文章编号: 1007-5321(2020) 06-0096-07 DOI: 10. 13190 / j. jbupt. 2020-089

基于 KM 算法的分布式无线节点任务分配方法

田兴鹏， 朱晓荣， 朱洪波
(南京邮电大学 通信与信息工程学院, 南京 210003)

摘要: 单个节点无法满足各种新颖的应用程序对时延或能耗的要求,为此提出了一种分布式无线节点任务协同分配方法,通过利用周围节点的空闲资源,来降低所有节点处理任务的总时延或总能耗. 首先根据层次分析法(AHP) 综合任务的多维属性,如计算负载、最晚完成时间等,确定任务执行的优先级; 然后建立时延和能耗的优化模型,并将其转化为二分图最大权值的匹配问题,采用 Kuhn Munkras(KM) 算法求解得到任务分配的最优解,实现终端节点在网络边缘高效地协同执行任务. 仿真结果表明,该算法能够有效地降低任务处理的时延和能耗.

关 键 词: 任务分配; 异构网络; 层次分析法; KM 算法
中图分类号: TP393 **文献标志码**: A

Distributed Wireless Node Task Allocation Method Based on KM Algorithm

TIAN Xing-peng， ZHU Xiao-rong， ZHU Hong-bo
(College of Telecommunications & Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: Aiming at the fact that a single node cannot meet the delay or energy consumption requirements of various novel applications, a distributed wireless node task collaborative allocation method is proposed to reduce the total delay or total energy consumption of all node processing tasks by utilizing the idle resources of surrounding nodes. Firstly, according to the analytic hierarchy process (AHP) , the priority of task execution is determined according to the multi-dimensional attributes of tasks, such as calculation load and latest completion time. Then, the optimization model of time delay and energy consumption is established, which is transformed into the problem of maximum weight matching of bipartite graph. The optimal solution of task allocation is obtained by using Kuhn Munkras(KM) algorithm, which realizes the efficient cooperation of terminal nodes at the edge of network. The simulation results show that the algorithm can effectively reduce the time delay and energy consumption of task processing.

Key words: task assignment; heterogeneous network; analytic hierarchy process; Kuhn Munkras algorithm

随着第 5 代移动通信系统(5G, the fifth generation of mobile communications system) 时代的到来,越来越多新颖的应用逐渐兴起,如图像识别、物联网数据流处理、移动健康计算等,这些应用的程序对时延要求较高,并且通常需要大量的计算资源进行处理,而物联网中各种终端设备的处理能力有限,若由单

收稿日期: 2020-07-23
基金项目: 国家自然科学基金项目(61871237) ; 江苏省高校“青蓝工程”和江苏省重点研发计划项目(BE2019017)
作者简介: 田兴鹏(1995—) , 男, 硕士生.
通信作者: 朱晓荣(1977—) , 女, 教授, 博士生导师, E-mail: xrzhu@ njupt. edu. cn.

个设备执行应用程序会产生较大的能耗和时延[1]. 因此,目前常见的一种解决方案是将计算密集型任务卸载到资源丰富的云端执行. 但由于远端云与终端设备之间的信道链路不稳定且距离较远,将计算任务卸载到云端执行会产生较大的传输时延. 由于边缘计算靠近数据产生的源头,具有邻近性、低时延、高宽带和位置认知等特点,在最近几年得到了快速发展[2]. 然而在即将到来的5G 网络中,将会有数以亿计的智能设备连接到网络,如自动驾驶汽车、智能家居设备、可穿戴设备、手机、急剧增长的各种嵌入式设备等[3]. 随着终端节点数量的爆炸式增长,边缘服务器将产生巨大的链路负担和任务负载,导致出现传输能耗大、任务执行时延高等各种问题. 因此,利用周边节点的计算、通信等资源,在分布式节点中进行任务分配,各节点相互协作来完成任务,成为一个新的解决途径.

