工业物联网中基于边缘计算的跨域计算资源分配与任务卸载

周鹏[1,2]，徐金城[1,2]，杨博[1,2]
（1. 上海交通大学自动化系，上海 200240；2. 系统控制与信息处理教育部重点实验室，上海 200240）

摘　要：在工业物联网中，现场设备的计算能力有限，基于边缘计算的任务卸载可以有效缓解现场设备的计算压力，提供低时延计算服务。此外，由于网络中不同区域的边缘服务器负载不同，需要合理安排任务卸载以及分配边缘服务器计算资源，从而降低任务完成时延，实现负载均衡。因此，研究了工业物联网中基于边缘计算的任务卸载和资源分配，提出了一种工业物联网中计算任务跨域卸载模型，并构建了一个最小化任务完成时间的混合整数非线性优化问题。将该问题分解为资源分配与任务卸载两个子问题，基于两个子问题特征，通过迭代交替求解，得到资源分配最优解与任务卸载策略。实验结果表明，与不跨域方法相比，所提方法有效地减轻了边缘服务器的负载不均衡，减少了任务完成时延。

关键词：工业物联网；资源分配；任务卸载；跨域
中图分类号：TP301.6
文献标识码：A
doi: 10.11959/j.issn.2096−3750.2020.00143

Cross-domain task offloading and computing resource allocation for edge computation in industrial Internet of things

ZHOU Peng[1,2], XU Jincheng[1,2], YANG Bo[1,2]
1. Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
2. Key Laboratory of System Control and Information Processing, Ministry of Education, Shanghai 200240, China

Abstract: In the industrial Internet of things, limited by the computing capacity of field devices, the task offloading based on edge computing can effectively alleviate the computing burden of field devices and provide low-latency computing services. Moreover, because the load of edge servers are different in different areas of the network, it is necessary to reasonably arrange task offloading and allocate computing resources of edge servers, thereby reducing task completion delay and achieving load balance. Thus, the task offloading and resource allocation for edge computing in the industrial Internet of things was studied, a cross-domain offloading model for computing tasks in the industrial Internet of things was proposed, and a mixed integer nonlinear optimization problem that minimizes task completion time was formulated. The problem was decomposed it into two sub-problems of resource allocation and task offloading, based on the characteristics of the two sub-problems, the optimal solution of resource allocation and task offloading strategy were obtained through iterative and alternating solution. The experimental results show that compared with the non-cross-domain strategy, the load imbalance of the edge server and the task completion delay are reduced effectively by the proposed strategy.

Key words: industrial Internet of things, resource allocation, task offloading, cross-domain

1　引言

近年来，随着通信和网络技术的不断发展，物联网垂直应用领域的研究也在快速推进，产生了许多新的应用场景[1]。随着工业 4.0、“中国制造 2025”的提出，工业物联网（IIoT, industrial Internet of things）成为解决智能工厂中许多问题的关键技术[2]，具体问题包括工业大数据采集、分析与应用、设备

收稿日期：2019−09−12；**修回日期**：2020−02−07
基金项目：国家重点研发计划重点专项（No.2018YFB1702300）
Foundation Item: The National Key R&D Program of China Key Special Projects (No.2018YFB1702300)

监测与维护、生产计划编排等。在工厂中，物联网设备会产生大量工业数据，部分业务对数据处理的实时性有一定要求。

工业场景中产生的工业数据需要进行存储、处理和分析，传统的工业数据处理方式是将数据传输到集中式云服务器计算，云服务器的计算能力强，能够处理计算量较大的任务[3]。但是，现有的工业环境中物联网设备数量多，把计算任务统一上传到云服务器计算会出现网络数据拥塞的情况，并且云服务器布置在设备的远端，设备在上传数据时产生的传输时延会极大地增加任务的完成时间，从而影响任务的实时性。因此，一个实时、高效的数据处理系统对 IIoT 是十分重要的。

作为云计算的拓展和补充，边缘计算受到越来越多人的重视[4]。针对由于云计算采用集中式计算造成的网络拥塞以及传输时延较大的问题，边缘计算采用了分布式计算方式，由分布在网络中的多个服务器接受用户的计算任务，从而降低设备上传数据至云服务器的需求，减小了网络拥塞[5]。同时，将服务器布置在距离设备更近的网络边缘节点，避免数据在远距离通信中产生高传输时延，边缘服务器可以在更短的时间内对用户的请求和任务做出响应，进一步保证了数据计算任务的实时性[6]。目前，关于边缘计算技术在工业场景或物联网场景中的应用有如下相关研究。在文献[7]中，研究了混合边缘云计算环境下的多跳计算任务卸载问题，通过博弈方法实现了满足服务质量要求的卸载方法。在文献[8]中，从时间代价和能量消耗代价两个方面进行优化，实现大规模绿色节能计算资源的最优分配。在文献[9]中，为了满足实时性要求，提出了在混合计算结构下的一种智能资源规划策略。综上所述，边缘计算能为现场设备提供较充足的计算资源，并通过部署在距离现场设备更近的网络边缘来减轻网络负载，进而满足任务服务质量的要求，减少不同场景下的系统开销。本文研究在工业场景下，基于边缘计算技术的计算资源分配和计算任务卸载问题，通过最优分配边缘服务器的计算资源和卸载任务，减小了工业现场设备计算任务的计算完成时延。

