智慧城市预警系统云边协同计算场景下的卸载决策优化

朱思峰[1], 胡家铭[1], 杨诚瑞[1], 柴争义[2]
(1. 天津城建大学计算机与信息工程学院, 天津 300384; 2. 天津工业大学计算机科学与技术学院, 天津 300387)

摘 要: 针对智慧城市预警系统存在的传感器设备(sensor device,SD)的计算与存储能力不足、预警数据处理实时性差等问题,基于边缘计算技术,提出了云边协同的城市预警系统任务卸载模型。该模型引入了云边协同缓存策略,并依次设计了时延模型、能耗模型和负载失衡度模型;将任务卸载问题转化为多目标优化问题,给出了一种基于 MOEA/D 算法的卸载决策方案,并通过对比实验进行了验证。实验结果表明:该卸载方案能够在保证总时延与总能耗较小的情况下使负载达到均衡,并且优于其他基准方案。

关键词: 智慧城市; 预警系统; 云边协同; 任务卸载; 边缘缓存; 多目标优化
中图分类号: TP 391 **文献标志码**: A **文章编号**: 0254 - 0037(2023)09 - 1007 - 09
doi: 10.11936/bjutxb2022010001

Offloading Decision Optimization in the Cloud Edge Collaborative Computing Scenario of Smart City Early Warning System

ZHU Sifeng[1], HU Jiaming[1], YANG Chengrui[1], CHAI Zhengyi[2]
(1. School of Computer and Information Engineering, Tianjin Chengjian University, Tianjin 300384, China;
2. School of Computer Science and Technology, Tiangong University, Tianjin 300387, China)

Abstract: To address the problems of insufficient computing and storage capacity of sensor devices (SD) and poor real-time processing of early warning data in smart city early warning system, a task offloading model of cloud-edge collaborative urban early-warning system was proposed based on edge computing technology. The cloud-edge collaborative cache strategy was introduced, and the delay model, energy consumption model and load imbalance model were designed in turn. The task offloading problem was transformed into a multi-objective optimization problem, and an offloading decision scheme based on MOEA/D was given, which was verified by comparative experiments. Results show that this offloading scheme can achieve load balancing with less total delay and total energy consumption, and is superior to other benchmark schemes.

Key words: smart city; early warning system; cloud-edge collaborative; task offloading; edge cache; multi-objective optimization

城市物联网与通信技术飞速发展,促使城市现代化水平不断提高,城市规模扩大,流动人口增多,城市所面临灾害风险的复杂性与不确定性随之增加[1-2],科学有效地提升城市灾害风险感知与响应能

收稿日期: 2022-01-04; 修回日期: 2022-08-24
基金项目: 国家自然科学基金资助项目(61972456);天津市自然科学基金资助项目(22JCZDJC00600)
作者简介: 朱思峰(1975—), 男, 教授, 主要从事边缘计算、人工智能算法及应用方面的研究,E-mail: zhusifeng@163.com
通信作者: 柴争义(1976—), 男, 教授, 主要从事智能物联网、边缘计算方面的研究,E-mail: tgu_chaizhengyi@163.com

力十分重要[3]。

近年来,城市中部署的安全监测预警传感器设备(sensor device,SD)数量激增,此类设备会产生海量数据,并且数据处理时延敏感性强[4-5],但 SD 数据处理能力有限,通常需要将采集数据上传到云服务器进行集中处理[6]。此外,SD 与云服务器之间存在数据传输距离长、网络带宽受限等不足[7-8],无法满足现有应急预警系统数据高效实时处理的要求。边缘计算场景中的边缘服务器(edge servers, ES)靠近 SD 边缘,具有较强的计算存储能力[9-10],能满足时延敏感的预警数据处理需求。另外,城市中部署了需要数据处理框架(例如:AI 视频处理框架)才能完成数据处理的 SD[11],利用边缘缓存技术[12]可缓存相关数据处理框架,使数据在边缘端得到及时有效的处理。

执行数据计算卸载任务时,数据处理时效性极大影响城市灾害预警效果,此外,能耗也是城市建设需要考虑的关键问题[13],因此,应急预警数据计算卸载过程中须考虑时延和能耗[14]。另外,为实现城市全面精准的预警监测,SD 放置数量激增,ES 数据处理负担变重,可能会出现某些 ES 过载导致设备故障、某些 ES 欠载导致计算资源浪费的情况,应考虑 SD 的计算任务在多个 ES 间合理分配,实现 ES 间均衡负载[15],使得城市预警系统更加合理高效地进行数据处理。

综上所述,本文把 ES 作为 SD 与云服务器间的“桥梁”,考虑 ES 任务处理框架缓存,构建城市预警系统模型,并创建任务计算卸载时延、能耗和负载失衡度模型,应用多目标优化算法得出解决方案,满足城市应急预警数据及时处理需求,减少系统能耗,实现 ES 间均衡负载。

1 城市预警系统模型

系统模型如图 1 所示,共分为 3 层。第 1 层为终端设备层,包括采集、处理与上传预警数据的 SD 和接收来自 ES 层和部门云服务器层回传的预警信号的接收器设备,SD 由 $U=\{u_1,u_2,\cdots,u_i,\cdots,u_N\}$ 表示,其中 $i\in\{1,2,\cdots,N\}$。为了简化时延和能耗模型,本文假设 SD 同样充当接收器设备角色。第 2 层为 ES 层,ES 由 $S=\{s_1,s_2,\cdots,s_j,\cdots,s_M\}$ 表示。其中:$j\in\{1,2,\cdots,M\}$;$s_j=\{F_j^s,C_j^s,Q_j^s\}$,$F_j^s$ 表示 s_j 的计算能力,C_j^s 表示 s_j 的计算资源,Q_j^s 表示 s_j 的存储资源。在时隙 Δt 内,每一个 SD 都只产生一个待处理任务,任务表示为 $K=\{k_1,k_2,\cdots,k_i,\cdots,k_N\}$。其中:$i\in\{1,2,\cdots,N\}$;$k_i=\{d_i^{up},\omega_i,C_i,Q_i\}$,$d_i^{up}$ 表示 u_i 上传或本地执行任务 k_i 的数据量,ω_i 表示计算任务 k_i 所需计算量,C_i 表示执行任务 k_i 所需的计算资源,Q_i 表示任务 k_i 所需的存储资源。第 3 层为部门中心云层,云服务器由 Cloud 表示,可为 SD 提供计算卸载服务,为 ES 提供 k_i 的任务处理框架。本文仅考虑 Δt 时刻的任务计算卸载状况。ES 存储空间有限,不能缓存所有任务框架,并且 ES 缓存的任务处理框架需要经过较长时间才会释放,故本模型暂不考虑缓存释放。符号 $g_{ij}=\{0,1\}$ 表示 s_j 是否已缓存任务 k_i 所需任务处理框架。$g_{ij}=1$ 表示 s_j 已缓存任务 k_i 所需任务处理框架;$g_{ij}=0$ 表示 s_j 未缓存任务 k_i 所需任务处理框架,需要从 Cloud 下载。