在分布式系统中进行任务分配一直是研究的热点, Funai 等[4-5] 以实验的方式评估了在同构的分布式协作网络中进行任务分配的性能,提出了一种迭代的任务分配算法,该算法在考虑网络通信开销的情况下,优化分布式节点间的任务分配. 针对多跳网络中的任务分配,Funai 等[5] 考虑了通信和计算的比例对系统性能的影响,并在计算时延和网络生命周期方面给出了折中方案. 对于移动边缘计算中的任务卸载,Chen 等[6] 提出了一种用于 5G 移动边缘计算的端到端(D2D,device to device) 人群框架. 在该框架中,处于网络边缘的节点利用周围节点进行计算和通信资源的共享,同时提出了一种基于图匹配的任务分配策略,然而其仅考虑了单轮的任务分配,各个节点中仅存在单个待分配的任务. Fan 等[7-8] 将单个计算密集型任务分解为多个具有依赖关系的子任务,以最小化任务完成时间为目标,将异构网络中任务分配问题转化为一个路径搜索问题,以任务在不同节点执行的时间作为路径成本,采用 A* 算法进行路径搜索,找到任务的最佳分配. Wang 等[8] 提出了一种基于分数激励机制的任务分配算法,分数用来奖励或惩罚参与任务执行的节点,由头节点负责更新,根据任务得分,网络中各个节点可以相互协作来完成复杂的任务. 对多个独立的任务,Yin 等[9] 考虑每个任务的执行需要不同的资源,将网络中的节点进行分簇,提出了一种两阶段的分配算法,以延长网络的存活时间为目标,将任务依次在集群间和集群内进行分配. Pu 等[10] 提出一种完全分布式的自组织网络中的任务分配方法,由于不存在中心节点,每个节点都要与周围的节点频繁地交换信息,所以产生较大的通信能耗. Sahni[11-12] 等均采用生物启发式算法,如遗传算法、粒子群算法,在分布式系统中进行任务分配,这类基于生物启发式的算法通常需要输入一些控制参数,如遗传算法中的交叉概率和变异概率,而且搜索算法在全局解空间中进行搜索,当任务规模与节点规模较大时,算法通常具有较大的时间开销.

针对上述研究的不足,笔者考虑在一个异构的分布式系统中进行任务分配,系统中每个节点可与周围的节点通过无线连接组成一个协作网络,共同完成各个节点产生的任务. 对每个节点产生的多个任务首先通过层次分析法对任务进行优先级划分,然后各个节点将按照优先级依次将任务信息传递给中心节点,中心节点处建立时延和能量损耗的最优任务卸载模型,通过 KM(Kuhn Munkras) 算法将任务分配给系统中合适的节点执行,使得整个系统的能耗或任务执行的时间最少.

1　系统模型

1.1　网络模型

图 1 所示为分布式协同计算网络架构. 该系统由 1 个中心节点和 m 个相互协作异构的分布式节点 2 部分组成. 中心节点与分布式节点通过无线连接进行通信,中心节点收集并更新分布式节点的状态信息,如剩余能量、空闲时间、计算速度、功耗等,基于这些信息,对各节点提交的任务进行分配. 中心节点不参与任务的执行. 分布式节点是具有计算、通信、存储等功能的异构设备. 同样,

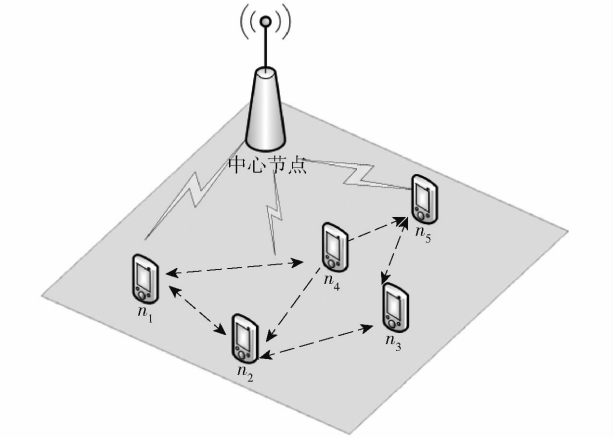

图 1　分布式无线节点网络模型

分布式节点之间通过无线连接,各节点地位平等,每个节点既可以发起任务,也可以参与完成其他节点的任务,从而在网络边缘共享计算和通信资源. 从图1可见,若节点 n_1 产生多个计算密集型任务,而目前它的计算资源被其他应用程序大量占用,此时,节点 n_1 将任务信息提交给中心节点,中心节点结合状态信息,将任务在与 n_1 直接相连的(包括 n_1) n_2、n_4 中进行分配,以便于节省任务的执行时间或者能耗.

用一个无向图 $G=\{N,E\}$ 表示多个分布式节点构成的网络. $N=\{n_1,n_2,\cdots,n_m\}$ 表示网络中的 m 个节点,$E=\{e_{ij}\}$ 表示分布式节点之间的连接. 对于节点 n_i,f_i 表示其工作频率(单位时间内的CPU周期),P_i^{t},P_i^{r} 分别表示 n_i 的发送功率和接收功率. P_i^{e} 表示单位时间内 n_i 计算消耗的能量. d_{ij} 为节点 n_i 和节点 n_j 之间的传输速度,E_i^{r} 表示节点 n_i 的剩余能量.

