DOI:10.19651/j.cnki.emt.2106482

基于边缘计算的配网在线监测任务分配机制*

刘 哲[1] 周 磊[2] 郑焕坤[3] 陈长金[1] 闫佳文[1]
(1.国网河北省电力有限公司培训中心 石家庄 050000;2.北京科东电力控制系统有限责任公司 北京 100192;
3.华北电力大学 保定 071000)

摘 要:边缘计算在物联网领域有着广阔的应用前景,尤其是智能配电网中的电缆实时在线监测业务。然而,由于边缘节点的资源和能力相对有限,难以高度全面地满足电缆在线监测任务的高实时性要求。对此,提出了一种基于边缘计算的有线实时在线监测业务任务分配机制,在有效利用和优化边缘节点的资源和能力的基础上,进行动态任务分配。考虑到配网的线性分布特性、边缘节点状态、任务处理开销以及延迟敏感任务的调度策略,建立了基于边缘计算的任务分配模型。其次,提出了一种基于任务分配策略的优化方法。仿真结果表明,所提出的任务分配机制能够有效降低配网有线实时在线监测业务的平均延迟,进一步提高智能配网的安全性和可靠性。

关键词:在线监测;边缘计算;粒子群算法;任务分配
中图分类号:TM71 **文献标识码**:A **国家标准学科分类代码**:510.8050

Distribution network online monitoring task allocation mechanism based on edge computing

Liu Zhe[1] Zhou Lei[2] Zheng Huankun[3] Chen Changjin[2] Yan Jiawen[1]
(1. State Grid Hebei Electric Power Co., Ltd. Training Center, Shijiazhuang 050000, China;
2. Beijing Kedong Electric Power Control System Co., Ltd., Beijing 100192, China;
3. North China Electric Power University, Baoding 071000, China)

Abstract: Edge computing has a promising application prospect in the field of Internet of Things, especially cable real-time online monitoring business in a smart grid. However, for the relatively limited resources and capabilities of the edge nodes, such as computing resource and storage resource, it is hard to highly and comprehensively satisfy the high real-time requirements of cable online monitoring tasks. To solve this problem, effectively dynamic task allocation is needed on the basis of efficient utilization and optimization of resource and capabilities of the edge nodes. In this article, a task allocation mechanism for cable real-time online monitoring business based on edge computing is proposed. First, considering the linear distribution characteristics of the cable, the statuses of edge nodes, the processing overhead of tasks, and the scheduling strategy of delay-sensitive tasks, we establish a task allocation model based on edge computing. Second, a task allocation strategy based on improved discrete particles warm optimization is proposed. In our strategy, we focus on the task queuing problem in edge nodes and the optimized task allocation problem among edge nodes. Simulation results show that the task allocation mechanism proposed in this article can effectively reduce the average delay of cable real-time online monitoring businesses, and further improve the security and reliability of the smart grid.

Keywords: real-time online monitoring; edge computing; particle swarm algorithm; task allocation

0 引 言

随着智能配电网建设的不断推进,对电力通信系统提出了更高的要求,它不但要满足大量数据信息流的接入实现实时采集、分析和管控,同时还需适应未来各种设备、应用、数据的不断增长,边缘计算在电力通信领域具有广泛的应用前景,特别是对于有线实时在线监测业务[1-2]。边缘计算有效降低了电网的不确定性,减少了配网电力电缆实时

收稿日期:2021-04-22
* 基金项目:国家电网配网带电作业仿真平台开发(03025724)、中央高校基本科研业务费专项资金(2014MS87)资助

在线监测的时延,进一步提高了智能配电网的安全性和可靠性[3]。基于边缘计算的电缆实时在线监测业务的网络架构包括 3 层:云层、边缘层和电缆实时在线监测终端接入层[4]。在该网络中,电缆实时在线监测设备负责收集数据并向边缘节点发送任务。大多数任务将在离它们更近的边缘节点上处理。此外,云层中的边缘节点管理和控制系统可以部署在边缘云附近,这确保了高效的任务分配。然而,边缘节点的资源和能力相对有限[5]。同时,大多数有线实时在线监控业务对延迟敏感。如果应用不合理的任务分配机制,边缘节点的任务负载会不平衡,有些任务无法及时完成。此外,不合理的任务分配机制容易造成边缘节点资源消耗严重。因此,需要合理的任务分配机制来充分利用边缘节点资源[6-7],从而保证延迟敏感任务的完成,减少总延迟,提高智能电网的安全性和可靠性[8]。

目前,边缘计算中任务分配策略算法的研究主要集中于基于博弈论任务分配、面向人工智能的任务分配算法和集中任务分配算法,文献[9-10]主要对基于边缘计算的集中式任务分配策略进行研究,通过合理的分配任务处理和计算资源能够显著降低边缘节点的开销。考虑到多用户的特点,文献[11]提出了一种基于博弈算法的分布式任务分配策略,有效提高了系统总体传输性能,降低时延。面向多小区业务覆盖模型,为提高网络的存储空间,扩大服务范围,文献[12]提出了一种基于博弈论的移动边缘计算任务调度策略,该方法明显改善了任务缓存时延对系统的负面影响。文献[13-14]重点将人工智能和大数据技术引入边缘计算任务分配和缓存机制中,文献[13]采用强化学习和深度强化学习的方式综合优化边缘计算节点任务分配机制,在降低网络通信时延的基础上,也有效地避免了网络拥塞的产生。文献[14]提出了基于边端协同的深度学习按需加速框架,通过协同优化模型分割和模型精简策略,实现时延约束下的高精度模型推理.