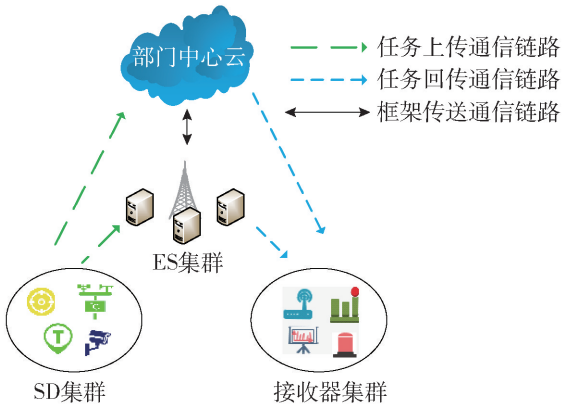

图 1 系统模型

Fig. 1 System model

假设本系统仅考虑一个时隙 Δt 内任务计算卸载状况,在时隙 Δt 内每个 ES 和 Cloud 都会收到数量不等的计算任务。用 $x_{ij}=\{0,1\}$ 表示任务 k_i 是否卸载到 s_j 或 Cloud 执行。$x_{ij}=0$ 表示任务 k_i 在本地执行;$x_{ij}=1,j\in\{1,2,\cdots,M\}$,表示任务 k_i 卸载到 s_j 执行;$x_{ij}=1,j=M+1$,表示任务 k_i 卸载到 Cloud 执行。

1.1 时延模型

本文主要考虑 6 种时延,即任务在本地的计算时延、任务卸载至 ES 的传输时延、任务在 ES 上的执行时延、Cloud 下载任务处理框架至 ES 的任务处理框架的下载时延、任务卸载至 Cloud 的传输时延和 Cloud 回传计算结果至终端设备层的回传时延。考虑到终端设备层与 ES 之间的传输距离较短,并且回传至终端设备层的数据量较小,故本文不考虑 ES 层回传计算结果至终端设备层的回传时延。

任务 k_i 在 u_i 上的本地计算时延 t_i^{local} 表示为

$$t_i^{\text{local}} = \frac{\omega_i}{f_i^{\text{local}}} \tag{1}$$

式中 f_i^{local} 表示 u_i 的计算能力。

任务 k_i 卸载至 s_j 的传输时延 t_{ij}^{tran} 表示为

$$t_{ij}^{\text{tran}} = \frac{d_i^{\text{up}}}{R_{ij}^{\text{tran}}} \cdot x_{ij} \tag{2}$$

式中 R_{ij}^{tran} 为 u_i 将任务 k_i 卸载至 s_j 的传输速率。

任务 k_i 在 s_j 上的计算执行时延 t_{ij}^{s} 表示为

$$t_{ij}^{\text{s}} = \frac{x_{ij} g_{ij} \omega_i}{F_j^{\text{s}}} + x_{ij} \cdot (1 - g_{ij}) \cdot \left(\frac{\omega_i}{F_j^{\text{s}}} + t_{\text{Cloud},s_j}^{\text{app}} \right) \tag{3}$$

式中 $t_{\text{Cloud},s_j}^{\text{app}}$ 为 Cloud 下载任务处理框架至 s_j 的下载时延，表示为

$$t_{\text{Cloud},s_j}^{\text{app}} = \frac{d_{\text{Cloud},s_j}^{\text{app}}}{R_{\text{Cloud},s_j}^{\text{app}}} \tag{4}$$

式中：$d_{\text{Cloud},s_j}^{\text{app}}$ 为 Cloud 下载至 s_j 的任务处理框架数据量大小；$R_{\text{Cloud},s_j}^{\text{app}}$ 为 Cloud 下载至 s_j 的任务处理框架传输速率。本文不考虑 s_j 向 Cloud 请求任务处理框架的传输时延。

任务 k_i 卸载至 Cloud 的传输时延 $t_{ij}^{\text{tran\_Cloud}}$ 表示为

$$t_{ij}^{\text{tran\_Cloud}} = \frac{x_{ij} d_i^{\text{up}}}{R_{ij}^{\text{tran\_Cloud}}} \tag{5}$$

式中 $R_{ij}^{\text{tran\_Cloud}}$ 为 u_i 将任务 k_i 卸载至 Cloud 的传输速率。

Cloud 将计算结果回传至 u_i 的回传时延 $t_{ji}^{\text{back\_Cloud}}$ 表示为

$$t_{ji}^{\text{back\_Cloud}} = \frac{x_{ij} d_{ji}^{\text{back}}}{R_{ji}^{\text{back\_Cloud}}} \tag{6}$$

式中：$R_{ji}^{\text{back\_Cloud}}$ 为 Cloud 将计算结果回传至 u_i 的回传速率；d_{ji}^{back} 为回传数据量大小。

任务 k_i 在 Cloud 上的处理时延 t_{ij}^{Cloud} 表示为

$$t_{ij}^{\text{Cloud}} = \frac{x_{ij} \omega_i}{F^{\text{Cloud}}} \tag{7}$$

式中 F^{Cloud} 表示 Cloud 的计算能力。

最终所有任务的执行总时延 T^{total} 表示为

$$\begin{cases} T^{\text{total}} = T^{\text{ES}} + T^{\text{Cloud}} + T^{\text{local}} \\ T^{\text{ES}} = \sum_{i=1}^{N} \sum_{j=1}^{M} (t_{ij}^{\text{tran}} + t_{ij}^{\text{s}}) \cdot x_{ij} \\ T^{\text{Cloud}} = \sum_{i=1}^{N} \sum_{j=M+1} (t_{ij}^{\text{tran\_Cloud}} + t_{ij}^{\text{Cloud}} + t_{ji}^{\text{back\_Cloud}}) \cdot x_{ij} \\ T^{\text{local}} = \sum_{i=1}^{N} \sum_{j=1}^{M+1} t_i^{\text{local}} \cdot (1 - x_{ij}) \end{cases} \tag{8}$$