1.2 任务模型

分布式节点中,某个节点产生了 k 个任务,$T=\{t_1,t_2,\cdots,t_k\}$,每个任务具有不同的属性,因此需要在分布式节点中对任务进行优先级排序,保证最紧急的任务获得最高的优先处理级别. 对每个任务,通过一组参数建模$\{l_i,o_i,w_i,s_i\}$,其中,l_i 为任务输入数据的大小,o_i 为任务输出数据的大小,w_i 为任务的计算负载(需要的CPU周期数),s_i 为任务的最晚完成时间. 以下任务均假设任务输出数据的大小比输入数据的大小小得多. 上述假设适用于各种无线分布式计算的应用场景. 例如,在图像识别任务中,输入数据需要的是完整的图像,任务执行完得到的结果仅包含识别的结果.

1.2.1 任务优先级模型

根据任务的属性,通过层次分析法(AHP,analytic hierarchy process)对任务进行优先级排序. AHP是一个多标准决策/多属性决策模型,在各个领域都有应用,是一种适用于解决基于优先级调度的方法[13].

如图2所示,任务优先级模型包括目标层 A、准则层 B、方案层 C. 由于任务的输出数据通常很小,所以在对任务进行优先级划分时,主要考虑输入的数据大小 l_i、计算负载 w_i 和最晚完成时间 s_i 这3个因素. 其中任务的最晚完成时间相比输入数据大小和计算负载更重要,因此,在层次分析模型中,任务的最晚完成时间在优先级划分中所占的权重最高.

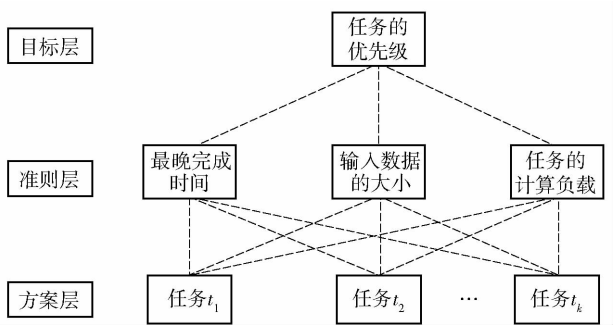

图2 任务优先级模型

1) 构造判断矩阵

判断矩阵表示同一层次中各个因素对于上一层次某个因素的重要程度,可以通过对同一层次的因素进行两两比较得到. 如 A 层次中的因素 A_k 与 B 层次中的 n 个因素有关,则可以建立 $B_1B_2B_n$ 对 A_k 的判断矩阵

$$\begin{bmatrix} b_{11} & b_{12} & \vdots & b_{1n} \\ b_{21} & b_{22} & \vdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & b_{nn} \end{bmatrix}$$

其中

$$b_{ij}=\begin{cases} \dfrac{1}{b_{ji}}=x,x=\{1,2,\cdots,9\}, & i\neq j \\ 1, & i=j \end{cases} \tag{1}$$

需要4个判断矩阵,包括1个目标层判断矩阵 $\boldsymbol{A}^B$ 和3个准则层的判断矩阵 $\boldsymbol{B}_1^C$,$\boldsymbol{B}_2^C$,$\boldsymbol{B}_3^C$.

2) 层次排序

分别求出 $\boldsymbol{A}^B$ 矩阵和3个准则层矩阵的最大特征值对应的特征向量,$\boldsymbol{\Theta}=[\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\boldsymbol{\theta}_3]^{\mathrm{T}}$,$\boldsymbol{\phi}^i=[\boldsymbol{\phi}_1^i,\boldsymbol{\phi}_2^i,\cdots,\boldsymbol{\phi}_k^i]^{\mathrm{T}}$,$\boldsymbol{\phi}_k^i$ 表示节点的第 k 个任务对决策层中的第 i 个因素的权重. 将特征向量中各分量的大小作为对下一层次中各因素排列的根据,其值越大,表示下一层次中的因素对于上一层次的某个因素越重要. 所有任务权重所对应的矢量组成矩阵,有

$$\boldsymbol{\phi}=[\boldsymbol{\phi}^1,\boldsymbol{\phi}^2,\boldsymbol{\phi}^3]=\begin{bmatrix} \boldsymbol{\phi}_1^1 & \boldsymbol{\phi}_1^2 & \boldsymbol{\phi}_1^3 \\ \vdots & \vdots & \vdots \\ \boldsymbol{\phi}_k^1 & \boldsymbol{\phi}_k^1 & \boldsymbol{\phi}_k^3 \end{bmatrix} \tag{2}$$

