇头生命周期高于VMaSC和Nhop算法,主要因为其在簇头选择过程中引入了熵、加权移动性和相对位置三个加权因子,这些因子能够更全面地衡量簇头的稳定性. 但该算法的权重未经过专门优化,导致簇头选择未必最优. 此外,FMC-PMC算法在分簇时,首先根据虚拟化网络功能的流行程度动态调整聚类数,以优化计算资源分配,使得车辆总能耗最小化. 这种方法可能在降低网络能耗方面表现良好,但未必能保证簇头的长期稳定性.

GACC算法的簇头生命周期较低,主要由于其分簇机制未着重考虑簇头的稳定性. 算法通过遗传算法优化簇内所有节点的效用之和,效用函数综合考虑簇大小、簇内数据传输延迟和节点与父节点的速度差. 在此优化目标下,算法优先生成更大规模的簇以提升整体效用. 然而,由于网络拓扑快速变化,且簇头未被特殊约束,其角色频繁更换. 这种动态调整机制导致簇头在分簇结果中的持续时间较短. GACC算法的主要目标是形成规模更大的簇,但未能充分关注簇头的稳定性.

HHOCNET算法的簇头生命周期较低且波动较大,主要原因在于其基于Harris Hawk优化算法的频繁动态调整. 该算法每轮优化时都会根据簇成员的数量和簇头到簇成员的距离对簇结构进行重新优化,因此簇头容易被频繁替换. 该算法未特别考虑簇头稳定性,仅基于当前的簇配置进行优化. 在高动态性场景下,这种局限性造成簇头生命周期的波动.

(2)平均簇成员生命周期测试与评估

平均簇成员生命周期测试结果如图6所示. 对于本文算法、VMaSC算法、Nhop算法、FMC-PMC算法、GACC算法,簇成员生命周期随着道路限速增加而逐渐降低. 对于HHOCNET算法,簇成员生命周期在不同限速下相对平稳,但是在所有算法中数值较低.



(*a*) 传输范围为100 m的平均簇成员生命周期 (*b*) 传输范围为200 m的平均簇成员生命周期 (*c*) 传输范围为300 m的平均簇成员生命周期

图6 平均簇成员生命周期测试结果

对比图6(*a*)~图6(*c*)可以看出,本文算法的簇成员生命周期随传输范围的增加而显著提升,Nhop、VMaSC、FMC-PMC、GACC算法在较大传输范围下簇成员生命周期有适度提升,而HHOCNET算法的簇成员生命周期随传输范围的增加,提升不显著且稳定性欠佳.

由图6可知,本文算法相比其他四种基线算法具有更高的簇成员生命周期,证明了本文提出的稳定节点跟随策略的有效性. 每个非稳定节点在单跳邻居中选择一个稳定节点跟随并与其同属一个簇,稳定节点跟随策略考虑了相对移动性,使得非稳定节点选择与其连接质量较高的稳定节点跟随. 此外,稳定节点跟随策略还考虑了历史归属信息,即使节点与跟随的稳定节点断开连接,在选择新的跟随的稳定节点时,会优先考虑与其之前同属一个簇的稳定节点.

VMaSC算法和Nhop算法的平均簇成员生命周期相似,均低于本文算法. 这两种算法主要关注簇头相对多跳邻居的平均移动速度. 然而,在车辆速度较高的网络环境中,与距离多跳的簇头保持稳定的连接并不容易,因此簇成员与簇头断开连接的情况会频繁发生. 当簇成员与其簇头断开连接或其所在簇不存在时,簇成员需要重新加入新的簇中或自己成为簇头. 在本文算法中,若簇成员是非稳定节点,则只需要保持与其跟随的稳定节点之间的连接即可,维护簇成员与其跟随的稳定节点之间的单跳连接较为容易. 若簇成员是稳定节点,则只需要保持与其簇头之间的多跳邻居关系即可,稳定节点的速度变化较小,因此维持稳定节点之间的多跳连接较为容易.

