云南大学

研究生开题报告

（原稿）

题　　目	车联网环境下能耗优先的任务调度算法研究
研究类型	基础研究□　应用研究√　其他□
学院/研究院	信息学院
学　　号	12021115016
姓　　名	李涵
专　　业	计算机软件与理论
导　　师	张学杰
层　　次	博　士□　　硕　士√
学习形式	全日制√　　非全日制□
时　　间	

基于智能算法的车联网环境下的任务调度算法研究

1. 研究意义

随着万物互联时代的到来，网络边缘设备产生的数据量快速增加，带来了更高的数据传输带宽需求，同时，新型应用也对数据处理的实时性提出了更高要求，传统云计算模型已经无法有效应对，因此，边缘计算应运而生，边缘计算的基本理念是将计算任务在接近数据源的计算资源上运行。

边缘计算的一大应用场景就是智能汽车。随着机器视觉、深度学习和传感器等技术的发展，汽车的功能不再局限于传统的出行和运输工具，而是逐渐变为一个集合娱乐为一身的计算系统，我们称这样新型的汽车为智能网联车（connected and auto nomous vehicles，CAVs）。车辆将被安装越来越多的传感器设备，以用来监测汽车的各种状态。这将会产生大量的数据。

智能网联车的出现催生出了一系列新的应用场景，例如自动驾驶、车联网以及智能交通。Intel 在 2016 年的报告指出一辆自动驾驶车辆一天产生的数据为 4TB，这些数据无法全部上传至云端处理，需要在边缘节点（汽车）中存储和计算。

清洁能源是未来的一个重要趋势。电能作为清洁能源的代表，越来越多的人选择了使用电能的电动汽车。电动智能车辆将成为未来城市交通的组成部分[1]。然而，在电池技术发生重大变革之前，电池的容量始终是电动智能汽车的短板。有限的电池容量会导致“续航焦虑”的问题，因为车辆的能量不可能得到快速的补充。

人们提出了许多解决里程问题的方法，例如考虑整车的能量， 对整车的能量进行管理[2]。车辆滑行是驾驶模式的一部分，使用滑行可以减少能耗[3]。

由于行驶过程中需要处理的数据量过大，中央处理器将消耗相当大的能量，这对行驶里程有很大影响。一些研究建议将任务发送到移动边缘计算服务器以节省能源。但如果所有数据都上传到云服务器进行处理，响应时间可能太长，无法满足低延迟要求[4]。文献[5]在论文中提出了一种保证任务低延迟性能的系统。此外，目前的带宽和存储根本无法满足传输所有数据的要求。

为了解决这个问题，文献[6]将重点放在车辆边缘云中的虚拟机资源分配上。文献[7]考虑了动态需求和资源约束，并将所有任务划分为四种类型的待定列表。然后，每个列表中的任务将根据其功能卸载到不同的节点。

2. 研究现状

2.1 车联网的计算模型

车联网的计算模型是一个分布式计算模型，它包含了多个组成部分，包括车辆、路边设施（RSU）、数据中心等，可以实现车辆之间、车辆和路边设施之间、车辆和数据中心之间的数据共享和协同处理。

首先，车联网中的车辆可以通过各种传感器收集大量的数据，包括车辆状态、位置信息、驾驶行为、环境信息等等。这些数据可以通过车载计算机进行处理和分析，然后传输到数据中心或其他车辆进行共享和协同处理。

其次，路边设施也可以通过传感器收集相关数据，例如交通信号灯、路况监测设备等。这些数据可以和车辆收集的数据一起传输到数据中心进行处理和分析。

最后，数据中心作为车联网的核心，它可以接收、处理、存储和共享车辆和路边设施收集的数据。在数据中心中，可以使用各种计算模型进行数据分析、模型训练和预测等。

车联网的数据中心由远程云、边缘云和车辆云三层云结构组成。其中远程云是传统云计算中的云资源，边缘云位于网络的边缘，可以提供更低延迟和更高带宽的服务，车辆云与边缘云通过无线通信连接。

远程云（Remote Cloud）通常具有超大的规模，能赋予用户超强的计算能力，对大规模数据聚合、数据挖掘，分析、优化和存储复杂数据，远程云可以在短的时间内计算得到结果。车辆应用产生的计算任务可以通过网络传输到远程云，在远程云计算并将结果返回给车辆。由于远程云超强的计算能力，因此能够计算和分析复杂的计算任务，并能够提供较高的计算性能，但是因为远程云位于网络的远端，将车辆计算任务传输到远程云并得到相应的返回结果需要很大的传输时间开销。因此对于数据密集型和延迟敏感型的计算任务，即使计算延迟较小，较大的传输延迟仍会使任务总御载延迟较大。同时，车辆产生的大量数据全部上传到远程云会占用大量的网络带宽，从而造成网络性能和稳定性的下降。

边缘云（Edge Cloud）位于远程云和车辆云之间，由拥有计算能力的边缘设备组成，它是移动边缘计算中的重要组成部分，是解决远程云传输延迟高的一种有效替代方案。边缘云可由边缘服务器、RSU、BS、无线接入点（Access Point，AP）以及移动设备等其他具备计算、存储和通信能力的边缘设备组成。因为边缘云更靠近车辆，处在网络的边缘，车辆到 EC 的传输延迟可以大大降低。因此边缘云可以为车辆提供低延迟计算、高速缓存、位置感知、紧急管理等服务，并能用于实时交互，对于数据密集型和延迟敏感型的车辆计算任务，例如：增强现实，实时视频分析，人类行为识别等任务，它们需要极低的延迟以便快速响应和决策，此时边缘云比远程云具有更大的优势。边缘计算作为云计算的延伸，未来将与云计算协同配合，为用户提供更高质量的服务。