随着越来越多的物联网设备接入工业网络，对网络架构提出了挑战。在传统的工业控制网络架构中，每一台设备都有各自独立的控制平面和数据平面，控制平面负责控制数据转发，数据平面完成数据转发。设备没有独立于设备自身的控制平面，意味着每当一台新设备接入网络中或原有的设备需要更新、部署新的功能时，就需要对其控制平面进行单独配置，以完成网络协议处理和计算等功能。在对灵活性要求较高的工业环境中，传统网络架构很难在较短时间内实现新功能的部署或新设备接入，将会极大地影响项目进度。为了克服传统网络架构的上述缺点，人们提出了一种新的网络架构——软件定义网络（SDN, software defined network）。SDN 将设备的控制平面与数据平面分离，设备仅具有数据平面，控制平面独立于设备。SDN 通过集中式的控制平面，对全网络的资源进行调配和优化；针对网络功能的部署，只需要利用软件定义新的规则，并接入集中控制器上的开放接口，便可以在网络中运行。SDN 的引入提高了网络的灵活性，减小了网络硬件的限制，实现了网络的自动化部署和调整[10]。

此外，簇域网络（clustered network）技术[11]已被发展和验证是适合 IIoT 架构的网络拓扑结构。越来越多的研究表明，簇域网络或分组网络（grouped network）的层次结构使得网络拓扑的更新与升级更灵活，逻辑上的集中控制也使其在集中管理方面具有明显优势，这种层次结构在 IIoT 中被广泛采用。簇域或分组工业无线网络的研究发展，为 IIoT 的高性能自适应系统设计提供了初步基础。IIoT 架构底层可由多个簇域网络组成，这些簇域网络是由各种底层实体如机器、智能终端抽象而成。簇域网络结构的 IIoT 架构通过整合上述边缘计算、SDN 等技术来解决 IIoT 中的通信传输时延、带宽约束、计算资源分配等问题[12]。

结合边缘计算与 SDN 技术，本文考虑一种基于簇域网络结构的工业场景。该场景根据实际工业现场设备、无线网络设备和边缘服务器的部署及工作特点，将网络划分成不同的簇域。由于工业场景对实时性和可靠性的要求较严格，需要实时存储工业数据并处理计算任务，因此，需要研究基于簇域网络结构的低时延计算任务卸载方法和资源分配策略。在上述工业场景下，本文解决了如下两个关键问题：1) 在簇域结构的异构工业网络中，不同终端设备、边缘设备的计算能力不同，针对多种任务如何在本地计算、卸载到边缘服务器中进行动态选择；2) 每个边缘服务器的计算资源有限，在任务的多种计算方式下，如何分配计算资源以实现总任务执行时间最优。基于此，本文研究在簇域网络结构

下的最优计算资源分配与计算任务卸载问题，提出了计算任务的跨域卸载方法，现场设备的计算任务可跨域卸载至其他域的边缘服务器上进行计算，进而优化任务完成时间和计算资源利用率。

为了实现 IIoT 中计算任务的实时处理，本文首先对工业环境下的计算任务处理过程进行建模，包括数据传输模型、计算模型和任务卸载模型。将计算任务的处理分为计算能力分配和计算任务卸载两个子问题，目标是使得计算任务的完成时间最小化。为了解决上述两个子问题，首先，证明了计算能力分配子问题是凸优化问题，通过最优性条件求得最优计算能力分配策略；然后，将计算任务卸载构建为整数线性规划问题进行求解，通过分支定界法和启发式模拟退火算法得到最优方案与次优方案；最后，仿真分析表明，本文的跨域任务卸载方法是有效的，与不跨域卸载方法相比减少了任务完成时延。

2　系统模型与问题构建

2.1　问题场景

基于边缘计算的跨域资源分配与卸载框架如图 1 所示。在图 1 中，工厂布置多个了 SDN 交换机，现场设备与交换机之间通过无线传输，交换机之间通过有线链路连接。考虑交换机的无线传输覆盖能力有限，设备的传输能力也有限，只能通过最近的交换机连接至网络，因此，交换机和邻近设备形成一个个域。假设每个设备只与一个 SDN 交换机连接，并且域和域之间不重叠。整个 IIoT 被分为若干个簇域网络，每个簇域网络由一个 SDN 交换机、配置在交换机边的一个边缘服务器以及若干个与其相连的设备组成。在一个簇域网络中，设备只能与本地 SDN 交换机通信进行数据传输，设备的任务可直接卸载至本地配置的边缘服务器上进行处理。当本地簇域网络的计算能力不足、计算任务负载过大时，设备可先将任务数据传输至本地交换机，再通过交换机之间的通信链路将任务数据传输至其他簇域网络中的交换机，在其他簇内完成计算。通过这种通信方式，本地簇域网络中的设备可以将任务卸载至其他簇域网络的边缘服务器中进行处理。

在图 1 中，SDN 控制器通过专有通信链路与所有交换机连接，采用 SDN 的专用 OpenFlow 协议进行通信。SDN 控制器负责收集、获取全网信息，控制任务传输和卸载。现场设备的重要信息如设备故障信息、运行状态信息等，会通过与基站相连的交换机、基站传输并存储到云平台上，在云平台上对