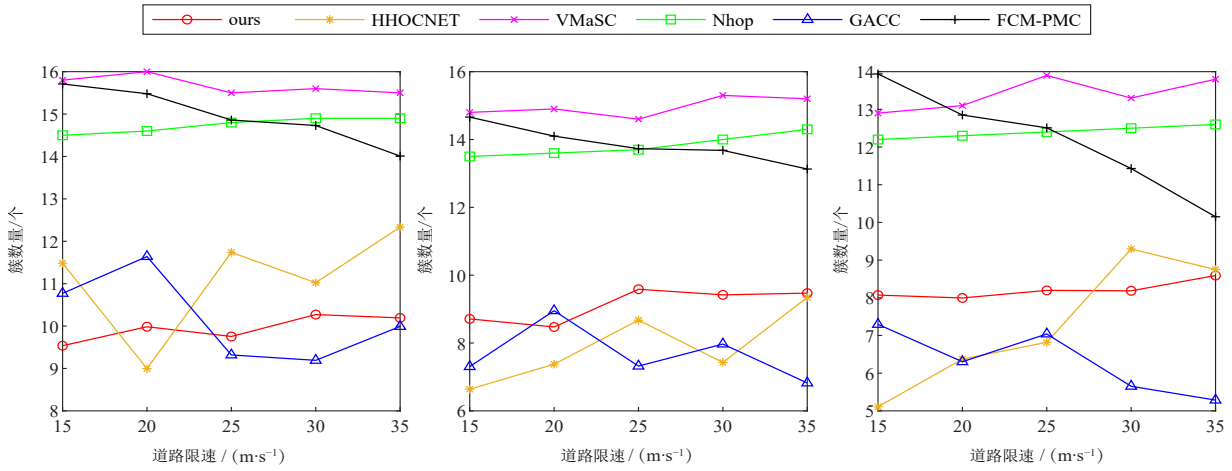
FMC-PMC算法的簇成员生命周期高于VMaSC和Nhop算法,主要因其引入熵和加权移动性两个指标. 熵衡量局部网络稳定性,熵值高表明邻居位置变化较小;加权移动性反映邻居变动情况,减少高频变动对簇结构的影响,进而延长簇成员生命周期. 然而,FMC-PMC算法的簇成员生命周期低于本文算法,主要因其采用模糊C均值聚类. FCM算法基于隶属度进行软聚类,导致簇的边界模糊,降低簇结构的稳定性. 在动态环境下,这种模糊性增加簇成员变动频率,从而缩短其生命周期.

GACC算法和HHOCNET算法的簇成员生命周期较低,主要源于它们优化目标的局限性. 这两种算法都侧重于优化簇头拥有的成员数量. 然而,它们并未充分考

虑车辆的移动速度和动态拓扑变化. 在实际应用中,车辆的相对速度对簇头与簇成员之间的连接稳定性至关重要. 当车辆以较高速度移动时,连接可能迅速断开,导致频繁的簇成员更新. 此外,过于注重成员数量的优化可能使得在实际运行中面临更多的簇成员变动.

5.2.3　簇数量测试与评估

各种算法的簇数量测试结果如图7所示. 随着传输范围增大,簇数量总体上呈现下降趋势. 这是因为更大的传输范围使车辆之间更容易保持连接,从而减少了簇的数量.



(a) 传输范围为100 m的簇数量　(b) 传输范围为200 m的簇数量　(c) 传输范围为300 m的簇数量

图7　簇数量测试结果

本文算法的簇数量相对较少,表现稳定,在所有通信范围内,其簇数量均在较低水平,原因主要有两点:其一,本文算法的簇头选举策略考虑了覆盖节点数量这一指标,优先选择预估簇成员多的节点作为簇头;其二,在本文算法提出的稳定节点跟随策略中,节点无需了解距离自己多跳的簇头节点的情况,只需要在单跳邻居中选择稳定节点跟随,即可加入相应的簇. 稳定节点跟随策略使得节点更容易加入簇,减少了孤立节点的数量.