车辆云（Vehicular Cloud，VC）是汽车组成的资源系统，随着汽车工业中技术的演变和智能网联车辆的发展，车辆已被赋予更多的计算、存储、通信和传感资源。城市地区有大量的车辆，将空闲的车辆资源加以利用，可以为移动设备提供巨大的资源和价值，通过整合车辆空闲的计算资源形成车辆云，为其他车辆或者移动用户提供服务，可以大大提高服务和应用的质量。智能网联车辆包含使用无线通信议的设备，车辆能以较低的通信延迟接发计算任务，充分利用车辆上空闲的计算、存储资源，在保证车辆安全行驶的前提下，降低任务处理延迟，提高用户的驾驶体验。车辆云中车辆的角色不是固定的，当车辆资源空闲时可以作为服务提供者，接受来自其他车辆或是移动设备的任务请求；当车辆需要卸载任务时，可以作为服务请求者，将任务卸载至其他拥有空闲计算资源的车辆，来实现资源的整合和资源利用率的提高。

2.2 任务模型

车辆应用程序按照其关键程度分为三类：关键应用程序（Crucial Applications，CAs）、高优先级应用程序（High-Priority Applications，HPAs）和低优先级应用程序（Low-Priority Applications，LPAs）。

CAs 是和车辆安全相关的应用，是保证车辆和乘客安全的关键应用程序，如车辆控制、系统监控和事故预防等。由于 CAs 和安全紧密相关，因此享有最高的优先级，车辆制造商必须保留充足的计算资源给这部分应用，不能因为 HPAs 和 LPAs 的存在而影响 CAs 的正常运行，同时这类任务也不允许卸载，只允许在本地执行，不属于计算任务卸载的范畴。该类任务的实例是：车辆控制、碰撞预警、红绿灯警告、网上车辆诊断、道路湿滑检测等。

HPAs 包括与驾驶相关的应用和可选的安全增强应用，这类应用程序对车辆而言是重要但不是必须的，拥有较高的执行优先级，例如实时路径规划和路况提醒等。这类应用允许出现延迟或卸载失败的情况，但不会影响车辆安全。该类任务的实例是：地图导航、平视显示器、视野增强、车辆传感等。

LPAs 是一类为乘客提供娱乐服务的应用程序，它的优先级较低，例如语音识别，它允许驾驶员发出各种声音命令，通过语音识别命令计算机做一些响应，而不会使驾驶员分心。该类任务的实例是：虚拟现实、语音识别、视频处理、在线游戏等。

HPAs 和 IPAs 已经被部署到越来越多的车辆上，由于 HPAs 和 LPAs 不会影响到车辆的安全，因此可以将其进行卸载，来提高资源的利用率。

2.3 通信模型

计算任务卸载过程主要通过无线通讯网络将计算任务及其相关参数从服务请求者传输至服务提供者。

随着车联网的发展，车载网络技术已经相对成熟，车辆之间（Vehicle--To-Vehicle，V2V）的通信可以基于专用短程通信（Dedicated Short-Range Communications，DSRC）来实现，DSRC 是基于 IEEE802.11p 设计的，在 300m 覆盖内 DSRC 的数据速率可以达到 27Mb/s 左右。

车辆与固定基础设施之间（Vehicle-To-Infrastructure，V2I）的通信则可以使用 Long-Term Evolution（LTE）和 DSRC，与 DSRC 相比，LTE 的覆盖范围更广，服务质量更具有保障，LTE 还支持 350km/h 的高移动性的用户设备，可以很好地适应高速移动的车辆。

此外，迅速发展的 5G 通信网络提供了一种全新的网络架构，提供 10Gbps 以上的峰值速率、更佳的移动性能、毫秒级时延和超高密度连接，国际电信联盟无线电通信局定义了 5G 的三大典型应用场景为增强型移动宽带（eMBB）、超低时延高可靠通信（uRLLC）和海量大规模连接物联网（mMTC）。其中 uRLLC 由于其即时、可靠、高效数据传输的特点，如车联网、自动驾驶、自动化无人机等都是其主要实际应用，基于 5G 的车联网络将是未来车辆计算卸载的通信技术之一。

DSRC、LTE、5G 等车联网络技术为 VEC 计算任务卸载提供了相对可靠的网络传输通道，为高速移动下车辆的计算任务卸载提供可靠的通信保障。

2.4 任务卸载

任务卸载是指将移动设备（如智能手机、车载终端等）上的计算任务分解成若干个子任务，其中一部分在本地完成，另一部分则通过网络卸载到云端或者其他设备上进行计算，最终将结果返回到本地设备，从而提高移动设备的计算性能和能耗效率。

根据不同的任务卸载方式和卸载目标，可以将任务卸载分为以下几种类型：

（1）本地卸载：任务完全在本地设备上完成，不需要使用其他远程设备或云端资源；

（2）远程卸载：将部分任务卸载到远程设备或云端上进行计算，通过网络返回计算结果；

（3）协同卸载：多个本地设备相互协作，将部分任务卸载到其他本地设备上进行计算，以达到计算效率和能耗的均衡优化；

（4）分布式卸载：将任务分解成多个子任务，通过分布式计算的方式在多个设备

上并行计算，最后合并结果并返回。

根据卸载的目标，任务卸载还可以分为以下两类：

（1）计算卸载：主要目的是将计算密集型任务卸载到云端或其他远程设备上，减少本地设备的计算负担，提高计算性能和能耗效率；

（2）通信卸载：主要目的是将数据密集型任务卸载到云端或其他远程设备上进行处理，减少本地设备的数据传输负担，提高通信性能和能耗效率。

3. 研究的主要内容

作为下一代智慧交通的电动智能汽车有许多的问题需要解决，其中电池问题尤为重要，这关系着汽车的行驶里程。而任务卸载是其中较有希望的一个减少能源消耗，提高能源利用率的一种方式。

