应用于智能制造的边缘计算任务调度算法研究

Task scheduling strategy of edge computing system in intelligent manufacturing

王　硕，郑爱云，刘怀煜

WANG Shuo，ZHENG Ai-yun，LIU Huai-yu

（华北理工大学 机械工程学院，唐山 063210）

摘　要：边缘计算弥补了传统的云计算模式在智能设备和数据量的爆炸式增长过程中的缺陷，任务调度策略是解决边缘计算与智能制造结合的关键问题之一。首先，结合智能制造应用相关特性，引入了一个面向智能制造的分布式边缘计算系统架构，通过调度策略使任务请求由云计算层迁移到边缘计算层。其次，以该架构调度策略为基础，提出一种基于遗传思想的粒子群调度算法（GAPSO），该算法以粒子群算法为基础，融合杂交和变异的迭代方式，解决局部优化解等问题。最后，为验证该模型的准确性，通过MATLAB进行仿真实验，计算任务数从50增加到600时任务的完成时间和智能终端能源消耗。实验结果表明，GAPSO的时延相比粒子群算法和轮询调度算法，在任务数为600时，完成时间分别减少了5.8%和15.4%，能源消耗分别减少了4.6%和14.1%。

关键词：边缘计算；任务调度；遗传算法；粒子群算法

中图分类号：TP393　　**文献标识码**：A　　**文章编号**：1009-0134(2020)12-0098-08

0　引言

发展智能制造是世界上发达国家或地区制造业发展的内在要求，也是中国制造业转型升级的主攻方向，而工业物联网技术是实现智能制造的核心所在。近几年，工业物联网技术在制造业领域取得巨大的进步。然而，随着信息技术、通信技术和工业物联网技术的发展，大量的物联网设备被应用到智能工厂中。传统的云计算模式，数据需要从物联网设备传输到云端，这一过程会消耗核心网络的大量带宽和能量。同时，由于云服务器距离物联网设备较远，请求响应时间太长，难以满足延迟敏感的物联网应用需求。因此，传统的集中式信息处理模式开始向集中式管理和分布式自治相结合的模式转变[1]，逐渐催生出了边缘计算[2]、雾计算[3,4]和移动边缘计算[5]等新的计算模式。

虽然，边缘计算减少了物联网设备到云服务器之间的传输距离，提高了系统的实时性。但是，我们仍然需要利用计算资源调度策略进一步提高系统的实时性。任务分配是边缘计算中被广泛讨论的热点，它通过解决由终端应用程序生成的计算任务如何在边缘服务器之间分布的问题，使资源受限的边缘服务器能够向终端设备提供满意的服务。

延迟是判断工业物联网的一个主要性能指标，智能制造中大多数工业物联网应用属于时间敏感应用，需要实时性要求。为了减少计算时延，Fan和Ansari设计了一种基于边缘计算的物联网应用程序感知工作负载分配方案。通过为每个终端的不同类型请求选择cloudlet并为每个cloudlet的不同应用程序分配计算资源，可以最小化IoT应用程序请求的响应时间[6]。Yin等在采用容器虚拟技术的智能制造环境中引入了fog计算，通过任务调度和资源分配实现重新分配机制来保证任务的实时性能，进一步降低了任务的计算延迟[7]。Yang等使用NFV技术解决了移动边缘云的资源分配问题，提出一种动态资源分配框架，满足了低延迟、低成本的灵活资源分配需求[8]。该框架包括一种用于动态执行资源分配的快速启发式增量分配机制和一种保持最优成本的再分配优化算法。Dai等对自动驾驶和移动边缘计算进行研究，并提出了一种考虑自动驾驶人物特征的任务调度算法，通过任务的置换和重组，使任务迁移选择更合适的MEC服务器来减少延迟[9]。Deng等研究了单跳雾-物联网结构的最坏时延和最优效用的工作负载调度问题，提出了工作负载动态调度算法（WDSA）[10]。提出的WDSA算法在保证任务处理的最坏情况延迟的前提下，可以使平均吞吐量效用最大化。Fan等提出了一种fog网络的工作负载均衡方案，通过将物联网设备与合适的BSs相关联，使通信和处理过程中的数据流延迟最小化[11]。Zeng等研究了雾计算支持下的SDN嵌入式系统的最小化任务完成时间问题，包括任务调度、资源管理和中断请求。将其描述为一个混合整数非线性规划问题，并给出了求

收稿日期：2019-10-10

作者简介：王硕（1994-），男，河北承德人，硕士研究生，研究方向为边缘计算和工业物联网技术应用。

解方法[12]。Rodrigues等提出了一种在使用两个cloudlet服务器的场景中最小化服务延迟的方法[13]。该方法同时关注计算和通信延迟，通过虚拟机迁移来控制处理延迟，通过传输功率控制来改善传输延迟。Sun等为了最小化移动用户将应用程序工作负载卸载到地理上分散的cloudlet所花费的平均响应时间，提出了一种称为LEAD的延迟感知工作负载卸载策略[14]。Wu等研究边缘服务器在有限的能量预算和时隙预算下，如何最优选择提供计算卸载服务的设备，并提出了一种高效的设备优化选择算法[15]。