Nhop算法的簇数量略小于VMaSC算法的簇数量,原因在于Nhop算法在设定的跳数范围内形成簇时,节点选择距离自己跳数小的簇头加入. 然而,在VMaSC算法中,节点选择与自己移动性相似的簇头加入,这种以稳定性为导向的簇形成策略可能导致簇大小不如基于简单多跳通信的Nhop算法大.

FCM-PMC算法的簇数量相对较多,但会随着车辆速度的提高而逐渐减少. 这种分簇方式是为了降低整体网络能耗而优化得到的. FCM-PMC算法的簇数量随道路限速增加而减少. 较高车速导致成员车辆更频繁更换簇,增加簇头的管理负担. 同时,虚拟化网络功能的迁移频率上升,加重了CPU负载和延迟. 因此,算法通过减少簇数量来降低簇头负载和迁移成本.

GACC和HHOCNET算法形成簇数量较少,主要源于它们的优化目标和分簇机制设计. GACC算法通过遗传算法优化簇内节点的效用和,效用函数中的簇大小是重要因素,较大的簇能够提高效用值,从而促使算法生成规模更大的簇以减少簇的数量. HHOCNET算法则通过HHO算法控制簇成员的分布,明确定义了簇头的最佳簇成员数量,并通过优化使簇成员数量接近目标值,避免了生成过多的簇. 两种算法都注重提升簇规模,从而共同导致了簇数量较少的结果.

5.2.4　分簇开销测试与评估

图8展示了节点传播范围为200 m的设置下各种算法的分簇开销. 由于HHOCNET和GACC算法不涉及车辆之间的控制消息传输,实验仅测试和对比本文算法、VMaSC算法、Nhop算法、FMC-PMC算法的分簇开销. 由图8可知,随着道路限速的增加,分簇开销也随之增加. 这是因为车辆速度增加,引起簇稳定性下降,节点重新加入簇造成额外的开销. 此外,本文算法的分簇开销随着道路限速的增加相较基线算法更为平缓,说明了本文算法具有较高的鲁棒性.

与基线算法相比,本文算法的分簇开销较小. 第一个原因是本文算法高效的邻居发现机制大幅减少了HELLO消息的数量. 在基线算法中,每个节点需要转发接收到的所有HELLO消息以维护多跳邻居列表,造成大量的HELLO消息在网络中传播. 然而,在本文算法中,非稳定节点无需转发HELLO消息,只是与单跳邻居节点交换HELLO数据包以连接到簇. 稳定节点转发来自其他稳定节点的HELLO消息,减少了多跳传播的HELLO消息数量. 第二个原因是本文算法的稳定节点跟随机制,使得节