通过上述研究分析发现,尽管通过上述传输机制能够明显的提高边缘计算设备内容缓存性能,降低网络通信时延。但这些算法往往只能获得局部最优解,尤其是集中任务分配策略[15],同时不利于提高系统性能,很难保证延迟敏感任务的完成,容易局部陷入收敛,限制任务分配效率的提高[16-17]。此外,这些算法,任务的排队问题往往被忽略,同时并不关注陪配网有线实时在线监测业务的任务分配。为充分利用实时在线监控业务和边缘计算的特点,需要一种新的针对这种独特场景的任务分配策略。对此,本文研究了基于边缘计算的有线实时在线监测业务的任务分配机制。首先,给出了有线实时在线监测业务的边缘计算任务分配模型。综合考虑电缆的线性分布特性、边缘节点的实时状态、任务的处理开销以及延迟敏感任务的处理策略,提出了有线实时在线监测业务边缘计算任务分配策略。在任务分配策略中,将边缘节点的任务排队问题进行重点考虑,并提出了一种多优先级任务排队算法。同时,提出了解决任务分配策略的粒子群优化算法。仿真结果表明,本文所提的算法在降低边缘节点的时延的同时具有较好的收敛性。

1　基于边缘计算的任务分配模型

本节重点对基于边缘计算的配网电力电缆有线实时在线监测业务的任务分配模型进行分析。如图 1 所示,电缆线路上分布有多个边缘节点,每个节点覆盖有效覆盖范围内的一些设备,设备周期性地或随机地生成任务。任务将在边缘节点排队,直到被处理。可以将任务分配给有效覆盖范围内的边缘节点。

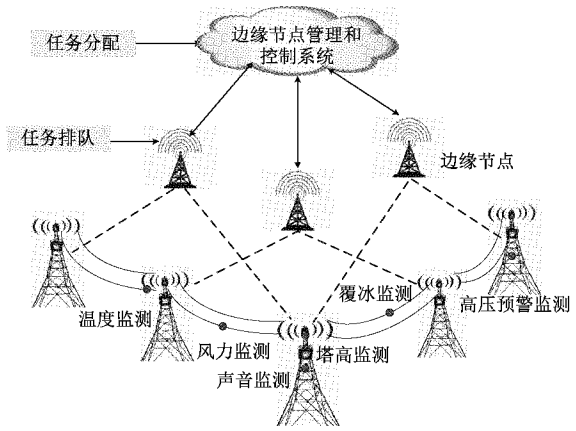

图 1　面向配网的任务分配场景

面向配网电力电缆线路监测任务的分配模型如图 2 所示,根据它做出如下假设。

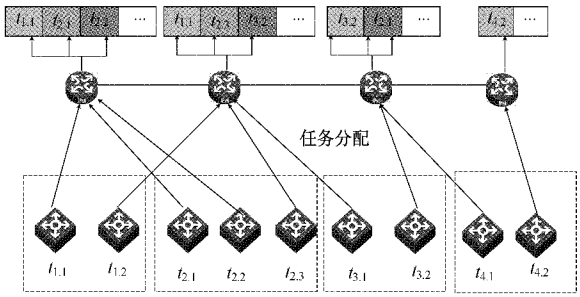

图 2　面向配网电力电缆线路监测任务的分配模型

1)基于配网电力电缆的线性特性,传输电缆可以近似为一条直线,相邻边缘节点之间的距离相同。

2)考虑到电力系统的实际运行环境,假设边缘节点的计算资源和存储资源相对足够,同时,两个相邻节点之间的覆盖存在部分重叠,保证每个任务都能被边缘节点采集并处理。

3)有许多相互依赖的任务。在这种情况下,相互依赖的任务来自同一个设备,并且有相关的延迟要求。考虑到大多数任务是延迟敏感的,来自同一设备的相互依赖的任务将由边缘节点识别并在同一节点上处理。

1.1　通用任务分配模型

设一根电缆上的边节点数为 N,所有节点任务总数围

为 M。对于节点 i 的任务数为 n_i。节点 i 下的任务 j 为 s_{ij}，包括如下任务参数：任务开始时间 t_{sij}、任务截止时间 t_{eij}、任务运行成本 w_{ij}、任务分配节点 k。