然而，仅仅以延迟为指标，不能满足智能制造场景下对边缘计算的要求。因此，大部分学者在研究计算资源调度时，不仅考虑延迟，还考虑到能耗、负载均衡和稳定性等多个指标。Yang等提出了一种改进的二进制粒子群优化算法（MBPSO），实现了包括任务处理时间、能量消耗和网络生命周期在内的整体性能优化[16]。Deng等研究了雾计算中功耗和传输时延的权衡问题，并建议在服务延迟受限的情况下，在雾和云之间将工作负载分配到最小功耗[17]。Chetty等利用软件定义网络的思想，研究任务卸载问题，并将问题转化为任务分配和资源调度两个子问题，提出了一种高效的卸载方案，称为软件定义任务卸载（SDTO），以节省用户设备电池寿命的同时最小化延迟[18]。Diao等使用三种算法联合优化了计算资源、功率和信道分配，使所有用户的能耗和时延加权和最小[19]。Van等研究了移动边缘云中用户会话分配的调度算法。通过对比发现，基于MECs占用水平的调度在各种场景下都具有较强的鲁棒性，可以与分数保护通道策略一起应用，为订阅者提供服务[20]。Jian等提出了一种基于fog网络的工作负载均衡方案，通过将物联网设备与合适的BSs相关联，使通信和处理过程中的数据流延迟最小化[21]。Miao等利用基于LSTM算法的计算任务预测、基于任务预测的移动设备计算卸载策略和基于边缘云调度方案的任务迁移来辅助优化边缘计算卸载模型[22]。算法在计算数据量和子任务量增加的情况下有效地减少总任务延迟，从而保证了时延敏感任务的高效处理。为了满足智能制造的实时性要求，Li等设计了一种基于贪心策略和阈值策略的两阶段算法来调度边缘层的计算资源，使边缘计算的智能制造计算服务具有良好的实时性、满意度和能耗性能[23]。Wang等研究了基于雾计算场景的任务调度策略。提出了一种基于混合启发式（HH）算法的任务调度策略，主要解决了终端设备计算资源有限、能耗高的问题，使该方案对终端设备的实时高效处理任务具有可行性[24]。Sun等主要研究移动边缘计算在车联网中的应用，并将卸载决策问题转化为背包问题，提出了一种新的多目标VEC任务调度算法，通过观察VEC系统的状态，对通信和计算资源的分配做出最优决策[25]。

1 系统模型和问题描述

1.1 系统架构和模型

随着工业物联网技术在智能制造中的广泛的应用，越来越多的时延敏感和计算密集型任务需要处理。边缘计算技术的出现，将会很好的解决这个问题。与云服务器相比，边缘服务器更靠近数据源侧。但是与云服务器相比，边缘节点的计算能力有限且各不相同，如何有效利用边缘节点的计算资源是本研究的目的。首先，我们参照边缘计算产业联盟（ECC）和工业互联网产业联盟（AII）在2018年联合发布的《边缘计算参考架构3.0》建立了一个面向智能制造的分布式边缘计算系统架构。如图1所示，系统架构由四层组成：现场设备层、边缘计算层、云层和应用层。在边缘计算系统中，对于时间敏感的任务在边缘层处理，容忍延迟和密集型任务则在云服务器中处理。

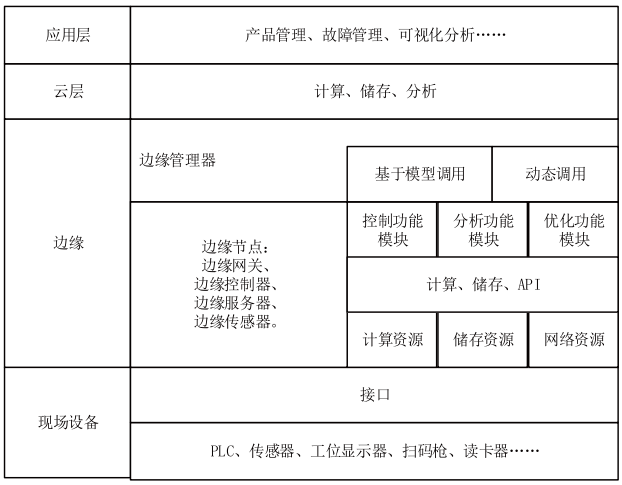

图1 面向智能制造的分布式边缘计算系统架构

现场设备层：现场设备层由生产现场中的所有智能设备构成，主要包括智能生产线中的生产设备、传输设备、传感设备和各种手持智能设备。生产线中的各种设备通过无线方式连接，形成一个大容量、广覆盖的可接入网络，并向边缘层提供数据和发送任务。

边缘层：边缘层位于设备层和云计算层之间，更靠近数据源侧。边缘服务器主要被布置在工业现场生产设备周围，通过有线网络或局域网与生产设备层连接。边缘服务器主要是指交换机和路由器以及其他智能终端设备，它们具有一定的计算能力、通信能力和存储能力，并在云与终端设备之间进行工作。时延敏感任务在边缘层处理，容忍时延和密集型任务在云服务器上执行。边缘计算的出现解决了云计算模式的高延迟，但是边缘服