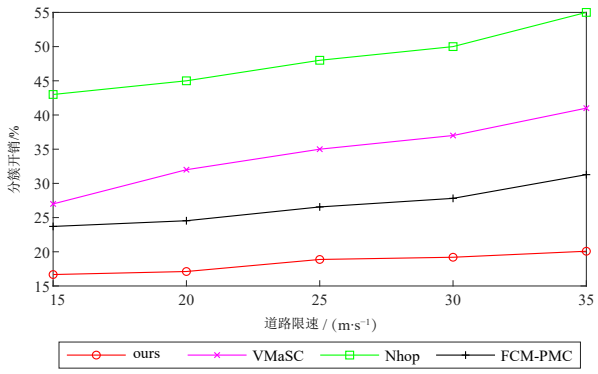


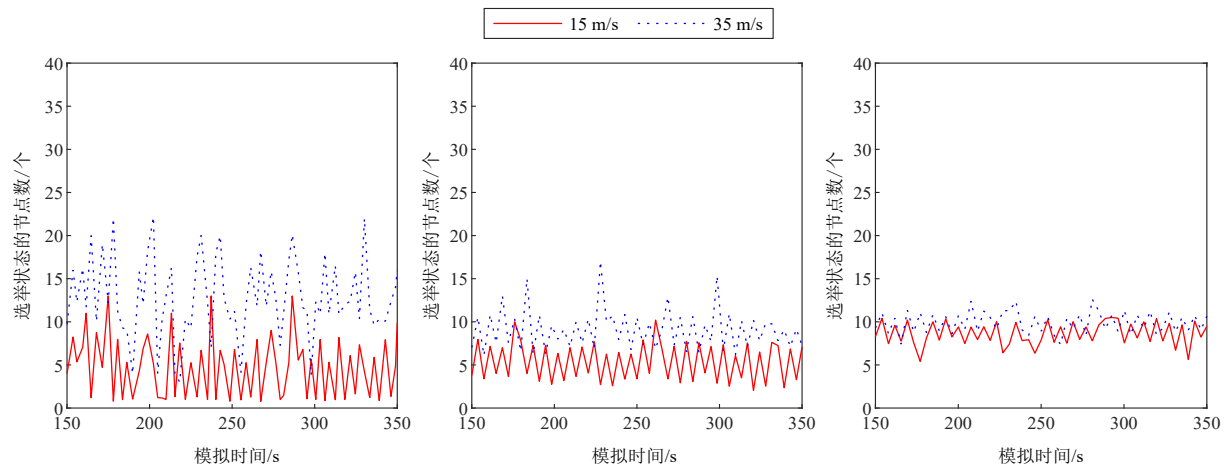
图8 分簇开销测试结果

点通过单跳邻居中的稳定节点连接到簇，而非多跳连接到簇头．稳定节点跟随机制提高了簇稳定性，减少了节点重新加入簇的次数，从而降低了分簇开销．

5.2.5 EL状态节点数测试与评估

由于其他基线算法中未引入节点簇状态这一变量，实验仅测试和对比Nhop算法、VMaSC算法、本文算法模拟过程中处于EL状态的节点数量．图9展示了三种算法在道路限速为15 m/s和35 m/s时处于EL状态的节点数量随着模拟时间的变化．由图9可知，道路限速较大时，处于EL状态的节点数量较多．这是因为随着车辆速度的增加，节点之间的连接中断概率增加，导致更多的节点与所属的簇断联并寻求新的簇加入．

对比图9(a)~图9(c)可知，在不同道路限速下，本文算法的EL状态节点数均小于VMaSC和Nhop算法的EL状态节点数，这表明本文算法划分的簇具有更高的稳定性，节点不容易与簇断联．这得益于本文提出的稳定节点跟随策略，每个节点从单跳邻居中选择最适合的稳定节点进行跟随，节点与稳定节点之间的连接相较与簇头之间的多跳连接更为稳定和持久．此外，本文算法获得的平均簇成员数量大于基线算法的簇成员数量，簇头节点不易与所有簇成员丢失连接，因此簇头节点很少转换到EL状态．



(a) Nhop算法处于EL状态的节点数量 (b) VMaSC算法处于EL状态的节点数量 (c) 本文算法处于EL状态的节点数量

图9 处于EL状态的节点数量测试结果

值得注意的是，相较VMaSC和Nhop算法，本文算法在道路限速较低情况下处于EL状态的节点数与道路限速较高情况下处于EL状态的节点数的差距较小．这表明本文算法在不同车辆速度的环境中都可以获得较高的簇稳定性，算法具有一定的鲁棒性．