则各个任务的优先级向量 $\boldsymbol{\lambda}$ 为

$$\boldsymbol{\lambda}=\boldsymbol{\phi}\boldsymbol{\Theta}=\begin{bmatrix} \boldsymbol{\phi}_1^1 & \boldsymbol{\phi}_1^2 & \boldsymbol{\phi}_1^3 \\ \vdots & \vdots & \vdots \\ \boldsymbol{\phi}_k^1 & \boldsymbol{\phi}_k^1 & \boldsymbol{\phi}_k^3 \end{bmatrix}[\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\boldsymbol{\theta}_3]^{\mathrm{T}}=$$

$$\begin{bmatrix} \sum_{i=1}^{3} \boldsymbol{\phi}_1^i \times \boldsymbol{\theta}_i \\ \sum_{i=1}^{3} \boldsymbol{\phi}_2^i \times \boldsymbol{\theta}_i \\ \vdots \\ \sum_{i=1}^{3} \boldsymbol{\phi}_k^i \times \boldsymbol{\theta}_i \end{bmatrix} \tag{3}$$

向量 $\boldsymbol{\lambda}$ 指示了任务执行的次序.

1.2.2　任务分配模型

各个分布式节点对任务优先级进行排序后,每次将优先级最高的任务信息传输至中心节点,进行一轮任务分配,直到所有节点的任务分配完成,如图 3 所示. 在分配过程中,中心节点需要考虑各个分布式节点消耗的时间和能量.

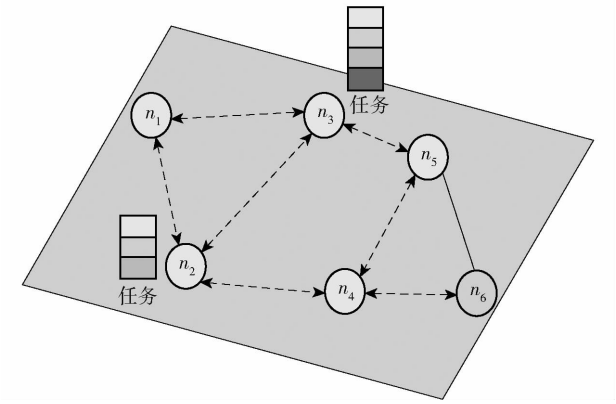

图 3　任务分配模型

对节点 n_i,其任务既可以在本地执行,也可以由相邻节点执行. 对中心节点,使用 t_i 表示节点 n_i 当前待分配的优先级最高的任务. 任务在本地执行时,需要的时间为

$$T_i^{\mathrm{l}} = T_i^{\mathrm{b}} + T_i^{\mathrm{c}} \tag{4}$$

其中: T_i^{b} 为节点 n_i 开始处理所分配任务需要的时间,若节点当前空闲,则为零,即表示无需等待. 计算时间为

$$T_i^{\mathrm{c}} = \frac{w_i}{f_i} \tag{5}$$

任务 t_i 在本地执行时,需要的能量为

$$E_i^{\mathrm{l}} = E_i^{\mathrm{c}} = T_i^{\mathrm{c}} P_i^{\mathrm{c}} \tag{6}$$

相应地,任务 t_i 在节点 n_j 执行时所需的时间为

$$T_{ij}^{\mathrm{o}} = T_{ij}^{\mathrm{s}} + T_{ij}^{\mathrm{c}} + T_j^{\mathrm{b}} \tag{7}$$

其中: T_{ij}^{s} 为传输数据需要的时间,即

$$T_{ij}^{\mathrm{s}} = \frac{l_i + o_i}{d_{ij}} \tag{8}$$

T_{ij}^{c} 为计算需要的时间,即

$$T_{ij}^{\mathrm{c}} = \frac{w_i}{f_j} \tag{9}$$

同理,任务 t_i 在节点 n_j 执行时所需的能量包括传输能量和计算能量,即

$$E_{ij}^{\mathrm{o}} = E_{ij}^{\mathrm{s}} + E_{ij}^{\mathrm{c}} \tag{10}$$

$$E_{ij}^{\mathrm{s}} = \frac{l_i}{d_{ij}}(P_i^{\mathrm{t}} + P_j^{\mathrm{r}}) + \frac{o_i}{d_{ij}}(P_j^{\mathrm{t}} + P_i^{\mathrm{r}}) \tag{11}$$