务器之间的任务分配问题任然要解决。如果任务没有分配给合适的服务器，会造成边缘服务器负载不均衡，从而增加整个系统的时延。

云计算层：云计算层包含云数据中心、云存储、云计算设备，由高性能服务器组成，具有强大的计算和存储能力。云服务器可以对生产设备采集到的数据进行筛选、存储和分析，为智能生产线提供远程服务。例如，可以远程实时了解生产线生产情况。此外，云平台还将会提供更为全面的服务。

应用层：智能生产线的应用包括产品检测、故障诊断、设备维护、实时监控、库存管理等，企业可以通过电脑和手机使用这些应用。智能生产线应用可以有效提高生产效率和生产质量。此外，生产线应用面向的对象是企业的员工和管理者，为他们提供了不同的管理权限和功能界面，以便于直观的、实时了解生产状况。

1.2 延迟模型和能耗模型

边缘计算中的任务调度问题可以描述为若干智能终端随机生成n个独立任务分配给m个边缘服务器。为了方便表示：我们将终端设备集表示为U={U_1，U_2，U_3，…U_i…U_g}，其中，i表示终端设备的序列号，g表示共有g个终端设备。终端设备U_i 的属性可以用三元组表示：Pu={P_{ir}，P_{ie}，E_{i1}}，其中，P_{ir}表示终端设备的传输功率；P_{ie}表示终端设备的空闲功率；E_{i1}表示终端设备的剩余能量。任务集T，表示为T={T_1，T_2，T_3，…T_i…T_n}，T_i表示终端设备产生的第i个任务。任务的属性用四元组表示为：P_T={D_i，θ_i T（i，max），T（i，exp）}。其中，D_i表示任务T_i的数据量大小，θ_i表示任务T_i的计算强度，T（i，max）表示最大完成任务的时间，T（i，exp）表示期望完成任务的时间。边缘服务器集E，E={E_1，E_2，E_3，…E_j…E_m}，E_j表示第j个边缘服务器，边缘服务器E_m的属性用三元组表示为：PE={C_j，K_j，B_j}，其中，C_j表示边缘服务器E_j的计算资源，K_j表示边缘服务器E_j的储存能力，B_j表示边缘服务器E_j的带宽。任务调度问题的可行解表示为x={x_1，x_2，x_3，…x_i…x_n}，其中，x_i 表示第i个任务对应的服务器ID，任务T和边缘服务器E的映射关系有N×M阶矩阵s表示。当s_{ij}=0时，表示任务T_i不在服务器E_j上执行，当s_{ij}=1时，表示任务T_i在服务器E_j上执行。任务调度的目标是寻找最优解x使时间成本和能耗成本最小化。

在边缘计算系统中，任务的完成时间包括任务执行时间T_{ij}^{com}和传输时间T_{ij}^{rt}，任务处理时间与任务的计算强度和服务器的计算资源有关，任务T_i在边缘服务器E_j处的执行时间可以表示为：

$$T_{ij}^{com}(\mathrm{I},\mathrm{J},\mathrm{D}_i)=D_i\theta_i/C_j \tag{1}$$

任务的数据量大小则影响任务的传输时间，任务T_i传输到边缘服务器E_j的传输时间可以表示为：

$$T_{ij}^{tr}(\mathrm{u},\mathrm{J},\mathrm{D}_i)=\frac{D_i}{r_{ij}} \tag{2}$$

其中，R_{ij}表示设备终端到边缘服务器的传输速率：

$$r_{ij}=w*\log_2\left(1+\frac{h*P}{\sigma}\right) \tag{3}$$

其中w为网络带宽，σ为噪声功率，h为信道功率增益，p为传输功率。

因此，任务执行的总时间的数学模型为：

$$T_{ij}(u,J,D_i)=s_{ij}(T_{ij}^{com}+T_{ij}^{tr}) \tag{4}$$

从终端设备的角度，将终端设备U_i上的能耗分为两部分，一部分是传输能耗，另一部分是等待能耗，终端设备能源消耗的数学模型为：

$$E_{ij}(u,J,D_i)=s_{ij}(T_{ij}^{tr}*P_{ir}+T_{ij}^{com}*P_{ie}) \tag{5}$$

任务调度在边缘计算模式下的目的是根据某种调度策略将多个任务分配给多个边缘服务器，尽可能满足终端设备的任务请求，同时减少任务完成的时间和终端设备的能源消耗。

为了简化问题的复杂性，降低解决问题的难度，提出如下假设：各个任务独立，任务之间不存在约束关系；每个任务只能分配给一个边缘服务器，不允许重复分配任何任务。计算过程中不考虑终端设备移动性的影响。所有的边缘服务器都是静态的，执行过程中的任务不能被中断。目标函数主要考虑终端设备执行所有任务的开销，包括所有任务的完成时间和所有终端设备的能耗；在有延迟和能耗约束的边缘计算中，任务调度的目标函数为：

$$f(x)=\min\sum_{i=1}^{n}\left\{\lambda\sum_{j=1}^{m}\left[s_{ij}*T_{ij}\right]+(1-\lambda)\sum_{j=1}^{m}\left[s_{ij}*E_{ij}\right]\right\} \tag{6}$$

$$\sum_{j=1}^{m}s_{ij}=1,i=1,2,\cdots,n \tag{7}$$

$$s_{ij}*T_{ij}\leqslant T_{i,max} \tag{8}$$

$$s_{ij}*E_{ij}\leqslant E_{il} \tag{9}$$

其中，式(7)为任务调度决策的约束条件，即每个任务只能分配给一个边缘服务器；式(8)为每个任务的延迟约束；式(9)为每项任务的能源消耗限制；λ为延迟和能源约束权重系数。