式中：T^{ES} 为任务卸载到 ES 层的传输和计算总时延；T^{Cloud} 为任务卸载到 Cloud 层的任务传输、计算和回传总时延。

1.2　能耗模型

任务 k_i 在 u_i 本地计算的能耗 e_i^{local} 表示为

$$e_i^{\text{local}} = t_i^{\text{local}} \cdot p_i^{\text{local}} \tag{9}$$

式中 p_i^{local} 为任务 k_i 在 u_i 上的本地计算的能耗系数。

任务 k_i 卸载至 s_j 的传输能耗 e_{ij}^{tran} 表示为

$$e_{ij}^{\text{tran}} = t_{ij}^{\text{tran}} \cdot p_{ij}^{\text{tran}} \tag{10}$$

式中 p_{ij}^{tran} 为任务 k_i 卸载至 s_j 的传输能耗系数。

任务 k_i 在 s_j 上的执行能耗 e_{ij}^{s} 表示为

$$e_{ij}^{\text{s}} = t_{ij}^{\text{s}} \cdot p_{ij}^{\text{s}} \tag{11}$$

式中 p_{ij}^{s} 为任务 k_i 在 s_j 上的计算能耗系数。

任务 k_i 卸载至 Cloud 的传输能耗 $e_{ij}^{\text{tran\_Cloud}}$ 表示为

$$e_{ij}^{\text{tran\_Cloud}} = t_{ij}^{\text{tran\_Cloud}} \cdot p_{ij}^{\text{tran\_Cloud}} \tag{12}$$

式中 $p_{ij}^{\text{tran\_Cloud}}$ 为任务 k_i 卸载至 Cloud 的传输能耗系数。

任务 k_i 在 Cloud 上的执行能耗 e_{ij}^{Cloud} 表示为

$$e_{ij}^{\text{Cloud}} = t_{ij}^{\text{Cloud}} \cdot p_{ij}^{\text{Cloud}} \tag{13}$$

式中 p_{ij}^{Cloud} 为任务 k_i 在 Cloud 上的计算能耗系数。

Cloud 将任务计算结果回传至终端设备层的能耗 $e_{ji}^{\text{back\_Cloud}}$ 表示为

$$e_{ji}^{\text{back\_Cloud}} = t_{ji}^{\text{back\_Cloud}} \cdot p_{ji}^{\text{back\_Cloud}} \tag{14}$$

式中 $p_{ji}^{\text{back\_Cloud}}$ 为 Cloud 回传至终端设备层的回传能耗系数。

总能耗 E^{total} 表示为

$$\begin{cases} E^{\text{total}} = E^{\text{ES}} + E^{\text{Cloud}} + E^{\text{local}} \\ E^{\text{ES}} = \sum_{i=1}^{N} \sum_{j=1}^{M} (e_{ij}^{\text{tran}} + e_{ij}^{\text{s}}) \cdot x_{ij} \\ E^{\text{Cloud}} = \sum_{i=1}^{N} \sum_{j=M+1} (e_{ij}^{\text{tran\_Cloud}} + e_{ij}^{\text{Cloud}} + e_{ji}^{\text{back\_Cloud}}) \cdot x_{ij} \\ E^{\text{local}} = \sum_{i=1}^{N} \sum_{j=1}^{M+1} e_i^{\text{local}} \cdot (1 - x_{ij}) \end{cases} \tag{15}$$

式中：E^{ES} 为任务卸载到 ES 层的传输和计算总能耗；E^{Cloud} 为任务卸载到 Cloud 层的传输、计算和回传总能耗；E^{local} 为任务本地计算的任务总能耗。

1.3　负载失衡度模型

本文应用负载均衡机制管理 ES 负载情况。s_j 接收到的卸载任务数量 q_j 表示为

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

基于 q_j、s_j 的任务平均占用数量表示为

$$A = \frac{\sum_{j=1}^{M} q_j}{M} \tag{17}$$

负载失衡度可以表示为

$$L^{\text{imblan}} = \frac{\sum_{j=1}^{M} | q_j - A |}{M} \tag{18}$$

1.4 多目标优化模型

在智慧城市预警系统场景下,边缘计算卸载优化问题的优化目标是使系统总时延、总能耗、负载失衡度达到最小。该模型为三目标优化问题模型,属于多目标优化问题。建立计算卸载决策多目标优化模型,模型可表示为

$$\begin{aligned}
&\min\ T^{\text{total}}(X), \min\ E^{\text{total}}(X), \min\ L^{\text{imblan}}(X)\\
&\text{s. t.}\\
&\quad 1 \leqslant i \leqslant N\\
&\quad 1 \leqslant j \leqslant M+1\\
&\quad 0 \leqslant \sum_{j=1}^{M+1} x_{ij} \leqslant 1\\
&\quad \sum_{i=1}^{N} C_i \cdot x_{ij} \leqslant C_j^{\text{s}}\\
&\quad \sum_{i=1}^{N} Q_i \cdot x_{ij} \leqslant Q_j^{\text{s}}
\end{aligned} \tag{19}$$

对于约束条件的说明如下。

1) $1 \leqslant i \leqslant N$:系统中有 N 个 SD。

2) $1 \leqslant j \leqslant M+1$:系统中有 M 个 ES 和 1 个 Cloud。

3) $0 \leqslant \sum_{j=1}^{M+1} x_{ij} \leqslant 1$:$u_i$ 产生的任务 k_i 最多卸载到 1 个 ES 或 1 个 Cloud。

4) $\sum_{i=1}^{N} C_i \cdot x_{ij} \leqslant C_j^{\text{s}}$:卸载到 s_j 的卸载任务消耗的总计算资源小于或等于 s_j 的计算资源上限。

5) $\sum_{i=1}^{N} Q_i \cdot x_{ij} \leqslant Q_j^{\text{s}}$:卸载到 s_j 的卸载任务消耗的总存储资源小于或等于 s_j 的存储资源上限。