对于边缘节点 i，其计算能力设置为 v_i。将一个任务从节点 i 分配到节点 k，节点 i 分配给节点 k 的传输延迟为 t_{ik}，任务分配后，节点 k 上 s_{ij} 任务队列前 t 时刻的总开销为 $W_{kij}(t)$。

首先，应该计算任务的延迟时间。对于电缆实时在线监控设备生成的任务，延迟时间包括发送延迟、传输延迟、排队延迟和处理延迟。考虑到发送延迟非常小，假设发送延迟为 0。假设排队延迟为 $W_{kij}(t)(m\tau+t_{i,k})/v_k$，处理延迟为 w_{ij}/v_k。

假设每执行一次任务分配占用时间为 τ，在整数时间点 $m\tau$，完成的节点任务分配。假设任务 s_{ij} 分配给节点 k。任务的总延迟如下：

$$\delta t_{ij}=m\tau-t_{sij}+t_{i,k}+\frac{W_{kij}(t)(m\tau+t_{i,k})}{v_k}+\frac{w_{ij}}{v_k} \tag{1}$$

式(1)是任务的总延迟。任务的传输延迟为 $t_{i,k}$，考虑到实际情况，如果节点 i 和节点 k 是不同的节点，则给出如下约束：

$$t_{i,k}>t_{i,i} \tag{2}$$

所有任务的总延迟和平均任务延迟可以表示为：

$$\delta T_s=\sum_{i=1}^{i=N}\sum_{j=1}^{j=n_i}\left(m\tau-t_{sij}+t_{i,k}+\frac{W_{kij}(t)(m\tau+t_{i,k})}{v_k}+\frac{w_{ij}}{v_k}\right) \tag{3}$$

$$\delta T_{avg}=\frac{\delta T_s}{M} \tag{4}$$

考虑到任务应在期限内完成，给出了如下约束条件：

$$\delta t_{ij}\leqslant t_{eij}-t_{sij} \tag{5}$$

考虑到节点的分布，给出如下约束：

$$\begin{cases}i-1\leqslant k\leqslant i+1\\1\leqslant k\leqslant N\end{cases} \tag{6}$$

因此，任务分配问题可以表述如下：

$$\text{minimize } \delta T_{avg} \tag{7}$$

$$constraint:\delta t_{ij}\leqslant t_{eij}-t_{sij} \tag{8}$$

$$\begin{cases}i-1\leqslant k\leqslant i+1\\1\leqslant k\leqslant N\end{cases} \tag{9}$$

1.2　抢占式任务分配模型

考虑到存在一些延迟敏感的任务，在这一部分，给出了实时在线监控业务边缘计算抢占式任务分配模型。引入了 3 个变量：调度时间阈值 t_d、调度时间开销 t_p 以及任务分配后节点 i 是否执行优先调度标志 p_i（$p_i=0$ 表示节点 i 没有执行优先调度，$p_i=1$ 表示节点 i 执行了优先调度）。

对于优先任务，将首先处理它，并生成排队延迟等于 0，需抢占调度时间。对于最初位于节点 i 的任务 j，如果它在时间 $m\tau$ 被分配给节点 k，抢先调度约束如下：

$$t_{eij}-(m\tau+t_{i,k})\leqslant t_d \tag{10}$$

任务的总延迟可以表示为：

$$\delta t_{ij}=\begin{cases}m\tau-t_{sij}+t_{i,k}+t_p+\dfrac{w_{ij}}{v_k}, t_{eij}-(m\tau+t_{i,k})\leqslant t_d\\m\tau-t_{sij}+t_{i,k}+t_p+\dfrac{w_{ij}}{v_k}+\dfrac{W_{kij}(m\tau+t_{i,k}+p_it_p)}{v_k},\\t_{eij}-(m\tau+t_{i,k})>t_d\end{cases} \tag{11}$$

所有任务的总延迟和平均任务延迟可以表示为：

$$\delta T_s=\sum_{i=1}^{i=N}\sum_{j=1}^{j=n_i}\delta t_{ij} \tag{12}$$

$$\delta T_{avg}=\frac{\delta T_s}{M}=\sum_{i=1}^{i=N}\sum_{j=1}^{j=n_i}\frac{\delta t_{ij}}{M} \tag{13}$$

此时的时间约束和节点分布约束为：

$$\delta t_{ij}\leqslant t_{eij}-t_{sij} \tag{14}$$

$$\begin{cases}i-1\leqslant k\leqslant i+1\\1\leqslant k\leqslant N\end{cases} \tag{15}$$

因此，任务分配问题可表述如下：

$$\text{minimize } \frac{\delta T_s}{M}=\sum_{i=1}^{i=N}\sum_{j=1}^{j=n_i}\frac{\delta t_{ij}}{M} \tag{16}$$

$$constraint:\delta t_{ij}\leqslant t_{eij}-t_{sij} \tag{17}$$

$$\begin{cases}i-1\leqslant k\leqslant i+1\\1\leqslant k\leqslant N\end{cases} \tag{18}$$