但是其中很多问题需要解决，例如，如何衡量计算资源，如何实时的分配任务，如何分配任务更加公平。

车联网中任务卸载主要应用于以下几个方面：

（1）智能驾驶：智能驾驶需要大量的计算和数据处理能力，通过任务卸载技术，可以将一部分计算和数据处理任务卸载到云端或其他设备上，提高计算性能和能耗效率；

（2）车辆监控：车辆监控需要实时收集和处理大量的数据，并进行数据分析和决策，通过任务卸载技术，可以将一部分数据处理任务卸载到云端或其他设备上，提高数据处理效率；

（3）车辆诊断：车辆诊断需要进行大量的数据分析和处理，通过任务卸载技术，可以将一部分数据处理任务卸载到云端或其他设备上，提高诊断效率和精度；

（4）车联网服务：车联网服务需要进行大量的数据计算和处理，通过任务卸载技术，可以将一部分计算任务卸载到云端或其他设备上，提高服务性能和能耗效率。

在最近流行的边缘计算技术中，我们不仅可以将任务分配给附近的边缘服务器，还可以分配给附近资源丰富的用户。

我们建议将任务分配给附近的车辆，以最小化系统的能耗。

我们将移动边缘计算场景车联网中任务卸载的问题抽象为一个数学规划的形式，其中，目标是最小化所有能源消耗的平方，这可以兼顾能耗最小化与公平。

约束包括：所有的任务必须有车辆来做，所有的车辆能够在规定时间内完成任务，以及为了保证通讯的质量，参与资源共享的智能汽车的距离要在一定的距离内。

这个卸载过程形成了一个三层的结构，包括车载云、路边单元的边缘服务器和中央云处理器。边缘服务器是车辆云的控制器。中央云则负责全局的调度。

该问题是一个非线性整数规划问题，没有多项式时间范围内的解决办法。

3.1 网络结构

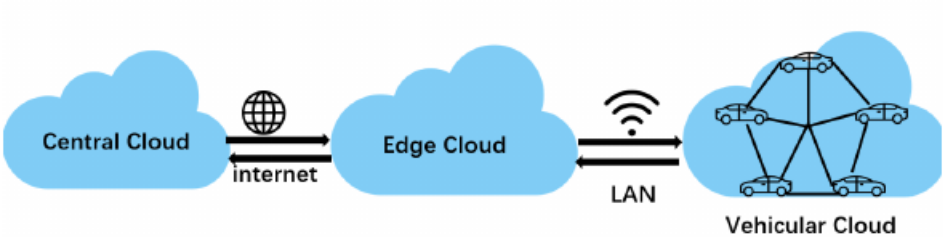

图 1：车联网结构图

在本文中，如图 1 所示，层次结构由车辆云、边缘云和中心云组成。中心云负责全局调度，它执行的任务包括复杂的计算和全局决策。边缘云是车辆云的控制器，负责创建、维护和删除车辆云。车辆云由共享其计算资源的车辆组成。

每辆车都可以访问边缘云并与其进行通讯。车辆可以通过将任务卸载到其他车辆的方式来节省能源。这样可以提高整体资源利用率。

任务卸载过程如下：首先，实时流量信息被传输到云服务器进行分析。实时信息包括目的地、当前位置、时间、车速等。云服务器在汇总数据后，进行大数据分析以获取道路拥堵情况，然后推断未来某个时间段内车辆的位置信息，并将结果返回到车辆附近的边缘云。

这些车辆在未来一段时间内将会参与资源共享。然后，边缘云根据这些信息分配任务，并将车辆分为提供者和请求者，他们在 T 时间段共享资源。

3.2 计算模型

我们将资源共享时间定义为 $\mathcal{T}=\{1,\ldots,T\}$. 。车辆数量定义为 N 。

我们使用元组 $J_{it}=\{r_{it},r'_{it},d_{it}\}$ 来表示时间为 t 时，车辆 i 的任务大小，d_{it} 是任务的数据大小。

任务分为必须在本地执行的任务，以及可以分配出去的任务。这一点并不抽象， 例如有些涉及隐私的任务，或者是要求实时性的任务，就必须在本地执行。

在时间 t 时，r_{it} 时车辆 i 必须要在本地执行的任务，r'_{it} 是车辆 i 能够分配给别人的任务， 并且，它们的值是由需要的指令条数来表示的。

为了更好的描述这个模型，我们定义 $\mathbf{X}_t=\left\{x_{ij}\right\}\in\{0,1\}^{N\times N}$ 为分配矩阵，如果 $x_{ij}=1$，代表车辆i执行车辆 j 所卸载的任务。注意，如果$x_{ii}=1$，车辆 i 所有的任务都在本地