6 结论

针对现有VANET分簇算法簇稳定性低、分簇开销大的问题，本文提出了一种端云协同的VANET分簇算法．首先，提出一种云侧稳定节点识别算法，车辆通过RSU将自身特征数据上传至云，云侧对车辆进行稳定性分类，并将分类结果反馈给车辆．本文将簇头节点限制在稳定节点范围内，从而简化簇头选举过程并提高簇稳定性．在选举过程中，考虑了稳定节点的相对移动性和覆盖节点数量等因素，进一步增强了簇稳定性，并确保簇头能够有效管理更大范围内的节点．其次，本文提出了一种新的邻居发现和更新机制，非稳定节点不参与HELLO消息的转发，仅稳定节点需要转发来自其他稳定节点的HELLO消息．实验表明：本文提出的端云协同的VANET分簇算法在簇稳定性、簇数量以及分簇开销等关键指标上显著改善，验证了算法的有效性，并展示了其在实际交通场景中的应用潜力．

参考文献

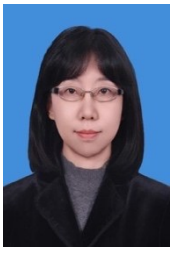
[1] MILES J C, 陈干．智能交通系统手册[M]. 王笑京译．北京: 人民交通出版社, 2007: 201-203.
MILES J C, CHEN G. Intelligent Transportation Systems Handbook[M]. WANG X J, trans. Beijing: China Commu-