2　基于边缘计算的任务分配策略

在本节中，将给出基于 IPSO 算法的任务分配策略。首先，提出了一种 MPTQ 算法来解决任务排队问题。其次，给出了通用任务分配策略和优先任务分配策略。针对粒子群优化算法的不足，结合配网线路实时在线监测业务的特点，提出了一种 IPSO 算法。

2.1　MPTQ 排队算法

排队延迟会影响任务的总延迟。边缘节点上不同的排队方式对任务延迟有很大影响。基于最早截止时间优先级，考虑任务的截止时间和处理开销，提出了一种 MPTQ 算法。假设边缘节点的运行速度为 v，在时间 t_f，有一个长度为 M 的任务队列 s。假设任务开始时间为 t_{si}，任务截止时间为 t_{ei}，任务运行成本为 w_i。边缘节点请求的平均延迟为：

$$\delta T_{avg}=\frac{1}{M}\sum_{i=1}^{i=N}\left(t_f-t_{si}+\sum_{j=i}^{j=N}\frac{w_i}{v_k}\right) \tag{19}$$

时间约束如下：

$$t_f-t_{si}<t_{ei} \tag{20}$$

当前的目标是通过在满足式(20)的条件下改变排队顺序来最小化式(19)的任务平均延迟。本文提出了一种混合优先级排队算法。首先，根据截止日期对任务队列进行排序。之后连续比较两个相邻任务的处理开销。如果后方任务成本小，两个任务在队列中互换，任务仍然可以满足延

迟。需求交换任务位置,并递归执行任务队列上的交换函数。MPTQ 算法如算法 1 所示。假设一个任务队列的数量为 n,算法中的函数 $Taskswap$ 是一个双循环,所以算法的时间复杂度为 $O(n^2)$。

算法 1 MPTQ 算法

1. 变量的初始化:任务节点数 M,任务队列 s
2. 输出:s
3. 调用函数任务分配 $Taskswap(s,l_1,l_2)$
4. 判断是否满足 $l_1>l_2$,如果满足,结束,否则执行任务交换 $Taskswap\ (s,l_1,\ l_2-1)$。
5. 判断节点 i 是否在(l_1,l_2)范围内,如果满足则继续,否则跳出循环,结束。
6. 如果 $s[i]\cdot w>s[i+1]\cdot w$ 同时任务 $s[i]$ 和 $s[i+1]$ 仍可在交换后的结束时间内完成,则交换 $s[i]$ 和 $s[i+1]$ 否则执行任务交换 $Taskswap\ (s,l_1,i-1)$, $Taskswap\ (s,\ i+1,\ l_1)$
7. 函数结束
8. 按照任务 s 结束时间进行升序排列
9. 返回 s

2.2　基于 IPSO 算法的通用任务分配策略

在这一部分,本文提出了基于 IPSO 算法的通用任务分配策略。粒子群优化算法容易局部收敛,而本文研究的任务分配问题是一个离散问题,这意味着经典的粒子群优化算法很难取得好的效果。改进方案如下。

1)为了解决模型的离散化问题,提出了粒子群算法的离散化策略。

2)结合遗传算法的变异思想,粒子进行变异后达到迭代,有助于提高算法的求解能力。

3)为了扩大粒子解的范围,解决局部收敛问题,将种群划分为若干个子种群,记录每个子种群的最大适应度和最优位置。这些子种群将并行执行,这有助于提高算法的稳定性。

4)为了提高种群的多样性,粒子被分为两类:导航粒子和探索粒子。每个亚群都有两种粒子。两种粒子的速度迭代公式会有所不同。

IPSO 算法描述如下。

本文需要解一个向量 $\vec{x}=(x_1,x_2,\cdots,x_n)$ 来最大化函数 $f(\vec{x})=f(x_1,x_2,\cdots,x_n)$。设定 q 个子种群,每个子种群大小为 m,同时每个子群包含导航粒子和探索粒子。位置矩阵 $\boldsymbol{X}_j(i)$ 和速度矩阵 $\boldsymbol{V}_j(i)$ 如下:

$$\boldsymbol{X}_k(i)=\begin{pmatrix} x_{11}(i) & x_{12}(i) & \cdots & x_{1n}(i) \\ x_{21}(i) & x_{22}(i) & \cdots & x_{2n}(i) \\ & & \cdots & \\ x_{m1}(i) & x_{m2}(i) & \cdots & x_{mn}(i) \end{pmatrix} \tag{21}$$

$$\boldsymbol{V}_k(i)=\begin{pmatrix} v_{11}(i) & v_{12}(i) & \cdots & v_{1n}(i) \\ v_{21}(i) & v_{22}(i) & \cdots & v_{2n}(i) \\ & & \cdots & \\ v_{m1}(i) & v_{m2}(i) & \cdots & v_{mn}(i) \end{pmatrix} \tag{22}$$