在求解多目标组合优化问题的过程中,某一个目标函数值的优化可能伴随着其他目标函数值的劣化。这种多目标优化问题的目标之间存在相互制约关系,通常不存在一个解使得所有目标均达到最优,一般会得到一个解集,被称作 Pareto 最优解集。本文利用基于分解的 MOEA/D 算法求解多目标优化问题的优势对问题模型进行求解。

2 利用 MOEA/D 算法求解卸载优化问题

MOEA/D 算法将多目标问题分解为多个标量的单目标优化问题,应用均匀的权重向量和参考点能使算法所得解的多样性更佳,计算效率更高[16]。权重向量主要负责将目标空间进行划分,种群的选择、进化、更新操作则在由多个权重向量构成的邻域中进行。权重向量确定后,多目标优化算法应用切比雪夫方法来比较进化解的优劣,指导种群进化方向。

2.1 问题编码

本文采用二进制编码和整数编码相结合方式对问题进行求解,对优化个体的约束检查和目标函数的计算均采用二进制编码方式,方便计算。对于个体进行进化操作时,采用整数编码方式。

采用二进制编码方案,个体 $\boldsymbol{X}$ 的编码表示为

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & \cdots & x_{1j} \\ \vdots & & \vdots \\ x_{i1} & \cdots & x_{ij} \end{pmatrix} \tag{20}$$

式中 $x_{ij} \in \boldsymbol{X}, i = \{1,2,\cdots,N\}, j = \{1,2,\cdots,M+1\}$。$x_{ij}=1$ 分为 2 种情况:当 $j \in \{1,2,\cdots,M\}$ 时,代表 u_i 将任务 k_i 卸载到 s_j 上执行;当 $j = M+1$ 时,代表 u_i 将任务 k_i 卸载到 Cloud 上执行。$x_{ij}=0$代表 u_i 将任务 k_i 在本地执行,$j \in \{1,2,\cdots,M+1\}$。

种群 O 的种群规模为 H,其编码表示为

$$O = \{\boldsymbol{X}^h \mid 1 \leqslant h \leqslant H\} \tag{21}$$

式中 h 代表种群 O 中第 h 个体 $\boldsymbol{X}$。

采用整数编码,同样种群规模为 H 的种群 O 的编码表示为

$$O = \{y_i^h \mid 1 \leqslant i \leqslant N, 1 \leqslant h \leqslant H\} \tag{22}$$

式中:y_i^h 中的 h 表示种群中第 h 个个体,y_i^h 中的 i 表示第 i 个终端设备 u_i;$y_i^h = 0$ 表示任务 k_i 在本地执行,当 $j \in \{1,2,\cdots,M\}$ 时,$y_i^h = j$,任务 k_i 卸载到 s_j 上执行,当 $j = M+1$ 时,$y_i^h = j$ 代表任务 k_i 卸载到 Cloud 上执行。

2 种编码方式的转换公式表示为

$$y_i^h = \begin{cases} 0, & \sum_{j=1}^{M+1} x_{ij} = 0 \\ j, & x_{ij} = 1 \end{cases} \tag{23}$$

采用二进制编码方案,编码方式表示为

$$\boldsymbol{G} = \begin{pmatrix} g_{11} & \cdots & g_{1j} \\ \vdots & & \vdots \\ g_{i1} & \cdots & g_{ij} \end{pmatrix} \tag{24}$$

式中 $\boldsymbol{G}$ 表示 ES 是否缓存任务 k_i 的处理框架，$g_{ij} \in \boldsymbol{G}, i = \{1,2,\cdots,N\}, j = \{1,2,\cdots,M\}$。如果 $g_{ij} = 1$，表示 s_j 已缓存任务 k_i 的处理框架；如果 $g_{ij} = 0$，表示 s_j 未缓存任务 k_i 的处理框架。具体分类情况如下。

1）$(x_{ij} \in \boldsymbol{X}, x_{ij} = 0) \cap (g_{ij} \in \boldsymbol{G}, g_{ij} = 0)$ 表示 s_j 未缓存任务 k_i 的处理框架且任务 k_i 在本地执行。

2）$(x_{ij} \in \boldsymbol{X}, x_{ij} = 1) \cap (g_{ij} \in \boldsymbol{G}, g_{ij} = 0)$ 表示 s_j 未缓存任务 k_i 的处理框架且任务 k_i 卸载至 s_j 执行。

3）$(x_{ij} \in \boldsymbol{X}, x_{ij} = 1) \cap (g_{ij} \in \boldsymbol{G}, g_{ij} = 1)$ 表示 s_j 已缓存任务 k_i 的处理框架且任务 k_i 卸载至 s_j 执行。

另外，以上 3 种情况中 $j \in \{1,2,\cdots,M\}$，因为只讨论 ES 是否缓存任务处理框架。

2.2　算法流程

MOEA/D 算法的流程见算法 1。

算法 1　MOEA/D

输入：多目标优化问题（见式（19））；最大迭代次数 $g_{\max}$；子问题数量 H；权重向量 $\lambda^1, \lambda^2, \cdots, \lambda^H$；每个权重向量邻域内权重向量个数 T

输出：外部存档 EP

初始化

1　外部存档 EP = ∅。

2　计算 2 个权重向量之间的欧氏距离，然后算出每个权重向量邻域内最近的 T 个向量，对于每一个 $h = 1,2,\cdots,H$，设置 $B(h) = h_1, h_2, \cdots, h_T, \lambda^{h_1}, \lambda^{h_1}, \cdots, \lambda^{h_T}$ 表示权重向量 λ^h 邻域内最近的 T 个向量。

3　随机生成初始种群 O，并对种群中的每个个体 $\boldsymbol{X}^h$ 计算适应度。

4　初始化参考点 $\boldsymbol{z} = (z_1, z_2, z_3)^{\mathrm{T}}$。

更新

5　for $h = 1,2,\cdots,H$ do

6　　从 $B(h)$ 中随机选择 2 个序号 α、β，然后利用遗传算子对个体 $\boldsymbol{X}^\alpha$、$\boldsymbol{X}^\beta$ 进行操作，产生一个新的个体 $\boldsymbol{Y}$。