为了清楚地演示这个问题，我们考虑一个极端情

表1 一个简单的任务分配实例

Task	D（M）	θ（cycles/s）	计算能力（cycles/s）	传输速度（M/s）	最大时延（s）	Edge1时延（s）	Edge2时延（s）
Task1	40	300	1G	10M/s	15s	12.12s	0
Task2	18	300	1.2G	10M/s	15s	0	4.57s
Task3	25	300	1G	10M/s	15s	19.69s	0

况，即只有两个边缘服务器：Edge1和Edge2，并且按顺序上传了三个任务。系统的核心是一个任务分配模块，又被称为任务调度器，用于决定任务执行的服务器。在某时刻，任务Task1被上传并且分配给Edge1。根据轮询调度算法任务Task2上传并且分配给Edge2，任务Task3上传并且分配给Edge1。因此，此时Edge1有两个任务Task1和Task3需要处理，而Edge2有1个任务Task2需要处理。各个任务和边缘服务器的属性参数如表1所示。

从表1中可以得出，Edge1处理完成Task1和Task3共需要19，69s。Edge2处理完成Task2需要时间4.57s。因此，该系统延迟为19.69s。

2 任务调度算法设计

任务调度是一个NP难问题，本文设计了基于遗传思想的改进粒子群算法来解决调度问题。粒子群优化（PSO）算法具有全局搜索能力强、收敛速度快和易编程等特点。但是，粒子群算法容易陷入局部最优问题。因此，在粒子群算法中加入了遗传算法的交叉操作和变异操作。另外，由于边缘计算中任务调度问题的参数是离散的，标准的粒子群优化算法不能用于求解优化问题，因此，采用离散粒子群优化算法来解决任务调度问题。

2.1 染色体编码

染色体采用实数编码，即每一位都是用正整数表示，每一位代表一个基因，若干基因组成一个染色体，染色体在粒子群算法中表示位置信息。粒子群中每个粒子代表一种解决方案，一个粒子有位置信息和速度信息，因此，粒子的维度是任务的个数。例如表1所示的粒子，任务1、2、3分别被分配给了边缘服务器2、3和4。

表2 简单的粒子实例

Task	1	2	3	4	5	6	7	8	9	10
Server	2	3	4	1	2	2	5	3	1	5

2.2 粒子群位置和速度信息更新

为了提高基本粒子群优化算法的性能，我们引入了自适应惯性权重ω，惯性权重ω对粒子的搜索能力起着重要作用，可以提高粒子群优化算法的性能。群优化算法中的惯性权值ω是一个线性递减函数，如式(10)所示。

$$\omega^{k+1}=\omega_{\max}-(\omega_{\max}-\omega_{\min})\times k/N \tag{10}$$

其中，ω_{max}是最大惯性权值，ω_{min}是最小惯性权值，k为迭代次数，N是最大迭代次数。

为了平衡粒子的全局和局部搜索能力，本文对惯性权值ω，做了调整，使惯性权值可以通过适应度函数值的大小，改变自身大小。新的惯性权值函数为：

$$\omega^{k+1}=\omega_{\max}-(f^{k}-b)(\omega_{\max}-\omega_{\min})/(avgf-b) \tag{11}$$

其中，f是适应度函数值，b是最优个体适应度函数值，avgf是适应度函数的平均值。

则粒子速度更新方程为：

$$v^{k+1}=\omega^{k}\times v^{k}+c_1\times r_1\times(p^{k}-x^{k})+c_2\times r_2\times(g^{k}-x^{k}) \tag{12}$$

其中，x是粒子的位置，v是粒子的速度，p是当前最优个体，g是当前最优群体。

粒子群的位置更新公式为：

$$x^{k+1}=x^{k}+round(v^{k}) \tag{13}$$

2.3 适应度函数

适应度函数是粒子群算法的目标函数，通过适应度函数判断当前位置和速度的好坏，不断搜索和更新的个人最优值和全局最优值，直到算法终止。适应度函数值越小，解越好。根据时间模型和能耗模型，适应度函数公式为：

$$f(x)=\min\sum_{i=1}^{n}\left\{\lambda\sum_{j=1}^{m}\left[s_{ij}*T_{ij}\right]+(1-\lambda)\sum_{j=1}^{m}\left[s_{ij}*E_{ij}\right]\right\} \tag{14}$$