执行。

我们将车辆 i 的容量定义为 C_{it} 在时间为 t 时。在时间 t，当车辆 i 被选为提供者（P），所有需求者需要的资源需要少于这辆车的容量。公式化表达如下：

$$\sum_{j=1}^{N} x_{ij}^{t} \cdot r_{jt}' \le C_{it}, i = 1, \ldots, N \tag{1}$$

一个关键的步骤是怎样衡量车辆的计算资源，在本篇文章中，计算资源定义在每秒能够执行的指令条数上。

公式化表达如下：

$$C_{it} = M_{it} \cdot \Delta T - r_{it} \tag{2}$$

其中，ΔT 是资源共享的持续时间，并且它是一个比建立车辆边缘网络的更小的时间粒度，在 ΔT 秒后，分配矩阵会发生变化。

对于一个单核的 CPU，每秒能够执行的指令条数（m_{it}） 和 CPU 的频率（f_{it}）有如下的关系:

$$M_{it} = v_i \cdot f_{it} + \theta_i \tag{3}$$

其中，v_i 和 θ_i 是待估计的参数。

最后，公式（2）中的 CPU 的容量 C_{it} 在能被如下公式计算：

$$C_{it} = (v_i \cdot f_{it} + \theta_i) \times \Delta T - r_{it} \tag{4}$$

3.3 能耗模型

当我们考虑车辆的能源消耗时，我们将它分为两个部分（1）计算所需要的能量（2）传输所需要的能量。

根据[9][10][11][12]，计算所需要的能耗能够被下列公式计算：

$$E = \lambda_i \cdot f_{it}^{3} \cdot \Delta T \tag{5}$$

其中 f_{it} 是 CPU 在 t 时刻的频率。如果一辆车被选为了需求者（R），这辆车的频率会下降 f_{it}'。因为任务被分配出去了，所以他消耗的能量会减少。消耗的能量能够被以下公式计算：

$$E_i^{save} = (\lambda_i \cdot f_{it}^{3} - \lambda_i \cdot f_i'^{3}) \times \Delta T, \forall i \in R \tag{6}$$

传输能耗和传输的时间有线性关系，传输的时间取决于数据大小和传输速率的比值（b_{ij}）：

$$E = P_0 \cdot \frac{d_{it}}{b_{ij}} \tag{7}$$

其中，P_0 是传输率。

最大的传输速率（b）可以通过香农公式计算：

$$b_{ij} = W \log(1 + \mathrm{SNR}) \tag{8}$$

其中，SNR 是信噪比，W 是频道的带宽。因为它和每一个场景相关，我们将它考虑为一

个常数。

对于每一辆车，接收信息所消耗的能量是：

$$E_i^{rec} = \sum_{j=1, j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\mathrm{SNR})}, \quad i = 1, \ldots, N \tag{9}$$

对于每一辆车，发送信息所需要的能量是：

$$E_i^{send} = \sum_{i=1, i\neq j}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+SNR)}, \quad j = 1, \ldots, N \tag{10}$$

定义 E_t^{blnc} 是车辆 i 在时刻 t 所消耗的能量总和: 可以被这么计算

$$E_{it}^{blnc} = \sum_{j=1, j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\mathrm{SNR})} + \sum_{j=1,, j\neq i}^{N} x_{ji}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+\mathrm{SNR})} + \lambda_i \cdot f_{it}^3 \cdot \Delta T, \quad i = 1, \ldots, N \tag{11}$$

我们算法的设计目标是:最小化所有车辆能量消耗的平方。这个目标既能够考虑最小化能量消耗，又能考虑公平。也就是说平衡了能量消耗和公平。公式表示如下文：

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} (E_{it}^{blnc})^2 \tag{12}$$

该问题最终被建模为如下形式：

$$\min \quad \sum_{t=1}^{T} \sum_{i=1}^{N} (E_{it}^{blnc})^2 \tag{13}$$

$$\text{s.t.} \quad \sum_{j=1, j\neq i}^{N} x_{ij}^t \cdot P_0 \cdot \frac{d_{jt}}{W \log(1+\mathrm{SNR})} + \sum_{j=1, j\neq i}^{N} x_{ji}^t \cdot P_0 \cdot \frac{d_{it}}{W \log(1+\mathrm{SNR})} + \lambda_i \cdot f_{it}^3 \cdot \Delta T, \quad i = 1, \ldots, N \tag{14}$$

$$\sum_{j=1}^{N} x_{ij}^t \cdot r'_{jt} \leq C_{it} \tag{15}$$

$$\sum_{i=1}^{N} x_{ij}^t \geq 1 \tag{16}$$

$$f_{it} \geq 0 \tag{17}$$

$$x_{it}^t \in \{0, 1\} \tag{18}$$

正如前面所提到的，公式（14）说明了对于每一辆车辆 i，需求的总和不能超过他的容量。公式（15）说明了车辆 i 的任务，必须要被执行，无论是本地执行还是分配给其他车辆。

3.4 分配矩阵实例

$$\mathbf{X}=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&1\\0&0&0&1&0\\0&0&0&0&0\end{bmatrix}$$

这是一个分配矩阵的例子，在该实例中，车辆的集合是{A，B，C，D，E}，车辆A，B，D没有卸载自己的任务，或者负载其他车辆的任务。因此他们没有产生传输能耗。车辆C执行了车辆E的任务。每一列的和都大于等于1， 这满足了公式（18）这个约束。