设迭代总数为 $3L$,对于第 k 个子种群中的粒子 j,在第 i 次迭代时,速度更新方程如下:

$$\vec{v}_j(i)=\begin{cases} w\vec{v}_j(i-1)+c_1r_1[\vec{p}_j(i-1)-\vec{x}_j(i-1)]+ \\ c_3r_3[\vec{G}(i-1)+\vec{x}_j(i-1)],type=1 \\ w\vec{v}_j(i-1)+c_1r_1[\vec{p}_j(i-1)-\vec{x}_j(i-1)]+ \\ c_2r_2[\vec{g}_k(i-1)+\vec{x}_j(i-1)],type=2 \end{cases} \tag{23}$$

在式(23)中,$type=1$ 表示粒子是导航粒子,$type=2$ 表示粒子是探索粒子。w 是粒子的惯性系数,c_1 是个人最优加速因子,c_2 是子种群最优加速因子,c_3 是种群最优加速因子,$\vec{p}_j(i-1)$ 是第 i 次迭代前的粒子最佳位置。$\vec{g}_k(i-1)$ 是第 i 次迭代之前的第 k 个子种群最佳位置,并且 $\vec{G}(i-1)$ 是第 i 次迭代前的种群最佳位置。r_1、r_2 和 r_3 是随机系数,通常介于 0 和 1 之间。

对于本文中的任务分配问题,如果任务最初位于节点 i,它可以被分配给节点 i 、节点 $i-1$ 和节点 $i+1$。任务的初始节点是已知的,粒子位置 x 可以取为 1、0 和 1。因此,使用概率函数将速度转换为从 0～1 的数字。概率函数如下:

$$\vec{p}(i)=\frac{4}{3(1+\mathrm{e}^{-(|\vec{v}(i)|-1)})}-\frac{1}{3} \tag{24}$$

q 维度的粒子位置更新公式如下:

$$x_{jq}(i)=\begin{cases} x_{jq}(i)-1,r>x_{jq}(i),v<0,x_{jq}(i)\neq -1 \\ x_{jq}(i),r\leqslant x_{jq}(i) \\ x_{jq}(i)+1,r>x_{jq}(i),v>0,x_{jq}(i)\neq 1 \end{cases} \tag{25}$$

通过引入突变的思想,在粒子速度上增加了随机扰动:

$$\vec{v}_j(i)=\begin{cases} \vec{v}_j(i)+r-\dfrac{1}{2}, & type=1 \\ \vec{v}_j(i)+2r-1, & type=2 \end{cases} \tag{26}$$

在式(26)中,$type=1$ 表示粒子是导航粒子,$type=2$ 表示粒子是探索粒子。r 是随机系数,通常在 0 和 1 之间。

现在,描述 CTA-IPSO 算法的过程。首先,给出了算法运行中使用的变量。假设任务总数为 M,节点数为 N。为了使用该算法,所有要分配的任务被分组到长度为 M 的向量中。式(27)描述了向量:

$$\vec{s}=(s_1,s_2,\cdots,s_M) \tag{27}$$

对于向量中的第 j 个任务,假设任务开始时间设置为 t_{sj},任务截止时间为 t_{ej},任务运行成本为 w_j,任务初始节

点为 i_j，任务分配节点为 k_j。节点 k 上的第 j 个任务在时间 t 前的总成本为 $W_{(k_j)}(t)$。任务 j 的排队延迟为 t_{qj}。节点 i 的计算能力为 v_i，假设执行一次任务分配的时间间隔为 τ。

节点任务分配向量和总成本如下：

$$\vec{\boldsymbol{k}} = (k_1, k_2, k_3, \cdots, k_M) \tag{28}$$

$$\vec{\boldsymbol{W}} = [W_1(m\tau), W_2(m\tau), \cdots, W_N(m\tau)] \tag{29}$$

在时间 $t(m\tau < t < m\tau + \tau)$，节点 i 的总成本和任务 j 的排队延迟如下：

$$W_i(t) = \begin{cases} W_i(m\tau) - v_i t, & W_i(m\tau) > v_i t \\ 0, & W_i(m\tau) < v_i t \end{cases} \tag{30}$$

$$t_{qj} = \frac{W_{k_j}(m\tau + t_{i_j k_j}) + w_j}{v_{k_j}} \tag{31}$$

因此，适应度函数和延迟约束如下：

$$f(\vec{\boldsymbol{k}}) = \frac{M}{\sum_{j=0}^{M}(m\tau - t_{sj} + t_{i_j k_j} + t_{qj})} \tag{32}$$

$$\forall j \in M, t_{i_j k_j} + t_{qj} \leqslant t_{ej} - m\tau \tag{33}$$

可以看出，算法的目的是最大化适应度函数，平均延迟的倒数是为了便于处理任务的延迟约束。此外，一旦任务不满足延迟约束，将适应度函数限制为 0。

考虑到节点的分布，给出了如下约束：

$$\begin{cases} i_j - 1 \leqslant k_j \leqslant i_j + 1 \\ 1 \leqslant k_j \leqslant N \end{cases} \tag{34}$$