7　　根据约束修正个体 $\boldsymbol{Y}$，产生新个体 $\boldsymbol{Y}'$。

8　　更新参考点 $\boldsymbol{z}$。

9　　对于每一个 $j \in B(h)$，应用切比雪夫法更新其邻域解。

10　　更新外部存档 EP。

11　终止判定。若满足终止条件则输出外部存档 EP，否则转向步骤 5。

3　实验与分析

为了评价基于 MOEA/D 卸载决策方案的有效性，本文把系统的响应时延（总时延）、能量消耗（总能耗）和负载失衡度作为执行任务卸载的综合代价，在相同的条件下，将 MOEA/D 方案与基于 NSGA-Ⅱ的方案、基于 NSGA-Ⅲ的方案、随机卸载方案、全部本地执行方案分别从 SD 数量变化、ES 数量变化、系统是否已部署边 ES、ES 有无缓存任务处理框架等方面进行对比实验。

3.1　实验参数设置

本文在内存为 8 GB、CPU 为 2.1 GHz、安装 Win10 操作系统的笔记本电脑上应用 Matlab2021 软件进行仿真实验。

终端设备 u_i 的计算能力 f_i^{local} 服从高斯分布 $f_i^{\mathrm{local}} \sim \mathrm{CN}(\mu_1, \delta_1^2)$，其中均值 $\mu_1 = 6$，方差 $\delta_1^2 = 0.5$，单位 GHz；任务 k_i 卸载至 s_j 的传输速率 R_{ij}^{tran} 服从高斯分布 $R_{ij}^{\mathrm{tran}} \sim \mathrm{CN}(\mu_2, \delta_2^2)$，其中均值 $\mu_2 = 10$，方差 $\delta_2^2 = 1$，单位 Mbit/ms；任务 k_i 上传或本地执行的数据量 d_i^{up} 服从高斯分布 $d_i^{\mathrm{up}} \sim \mathrm{CN}(\mu_3, \delta_3^2)$，其中均值 $\mu_3 = 10$，方差 $\delta_3^2 = 3$，单位 Mbit；任务 k_i 的计算量 ω_i 服从高斯分布 $\omega_i \sim \mathrm{CN}(\mu_4, \delta_4^2)$，其中均值 $\mu_4 = 20$，方差 $\delta_4^2 = 3$；任务 k_i 所需计算资源 C_i 随机分布在 $[100,200]$；任务 k_i 所需存储资源 Q_i 随机分布在 $[100,300]$；s_j 的计算能力 F_j^{s} 随机分布在 $[40,50]$，单位 GHz；Cloud 下载任务处理框架至 s_j 的任务处理框架数据量 $d_{\mathrm{Clouds}_j}^{\mathrm{app}}$ 设为 300，单位 Mbit；Cloud 下载任务处理框架至 s_j 的框架传输速率 $R_{\mathrm{Clouds}_j}^{\mathrm{app}}$ 值设为 100，单位 Mbit/ms；s_j 的计算资源 C_j^{s} 服从高斯分布 $C_j^{\mathrm{s}} \sim \mathrm{CN}(\mu_5, \delta_5^2)$，均值 $\mu_5 = 1\,500$，方差 $\delta_5^2 = 115$；s_j 的存储资源 Q_j^{s} 服从高斯分布 $Q_j^{\mathrm{s}} \sim \mathrm{CN}(\mu_6, \delta_6^2)$，均值 $\mu_6 = 950$，方差 $\delta_6^2 = 300$；Cloud 的计算能力 F^{Cloud} 随机分布在 $[200,500]$，单位 GHz；任务 k_i 卸载至 Cloud 的传输速率 $R_{ij}^{\mathrm{tran\_Cloud}}$ 服从高斯分布 $R_{ij}^{\mathrm{tran\_Cloud}} \sim \mathrm{CN}(\mu_7, \delta_7^2)$，其中均值 $\mu_7 = 2.5$，方差 $\delta_7^2 = 0.2$，单位 Mbit/ms；Cloud 回传计算结果至 u_i 的数据量 d_{ji}^{back} 服从高斯分布 $d_{ji}^{\mathrm{back}} \sim \mathrm{CN}(\mu_8, \delta_8^2)$，其中均值 $\mu_8 = 5$，方差 $\delta_8^2 = 1$，单位 Mbit；Cloud 回传计算结果至 u_i 的回传速率 $R_{ji}^{\mathrm{back\_Cloud}}$ 服从高斯分布 $R_{ji}^{\mathrm{back\_Cloud}} \sim \mathrm{CN}(\mu_9, \delta_9^2)$，其中均值 $\mu_9 = 4$，方差 $\delta_9^2 = 0.2$，单位 Mbit/ms。

为了公平起见，3 种算法设置参数相同，其中种群规模 $H = 105$，交叉概率 $p_{\mathrm{c}} = 0.9$，变异概率 $p_{\mathrm{m}} = 1/D$，D 为变量维数。

3.2　实验结果及讨论

本文对各个实验进行 25 次独立运行，取平均效果，并且本文假设 SD 卸载的计算任务均需任务处

理框架才能完成任务计算,云服务已缓存全部任务处理框架,SD 拥有计算本设备产生的部分任务处理框架。本实验的 SD 数量为 140 个,ES 数量为 20 个。5 种方案的 Pareto 解集在 3 个约束目标上的分布情况如图 2 所示。

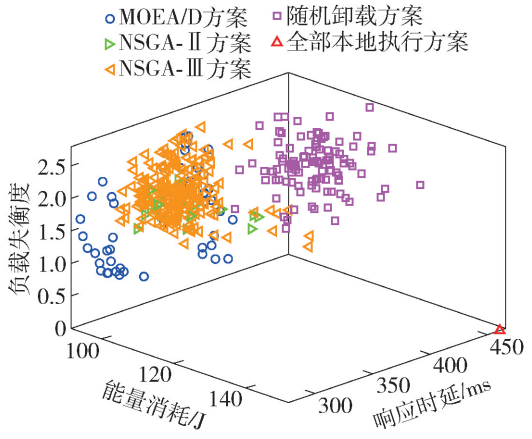

图 2 Pareto 解集在 3 个约束目标上的分布

Fig. 2 Distribution of Pareto solution set on three constrained targets

从图 2 可以看出:MOEA/D 方案得到的解在多数情况下优于其他 4 种方案;NSGA-Ⅱ方案和 NSGA-Ⅲ方案的效果相当;MOEA/D 方案、NSGA-Ⅱ方案和 NSGA-Ⅲ方案均优于随机卸载方案;MOEA/D 方案、NSGA-Ⅱ方案和 NSGA-Ⅲ方案在时延和能耗方面均优于全部本地执行方案。