2.4 杂交和变异操作

传统的粒子群优化算法容易陷入到局部最优，为了避免这一现象，引入遗传算法中的杂交和变异操作，增加粒子群的多样性。具体的杂交操作需要在每次迭代

中计算所有粒子的适应度值。粒子将根据适应度值从小到大进行排序。然后，根据杂交概率p_c确定杂交池的大小。将适应度值靠前的粒子放入杂交池，从中选择粒子进行杂交操作。对于变异操作，变异操作是指染色体基因中的某个位置上的值发生改变，从而产生新的个体。本文采用高斯变异，在变异时用一个均值为μ，方差为σ^2的正态分布的一个随机数来替换原有值。

2.5 基于遗传思想的粒子群算法流程

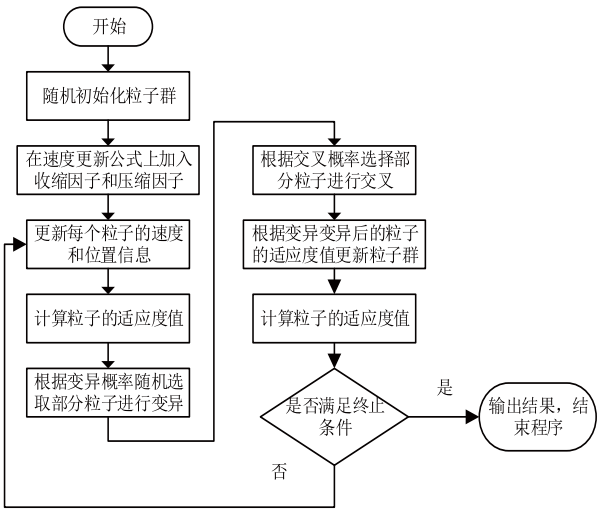

图2 GAPSO算法流程图

步骤一：随机初始化粒子群。随机生成粒子位置信息和速度信息，位置信息采用实数编码，用于表示边缘服务器编号。速度信息取值范围{-5～5}。

步骤二：在速度更新公式中加入惯性权重，如式(12)所示。

步骤三：根据式(12)对整个粒子群进行速度更新，根据式(13)进行位置更新。

步骤四：计算所有粒子的适应度函数值并进行排序。

步骤五：根据交叉概率选择部分粒子进行交叉。按照适应度函数值大小对粒子进行排序，根据杂交因子Pc确定杂交池大小，并随机选择需要杂交的粒子，加入到杂交池。从杂交池中选择两个的粒子进行杂交，产生子代粒子，通过计算适应度函数值与父代进行比较。如果子代适应度函数值小，则保留子代，并更新粒子的位置信息和速度信息。

步骤六：根据变异概率选择部分粒子进行变异。根据变异因子p_m确定变异池的大小，并随机选择需要变异的粒子加入到变异池。根据变异因子p_m逐一确定变异池中的粒子是否需要变异，如果需要变异，则采用高斯变异算子进行变异操作。

步骤七：根据交叉变异后的粒子的适应度值更新粒子群。

步骤八：计算所有粒子的适应度函数值。

步骤九：判断是否满足终止条件，不满足则返回到步骤四。

3 仿真实验设计和分析

3.1 仿真实验设计

为了验证本文提出的基于遗传思想的粒子群算法在边缘计算资源调度模型中的有效性，本文使用粒子群算法（PSO）和轮询调度算法（RR）与本文算法进行比较，并使用MATLAB2016a仿真平台进行实验。通过比较不同的调度算法调度结果的完成时间和能源消耗来比较算法的效果。实验运行环境为Windows8.1，64位操作系统，Intel Core i5-4500CPU，4GB内存。仿真实验中需要的参数如表3所示。

表3 算法参数表

参数	值/单位	描述
n	50～300	终端设备数量
m	5～10个	服务器数量
D_i	10～50Mb	任务i数据量大小
C_j	1～2G cycles/s	Edge的j计算能力
θ_c	300 cycles/s	计算强度
P_{rt}	0.1W	传输功率
P_{ie}	0.05W	空闲功率
B	100 MHz	带宽
ω	[0.4 0.9]	惯性权重因子
c_1，c_2	1.49445	学习因子
P_c	0.5	杂交概率
P_m	0.05	变异概率

为了保证解的质量，我们通过实验来确定种群的大小，设置了不同大小的调度问题Q，其中Q的每个值是由任务T和边缘服务器E的数量决定的。边缘服务器和任务数量分别选取了（5/20）、（10/20）和（10/50）三组。对于每个Q值进行5次实验，取时延的平均值，如图3所示。另外，图4显示了MATLAB的运行时间。在这些实验过程中迭代次数设为300次。从图中可以看出随着种群数量的增加，可以得到较好的解。然而，种群数量在30以后，优化效果不明显。且种群数量的增加导致算法运行时间增加。因此，确定种群数量为30。