为处理约束(28)，粒子位置矩阵为任务的节点偏移量。

$$\boldsymbol{X} = \begin{pmatrix} \vec{\boldsymbol{k}}_1 - \vec{\boldsymbol{i}}_1 \\ \vec{\boldsymbol{k}}_2 - \vec{\boldsymbol{i}}_2 \\ \cdots \\ \vec{\boldsymbol{k}}_m - \vec{\boldsymbol{i}}_m \end{pmatrix} \tag{35}$$

粒子的位置只能在{−1,0,1}中获取。只要得到偏移值，就可以从任务的初始节点得到最终的任务分配向量。

算法的执行步骤如下。

1)输入待分配的节点信息和任务队列，初始化与粒子群算法相关的参数。

2)根据任务队列获取粒子的初始位置矩阵和初始速度矩阵。

3)使用改进的离散粒子群算法(improved discrete particle swarm optimization)的循环迭代解。在每个循环中，更新粒子、子种群和种群的最大适应度和最大适应度位置。在适应度函数的计算中，首先通过混合优先级任务排队算法对每个节点的任务进行排队。然后根据迭代公式更新粒子速度和位置。

4)循环结束，计算最小平均延迟 $T_{\min}$ 和最优任务分配向量。粒子群算法本质上是双环的函数。前提是把每个粒子适应度计算过程看作一个单位运算。在此前提下，假设文章总数为 m，迭代次数为 l，算法的时间复杂度为 $O(m \cdot l)$。

2.3　基于 IPSO 算法的抢占任务分配策略

在这一部分，提出了基于 IPSO 算法的抢占任务分配策略，对于边缘计算节点上的任务排队问题，考虑任务抢占调度的情况，在确定需要优先调度的任务后，后续任务为非优先调度，后续任务可以继续使用混合优先级。任务排队算法对任务进行排队。如果节点上有多个抢先式调度任务，则它们的任务排序也可以使用混合优先级任务排队算法对任务进行排队。

对于不执行抢先调度的任务队列，执行 IPSO 算法，任务分配向量如下：

$$\vec{\boldsymbol{k}} = (k_1, k_2, k_3, \cdots, k_M) \tag{36}$$

任务 j 的排队延迟为：

$$t_{qj} = \frac{W_{k_j}(m\tau + t_{i_j k_j} + p_k t_p) + w_j}{v_{k_j}} \tag{37}$$

因此，适应度函数和延迟约束为：

$$f(\vec{\boldsymbol{k}}) = \frac{M}{\sum_{j=0}^{M}(m\tau - t_{sj} + t_{i_j k_j} + p_k t_p + t_{qj})} \tag{38}$$

$$\delta_{t_j} \leqslant t_{ej} - t_{sj} \tag{39}$$

该算法的步骤如下。

1)输入待分配的节点信息和任务队列，初始化粒子群算法相关参数。

2)确定需要抢先调度的任务，并为其分配任务。其他任务形成一个非重复的任务队列。

3)根据非初始任务队列获取粒子的初始位置矩阵和初始速度矩阵。

4)使用 IPSO 的循环迭代解。在每个循环中，更新粒子、子种群和种群的最大适应度和最大适应度位置。每个节点上的任务首先使用混合优先级任务排队算法进行排队。粒子速度和位置是根据迭代公式更新的

5)在周期结束时，结合抢占式调度和非抢占式调度的任务，计算最小平均延迟 $T_{\min}$ 和最优任务分配向量。

3　仿真结果分析

为了验证本文所提的任务分配策略的有效性，搭建仿真环境对其进行结果验证与分析。为降低仿真过程的复杂性，使用计算单元来评估任务处理开销。这里采用系统时间片的长度作为基本计算单元，一个时间片的长度为 10 ms，标准 CPU 频率为 1 GHz。基于不同配电线路实时在线监控业务特性[18-19]，设置不用的仿真参数，具体参数设置如表 1 所示。

为了验证本文所提出的 MPTQ 排队算法的有效性，选取较为典型的最早的截止队列(EDQ)算法在相同任务量的条件下，比较不同处理时延，具体仿真结果如图 3 所示，图中边缘节点的初始负载设置为 0。可以看出，MPTQ 算

表 1　系统仿真参数设置

配电业务	运算量	时延/ms
杆塔倾斜监测	5	320
温度监测	2	280
风力监测	2	400
线路压力监测	5	400
覆冰监测	6	600
电压报警检测	1	80
境视频监测	10	200

法的性能优于 EDQ 算法,当任务数为 20 时,算法 MPTQ 比算法 EDQ 的时延改进程度最大,平均延迟降低了 12%,这是由于相同任务量的条件下执行 MPTQ 算法可以综合考虑不同任务的处理开销和截止时间,从而尽可能完成任务队列的调整,降低任务处理时延。