$$E_{ij}^{\mathrm{c}} = T_{ij}^{\mathrm{c}} P_j^{\mathrm{c}} \tag{12}$$

在每一轮任务分配过程中,优化的目标为

$$\begin{aligned} \min_{u} \sum_{i=1}^{N} \Big\{ \alpha \Big(u_{ii} E_i^{\mathrm{l}} + (1-u_{ii}) \sum_{j \neq i} E_{ij}^{\mathrm{o}} \Big) + \\ (1-\alpha) \Big(u_{ii} \boldsymbol{T}_i^{\mathrm{l}} + (1-u_{ii}) \sum_{j \neq i} \boldsymbol{T}_{ij}^{\mathrm{o}} \Big) \Big\} \end{aligned} \tag{13}$$

$$\text{s. t. 1: } u_{ij} = 0, \forall e_{ij} \notin E$$

$$\text{s. t. 2: } \sum_{j=1}^{m} u_{ij} = 1$$

$$\text{s. t. 3: } \sum_{i=1}^{m} u_{ij} \leqslant 1$$

$$\text{s. t. 4: } \sum_{j=1}^{m} \Big(u_{ii} T_i^{\mathrm{l}} + (1-u_{ii}) \sum_{j \neq i} T_{ij}^{\mathrm{o}} \Big) \leqslant s_i$$

$$\text{s. t. 5: } \sum_{j}^{m} \Big(u_{ii} E_i^{\mathrm{l}} + (1-u_{ii}) \sum_{j \neq i} E_{ij}^{\mathrm{o}} \Big) \leqslant E_i^{\mathrm{r}}$$

$$\text{s. t. 6: } u_{ij} \in \{0,1\}$$

其中: α 为加权因子,取值范围为 [0 - 1]. α 取 0 时,优化目标为时延; α 取 1 时,优化目标为能耗. 式(13) 表示所有节点优先级最高的任务在本次任务分配过程中消耗的时间或能量最小; u_{ij} 为任务分配矩阵的元素, u_{ij} 为 1 表示任务 t_i 分配给节点 n_j 执行; 否则为 0. 约束 1 表示对节点 n_i 而言,它只能将任务分配给与节点 n_i 具有直接连接的节点中. 约束 2 表示每个任务只能被分配 1 次. 约束 3 表示在一轮任务分配过程中每个节点最多只能被分配 1 个任务. 约束 4 表示任务的执行时间满足任务最晚完成时间约束. 约束 5 表示任务 t_i 在节点 n_j 执行时,消耗的能量要小于节点 n_j 的剩余能量.

1.2.3　基于 KM 算法的任务分配

KM 算法是一种在多项式时间内查找二分图中最大权值匹配问题的算法[14]. 二分图中包括 2 个互不相交的顶点集 T 集和 N 集,并且图中每条可选边连接的 2 个顶点,一个在 T 中,另一个在 N 中. 式(13) 的问题需要将节点中优先级最高的任务在各节点之间进行分配,使得任务执行时间或能耗最小,同时满足上述约束条件. 因此,如图 4 所示,将

各个节点中优先级最高的任务作为 T 集(任务集),t_2 表示节点 n_2 的优先级最高的任务,所有的节点作为 N 集(节点集),任务在不同节点执行的时间或能量即式(13)作为权值,进行最大权值匹配. 其中 T 集和 N 集之间的可选边表示任务所在节点与待分配节点之间具有通信连接,即约束1. 特别地,t_2 与 n_2 的连接表示节点 n_2 的任务在本地执行. 对于约束4和5,即能量约束和任务最晚完成时间约束,需要对可选边进行删减. 以图4中节点 n_2 的任务为例,中心节点计算 n_2 的任务在 n_1,n_2,n_3,n_4 上执行时消耗的能量和时间,并与节点的剩余能量和任务最晚完成时间比较,若不满足约束4、5,则去除相应的可选边.

由于在任务分配过程中,节点仅会将任务分配给与自己具有直接联系的节点,所以若一个节点没有待分配的任务,其相邻节点也没有待分配的任务,则认为该节点为冗余节点,在创建节点集 N 时应去除. 图3中,节点 n_6 没有待分配的任务,其相邻节点也没有待分配的任务,因此 n_6 是冗余节点,应该去除. 完整的基于KM算法的任务分配模型如图4所示.