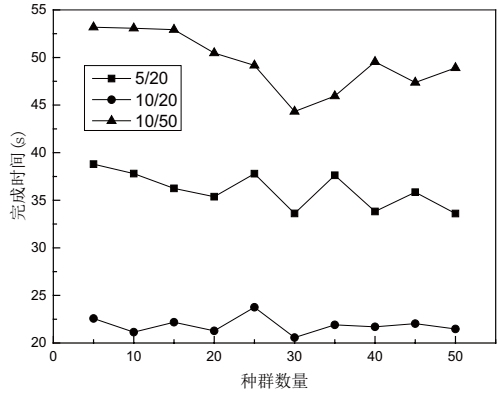

图3 种群数量对完成时间的影响

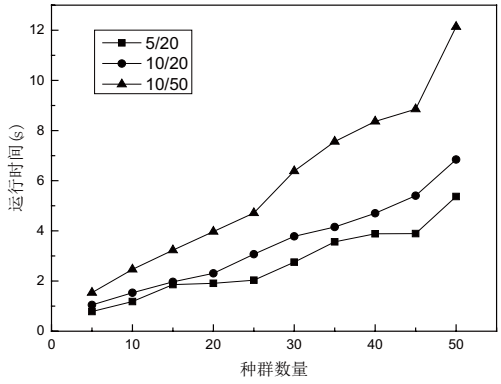

图4 种群数量对MATLAB运行时间的影响

为了确定迭代次数MAX的最佳值，同时考虑得到的解的质量。我们使用相同的Q值，保持种群数量为30，并分别改变MAX的值，如图所示。对于每一个考虑值N和MAX的组合，执行5个不同的实验，对应的时延平均值如图5所示，MATLAB运行时间如图6所示。根据实验，MAX的增加细化了延迟到一定的点，然而，它通常会大幅增加调度算法的运行时间。因此，MAX被设置为300，因为它提供了良好的质量解决方案与中等的运行时间。

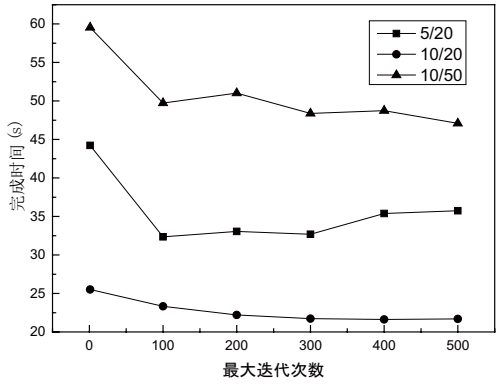

图5 最大迭代次数对完成时间的影响

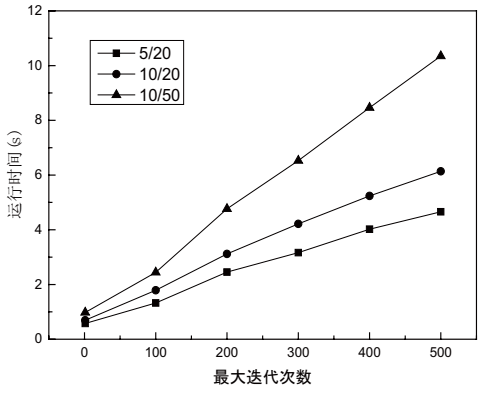

图6 最大迭代次数对MATLAB运行时间的影响

通过实验确定权重系数λ的值，设置种群数为30，最大迭代次数为300，任务数为200，服务器数为10。λ值从0.1增加到0.9，每次实验运行5次取平均值。图7～图9，显示了不同权重系数值对应的适应度函数值、完成时间和总消耗。从图中可以看出三个指标各自的趋势，横坐标为λ。当λ从0.1逐渐增加到0.9时，完成时间趋于缩短，总消耗呈现增加趋势。因为总的完成时间随着权重系数的增加，在适应度函数中变得越来越重要。因此，较大的权重系数可以较好地用于优化总完成时间。另外，从图中可以得到权重系数值得改变对能源消耗影响较小，而总的完成时间波动较大，且当λ=0.9时，完成时间相对较小。因此，权重系数值确定为λ=0.7。

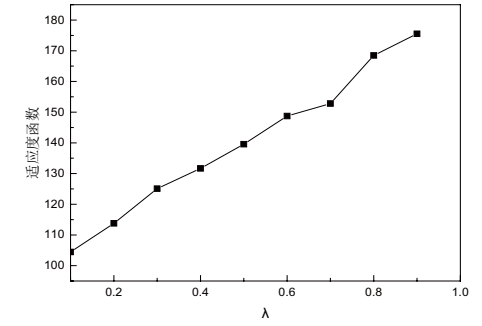

图7 不同λ值的适应度函数值

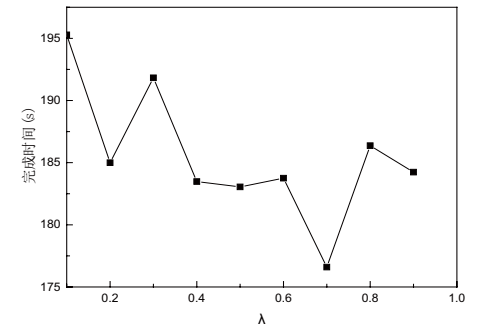

图8 不同λ值的完成时间

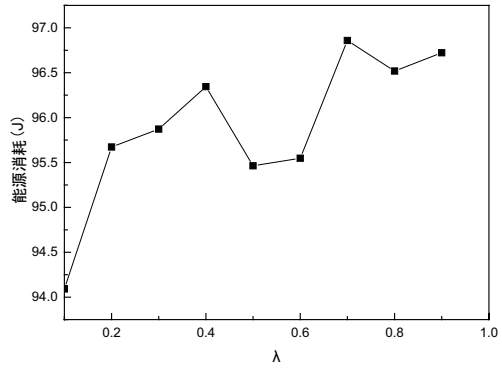

图9　不同λ值的能源消耗

3.2 仿真结果分析

本文从采用完成时间和能耗两个性能指标，第一个性能指标为处理完成所有任务的时间。第二个性能指标是处理所有任务所需的能耗。将提出的任务调度策略GAPSO与粒子群算法（PSO）和轮询调度算法（RR）两种调度算法进行了比较。