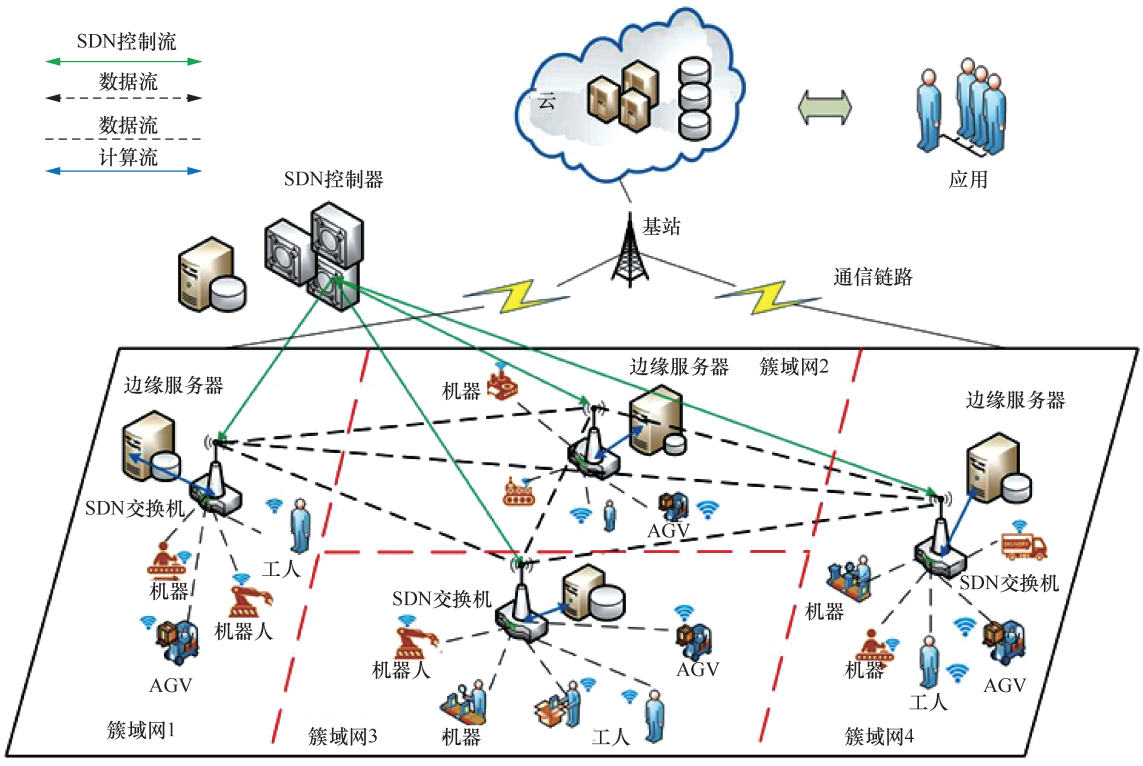


图 1　基于边缘计算的跨域资源分配与卸载框架

工业大数据进一步分析与处理。

上述 IIoT 系统的主要工作流程如下。

1) 当有计算任务时，现场设备发出计算任务服务请求，通过接入网络经 SDN 交换机传送至 SDN 控制器。SDN 控制器接收任务请求后，发出服务响应。现场设备在接收服务响应后，将任务的相关信息与设备的自身信息发送至 SDN 控制器，包括待处理任务的计算量大小、数据量大小、自身设备的计算能力。

2) 各个簇域网络中的边缘服务器将其信息发送至 SDN 控制器，包括当前的计算能力和数据传输率。SDN 控制器收集到上述信息后，在控制器中集中求解计算资源分配与任务卸载问题，得到最优资源分配和任务处理方式。现场设备和 SDN 根据控制器指令，建立设备与服务该设备任务的边缘服务器之间的通信链路。然后，任务在现场设备计算或被卸载至相应服务器进行处理。

3) 当计算完成后，将计算结果发送给各个设备。上述过程完成后，则当前阶段的现场设备计算任务完成，下个阶段新的计算任务产生，重复上述过程。

2.2　系统模型

将现场网络划分为 N 个簇域，相应地，边缘服务器为 N 个，其中，M 个设备属于这 N 个簇域网络，并且连接到相应的交换机上。在系统运行时，每个设备 D_i 产生任务 m_i，其计算需要的 CPU 周期为 w_i，数据量大小为 d_i，计算处理方式包括在自身设备计算、在本地簇域内的边缘服务器计算、卸载到其他簇域内的边缘服务器计算，每个任务可以任选一种。簇域内的边缘服务器 ES_j 与设备通信的带宽为 B_j，属于不同簇域的边缘服务器之间的通信带宽为 $B_j^{\gamma_i}$。设备 D_i 产生的计算任务 m_i 可以在本地设备计算，也可以上传到与其相连的服务器计算，或者卸载到其他服务器计算，变量标识如表 1 所示，3 种计算方式下的任务完成时间如下。

1) 本地任务计算完成时间

若计算任务在本地设备进行计算，定义设备 D_i 的计算能力为 C_i^{D}，则本地处理任务需要的时间为

$$t_i^{\mathrm{D}} = \frac{w_i}{C_i^{\mathrm{D}}} \tag{1}$$

2) 本地边缘服务器计算完成时间

任务上传到与设备相连的本地簇域网络上的边缘服务器上进行计算，任务完成时间分为两部分：任务传输时间和任务计算时间。假设任务计算结果的数据量很小，则忽略结果传回现场设备所需的时间。定义设备与服务器之间的通信带宽为 B_j，服务器的计算能力为 C_j^{E}，则完成时间为

$$t_i^{\mathrm{ES}} = \frac{d_i}{B_j} + \frac{w_i}{\kappa_i^{\gamma_i} C_{\gamma_i}^{\mathrm{E}}} \tag{2}$$

其中，γ_i 表示计算任务 m_i 的服务器位置，$\gamma_i \in \{1,2,\cdots,N\}$，在将整个网络划分为若干个簇域网络时，按照一定顺序对交换机和边缘服务器进行编号，$\gamma_i = n$ 表示第 i 个现场设备的计算任务 m_i 被卸载至第 n 个边缘服务器上处理计算，$\kappa_i^{\gamma_i}$ 表示服务器 γ_i 分配给任务 m_i 的计算能力的百分比。