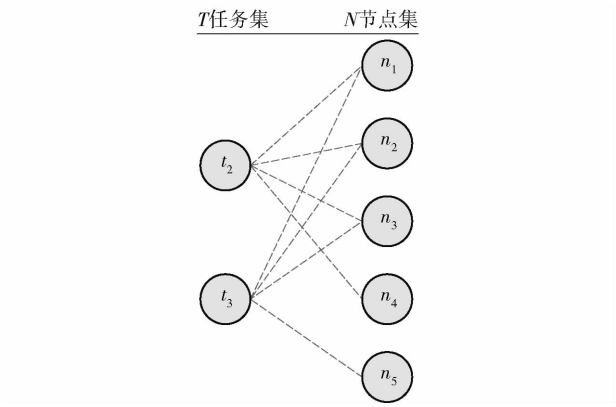

图4 KM算法的任务分配模型

将基于KM算法的任务分配方法总结如下.

输入: 待分配任务的属性 $t_i=\{l_i,o_i,w_i,s_i\}$,网络图 $G=\{N,E\}$,各个节点的属性$\{f_i,P_i,i,P_i,T_i\}$

1 根据图 G 和待分配的任务确定KM任务分配模型
2 for $t_i \in T$
3 for $n_j \in N$
4 if $e_{ij} \notin E$
5 令 $u_{ij}=0$
6 else
7 令 $u_{ij}=1$
8 end for
9 for $j=1:N$
10 if $u_{ij} \neq 0$
11 计算任务 t_i 在 n_j 的执行时间 T_{ij}
12 if $T_{ij} > s_i$
13 令 $u_{ij}=0$
14 计算任务 t_i 在 n_j 的执行能耗 E_{ij}
15 if $E_{ij} > E_j^r$
16 令 $u_{ij}=0$
17 end for
18 end for
19 运行标准KM算法

输出: 任务分配矩阵 $\boldsymbol{U}=\{u_{ij}\}$

1.2.4 时间复杂度分析

如上所述,在运行标准的KM算法之前,需要做一些预处理,步骤2~步骤18遍历每一个节点来计算当前节点的任务在其余节点处执行的时间和能耗,这部分的时间复杂度为 $O(n^2)$. KM算法的执行步骤包括: ① 利用预处理的结果初始化节点的顶标; ② 用匈牙利算法为每个节点寻找增广路径; ③ 若未找到,则修改可行顶标的值; ④ 重复②③直到找到二分图的最佳匹配为止. 这部分的时间复杂度为 $O(n^3)$,所以整体的时间复杂度为 $O(n^3)$.

2 仿真结果与分析

通过Java程序进行仿真,分别评估在时间最小化和能耗最小化时基于KM算法的任务分配方法的性能,并与贪心算法和任务在本地执行时进行比较. 表1所示为仿真参数的设置,其中,令任务执行所需的计算周期与其输入的数据大小成正比,每次仿真中,节点生成的任务数为0~4个,节点产生任务的概率为0.5.

表1 仿真参数

参数	值
发送功率/mW	100~200
接收功率/mW	100~200
计算功率/mW	1 000~1 500
节点之间传输速率/(Mbit·s^{-1})	2~3
计算速度/GHz	1.2~1.5
任务的计算负载(cycle·byte^{-1})	3 000
任务输入数据的大小/KB	500~1 000
任务输出数据的大小/KB	50~100
最晚完成时间/ms	250~500
每个节点生成的任务数/个	0~4

首先考虑能耗最小化时算法的性能. 图 5 和图 6 所示分别为任务的总能耗和平均能耗与节点数的关系. 从图 5 可见,随着节点数量的增加,所有节点生成的任务总数增多,因此任务消耗的总能量增加. 3 种方法中,所有任务在本地执行时消耗的能量最多,使用 KM 算法在节点之间进行任务分配消耗的能量最小,采用贪心算法的性能处于两者之间. 当节点数量比较少时,3 种方法进行任务分配的总能耗与任务的平均能耗接近,随着节点数量的增多,3 种方法分配任务消耗的总能量和平均能量的差距开始增加. 这是由于分布式节点在计算速度、功耗等方面是异构的,而所有任务在本地执行时,任务消耗的能量完全取决于任务所在节点的性能,无法利用周围的节点的优势,而这些周围节点可能具有更快的计算速度和更低的功耗. 对于贪心算法,每次分配任务时,总是选择当前开销最小节点作为目标节点,不从整体上加以考虑,不一定能获得最优解. 而 KM 算法通过递归回溯,能得到二分图权值的最佳匹配. 在任务生成率为 0.5,且节点数较少时,贪心算法分配当前任务对后续分配产生影响的可能性较小; 节点数增多时,分布式节点中总任务数增加,无法得到最佳匹配的任务数将增加,因此贪心算法与 KM 算法之间的能耗差距增加. 从图 6 可见,随着节点数的增加,任务的平均能耗下降,之后趋于平稳. 这主要是由于节点数增多时,每个节点相邻的节点数增加,所以运用 KM 算法和贪心算法分配任务时,可供选择的节点增加,单个任务的平均能耗降低. 当节点数继续增加时,每个节点的相邻节点逐渐平稳,且节点之间整体的异构性减小,所以平均能耗趋于平稳.