3.3 SD 数量变化对 3 个约束目标的影响

为研究 SD 数量变化对 3 个约束目标的影响,本实验将 SD 数量分别设置为 100、120、140、160、180、200、220、240、260、280 个,ES 数量设置为 20 个。

5 种方案随 SD 数量变化对响应时延的影响如图 3 所示。可以看出,随着 SD 数量增加,5 种方案所得系统的响应时延均呈上升趋势。这是因为在 ES 数量不变的情况下,SD 数量增加导致系统总任务量增加,需要消耗更长时间处理 SD 产生的计算任务,造成系统总的响应时延增大。通过对比 5 种卸载方案可以看出,随着 SD 数量增加,MOEA/D 方案的响应时延均明显小于 NSGA-Ⅱ方案、NSGA-Ⅲ方案、随机卸载方案、全部本地执行方案的总的响应时延。

5 种方案随 SD 数量变化对能量消耗的影响如图 4 所示。可以看出,随着 SD 数量增加,5 种方案所得系统总的能量消耗呈上升趋势。这是因为在 ES 数量不变的情况下,SD 数量增加导致了系统总任务量增加,需要消耗更多能量处理 SD 产生的计

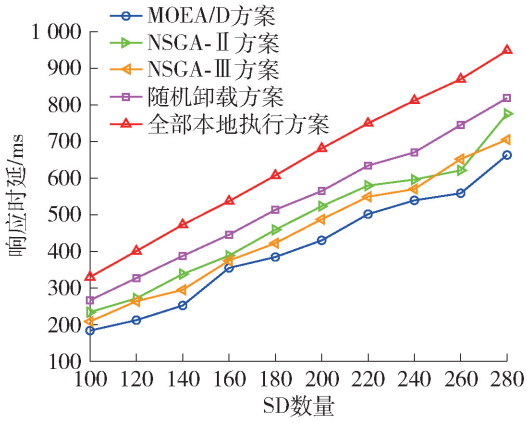

图 3 5 种方案随 SD 数量变化对响应时延的影响

Fig. 3 Effect of five schemes on response delay with the number of SD

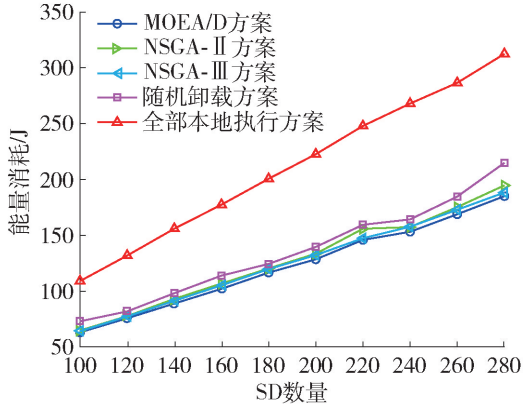

图 4 5 种方案随 SD 数量变化对能量消耗的影响

Fig. 4 Effect of five schemes on energy consumption with the number of SD

算任务,造成系统总的能量消耗增大。通过对比 5 种卸载方案可以看出,随着 SD 数量增加,MOEA/D 方案的能量均优于 NSGA-Ⅱ方案、NSGA-Ⅲ方案、随机卸载方案、全部本地执行方案的总的能量消耗。

5 种方案随 SD 数量变化对负载失衡度的影响如图 5 所示。可以看出,随着 SD 数量增加,虽然 5 种方案的负载失衡度均无明显趋势,但 MOEA/D 方案的负载失衡度均优于 NSGA-Ⅱ方案、NSGA-Ⅲ方案和随机卸载方案,因为全部本地执行方案不需要计算负载失衡度,所以值均为 0,并且不作为对比方案。

综上所述,随着 SD 数量的增加,MOEA/D 方案均能在 3 个约束目标上表现最优。

3.4 ES 数量变化对 3 个约束目标的影响

为研究 ES 数量变化对 3 个约束目标的影响,本

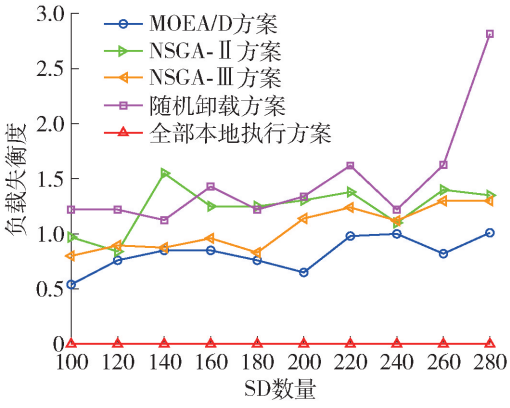

图 5　5 种方案随 SD 数量变化对负载失衡度的影响

Fig. 5　Effect of five schemes on load imbalance with the number of SD

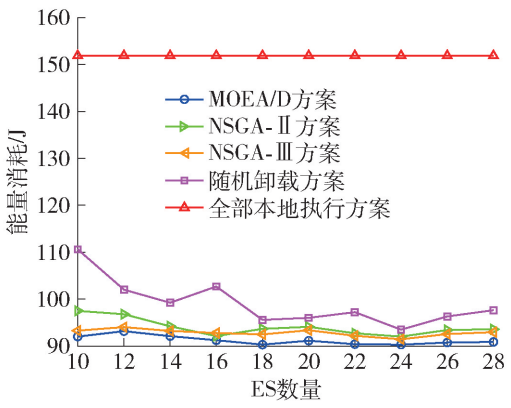

图 7　5 种方案随 ES 数量变化对能量消耗的影响

Fig. 7　Effect of five schemes on energy consumption with the number of ES

实验将 ES 数量分别设置为 10、12、14、16、18、20、22、24、26、28 个，SD 数量设置为 140 个。

5 种方案随 ES 数量变化对响应时延的影响如图 6 所示。可以看出，随着 ES 数量增加，MOEA/D、NSGA-Ⅱ、NSGA-Ⅲ和随机卸载这 4 种方案所得系统总的响应时延总体呈下降趋势，并且 MOEA/D 方案的响应时延均优于其他 4 种方案。另外，由于 SD 数量固定，系统产生的总的任务计算量固定，随着 ES 数量变化，全部本地执行方案的响应时延保持不变。