表 1　　变量标识

参数	含义
D_i	第 i 个设备
ES_j	第 j 个边缘服务器
M	设备的数量
N	边缘服务器的数量
m_i	第 i 个设备上产生的任务
w_i	任务 m_i 需要的计算量/M cycles
d_i	任务 m_i 的数据量大小/MB
C_i^{D}	设备 D_i 的计算能力/G cycles
C_j^{E}	边缘服务器 ES_j 的计算能力/G cycles
γ_i	整数变量，任务 m_i 在边缘服务器计算的位置，$\gamma_i \in \{1,2,\cdots,N\}$
B_j	ES_j 与其连接设备之间的通信宽带/MHz
$B_j^{\gamma_i}$	边缘服务器 j 与卸载域边缘服务器之间通信的带宽/MHz
x_i	0-1 变量，若任务 m_i 在设备计算，则 $x_i = 0$；否则，$x_i = 1$
β_i	0-1 变量，若任务 m_i 在服务器之间进行卸载，则 $\beta_i = 1$；否则，$\beta_i = 0$
$\kappa_i^{\gamma_i}$	任务 m_i 占用服务器 γ_i 计算能力的比例

3) 跨簇域边缘服务器计算完成时间

若任务卸载到其他服务器进行计算，则完成时间由 3 部分组成：任务从设备上传到与其相连的本地服务器的传输时间、本地服务器与卸载服务器之间的任务传输时间以及计算时间。定义服务器之间通信的带宽为 $B_j^{\gamma_i}$，并且为常数，则完成时间为

$$t_i^{\mathrm{ES}} = \frac{d_i}{B_j} + \frac{d_i}{B_j^{\gamma_i}} + \frac{w_i}{\kappa_j^{\gamma_i} C_{\gamma_i}^{\mathrm{E}}} \tag{3}$$

任务 m_i 的总完成时间为

$$t_i=(1-x_i)\frac{w_i}{C_i^{\mathrm{D}}}+x_i\left(\frac{w_i}{\kappa_i^{\gamma_i}}+\frac{d_i}{B_j}+\beta_i\frac{d_i}{B_j^{\gamma_i}}\right) \tag{4}$$

其中，x_i 为 0-1 变量，表示任务 m_i 是在本地设备计算还是上传到服务器，$x_i=0$ 表示在本地设备计算，$x_i=1$ 表示任务需要上传到服务器。则网络中所有任务完成时间之和为

$$t=\sum_{i=1}^{M}t_i=\sum_{i=1}^{M}\left[(1-x_i)\frac{w_i}{C_i^{\mathrm{D}}}+x_i\left(\frac{w_i}{\kappa_i^{\gamma_i}C_{\gamma_i}^{\mathrm{E}}}+\frac{d_i}{B_j}+\beta_i\frac{d_i}{B_j^{\gamma_i}}\right)\right] \tag{5}$$

需要满足的约束为

$$\sum_{i\in O_{\gamma_i}}\kappa_i^{\gamma_i}\leqslant 1 \tag{6}$$

其中，约束表示每个服务器分配的计算能力不能超过其自身总能力。

3 最优资源分配与计算任务卸载

为了最小化所有任务完成总时间，给出边缘服务器的最优资源分配策略和现场设备计算任务卸载方案。定义目标函数为

$$f(x,\gamma,\kappa,\beta)=\sum_{i=1}^{M}t_i=\sum_{i=1}^{M}\left[(1-x_i)\frac{w_i}{C_i^{\mathrm{D}}}+x_i\left(\frac{w_i}{\kappa_i^{\gamma_i}C_{\gamma_i}^{\mathrm{E}}}+\frac{d_i}{B_j}+\beta_i\frac{d_i}{B_j^{\gamma_i}}\right)\right] \tag{7}$$

目标函数是一个混合整数非线性函数，最优化问题的求解方法包括间接法、直接法和数值计算法等。因为本文的目标函数和约束条件有明显的解析表达式，可以通过最优性条件推导出解析解，满足间接法的使用条件，因此，本文采用间接最优化方法。首先，把原问题分为两个子问题进行求解：1) 最优计算资源分配问题。当现场设备的计算方式 x、β 及卸载服务器位置 γ 确定时，则原问题转换为关于服务器计算能力分配比例系数 κ 的凸函数，从而可以用 KKT（Karush-Kuhn-Tucher）条件求得最优计算能力分配策略，给出最优目标函数值 $f(\kappa^*)$ [13]。2) 计算任务卸载问题。基于最优计算能力分配策略 κ^*，确定计算任务的计算方式 x、β 及卸载服务器位置 γ。该子问题是一个 0-1 整数规划问题，本文采用分支定界法与启发式模拟退火算法求解每个计算任务的处理方式和计算位置。

3.1 最优资源分配问题

给定任务计算方式的一个可行解为 x^0、β^0、γ^0，原目标函数转换为关于 κ 的凸函数。假设共有 l 个现场设备选择将计算任务卸载至服务器处理，其中，p 个设备将任务卸载至其他簇域网络中的服务器进行处理，剩余设备选择本地计算，则目标函数为

$$f\left(x^0,\gamma^0,\kappa,\beta^0\right)=\sum_{i=1}^{l}\left(\frac{w_i}{\kappa_i C_j^{\mathrm{E}}}+\frac{d_i}{B_j}\right)+\sum_{i=1}^{p}\frac{d_i}{B_j^{\gamma_i}}+\sum_{i=M-l+1}^{M}\frac{w_i}{C_i^{\mathrm{D}}} \tag{8}$$