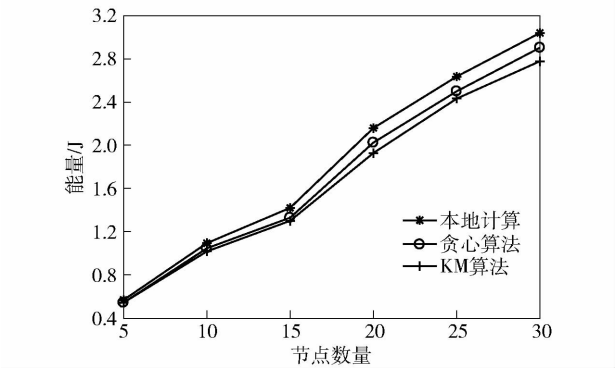

图 5　任务的总能耗与节点数的关系

图 7 所示为以时间最小化为目标时,任务的平均执行时间、节点数与任务生成率之间的关系. 当任务都在本地执行时,每个节点处有多个排队的任务,使得任务的等待时间大大增加. 对于贪心算法和 KM 算法,同上述分析,通过利用周围的空闲节点,能够减少任务的等待时间. 此外,任务生成率较低时,空闲节点数会增加,因此,任务的平均执行时间会进一步下降.

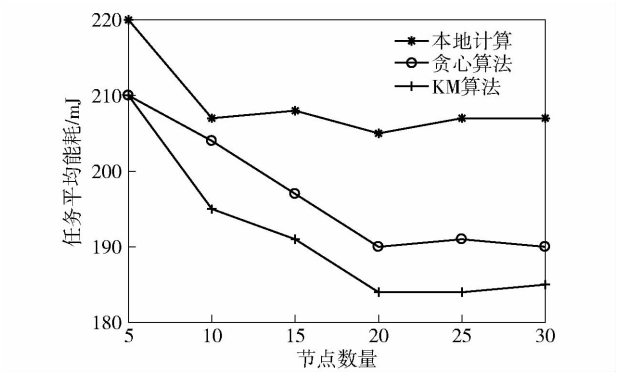

图 6　单个任务的平均能耗与节点数量之间的关系

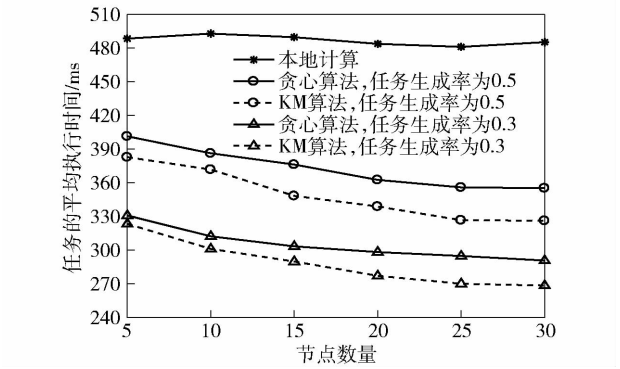

图 7　任务的平均执行时间、节点数量与任务生成率之间的关系

为了显示采用层次分析法进行任务优先级排序对任务完成的影响,在任务最晚完成时间为 250 ~ 500 ms,任务生成率为 0.5 的情况下,比较了在本地执行节点的多个任务,采用层次分析法进行优先级排序和任务按随机顺序执行时任务的失败率. 任务失败率指无法在任务最晚完成时间之前完成的任务数占总任务数的比例. 如图 8 所示,所有任务在本地执行时,每个任务都需要大量时间用于排队等待,导致任务失败率较高. 采用随机顺序执行时,能够利用周围空闲节点执行任务, 因此任务失败率有所下降,但是任务按随机执行,并没有考虑任务最晚完成时间这一属性,因此对一些紧急任务,无法在第一轮任务分配时被处理,可能导致任务无法在截止时间之前完成. 采用层次分析法通过考虑任务的紧急

程度来对任务进行优先级排序,进一步降低了任务失败率.