首先，利用完成时间验证三种任务调度策略的性能。实验进行20次，取平均值。任务号分别设置为50、100、150、200、250、300、350、400、450、500、550、600。结果如图10所示。

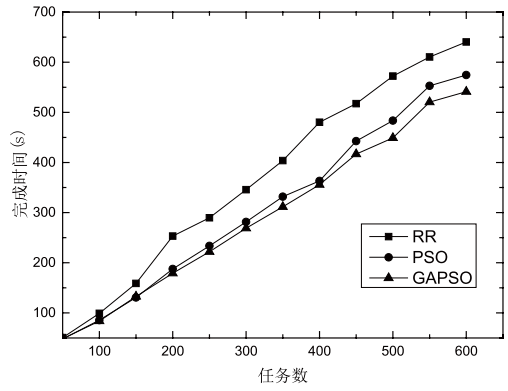

图10　三种任务调度算法的完成时间

图10展示了三种算法的任务完成时间如何随着任务数增加的变化而变化。由图可知GAPSO算法的完成时间最小，PSO的完成时间大于GAPSO，而RR的完成时间最大。这是因为GAPSO和PSO以最小完成时间为目标来进行任务调度。而RR算法依次将任务请求调度到不同的边缘服务器，调度过程中不考虑完成时间；因此，RR的完成时间性能最差。GAPSO算法是在PSO算法的基础上，利用遗传思想的得到的，具有PSO和GA的优点，因此，GASPO算法性能最优。其中，在任务数为600时，GAPSO与PSO和RR相比，完成时间分别减少了5.8%和15.4%。

图11则显示了三种任务调度策略在不同任务数量下的能耗仿真结果。

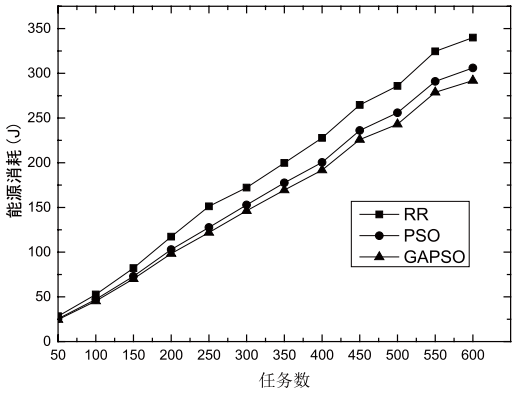

图11　三种任务调度算法的能源消耗值

随着任务的数量的增加，三种调度策略的能源消耗均呈现出上升的趋势，原因是增长的任务数量，终端设备需要消耗更多的能量来完成任务数据的发送和接收任务的结果。终端设备的能耗由终端设备的功率和运行时间决定，而终端设备的功率是固定的，因此，能耗的大小只与任务完成时间有关。从图中，可以看出RR算法对应的能耗最多，其次是PSO。能源消耗最少的是GAPSO。其中，在任务数为600时，GAPSO与PSO和RR相比，能源消耗分别减少了4.6%和14.1%。

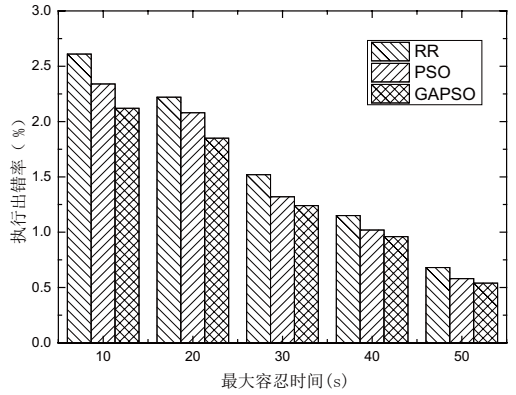

图12　违反最大容忍时间的任务所占百分比

为了评估算法的可靠性能力，我们利用执行出错率表示，执行出错率是指超出最大完成时间的任务在所有任务中所占的百分比。图12展示了在任务数为300的情况下，最大容忍时间从10增加到50，不同算法的执行出错率。从图中可以看出当最大容忍时间为50s时，执

行出错率最低为0.5%左右。当最大容忍时间降低到10s时，执行出错率急剧增加，但是GAPSO明显优于PSO和RR。因此，在最大容忍时间的限制下，GAPSO有更显著的性能。

4 结语

本文通过研究智能制造的相关特点，结合边缘计算特性，设计了一种面向智能制造的分布式边缘计算系统架构。同时，为了提高边缘计算的实时性，减少系统的能耗，建立了边缘计算系统模型和任务调度相关数学模型，并利用基于遗传思想的粒子群算法，解决边缘计算资源调度与任务分配问题。通过建立基于边缘计算的智能制造仿真环境，并将所提出的任务调度策略与其他两种策略进行比较，然后利用延迟和能量消耗两个性能指标进行了实验验证，表明了本文提出的应用于边缘计算模型的GAPSO算法，可以有效地减少任务处理的时间，同时，减少系统的能源消耗。

参考文献：

[1] 景轩,姚锡凡.大数据驱动的云雾制造体系架构[J].计算机集成制造系统,2019,25(9):2119-2139.