需要满足的约束为

$$\sum_{i\in O_{\gamma_i}}\kappa_i^{\gamma_i}\leqslant 1 \tag{9}$$

求 $f(x^0,\gamma^0,\kappa,\beta^0)$ 关于 $\{\kappa_1,\kappa_2,\cdots,\kappa_M\}$ 的 Hessian 矩阵为

$$\boldsymbol{H}=\begin{bmatrix}\dfrac{\partial^2 f}{\partial^2\kappa_1} & \dfrac{\partial^2 f}{\partial\kappa_1\partial\kappa_2} & \cdots & \dfrac{\partial^2 f}{\partial\kappa_1\partial\kappa_M}\\ \dfrac{\partial^2 f}{\partial\kappa_2\partial\kappa_1} & \dfrac{\partial^2 f}{\partial^2\kappa_2} & \cdots & \dfrac{\partial^2 f}{\partial\kappa_2\partial\kappa_M}\\ \vdots & \vdots & \ddots & \vdots\\ \dfrac{\partial^2 f}{\partial\kappa_M\partial\kappa_1} & \dfrac{\partial^2 f}{\partial\kappa_M\partial\kappa_2} & \cdots & \dfrac{\partial^2 f}{\partial^2\kappa_M}\end{bmatrix} \tag{10}$$

矩阵的每一项具体表示为

$$\frac{\partial^2 f}{\partial\kappa_i\partial\kappa_j}=\begin{cases}\dfrac{2w_i}{(\kappa_i^{\gamma_i})^3C_j^{\mathrm{E}}}, i=j\\ 0,\text{其他}\end{cases} \tag{11}$$

显然，式(10)的 Hessian 矩阵中对角线元素为正数，即式(10)为对称正定矩阵。又因为函数的约束是关于 κ_i 的线性约束，因此，函数 $f(x^0,\gamma^0,\kappa,\beta^0)$ 是关于计算能力分配系数 κ_i 的线性凸函数。

应用凸优化理论知识，可求得改进问题的最优值。改进函数的拉格朗日函数可构建为

$$L(\kappa,\nu)=f(x^0,\gamma^0,\kappa,\beta^0)-\sum_{j=1}^{N}\nu_j\left(1-\sum_{i\in O_{\gamma_i}}\kappa_i^{\gamma_i}\right) \tag{12}$$

利用 KKT 条件，最优值 κ 与 ν 需满足的条件为

$$\begin{cases}\nabla_\kappa L(\kappa,\nu)=0\\ \nu_j\left(1-\sum\kappa_i^{\gamma_i}\right)=0\\ \nu_j\geqslant 0\end{cases} \tag{13}$$

解式(13)的方程，可以得到最优计算能力分配策略 κ_i^* 为

$$\kappa_i^* = \frac{\sqrt{w_i}}{\sum\limits_{i\in O_j}\sqrt{w_i}} \tag{14}$$

目标函数最优值为

$$f(x^0,\gamma^0,\kappa,\beta^0)=\sum_{i=1}^{l}\sum_{i\in O_{\gamma_i}}\left[\frac{\sqrt{w_i}\sum\limits_{i\in O_{\gamma_i}}\sqrt{w_i}}{C_j^{\mathrm{E}}}+\frac{d_i}{B_j}\right]+$$

$$\sum_{i=1}^{p}\frac{d_i}{B_j^{\gamma_i}}+\sum_{i=M-l+1}^{M}\frac{w_i}{C_i^{\mathrm{D}}}=\sum_{j=1}^{N}\left[\frac{\left(\sum\limits_{i\in O_{\gamma_i}}\sqrt{w_i}\right)^2}{C_j^{\mathrm{E}}}+\sum_{i\in O_{\gamma_i}}\frac{d_i}{B_j}\right]+ \tag{15}$$

$$\sum_{i=1}^{p}\frac{d_i}{B_j^{\gamma_i}}+\sum_{i=M-l+1}^{M}\frac{w_i}{C_i^{\mathrm{D}}}$$

3.2　计算任务卸载问题

基于上述子问题进行分析，确定每一个在服务器计算的任务分配到计算能力比之后，需要确定每个任务的计算位置。

将求出的最优资源分配结果代入原始问题中，可以得到

$$t=\sum_{i=1}^{M}\left[(1-x_i)\frac{w_i}{C_i^{\mathrm{D}}}+x_i\left(\frac{\sqrt{w_i}\sum\limits_{i\in O_{\gamma_i}}\sqrt{w_i}}{C_j^{\mathrm{E}}}+\frac{d_i}{B_j}+\beta_i\frac{d_i}{B_j^{\gamma_i}}\right)\right] \tag{16}$$

原问题转变为关于 x_i、β^i 及卸载服务器位置 γ^i 的整数线性规划问题，根据前文建立的模型有 M 个任务、N 个服务器，如果使用穷举法，则算法的时间复杂度为 $O((n+1)^m)$，在网络规模较大的情况下，时间复杂度将很大。

第二个子问题中的整数线性规划问题属于组合最优化问题中的 NP 难问题，这类问题的求解方法主要分为精确算法和近似算法两类。在处理小规模问题时，精确算法能在短时间内给出最优解；当求解大规模组合优化问题时，理论上可以得到问题的最优解，但由于计算量过大、计算时间过长，使得精确算法在实际问题中难以直接应用。近似算法在合理计算时间内能够给出一个近似最优解，虽然不能保证得到问题的全局最优解，但问题求解速度较快，所给的近似最优解通常可以满足实际应用需求。其中，基于智能优化的近似算法是一类基于一定的优化搜索机制且具备全局优化特点的算法。考虑分支定界法是求解整数规划问题最常用的算法，不但可以求解纯整数规划问题，还可以求解混合整数规划问题，同时，通过搜索和迭代方法可以给出问题最优解。因此，采用分支定界法给出精确解。然而，在面向实际应用时，考虑现场设备、交换机、边缘服务器数量较大，造成组合优化问题规模、精确算法计算量和计算时间消耗过大，不适用于工程实践。考虑模拟退火算法是一种通用的随机搜索算法，理论上具有概率的全局优化性能，目前已在工程中得到了广泛应用。因此，本文同时给出一个基于模拟退火算法的启发式近似算法。