3.5 距离约束

为了保证需求者和提供者之间的服务的质量（QoS），下面的约束必须被满足:

$$\min l_{ij}^{t}<\delta,\quad i=1,\ldots,N,\forall x_{ij}^{t}=1 \tag{19}$$

其中，l_{ij}^{t}是车辆 i 和 j 之间的距离。

4. 研究方案

我们将移动边缘计算场景车联网中任务卸载的问题抽象为一个数学规划的形式，其中，目标是最小化所有能源消耗的平方，这可以兼顾能耗最小化与公平。

约束包括：所有的任务必须有车辆来做，所有的车辆能够在规定时间内完成任务，以及为了保证通讯的质量，参与资源共享的智能汽车的距离要在一定的距离内。

为了更好的描述计算资源和判断是否能够完成任务，我们使用 CPU 的频率来刻画计算资源。并且通过频率来计算小号的能量。

同时，为了满足多重情况的要求，任务可分情况和任务不可分情况都要讨论。

4.1 任务可分情况

当车辆进行某些同质任务时，可以将大任务随意拆成一些小任务，因为任务的粒度太小，因此可以近似看成任务可以随意分配，也就是任务可分的情况。

在任务可分时，根据本地留存的高优先级任务的大小又分为：高于平均任务大小，和低于平均任务大小。

4.2 任务不可分情况

在任务不可分时，这个问题不可能在多项式时间内解决，因此，本文打算通过智能算法来获得一个近似解，来满足靠近最优解和卸载方案计算时间的平衡。

下文将简要介绍常用的智能算法，例如基因算法，粒子群算法，离散粒子群算法，侏儒猫鼬算法。

4.2.1 基因算法（GA）

基因算法（Genetic Algorithm，GA）是一种优化不可导目标函数的数值优化方法。它模仿生物进化过程，使用生物进化概念搜索一个优化问题的全局最优解（极小值或极大值）。

生物只有经过许多世代的不断进化（evolution， 演化），才能更好地完成生存与繁衍的任务。遗传算法也遵循同样的方式，需要随着时间的推移不断成长、演化，最后才能收敛，得到针对某类特定问题的一个或多个解。

遗传算法是从代表问题可能潜在的解集的一个种群（population）开始的，初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度（fitness）大小选择个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover）和变异（mutation），产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码（decoding），可以作为问题近似最优解。

基因算法的基本步骤如下：

（1）初始化种群：随机生成初始种群，种群的个体数和染色体长度等参数需要根据具体问题进行设置；

（2）基因编码：将种群个体表示为染色体，并将染色体中的基因编码为二进制串或其他形式；

（3）评估适应度：根据具体问题，定义适应度函数，评估每个个体的适应度；

（4）选择操作：根据适应度函数的值，选择适应度较高的个体作为父代，并通过选择算子实现选择操作；

（5）交叉操作：在父代中选择两个个体进行交叉操作，生成新的个体；

（6）变异操作：对新生成的个体进行变异操作，产生新的个体；

重复步骤 3~6，直到满足某个停止条件（如达到最大迭代次数、满足一定精度要求等）。

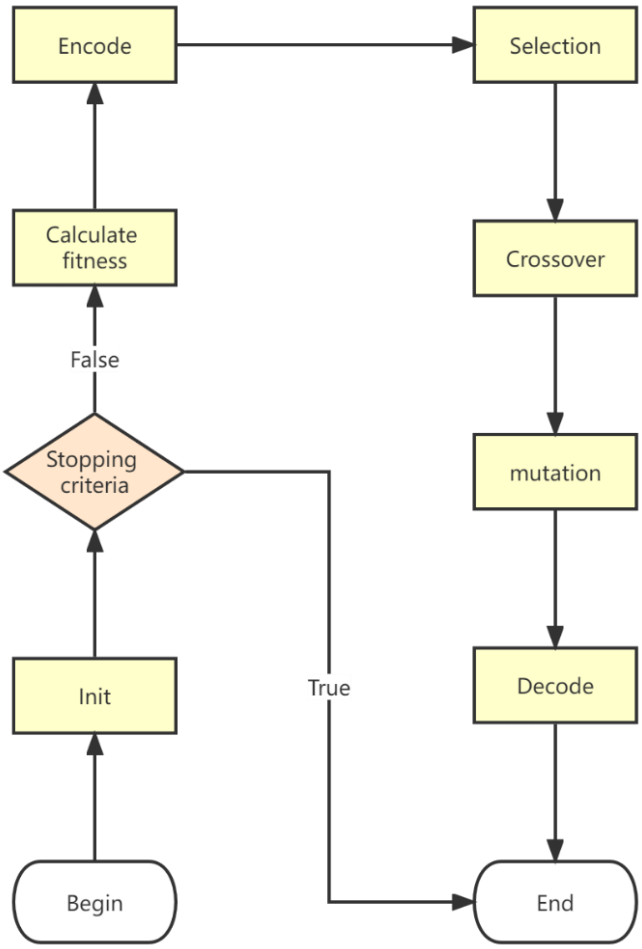

图 2：基因算法流程图

4.2.2 侏儒猫鼬算法（DMOA）

侏儒猫鼬算法（Dwarf Mongoose Optimization Algorithm, DMOA）是一种基于动物行为的启发式优化算法，是由阿德南·马扎里等科学家于 2019 年提出的。该算法的灵感源于侏儒猫鼬的捕猎行为，通过模拟侏儒猫鼬在野外活动的行为，来寻找最优解或次优解。

侏儒猫鼬算法的基本思想是，将每个解看作一个猎物，将侏儒猫鼬看作优化过程中的捕猎者。在每轮迭代中，侏儒猫鼬会根据自身的位置和状态，以及其他侏儒猫鼬的位置和状态，来选择下一步的行动。具体地，侏儒猫鼬算法通过不断地更新每个解的位置和状态，来实现优化过程。每个解的位置表示某个解的位置，状态表示解的质量和可行性。侏儒猫鼬算法在搜索过程中，不断尝试各种不同的位置和状态组合，计算适应度，并据此进行调整，最终找到最优解或次优解。