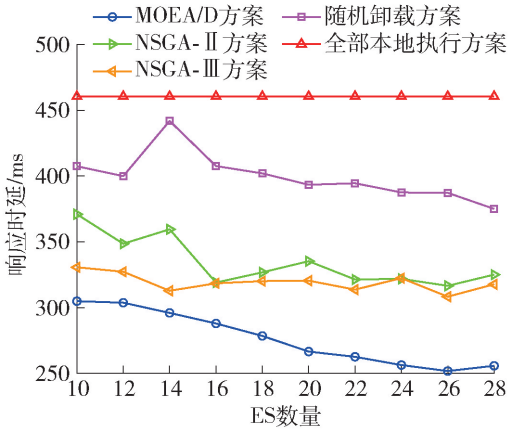

图 6　5 种方案随 ES 数量变化对响应时延的影响

Fig. 6　Effect of five schemes on response delay with the number of ES

5 种方案随 ES 数量变化对能量消耗的影响如图 7 所示。可以看出，随着 ES 数量增加，MOEA/D、NSGA-Ⅱ、NSGA-Ⅲ和全部本地卸载这 4 种方案所得系统总的能量消耗总体波动较小且呈现平稳趋势。随机卸载方案的能量消耗波动较大，MOEA/D 方案的能量消耗均优于其他 4 种方案。

5 种方案随 ES 数量变化对负载均衡度的影响如图 8 所示。可以看出，随着 ES 数量增加，虽然 5 种方案的负载失衡度均无明显趋势，但 MOEA/D 方案的负载失衡度均优于 NSGA-Ⅱ方案、NSGA-Ⅲ方案和随机卸载方案。因为全部本地执行方案不需要计算负载失衡度，所以值均为 0 且不作为对比方案。

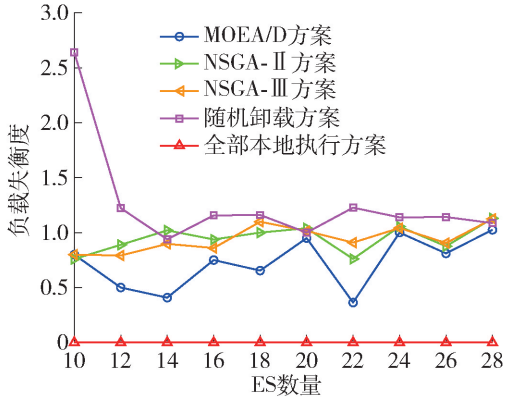

图 8　5 种方案随 ES 数量变化对负载失衡度的影响

Fig. 8　Effect of five schemes on load imbalance with the number of ES

综上所述，随着 ES 数量的增加，MOEA/D 方案均能在 3 个约束目标上表现最优。

3.5　ES 有无缓存框架对响应时延的影响

为研究 ES 有无缓存任务处理框架对响应时延的影响，将 SD 数量设定为 140 个，ES 数量设定为20 个。

5 种方案在 ES 有缓存任务处理框架和 ES 无缓存任务处理框架 2 种情况下，响应时延的变化如图 9 所示。可以看出，除全部本地执行方案外，ES 有缓存任务处理框架与无缓存任务处理框架相比，均可降低其他 4 种方案的响应时延。此外，MOEA/D 方案在 ES 有缓存任务处理框架情况下得到的响

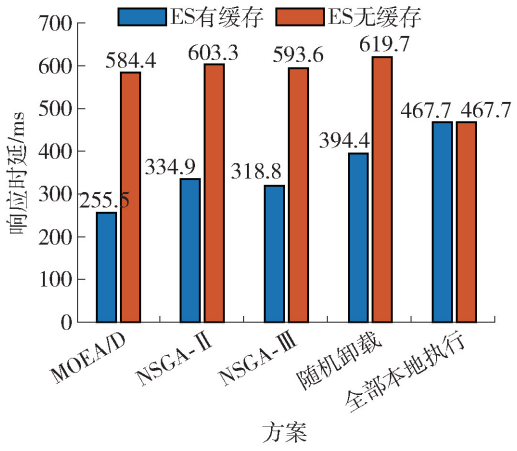

图 9 ES 有无缓存框架对响应时延的影响

Fig. 9 Effect of ES with or without caching framework on response delay

应时延均优于其他 4 种方案。MOEA/D 方案在 ES 无缓存任务处理框架情况下得到的响应时延均优于 NSGA-Ⅱ、NSGA-Ⅲ、随机卸载这 3 种方案,这是因为在无边缘缓存情况下,ES 计算传感器卸载的任务时,需要从 Cloud 下载任务处理框架,使得响应时延增加,导致 MOEA/D 方案劣于全部本地执行方案。

3.6 系统是否部署 ES 对响应时延和能量消耗的影响

为研究系统中是否部署 ES 对系统响应时延和能量消耗的影响,本文考虑系统中部署 ES 和系统中未部署 ES 这 2 种情况。2 种情况的 SD 数量均设定为 140 个。已部署 ES 时,将其数量设定为 20 个;未部署 ES 时,将其数量设定为 0 个。

4 种方案在系统中是否部署 ES 场景下响应时延变化如图 10 所示。可以看出,系统中部署 ES 场景下 4 种方案的响应时延均比系统未部署 ES 场景下 4 种方案的响应时延低。此外,MOEA/D 方案在系统中部署 ES 和未部署 ES 这 2 种场景下的响应时延均优于其他 3 种方案。

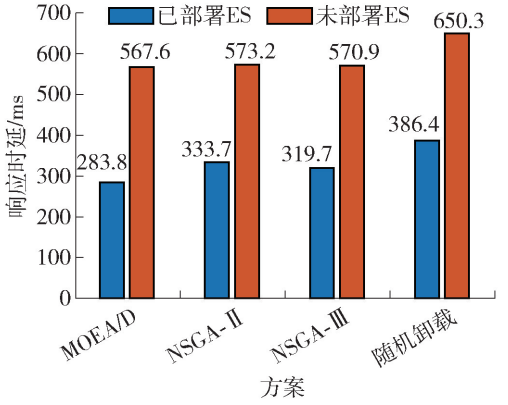

图 10 4 种方案在系统中是否部署 ES 场景下响应时延变化

Fig. 10 Response delay change in the scenario of whether ES are deployed in the system for the four schemes