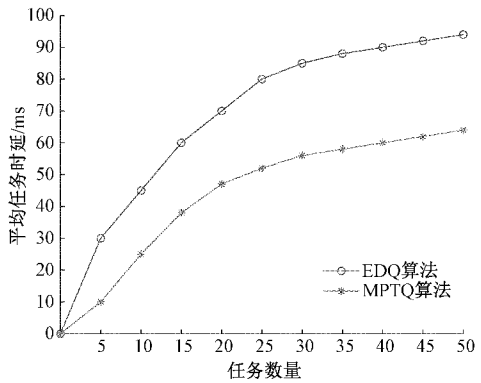

图 3　基于不同算法的任务量与时延之间的关系

为验证本文提出的面向配网通用任务的 CTA-IPSO 算法有效性,选取较为典型的基于遗传算法(CTA-GA)的公共任务分配策略和基于并行粒子群优化(CTA-PPSO)的公共任务分配策略进行对比分析。仿真的参数设置如下:电缆上配有 1、1.5 和 2 GHz 3 类 CPU 运行频谱的边缘节点,节点平均任务处理时延为 10 ms,每个设备根据表 1 生成不同的电缆实时在线监测服务任务,迭代次数为 200 次。

图 4 为基于通用任务分配算法的系统任务数量与平均时延之间的关系,从图中可以看出,CTA-IPSO 算法增强了粒子群算法的搜索能力,不易陷入局部最优解,与其他算法相比,CTA-IPSO 算法具有最小的任务平均延迟。在相同条件下,CTA-IPSO 算法计算的任务平均延迟比 CTA-PSO 算法减少了约 3.32%。与 CTA-PSO 算法相比,CTA-IPSO 算法计算的任务平均延迟降低了约 2.04%,验证了本文所提算法的有效性。

图 5 为基于通用任务分配算法收敛性分析,该结果取 50 次仿真结果的平均值。从图中可以看出 CTA-PSO 和 CTA-PPSO 算法收敛速度快,但计算的平均时延高,而 CTA-IPSO 算法不仅收敛速度快,而且计算的平均时延也较低。较低的平均时延与本文对粒子群算法的改进密不可分。

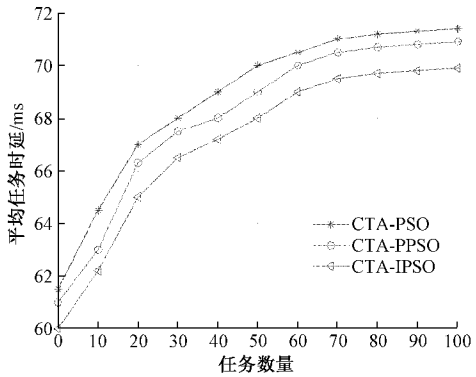

图 4　基于通用任务分配算法的系统任务数量与平均时延之间的关系

因此本文所提的基于通用的任务分配的 CTA-IPSO 的算法可以有效地减少时间延迟,运行时间更短,收敛速度更快。

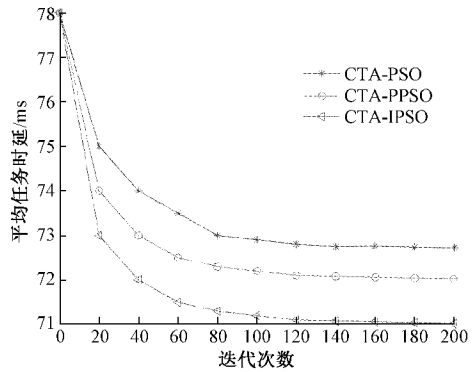

图 5　基于通用任务分配算法收敛性的比较

为验证本文所提的基于抢占式任务分配算法的有效性,选取较为典型的基于遗传算法(CTA-GA)的优先任务分配策略和基于并行粒子群优化(CTA-PPSO)的优先任务分配策略进行对比分析,这部分主要基于多组初始负载大的边缘节点进行仿真,抢占式调度的开销为 10 ms,时间阈值为 100 ms。

图 6 为基于抢占式任务分配算法时延分析,图 7 为基于抢占式任务分配算法收敛性分析,从仿真图中可以看出,在抢占式调度条件下,PTA-IPSO 算法仍然具有最小的任

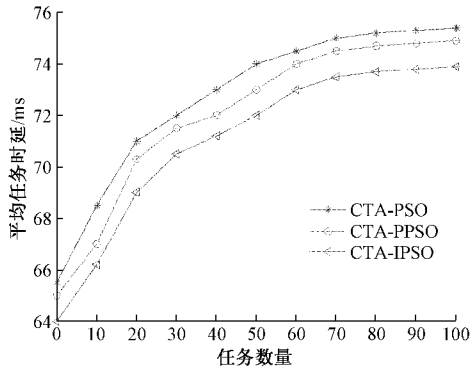

图 6　基于抢占式任务分配算法时延分析

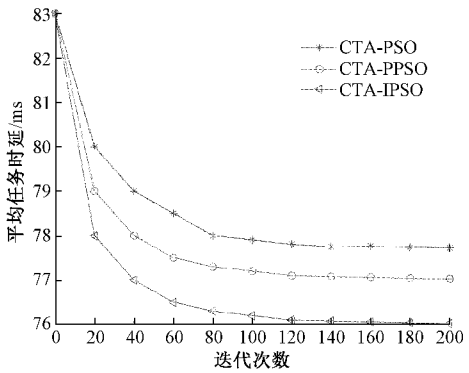

图 7　基于抢占式任务分配算法收敛性分析

务平均延迟，以及更低的运行时间和更快的收敛速度。在相同条件下，IPSO 算法计算的任务平均延迟比遗传算法减少了约 3.29%。与粒子群优化算法相比，IPSO 算法计算的任务平均延迟降低了约 2.18%。