首先，使用分支定界法[14]来求解终端设备计算任务卸载问题，依次分配任务 $\{\mathrm{m}_1,\mathrm{m}_2,\cdots,\mathrm{m}_M\}$ 的计算位置。定义原问题的子问题，用 Γ_m 表示 m 个待分支界定的子问题。用 $\mathbb{F}_y$ 表示已经被分支确定的 y 个任务的总完成时间，则剩下的 $m-y$ 个任务总完成时间的子问题为

$$\Gamma_{m-y}=\mathbb{F}_y+\sum_{i=y+1}^{M}\left[(1-x_i)\frac{w_i}{C_i^{\mathrm{D}}}+x_i\left(\frac{\sqrt{w_i}\sum\limits_{i\in O_{\gamma_i}}\sqrt{w_i}}{C_j^{\mathrm{E}}}+\frac{d_i}{B_j}+\beta_i\frac{d_i}{B_j^{\gamma_i}}\right)\right] \tag{17}$$

基于上述子问题的定义，分支过程是在子问题可行域进行深度优先搜索，首先搜索一个最小边界。每当搜索树上的一个叶节点被搜索到时，其他拥有更大最小边界的枝将被剪切掉。当所有叶节点都被搜索完或剪切完时，就找到了最优分配方法。

相比于其他启发式算法，模拟退火算法的全局收敛性和稳健性良好，求得全局最优解的概率较大，适合求解大规模组合规划问题。因此，本文采用基于模拟退火的启发式算法[15]求解终端设备的计算任务卸载问题，从而给出最优卸载方案。模拟退火启发式算法是基于 Monte-Carlo 迭代求解策略的一种随机寻优算法，它从某一较高初温出发，随着温度参数的不断下降，结合概率的突跳特性在解空间中随机寻找目标函数的全局最优解，即在局部最优解能概率性地跳出并最终趋于区域全局最优。

首先，定义解向量 $\boldsymbol{Z}=(z_{ij})_{M\times(N+1)}$，$z_{ij}\in\{0,1\}$，表示所有设备的计算方式和卸载方案。解向量第一列 j=1 表示设备是否选择本地计算方式，解向量第

$j=2,3,\cdots,N+1$ 列表示设备是否选择在第 $1,2,\cdots,N$ 个边缘服务器上计算，$z_{ij}=0$ 表示任务 m_i 不会在本地计算或卸载至第 $j-1$ 个边缘服务器上计算，$z_{ij}=1$ 表示任务 m_i 选择在本地计算或卸载至第 $j-1$ 个边缘服务器上计算。通过启发式模拟退火算法求解矩阵 $\boldsymbol{Z}$，解决计算任务卸载问题。算法流程为：首先随机产生一个初始解矩阵 $\boldsymbol{z}_{\mathrm{old}}$，模拟温度用 T 表示。在开始时，T 取最大值 $T_{\max}$；然后开始循环迭代，每次循环中产生新的解矩阵 $\boldsymbol{z}_{\mathrm{new}}$，通过计算整数线性规划问题表达式可计算新、旧解矩阵的任务总时延变化量 Δt。当 $\Delta t<0$ 时，接受 $\boldsymbol{z}_{\mathrm{new}}$；反之，则以概率 $\mathrm{e}^{-\frac{\Delta t}{T}}$ 接受 $\boldsymbol{z}_{\mathrm{new}}$。在每次迭代时，$T$ 按冷却系数 $\varepsilon(0<\varepsilon<1)$ 减小，当 $T=T_{\min}$ 时，则循环迭代终止，并输出最优解矩阵。

上述次优启发式方法在 SDN 控制器中实现，在 SDN 控制器收集任务信息、计算设备信息以及 SDN 信息之后，先随机产生一个任务卸载方案解矩阵，然后在较短时间内迭代收敛至近似最优解。获得近似最优解后，SDN 控制器将解信息发送广播至现场设备、边缘服务器及 SDN 交换机。随后，建立相应的通信链路，任务进行本地计算或卸载至相应服务器进行处理。最后，当计算完成后，将计算结果发送至各个设备，启发式方法减小了卸载方案的确定与任务完成时间。

4 仿真结果与分析

为了验证计算任务跨域卸载方法的有效性，首先比较本文方法与传统的计算卸载方法的性能差异，即计算任务不能跨域卸载计算，边缘服务器之间不能相互卸载任务；然后研究在利用上述两种方法求解的情况下，计算任务的总时延与任务数量的关系，验证了本文所提方法在性能上的提升；最后比较分支定界法和启发式模拟退火算法两种最优与次优方案在计算任务的总时延和算法执行时间方面的差异。

本文的仿真架构和算法实现均在 Matlab 环境下进行，系统的仿真参数设置如表 2 所示，表 2 中给出了设备计算能力、边缘服务器计算能力、任务计算量、数据量大小以及相应的数据传输率。其中，变化的参数是利用随机函数在给定范围内生成的，为了模拟实际网络中负载不均衡的情况，不同簇域间任务的计算量以及边缘服务器的计算能力大小会有比较明显的差别。