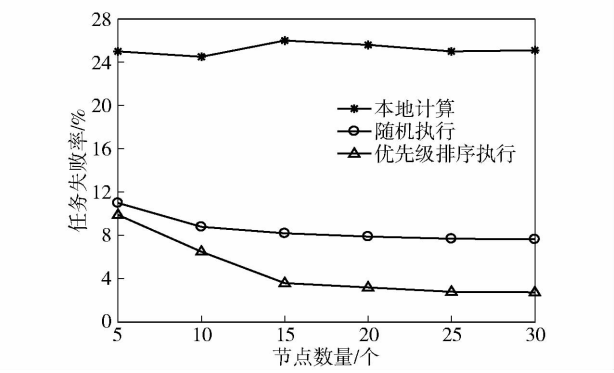

图8 任务失败率与节点数量的关系

3 结束语

综合考虑任务的时延和能耗,提出了一种基于KM算法的分布式无线网络中的任务分配方法,并将其性能与任务在本地执行和贪心算法进行了比较. 仿真结果表明,通过分布式节点之间的协作来共同执行节点产生的任务,能够有效地减少处理任务的时延或能耗. 下一步考虑在多跳的协作网络中分配任务,同时考虑节点参与任务计算处理的激励和意愿,并将其应用到实际场景中,以测试它们的实际性能.

参考文献:

[1] Satyanarayanan M, Chen Zhuo, Ha K, et al. Cloudlets: at the leading edge of mobile-cloud convergence [C]// International Conference on Mobile Computing. Austin: [s. n.], 2014: 1-9.

[2] 董思岐, 李海龙, 屈毓锛, 等. 移动边缘计算中的计算卸载策略研究综述 [J]. 计算机科学, 2019, 46(11): 32-40.
Dong Siqi, Li Hailong, Qu Yuben, et al. Survey of research on computation unloading strategy in mobile edge computing [J]. Computer Science, 2019, 46(11): 32-40.

[3] Swan M. Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0 [J]. Journal of Sensor & Actuator Networks, 2012, 1(3): 217-253.

[4] Funai C, Tapparello C, Heinzelman W. Mobile to mobile computational offloading in multi-hop cooperative networks [C]// 2016 IEEE Global Communications Conference. Washington, DC: IEEE, 2016: 1-7.

[5] Funai C, Tapparello C, Heinzelman W. Computational offloading for energy constrained devices in multi-hop cooperativse networks [J]. IEEE Transactions on Mobile Computing, 2020, 18(1): 60-73.

[6] Chen Xu, Pu Lingjun, Gao Lin, et al. Exploiting massive D2D collaboration for energy-efficient mobile edge computing [J]. IEEE Wireless Communications, 2017, 24(4): 64-71.

[7] Fan Wenhao, Tang Bihua, Liu Yuanan. Application multi-partitioning for offloading computation to multiple computing resources around mobile terminals [J]. International Journal of Grid and Distributed Computing, 2016, 9(6): 83-92.

[8] Wang Feng, Han Guangjie, Jiang Jinfang, et al. A task allocation algorithm based on score incentive mechanism for wireless sensor networks [J]. International Journal of Distributed Sensor Networks, 2015(5): 5-17.

[9] Yin Xiang, Zhang Kaiquan, Li Bin, et al. A task allocation strategy for complex applications in heterogeneous cluster-based wireless sensor networks [J]. International Journal of Distributed Sensor Networks, 2018, 14(8): 1-15.

[10] Pu Lingjun, Chen Xu, Mao Guoqiang, et al. An energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications [J]. IEEE Internet of Things Journal, 2019, 6(1): 84-99.

[11] Sahni Y, Cao Jiannong, Zhang Shigeng, et al. Edge Mesh: a new paradigm to enable distributed intelligence in internet of things [J]. IEEE Access, 2017, 5: 16441-16458.

[12] Yang Jun, Zhang Hesheng, Ling Yun, et al. Task allocation for wireless sensor network using modified binary particle swarm optimization [J]. IEEE Sensors Journal, 2014, 14(3): 882-892.

[13] Li Junjie. Analyzing key factors in taiwanese teachers teaching in China's mainland with analytic hierarchy process [J]. Journal of Interdisciplinary Mathematics, 2018, 21(2):307-316.

[14] Bruff D. The assignment problem and the Hungarian method [J]. Notes for Math, 2005, 20(5): 29-47.