nications Press, 2007: 201-203. (in Chinese)
[2] LIMBASIYA T, DAS D. Secure message transmission algorithm for Vehicle to Vehicle (V_2V) communication[C]// 2016 IEEE Region 10 Conference (TENCON). Piscataway: IEEE, 2016: 2507-2512.
[3] ALI KARABULUT M, SHAH A F M S, ILHAN H, et al. Inspecting VANET with various critical aspects - A systematic review[J]. Ad Hoc Networks, 2023, 150: 103281.
[4] ZANJIREH M M, LARIJANI H. A survey on centralised and distributed clustering routing algorithms for WSNs[C]// 2015 IEEE 81st Vehicular Technology Conference (VTC Spring). Piscataway: IEEE, 2015: 1-6.
[5] CAMPOLO C, MOLINARO A. On vehicle-to-roadside communications in 802.11p/WAVE VANETs[C]//2011 IEEE Wireless Communications and Networking Conference. Piscataway: IEEE, 2011: 1010-1015.
[6] ANWER M S, GUY C. A survey of VANET technologies[J]. Journal of Emerging Trends in Computing and Information Sciences, 2014, 5(9): 661-671.
[7] MUKHTARUZZAMAN M, ATIQUZZAMAN M. Clustering in vehicular ad hoc network: Algorithms and challenges[J]. Computers & Electrical Engineering, 2020, 88: 106851.
[8] AYYUB M, ORACEVIC A, HUSSAIN R, et al. A comprehensive survey on clustering in vehicular networks: Current solutions and future challenges[J]. Ad Hoc Networks, 2022, 124: 102729.
[9] KATIYAR A, SINGH D, YADAV R S. State-of-the-art approach to clustering protocols in VANET: A survey[J]. Wireless Networks, 2020, 26(7): 5307-5336.
[10] JABBAR M K, TRABELSI H. A review on clustering in VANET: Algorithms, phases, and comparisons[C]//2022 19th International Multi-Conference on Systems, Signals & Devices (SSD). Piscataway: IEEE, 2022: 444-451.
[11] LATIF S, MAHFOOZ S, JAN B, et al. A comparative study of scenario-driven multi-hop broadcast protocols for VANETs[J]. Vehicular Communications, 2018, 12: 88-109.
[12] GERLA M, TZU-CHIEH TSAI J. Multicluster, mobile, multimedia radio network[J]. Wireless Networks, 1995, 1(3): 255-265.
[13] CHEN G, NOCETTI F G, GONZALEZ J S, et al. Connectivity based k-hop clustering in wireless networks[C]// Proceedings of the 35th Annual Hawaii International Conference on System Sciences. Piscataway: IEEE, 2002: 2450-2459.
[14] BASU P, KHAN N, LITTLE T D C. A mobility based metric for clustering in mobile ad hoc networks[C]// Proceedings of the 21st International Conference on Distributed Computing Systems Workshops. Piscataway: IEEE, 2001: 413-418.
[15] ZHANG Z X, BOUKERCHE A, PAZZI R. A novel multi-hop clustering scheme for vehicular ad-hoc networks[C]// Proceedings of the 9th ACM International Symposium on Mobility Management and Wireless Access. New York: ACM, 2011: 19-26.
[16] UCAR S, ERGEN S C, OZKASAP O. VMaSC: Vehicular multi-hop algorithm for stable clustering in vehicular ad hoc networks[C]//2013 IEEE Wireless Communications and Networking Conference (WCNC). Piscataway: IEEE, 2013: 2381-2386.
[17] BASAGNI S. Distributed clustering for ad hoc networks[C]//Proceedings of the 4th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99). Piscataway: IEEE, 2002: 310-315.
[18] WOLNY G. Modified DMAC clustering algorithm for VANETs[C]//Proceedings of the 3rd International Conference on Systems and Networks Communications. Piscataway: IEEE, 2008: 268-273.
[19] CHATTERJEE M, DAS S K, TURGUT D. An on-demand weighted clustering algorithm (WCA) for ad hoc networks[C]//Global Telecommunications Conference (Globecom'00 - IEEE). Piscataway: IEEE, 2002: 1697-1701.
[20] DHURANDHER S K, SINGH G V. Weight based adaptive clustering in wireless ad hoc networks[C]//2005 IEEE International Conference on Personal Wireless Communications (ICPWC 2005). Piscataway: IEEE, 2005: 95-100.
[21] SENOUCI O, HAROUS S, ALIOUAT Z. An efficient weight-based clustering algorithm using mobility report for IoV[C]//Proceedings of the 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). Piscataway: IEEE, 2018: 614-620.
[22] ZHAO H T, TANG J W, ADEBISI B, et al. An adaptive vehicle clustering algorithm based on power minimization in vehicular ad-hoc networks[J]. IEEE Transactions on Vehicular Technology, 2022, 71(3): 2939-2948.
[23] KENNEDY J, EBERHART R. Particle swarm optimization[C]// Proceedings of ICNN'95 - International Conference on Neural Networks. Piscataway: IEEE, 1995: 1942-1948.
[24] HEIDARI A A, MIRJALILI S, FARIS H, et al. Harris Hawks optimization: Algorithm and applications[J]. Future Generation Computer Systems, 2019, 97: 849-872.
[25] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69: 46-61.
[26] GOLBERG D E. Genetic algorithms in search, optimization, and machine learning[J]. Machine Learning, 1988, 3: 95-99.
[27] SHAHZAD W, KHAN F A, SIDDIQUI A B. Clustering in mobile ad hoc networks using comprehensive learning particle swarm optimization (CLPSO) [M]//Communication and Networking. Berlin: Springer, 2009: 342-349.
[28] FAHAD M, AADIL F, REHMAN Z U, et al. Grey wolf optimization based clustering algorithm for vehicular ad-hoc networks[J]. Computers & Electrical Engineering, 2018, 70: 853-870.
[29] ALI A, AADIL F, KHAN M F, et al. Harris hawks optimization-based clustering algorithm for vehicular ad-hoc networks[J]. IEEE Transactions on Intelligent Transporta-