表 2 系统的仿真参数设置

参数	仿真值
N	[50,100]
M	5
w_i	[20,100]
d_i	[20,100]
C_i^{D}	[0.5,1]
C_j^{E}	[10,100]
B_j	[200,1 000]
$B_j^{\gamma_i}$	1 000 MHz

在工业场景中，大部分计算任务需要进行实时处理，工业对计算任务的时延性要求更高，同时本文的优化目标是最小化任务完成时间。因此，为了评估所提出算法的有效性和性能，本文把任务总完成时间作为重要性能指标。同时，使用算法执行时间性能指标，比较了最优与次优方法的算法时间复杂度。

仿真实验中不考虑将信息传输量当作性能指标，因为本文所提的方法是通过计算任务跨域卸载缓解和解决不同计算域的负载分布不均，优化边缘计算资源利用率，进而减少任务完成总时延。系统模型假设簇域内现场设备与交换机的无线通信分配固定的信道带宽，交换机之间的通信一般采用高速有线通信，不考虑通信资源对于任务卸载的影响。同时，本文关注任务总完成时间，因此，不考虑将信息传输量作为性能指标。在仿真中，现场设备与交换机之间的数据传输按照仿真参数给定的值进行传输。本文比较分析以下 3 种方法的性能差异，即分支定界法、启发式模拟退火算法和传统的计算卸载方法。

首先，将设备数量设为 100 个，网络被分为 5 个簇域，则每个簇域中有 20 台设备，不同卸载策略下各域任务完成时间的比较如图 2 所示。图 2 比较了任务不参与跨域卸载和任务进行跨域卸载的情况下，各域中任务的完成时间。其中，任务跨域卸载使用分支定界法求解。在任务不参与跨域卸载时，发现簇域 2 和簇域 4 中任务的完成时间较长，说明这两个簇域中的计算负载较大；而任务参与计算卸载之后，可以把簇域 2 和簇域 4 中的

任务卸载到其他簇域计算资源丰富的服务器中，从而利用整体网络的计算资源缓解了负载不均衡的状况。

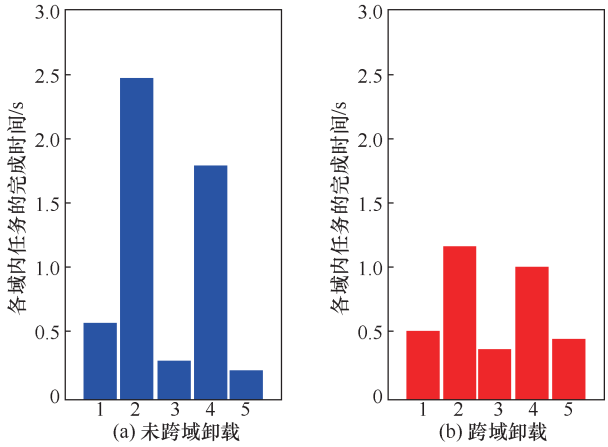


图 2　不同卸载策略下各域任务完成时间的比较

计算任务总时延与任务数量的关系如图 3 所示。图 3 比较了任务不参与跨域卸载和任务进行跨域卸载需要的总时延，任务不参与跨域卸载需要的总时延较高。在实际工业生产中，不同簇域网络中的任务计算量不同，任务若仅在当前域内进行计算，则全局网络中的计算资源分配是不均匀的，会造成计算资源的浪费。而进行跨域卸载，可以将计算量较大的任务卸载至其他簇域中比较空闲的服务器上进行计算，从而减小了任务的计算时延，总完成时间也相应地降低。

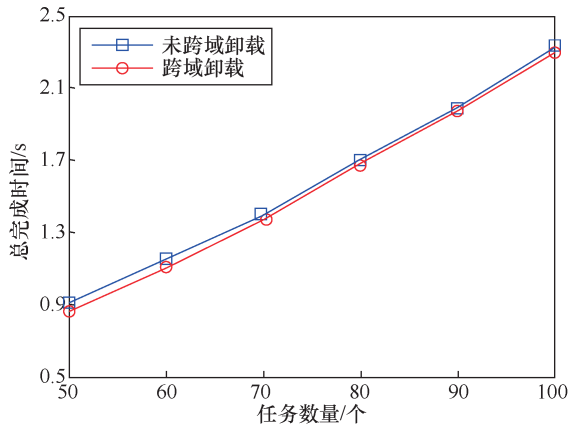


图 3　计算任务总时延与任务数量的关系

分支定界法与模拟退火算法的比较如图 4 所示。从图 4 中可以看出，两种方法的总时延都随着任务数量的增多而增加。分支定界法在不同任务数量下的性能都优于模拟退火算法，这是因为分支定界法是求解整数线性规划问题的精确算法，能够求得全局最优解。而模拟退火算法作为启发式算法，并不一定能找到全局最优解，但可以快速找到问题的近似最优解。

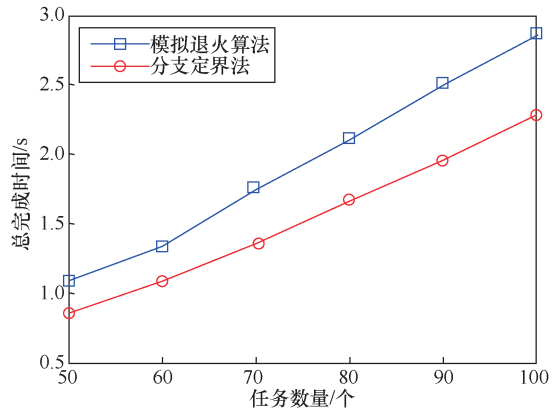


图 4　分支定界法与模拟退火算法的比较

当问题的规模为 n 时，分支定界法的时间复杂度为 $O(2^n)$，呈指数级增加；模拟退火算法的时间复杂度为 $O(kL_{\max}l(n))$，其中，k 为迭代次数，$L_{\max}$ 是与算法迭代次数相关的系数，$l(n)$ 是问题规模 n 的多项式函数，时间复杂度呈多项式增加。上述理论分析表明，随着问题规模的增加，模拟退火算法具有计算速度优势。