[2] Shi W, Cao J,Zhang Q,et al.Edge Computing:Vision and Challenges[J].IEEE Internet of Things Journal,2016,3(5):637-646.

[3] Chiang M,Zhang T. Fog and IoT: An Overview of Research Opportunities[J].IEEE Internet of Things Journal,2016,3(6):854-864.

[4] Bellavista P,Berrocal J, Corradi A, et al. A survey on fog computing for the Internet of Things [J].Pervasive and Mobile Computing,2019:71-99.

[5] Wang S, Zhao Y, Xu J, et al. Edge server placement in mobile edge computing[J].Journal of Parallel and Distributed Computing, 2019:160-168.

[6] Fan Q, Ansari N. Application Aware Workload Allocation for Edge Computing-Based IoT[J].IEEE Internet of Things Journal, 2018,5(3):2146-2153.

[7] Yin L, Luo J, Luo H, et al. Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing[J].IEEE Transactions on Industrial Informatics, 2018,14(10):4712-4721.

[8] Yang J, Lu Z, Wu J. Smart Toy Edge Computing oriented Data Exchange Based on Blockchain[J].Journal of Systems Architecture,2018,87:36-48.

[9] Dai H, Zeng X, Yu Z,et al.A scheduling algorithm for autonomous driving tasks on mobile edge computing servers [J].Journal of Systems Architecture,2019:14-23.

[10] Deng Y, Chen Z, Zhang D,et al.Workload scheduling toward worst-case delay and optimal utility for single-hop Fog-IoT architecture[J].Iet Communications,2018,12(17):2164-2173.

[11] Fan Q, Ansari N.Towards Workload Balancing in Fog Computing Empowered IoT[J].IEEE Trans on Network Science and Engineering, 2020,7(1):253-262.

[12] Zeng D, Gu L, Guo S, et al. Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System[J].IEEE Trans on Computers,2016, 65(12):3702-3712.

[13] Rodrigues T G, Suto K,Nishiyama H, et al. Hybrid Method for Minimizing Service Delay in Edge Cloud Computing Through VM Migration and Transmission Power Control[J].IEEE Transactions on Computers,2017,66(5):810-819.

[14] Sun X, Ansari N.Latency Aware Workload Offloading in the Cloudlet Network[J].IEEE Communications Letters,2017,21(7): 1481-1484.

[15] Wu Y,Shi J, Ni K, et al.Secrecy-Based Delay-Aware Computation Offloading via Mobile Edge Computing for Internet of Things[J]. IEEE Internet of Things Journal, 2019,6(3):4201-4213.

[16] Yang J, Zhang H, Ling Y, et al. Task Allocation for Wireless Sensor Network Using Modified Binary Particle Swarm Optimization [J]. IEEE Sensors Journal,2014,14(3):882-892.

[17] Deng R, Lu R, Lai C,et al.Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption [J].IEEE Internet of Things Journal,2016:1-1.

[18] Chetty N, Alathur S. An architecture for digital hate content reduction with mobile edge computing[J].Digital Communications and Networks,2019.

[19] Diao X, Zheng J, Wu Yuan, et al. Joint Computing Resource, Power, and Channel Allocations for D2D-Assisted and NOMA-Based Mobile Edge Computing[J].IEEE Access,2019:9243-9257.

[20] Van Do T, Do N H, Nguyen H T, et al. Comparison of scheduling algorithms for multiple mobile computing edge clouds[J]. Simulation Modelling Practice and Theory, 2019:104-118.

[21] Jian C, Chen J, Ping Jing, et al. An Improved Chaotic Bat Swarm Scheduling Learning Model on Edge Computing[J].IEEE Access, 2019: 58602-58610.

[22] Miao Y,Wu G, Li Miao, et al. Intelligent task prediction and computation offloading based on mobile-edge cloud computing[J]. Future Generation Computer Systems,2020: 925-931.

[23] Li X, Wan J, Dai H, et al.A Hybrid Computing Solution and Resource Scheduling Strategy for Edge Computing in Smart Manufacturing[J].IEEE Transactions on Industrial Informatics, 2019, 15(7): 4225-4234.

[24] Wang J, Li D.Task Scheduling Based on a Hybrid Heuristic Algorithm for Smart Production Line with Fog Computing[J]. Sensors,2019,19(5).

[25] Sun J, Gu Q, Zheng T, et al. Joint communication and computing resource allocation in vehicular edge computing[J].International Journal of Distributed Sensor Networks,2019,15(3).