tion Systems, 2023, 24(6): 5822-5841.
[30] CHAROENCHAI S, SIRIPONGWUTIKORN P. Genetic algorithm for multi-hop VANET clustering based on coalitional game[J]. Journal of Network and Systems Management, 2024, 32(1): 9.
[31] MYERSON R B. Graphs and cooperation in games[J]. Mathematics of Operations Research, 1977, 2(3): 225-229.
[32] QI G Q, DU Y M, WU J P, et al. Leveraging longitudinal driving behaviour data with data mining techniques for driving style analysis[J]. IET Intelligent Transport Systems, 2015, 9(8): 792-801.
[33] ZHANG L R, LAKAS A, EL-SAYED H, et al. Mobility analysis in vehicular ad hoc network (VANET)[J]. Journal of Network and Computer Applications, 2013, 36(3): 1050-1056.
[34] ZHENG J, TONG H, WU Y Y. A cluster-based delay tolerant routing algorithm for vehicular ad hoc networks[C]// 2017 IEEE 85th Vehicular Technology Conference (VTC Spring). Piscataway: IEEE, 2017: 1-5.
[35] WENZEL T P, ROSS M. The effects of vehicle model and driver behavior on risk[J]. Accident Analysis & Prevention, 2005, 37(3): 479-494.
[36] LI Z P, ZHANG R, XU S Z, et al. Study on the effects of driver's lane-changing aggressiveness on traffic stability from an extended two-lane lattice model[J]. Communications in Nonlinear Science and Numerical Simulation, 2015, 24(1/2/3): 52-63.
[37] CHEUNG E, BERA A, KUBIN E, et al. Identifying driver behaviors using trajectory features for vehicle navigation[C]//2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Piscataway: IEEE, 2018: 3445-3452.
[38] SU Z Z, WOODMAN R, SMYTH J, et al. The relationship between aggressive driving and driver performance: A systematic review with meta-analysis[J]. Accident Analysis & Prevention, 2023, 183: 106972.
[39] VAIANA R, IUELE T, ASTARITA V, et al. Driving behavior and traffic safety: An acceleration-based safety evaluation procedure for smartphones[J]. Modern Applied Science, 2013, 8(1): 88-96.
[40] BREIMAN L. Random forests[J]. Machine Learning, 2001, 45(1): 5-32.
[41] CAMPANILE L, GRIBAUDO M, IACONO M, et al. Computer network simulation with ns-3: A systematic literature review[J]. Electronics, 2020, 9(2): 272.
[42] UCAR S, ERGEN S C, OZKASAP O. Multihop-cluster-based IEEE 802.11p and LTE hybrid architecture for VANET safety message dissemination[J]. IEEE Transactions on Vehicular Technology, 2016, 65(4): 2621-2636.
[43] CHEN Y Z, FANG M Y, SHI S, et al. Distributed multi-hop clustering algorithm for VANETs based on neighborhood follow[J]. EURASIP Journal on Wireless Communications and Networking, 2015, 2015(1): 98.

作者简介

马 玲 女,1999 年 3 月出生于河南省. 现为东北大学计算机科学与工程学院博士研究生. 主要研究方向为无线通信网络、车联网.
E-mail: 2410822@stu.neu.edu.cn



杨晓春 女,1973 年 5 月出生于辽宁省. 现为东北大学计算机科学与工程学院教授、博士生导师. 主要研究方向为大数据管理和知识工程、数据质量管理、数据隐私保护和推荐系统. 中国电子学会会员编号:E190035815M.
E-mail: yangxc@mail.neu.edu.cn

王 斌 男,1972 年 11 月出生于辽宁省. 现为东北大学计算机科学与工程学院教授、博士生导师. 主要研究方向为算法的设计和分析、流数据查询处理和分布式系统.
E-mail: binwang@mail.neu.edu.cn

宋晓诗 男,1983 年 12 月出生于辽宁省. 现为东北大学计算机科学与工程学院副教授、博士生导师. 主要研究方向为 5G 及 B5G 无线通信网络、无线数据缓存网络、D2D 无线通信网络、认知无线网络、无线蜂窝网络. 中国电子学会会员编号:E190158661M.
E-mail: songxiaoshi@cse.neu.edu.cn

李发明 男,1988 年 1 月出生于黑龙江省. 现为东北大学计算机科学与工程学院讲师. 主要研究方向为数据管理、大数据计算.
E-mail: lifaming@mail.neu.edu.cn