当设备规模逐渐增加时，比较模拟退火算法与分支定界法的算法执行时间，模拟退火算法与分支定界法的算法执行时间比较如表 3 所示。从表 3 可以看出，当设备规模较小时，分支定界法的算法执行时间远小于模拟退火算法。当设备规模逐渐增加时，分支定界法的算法执行时间迅速增加，当设备规模达到 250 台时，分支定界法的算法执行时间已经大于模拟退火算法的执行时间。因此，仿真结果表明，模拟退火算法在计算速度上更优。

表 3　模拟退火算法与分支定界法的算法执行时间比较

设备规模/台	模拟退火算法	分支定界法
100	2.484 s	0.822 s
150	3.513 s	2.125 s
200	4.614 s	4.353 s
250	5.462 7 s	7.958 s
300	6.690 s	13.066 s

5　结束语

本文研究了基于边缘计算的 IIoT 中最优资源

分配与最优任务卸载问题，考虑实际工业中现场设备、无线网络、边缘服务器的部署和工作特点，将所有底层和边缘设备划分成不同域进行任务卸载。针对本地设备、边缘服务器的计算能力不同，本地簇域网络的数据传输率与跨域数据传输率不同以及不同簇域网络内计算任务量分布不均，提出现场设备的计算任务可跨域卸载至其他域的边缘服务器上进行计算，分别建立了本地计算与边缘计算方式的完成时间模型，进而得到计算模式选择与计算资源分配的精确解和近似解。仿真结果表明，与不跨域卸载的计算任务处理策略相比，本文所提方法在性能上有一定提升，更有效地利用了计算资源，进而减小了任务完成时延。

参考文献：

[1] 邬贺铨. 物联网技术与应用的新进展[J]. 物联网学报, 2017, 1(1): 1-6.
WU H Q. Technology and application progress on Internet of things[J]. Chinese Journal on Internet of Things, 2017, 1(1): 1-6.

[2] EMILIANO S, ABUSAYEED S, SONG H, et al. Industrial Internet of things: challenges, opportunities, and directions[J]. IEEE Transactions on Industrial Informatics, 2018, 14(11): 4724-4734.

[3] YANG C W, HUANG Q Y, LI Z L, et al. Big data and cloud computing: innovation opportunities and challenges[J]. International Journal of Digital Earth, 2017, 10(1): 13-53.

[4] 李林哲, 周佩雷, 程鹏, 等. 边缘计算的架构、挑战与应用[J]. 大数据, 2019, 5(2): 3-16.
LI L Z, ZHOU P L, CHENG P, et al. Architecture, challenges and applications of edge computing[J]. Big Data Research, 2019, 5(2): 3-16.

[5] YU W, LIANG F, HE X, et al. A survey on the edge computing for the Internet of things[J]. IEEE Access, 2017, 6: 6900-6919.

[6] KAUR K, GARG S, AUJLA G S, et al. Edge computing in the industrial Internet of things environment: software-defined-networks-based edge-cloud interplay[J]. IEEE Communications Magazine, 2018, 56(2): 44-51.

[7] HONG Z C, CHEN W H, HUANG H W, et al. Multi-hop cooperative computation offloading for industrial IoT-edge-cloud computing environments[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(12): 2759-2774.

[8] YU Y, BU X Y, YANG K, et al. Green large-scale fog computing resource allocation using joint benders decomposition, dinkelbach algorithm, ADMM, and branch-and-bound[J]. IEEE Internet of Things Journal, 2019, 6(3): 4106-4117.

[9] LI X M, WAN J F, DAI H N, 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.

[10] MASOUDI R, GHAFFARI A. Software defined networks: a survey[J]. Journal of Network and Computer Applications, 2016, 67: 1-25.

[11] ZHANG J, FENG X, LIU Z. A grid-based clustering algorithm via load analysis for industrial Internet of things[J]. IEEE Access, 2018, 6: 13117-13128.

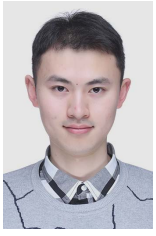
[12] SONG L M, CHAI K K, CHEN Y, et al. QoS-aware energy-efficient cooperative scheme for cluster-based IoT systems[J]. IEEE Systems Journal, 2017, 11(3): 1447-1455.

[13] PARDALOS P M. Convex optimization theory[M]. Massachusetts: Athena Scientific, 2011.

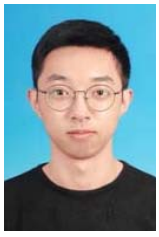
[14] BREMER K. Branch support and tree stability[J]. Cladistics, 1994, 10(3): 295-304.

[15] YAN W, XIE K G. Simulated annealing algorithm[J]. Journal of Mengzi Teathers College, 1999, 3(1): 95-98.

[作者简介]



周鹏（1995− ），男，安徽合肥人，上海交通大学硕士生，主要研究方向为工业物联网中的网络与计算优化。



徐金城（1996− ），男，江苏盐城人，上海交通大学硕士生，主要研究方向为工业物联网中的边缘计算、任务卸载和软件定义网络技术。

杨博（1980− ），男，上海人，上海交通大学教授、博士生导师，主要研究方向为物联网和能源互联网。