侏儒猫鼬通常生活在在一个母权制的家庭群体中，由一对阿尔法领导的夫妻生活在一起在猫鼬家族中，在每个年龄组中，雌性都比雄性地位高，而幼崽比它们的哥哥姐姐地位高。这些群体中的劳动分工和利他主义是哺乳动物中最高的，他们根据年龄和性别，不同的猫鼬充当警卫、保姆、攻击捕食者和攻击同种入侵者。

为了模仿它们的觅食行为。我们为优化过程建立了三个社会结构转换模型：阿尔法小组，侦察兵小组，保姆小组和童子军小组。

阿尔法小组出发去觅食。侦察兵寻找下一个睡觉的土堆，因为猫鼬不会回到先前睡觉的土堆，这保证了探险。保姆通常是附属的群体成员，和幼仔呆在一起，定期轮换，让母亲带领其他成员进行日常觅食。她通常在中午和晚上回来给幼崽喂奶。保姆的数量取决于人口规模。

优化从阿尔法小组出发(探索空间)觅食开始，将保姆和幼仔留在巢穴中。一旦找到觅食地点，阿尔法小组就一直觅食到中午，当他们返回交换保姆时,按照保姆交换标准来更换分工。

一旦保姆被交换，他们就不会回到先前觅食的地方，以避免过度放牧。侦察兵会发现一个新的觅食地点，并通知雌性首领，带领家族到达新的地点。保姆交换开始了一个新的探索阶段，紧接着是密集的开发，直到晚上，当部落回到一个新的睡觉的巢穴。

DMO 随机地为给定的优化问题创建一组候选解决方案然后通过，依靠 DMO 的探索和开发能力，模仿侏儒獴的半游牧行为和补偿性适应。

矮猫鼬从一个食物源或睡觉的土堆移动到另一个食物源或睡觉的土堆时，问题搜索空间的不同区域被探索搜索空间的有希望的区域被开发，因为 DMO 是模仿矮猫鼬不能捕捉大的猎物供家庭喂养，但是单独寻找足够的食物来满足个体。

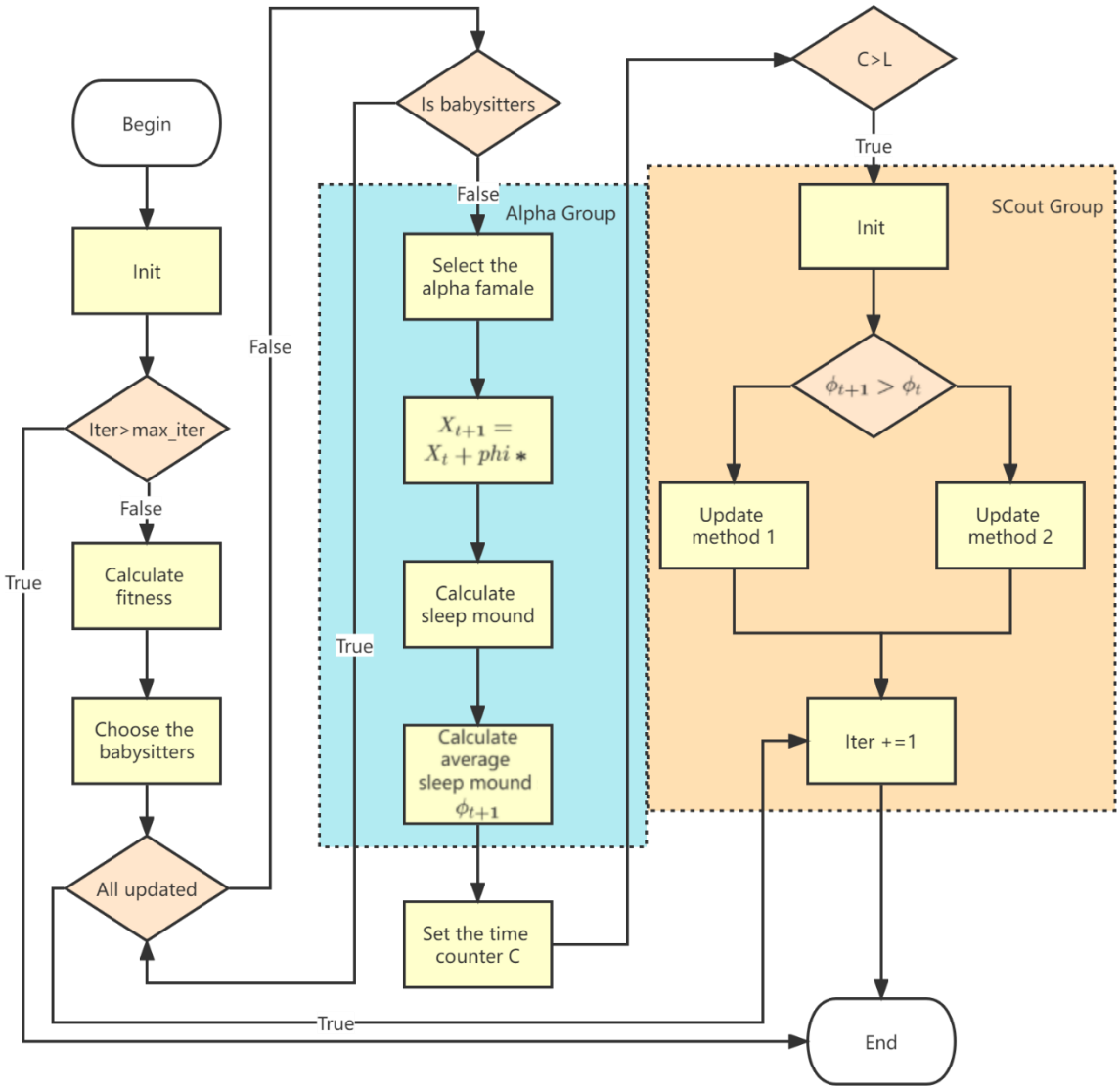

图 3：侏儒猫鼬算法流程图

4.2.3 粒子群算法（PSO）

粒子群算法（Particle Swarm Optimization, PSO）是一种基于群体智能的优化算法，最初由美国社会心理学家 Eberhart 和澳大利亚计算机科学家 Kennedy 于 1995 年提出。该算法源于对鸟群捕食行为的研究，通过模拟鸟群在搜索食物时的行为，寻找最优解或次优解。目前，粒子群算法已经被广泛应用于各种优化问题。