4 种方案在系统中是否部署 ES 场景下能量消耗变化如图 11 所示。可以看出,系统中部署 ES 场景下 4 种方案的能量消耗均比系统未部署 ES 场景下 4 种方案的能量消耗低。此外,MOEA/D 方案在系统中部署 ES 和未部署 ES 这 2 种场景下的能量消耗均优于其他 3 种方案。

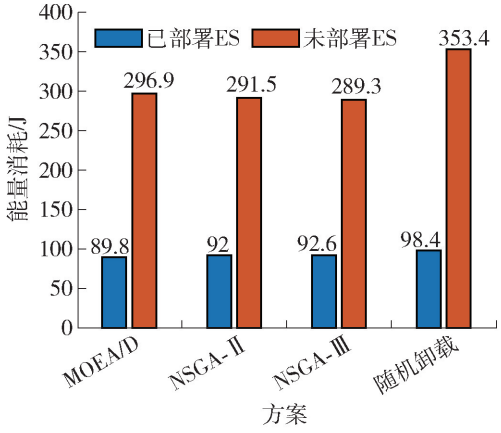

图 11 4 种方案在系统中是否部署 ES 场景下能量消耗变化

Fig. 11 Energy consumption change in the scenario of whether ES are deployed in the system for the four schemes

4 结论

1) 本文建立了智慧城市预警系统边缘计算场景下的任务卸载多目标优化问题模型,考虑了响应时延、能量消耗和负载失衡度这 3 个优化目标。

2) 应用 MOEA/D 算法对所构造的多目标优化问题模型进行求解,仿真结果表明,本文给出的 MOEA/D 卸载方案较优。

3) 下一步工作,将研究智慧城市预警系统中数据安全和隐私保护问题。

参考文献:

[1] 鲁钰雯, 翟国方. 人工智能技术在城市灾害风险管理中的应用与探索[J]. 国际城市规划, 2021, 36(2): 22-31, 39.
LU Y W, ZHAI G F. Applications and exploration of artificial intelligence technology in urban disaster risk management[J]. Urban Planning International, 2021, 36(2): 22-31, 39. (in Chinese)

[2] 石娟, 郑鹏, 常丁懿. 大数据环境下的城市公共安全治

理：区块链技术赋能[J]. 中国安全科学学报，2021，31(2)：24-32.
SHI J，ZHENG P，CHANG D Y. Governance of urban public safety in context of big data：block chain technology enablement[J]. China Safety Science Journal，2021，31(2)：24-32.（in Chinese）

[3] 郑宇. 城市治理一网统管[J]. 武汉大学学报(信息科学版)，2022，47(1)：19-25.
ZHENG Y. Unified urban governance models[J]. Geomatics and Information Science of Wuhan University，2022，47(1)：19-25.（in Chinese）

[4] 张伟东，高智杰，王超贤. 应急管理体系数字化转型的技术框架和政策路径[J]. 中国工程科学，2021，23(4)：107-116.
ZHANG W D，GAO Z J，WANG C X. Digital transformation of emergency management system：technical framework and policy path[J]. Strategic Study of CAE，2021，23(4)：107-116.（in Chinese）

[5] EVANGELOS M，LAZAROS K，ARIS D，et al. Public safety in smart cities under the edge computing concept[C]// IEEE International Mediterranean Conference on Communications and Networking. Piscataway：IEEE，2021：88-93.

[6] PERNA G，ROSMANINHO R，HUGO S，et al. On the performance of 5G for cloud-and edge-based emergency services in smart cities[C]// 2021 12th International Conference on Network of the Future. Piscataway：IEEE，2021：1-5.

[7] JIANG X T，YU F R，SONG T，et al. A survey on multi-access edge computing applied to video streaming：some research issues and challenges[J]. IEEE Communications Surveys & Tutorials，2021，23(2)：871-903.

[8] WU H M，ZHANG Z R，GUAN C，et al. Collaborate edge and cloud computing with distributed deep learning for smart city Internet of things[J]. IEEE Internet of Things Journal，2020，7(9)：8099-8110.

[9] ZHAO X B，ZENG Y，DING H W，et al. Optimize the placement of edge server between workload balancing and system delay in smart city[J]. Peer-to-Peer Networking and Applications，2021，14(6)：3778-3792.

[10] 张依琳，梁玉珠，尹沐君，等. 移动边缘计算中计算卸载方案研究综述[J]. 计算机学报，2021，44(12)：2406-2430.
ZHANG Y L，LIANG Y Z，YIN M J，et al. Survey on the methods of computation offloading in mobile edge computing[J]. Chinese Journal of Computers，2021，44(12)：2406-2430.（in Chinese）

[11] 栾清华，秦志宇，王东，等. 城市暴雨道路积水监测技术及其应用进展[J]. 水资源保护，2022，38(1)：106-116，140.
LUAN Q H，QIN Z Y，WANG D，et al. Review on monitoring technology of urban road waterlogging after rainstorm and its application[J]. Water Resources Protection，2022，38(1)：106-116，140.（in Chinese）

[12] 黄永明，郑冲，张征明，等. 大规模无线通信网络移动边缘计算和缓存研究[J]. 通信学报，2021，42(4)：44-61.
HUANG Y M，ZHENG C，ZHANG Z M，et al. Research on mobile edge computing and caching in massive wireless communication network[J]. Journal on Communications，2021，42(4)：44-61.（in Chinese）

[13] LEE K，SILVA B N，HAN K J. Algorithmic implementation of deep learning layer assignment in edge computing based smart city environment[J]. Computers & Electrical Engineering，2021，89：106909.

[14] XU X L，GU R H，DAI F，et al. Multi-objective computation offloading for Internet of vehicles in cloud-edge computing[J]. Wireless Networks，2020，26(11)：1611-1629.

[15] XU X L，HUANG Q H，YIN X C，et al. Intelligent offloading for collaborative smart city services in edge computing[J]. IEEE Internet of Things Journal，2020，7(9)：7919-7927.

[16] 李林林，刘东梅，王显鹏. 基于改进 MOEA/D 的多目标置换流水车间调度问题[J]. 计算机集成制造系统，2021，27(7)：1929-1940.
LI L L，LIU D M，WANG X P. Multi-objective permutation flowshop scheduling problem based on improved MOEA/D[J]. Computer Integrated Manufacturing Systems，2021，27(7)：1929-1940.（in Chinese）

（责任编辑　梁　洁）