4　结　　论

边缘计算在配网实时在线监测方面具有广阔的前景。然而，由于大多数有线实时在线监测业务对延迟敏感，边缘节点的资源和能力相对有限，急需一种合理的任务分配机制。对此，本文建立了一种基于边缘计算的有线实时在线监测业务任务分配模型。在该模型中，考虑了任务分配过程中的各种变量和约束条件，包括电缆的线性分布特性、边缘节点的实时状态、任务的处理开销以及特殊任务的调度策略，提出了一种基于 IPSO 算法的任务分配策略，包括公共分配策略和抢占式任务分配策略。仿真结果表明，与其他策略相比，该策略具有更低的任务延迟和运行时间，能够有效解决边缘节点任务分配问题。

参考文献

[1] 张淑清，杨振宁，张立国，等. 基于弹性网降维及花授粉算法优化 BP 神经网络的短期电力负荷预测[J]. 仪器仪表学报，2019，40(7)：50-57.

[2] 袁川来，廖庸邑，孔玲爽，等. 时间约束的改进分层模糊 Petri 网的配电网故障诊断方法[J]. 电子测量与仪器学报，2020，34(3)：131-139.

[3] 洪翠，付宁泽，郭谋发，等. 改进多分类支持向量机的配电网故障识别方法[J]. 电子测量与仪器学报，2019，33(1)：12-20.

[4] 张聪，樊小毅，刘晓腾，等. 边缘计算使能智慧电网[J]. 大数据，2019，5(2)：64-78.

[5] 陈顺，刘刚，熊原. 边缘计算中基于节点服务能力评价的多层级任务调度方法研究[J]. 科技传播，2020，253(4)：124-126.

[6] 薛建彬，安亚宁. 基于边缘计算的新型任务卸载与资源分配策略[J]. 计算机工程与科学，2020(6)：959-965.

[7] 董思岐，吴嘉慧，李海龙，等. 面向优先级任务的移动边缘计算资源分配方法[J]. 计算机工程，2020，46(3)：18-23.

[8] 张滨. 5G 边缘计算安全研究与应用[J]. 电信工程技术与标准化，2020，280(12)：6-12.

[9] 夏士超，姚枝秀，鲜永菊，等. 移动边缘计算中分布式异构任务卸载算法[J]. 电子与信息学报，2020，42(12)：68-75.

[10] 黄晓舸，崔艺凡，张东宇，等. 基于 MEC 的任务卸载和资源分配联合优化方案[J]. 系统工程与电子技术，2020，489(6)：188-196.

[11] ENNYA Z，HADI M Y，ABOUAOMAR A. Computing tasks distribution in fog computing：Coalition game model[C]. 2018 6th International Conference on Wireless Networks and Mobile Communications，2018：1-4.

[12] 林涛，秦冬阳，马同宽，等. 基于博弈论的移动边缘计算任务调度研究[J]. 计算机仿真，2018，35(11)：387-391.

[13] 徐文帅. 深度强化学习在移动边缘计算中的应用[D]. 成都：电子科技大学，2020.

[14] 周知，丁帅，陈旭. 边缘智能：边缘计算与人工智能融合的新范式[J]. 大数据，2019，5(2)：53-63.

[15] SAHNI Y，CAO J，YANG L. Data-aware task allocation for achieving low latency in collaborative edge computing[J]. IEEE Internet Things，2019，6(2)：3512-3524.

[16] PAN S，QIAO J，JIANG J，et al. Distributed resource scheduling algorithm based on hybrid genetic algorithm[C]. 2017 International Conference on Computing Intelligence and Information System，2017：24-28.

[17] FENG J，LIU Z，WU C，et al. A VE：Autonomous vehicular edge computing framework with ACO-based scheduling[J]. IEEE Transaction on Vehicular Technology，2017，66(12)：10660-10675.

[18] DESHPANDE J G，KIM E，THOTTAN M. Differentiated services QoS in smart grid communication networks[J]. Bell Labs Technical Journal，2011，16(3)：61-81.

[19] AL-ANBAGI I，EROL-KANTARCI M，MOUFTAH H T. Delay critical smart grid applications and adaptive QoS provisioning[J]. IEEE Access，2015，3：1367-1378.

作者简介

刘哲（通信作者），硕士，主要从事智能电网、电力系统及其自动化、配网大数据的研究。

E-mail：m15930235929@163.com