粒子群算法的基本思想是，将每个候选解看作一个粒子，在解空间中移动，并根据每个粒子的历史最优位置和全局最优位置进行调整，最终达到找到最优解或次优解的目的。具体地，粒子群算法通过不断更新每个粒子的位置和速度，来实现优化过

程。每个粒子的位置表示某个解的位置，速度表示粒子在解空间中搜索的方向和速度。每个粒子根据当前的位置和速度来计算适应度，并根据历史最优位置和全局最优位置进行调整。粒子群算法不断迭代直到满足停止条件，得到最优解或次优解。粒子群算法的基本流程如下：

（1）初始化：设置初始粒子位置和速度，以及其他算法参数。

（2）计算适应度：对每个粒子计算适应度函数值。

（3）更新粒子速度和位置：根据当前位置、历史最优位置和全局最优位置，更新粒子速度和位置。

（4）判断是否满足停止条件：如果满足，算法终止，否则，返回步骤 2 继续迭代。

（5）输出结果：输出最优解或次优解。

在第三步中，粒子群算法更新公式如下：

$$v_i = \omega \times v_i + c_1 \times \text{rand}() \times \left(\text{pbest}_i - x_i \right) + c_2 \times \text{rand}() \times \left(\text{gbest}_i - x_i \right)$$

其中，pbest 是本次迭代中最优的位置，gbest 是全局中最优的位置，ω 是 2 惯性因子, 值为非负

4.2.3 离散粒子群算法（DPSO）

离散粒子群算法（Discrete Particle Swarm Optimization, DPSO）是一种基于粒子群算法的优化算法。与粒子群算法相似，离散粒子群算法也是通过不断地迭代更新每个粒子的速度和位置，来寻找最优解或次优解。但不同之处在于，离散粒子群算法主要用于离散优化问题，例如，整数规划、组合优化、图论等问题。

离散粒子群算法的基本思想是，在解空间中将每个离散值看作一个状态，将整个解空间看作一个状态空间，粒子在状态空间中搜索最优解或次优解。每个粒子的状态表示为解的一组离散值，速度表示搜索的方向和速度，适应度表示解的质量。离散粒子群算法通过不断更新每个粒子的速度和位置，来实现优化过程。每个粒子在搜索过程中，不断尝试各种不同的离散值组合，计算适应度，并据此进行调整，最终找到最优解或次优解。

离散粒子群算法在粒子群算法的基础上更改了符号的运算规则，也就是重新定义了运算符。

5. 论文可能的创新点及预期成果

5.1 论文创新点

由于行驶过程中需要处理的数据量过大，中央处理器将消耗相当多的能量，这对行驶里程有很大影响。一些研究建议：

(1) 将任务发送到移动边缘计算服务器以节省能源。但如果所有数据都上传到云服务器进行处理，响应时间可能太长，无法满足低延迟要求。

(2) 将重点放在车辆边缘云中的虚拟机资源分配上。

(3) 考虑了动态需求和资源约束，并将所有任务划分为四种类型的待定列表。然后，每个列表中的任务将根据其功能卸载到不同的节点。

他们的工作没有考虑将任务卸载到其他车辆中，具有一定的局限性。我们的创新之处如下:

（1）我们考虑将任务卸载到附近的其他车辆中，这个过程由边缘服务器控制。

（2）我们将移动边缘计算场景车联网中任务卸载的问题抽象为一个数学规划的形式，以频率为基础计算能耗。为了更好的描述计算资源和判断是否能够完成任务，我们使用 CPU 的频率来刻画计算资源。其中，目标是最小化所有能源消耗的平方，这可以兼顾能耗最小化与公平。

5.2 预期成果

预期在国内外期刊及学术会议上发表论文 1-2 篇。其中 SCI、EI 收录 1 篇以上。

6. 论文前期已经完成的研究工作，和可能存在的相关问题

6.1 已经完成的研究工作

研究人主要从事智能算法研究等相关学习和研究，针对于所提出的车联网中任务卸载技术的研究，已在前期进行了一定的学习与研究。并已经开展了一下相关的前期研究和验证工作：

（1）本人了解已有的相关研究工作，并分析当前研究的现状。对相关工作的研究有一定的相关研究工作的知识积累。

（1）本人在智能算法的实现中，已经复现了整数粒子群算法，侏儒猫鼬算法。

6.2. 可能存在的问题

尽管已有相关研究理论和经验，并初步完成了相关的前期工作。但是由于任务的复杂性，因此在研究过程中，可能会遇到如下问题：

（1） 智能算法不收敛，或者是结果不好

（2） 可能改造的智能算法解决问题的效果不好，不适用于这个问题。

7. 论文进度工作安排

所提研究计划为一年，预期安排如下：

2023 年 3 月-2023 年 7 月　进行仿真环境配置：使用云服务器作为我们模型中虚拟服务提供商。将建模的问题使用启发式的方法解决，也就是确定问题的解决办法。

2023 年 8 月-2023 年 10 月　将问题解决办法用编程语言实现，并进行仿真实验。分析实验数据，将实验结果使用图表的方式进行描述，进行论文的撰写。

2023 年 11 月-2024 年 3 月　将得到的实验数据生成图表添加到我们的论文中，并且进一步完善我们的论文以进行投稿。

8. 参考文献：

[1] Zachary P Cano et al. “Batteries and fuel cells for emerging electric vehicle markets”. In: Nature Energy 3.4 (2018)， pp. 279–289.

[2] T. Constantijn J. Romijn et al. “A Distributed Op- timization Approach for Complete Vehicle Energy Management”. In: IEEE Transactions on Control Systems Technology 27.3 (2019)， pp. 964–980. doi: 10.1109/TCST.2018.2789464.

[3] Dalibor Barta et al. “Possibility of increasing vehicle energy balance using coasting”. In: Advances in Sci- ence and Technology. Research Journal 12.1 (2018).

[4] Virgilios Passas et al. “V2MEC: Low-Latency MEC for Vehicular Networks in 5G Disaggregated Architectures”. In: 2021 IEEE Wireless Communications and Networking Conference (WCNC). IEEE. 2021， pp. 1–6.

[5] Chen-Feng Liu et al. “Dynamic Task Offloading and Resource Allocation for Ultra-Reliable Low-Latency Edge Computing”. In: IEEE Transactions on Com- munications 67.6 (June 2019). Conference Name: IEEE Transactions on Communications， pp. 4132– 4150. issn: 1558-0857. doi: 10.1109/TCOMM.2019. 2898573.

[6] Rong Yu et al. “Toward cloud-based vehicular net- works with efficient resource management”. In: Ieee Network 27.5 (2013)， pp. 48–55.

[7] Chunhui Liu et al. “Adaptive Offloading for Time- Critical Tasks in Heterogeneous Internet of Vehicles”. In: IEEE Internet of Things Journal 7.9 (2020)， pp. 7999–8011. doi: 10.1109/JIOT.2020.2997720.

[8] Kyong Hoon Kim， Rajkumar Buyya， and Jong Kim. “Power Aware Scheduling of Bag-of-Tasks Applica- tions with Deadline Constraints on DVS-enabled Clusters”. In: Seventh IEEE International Sympo- sium on Cluster Computing and the Grid (CCGrid ’07). 2007， pp. 541–548. doi: 10 . 1109 / CCGRID . 2007.85.

[9] R. Ge， Xizhou Feng， and K.W. Cameron. “Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power- aware Clusters”. In: SC ’05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. 2005， pp. 34–34. doi: 10.1109/SC.2005.57.

[10] E. N. (Mootaz) Elnozahy， Michael Kistler， and Ramakrishnan Rajamony. “Energy-Efficient Server Clusters”. In: Power-Aware Computer Systems. Ed. by Babak Falsafi and T. N. Vijaykumar. Berlin， Hei- delberg: Springer Berlin Heidelberg， 2003， pp. 179– 197. isbn: 978-3-540-36612-6.

[11] J. Li and J.F. Martinez. “Dynamic power- performance adaptation of parallel computation on chip multiprocessors”. In: The Twelfth International Symposium on High-Performance Computer Architecture， 2006. 2006， pp. 77–87. doi: 10.1109/HPCA.2006.1598114.

[12] C. Piguet， C. Schuster， and J.-L. Nagel. “Optimizing architecture activity and logic depth for static and dynamic power reduction”. In: The 2nd Annual IEEE Northeast Workshop on Circuits and Systems， 2004. NEWCAS 2004. 2004， pp. 41–44. doi: 10 . 1109 / NEWCAS.2004.1359011.

[13] Jeffrey R. Sampson. “Adaptation in Natural and Artificial Systems (John H. Holland)”. In: SIAM Review 18.3 (1976)， pp. 529–530. doi: 10 . 1137 / 1018105. eprint: https://doi.org/10.1137/1018105. url: https://doi.org/10.1137/1018105.

[14] Vincent Roberge， Mohammed Tarbouchi， and Gilles Labonté. “Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning”. In: IEEE Transactions on industrial informatics 9.1 (2012)， pp. 132–141.

[15] Yourim Yoon and Yong-Hyuk Kim. “An Efficient Genetic Algorithm for Maximum Coverage Deploy- ment in Wireless Sensor Networks”. In: IEEE Trans- actions on Cybernetics 43.5 (2013)， pp. 1473–1483. doi: 10.1109/TCYB.2013.2250955.

[16] Anand Kannan et al. “Genetic algorithm based feature selection algorithm for effective intrusion detection in cloud networks”. In: 2012 IEEE 12th In- ternational Conference on Data Mining Workshops. IEEE. 2012， pp. 416–423.

[17] R Sivaraj and T Ravichandran. “A review of selec- tion methods in genetic algorithm”. In: International journal of engineering science and technology 3.5 (2011)， pp. 3792–3797.

[18] Cortex-A57. url: https : / / developer . arm . com / Processors/Cortex-A57 (visited on 07/20/2022).

[19] Tayebeh Bahreini， Marco Brocanelli， and Daniel Grosu. “VECMAN: A Framework for Energy-Aware Resource Management in Vehicular Edge Comput- ing Systems”. In: IEEE Transactions on Mobile

Computing (2021)， pp. 1–1. doi: 10 . 1109 / TMC . 2021.3089338.

[20] Zhigang Xu et al. “DSRC versus 4G-LTE for con- nected vehicle applications: A study on field experiments of vehicular communication performance”. In: Journal of advanced transportation 2017 (2017).