DOI: 10.13382/j.jemi.B2205589

面向边缘计算的人工鱼群搜索任务调度*

杨文杰 巨 涛 杨 阳 火久元
(兰州交通大学电子信息与工程学院 兰州 730070)

摘 要:如何将计算任务分配到合适的边缘计算资源上进行计算,以满足边缘计算环境下用户的计算需求、提高用户任务请求的服务质量,是边缘计算中面临的关键问题。本文提出一种基于人工鱼群搜索的边缘计算任务调度方法(AFETSA)。将人工鱼群搜索算法和边缘计算任务调度模型相结合,采用非线性递减函数动态地调整人工鱼的视野范围和步长,以提高启发式任务调度算法的全局搜索能力,降低任务的计算时延;同时与禁忌搜索算法进行融合,通过引入忌禁表,在每一次迭代中防止算法陷入局部最优,提高算法的寻优能力。CloudSim3.0 仿真平台实验评测结表明,本文所提 AFETSA 方法和已有的 AFSA、ACO 和 PSO 这 3 种调度算法相比,在任务执行时间、算法稳定性、负载均衡方面都有明显的提升,可充分利用边缘服务器计算资源,提升计算任务的计算性能,有效解决边缘计算中任务调度不均导致的时延过高和负载不均衡问题。

关键词:边缘计算;任务调度;人工鱼群算法;高斯分布函数;禁忌搜索算法
中图分类号:TP391 **文献标识码**:A **国家标准学科分类代码**:520.3099

Artificial fish swarm search task scheduling for edge computing

Yang Wenjie Ju Tao Yang Yang Huo Jiuyuan
(School of Electronics and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract: Allocating computing tasks to appropriate edge computing resources to meet the computing needs of users and improve the quality of service for user task requests is a key problem in edge computing. This paper proposes an edge computing task scheduling method (AFETSA) based on artificial fish swarm search. For improving the global search ability of the heuristic task scheduling algorithm and reducing the computation time delay, the artificial fish search algorithm was combined with the edge computing task scheduling model, and the field of view and step size of the artificial fish were dynamically adjusted by the nonlinear decreasing function. At the same time, for improving the optimization ability of the algorithm, the tabu search algorithm is fused, and the tabu list is introduced to prevent the algorithm from falling into local optimal in each iteration. The experimental evaluation results on the CloudSim3.0 simulation platform, show that compared with the existing task scheduling algorithms AFSA, ACO and PSO, the proposed task scheduling method in this paper has significant improvement in task execution time, algorithm stability and load balance. It can make full use of the computing resources of edge servers to improve the computing performance of computing tasks, and effectively solve the problem of high delay and load imbalance caused by uneven task scheduling in edge computing.

Keywords: edge computing; task scheduling; artificial fish algorithm; Gaussian distribution function; taboo search algorithm

0 引 言

随着万物互联时代的到来,移动智能设备和 5G 等无线通信技术得到了普及和发展,涌现出许多具有低延迟要求的计算密集型应用,如:智慧城市、增强现实、无人驾驶、智慧工业、环境监测等[1]。而云计算已经不能很好地满足时延敏感性应用的计算要求,因此边缘计算应运而

收稿日期:2022-06-16 Received Date:2022-06-16
*基金项目:国家自然科学基金(61862037, 62262038)、兰州交通大学天佑创新团队项目(TY202002)、兰州市人才创新创业项目(2021-RC-40)资助

生。边缘计算可弥补传统云计算的不足,为网络边缘的数据提供更直接的安全保障[2]。边缘计算模型将大部分的数据放在边缘设备上进行处理,不仅可节省数据传输带宽,减少传输时延,减小设备端的能耗问题,同时可以保护用户的数据隐私,防止这些敏感数据被泄露[3]。在边缘计算环境中,更多的服务请求从云计算中心移动到边缘设备端去处理,在进行处理时,边缘端能够缩短其响应时间,提高可靠性,因此,随着万物互联的发展,边缘计算将成为新兴万物互联应用场景下的重要计算模式[4]。

随着边缘计算的不断发展,边缘计算的任务调度问题也变得尤为重要。由于在边缘计算环境中,任务的状态、种类、大小随时变化,边缘计算资源的异构性、位置的分散性,以及不同应用对性能、稳定性、费用等需求的多样性,大大增加了任务调度的难度。如何利用合理的方式进行任务调度,实现资源的合理分配,以提高资源的利用率、降低时延以及能耗,是边缘计算任务调度必须要解决的重要问题。

1　相关工作

在边缘计算中,为了满足终端用户的任务请求,减少任务的完成时延以及能耗,将用户请求转化为任务调度问题。边缘计算环境下的任务调度是以任务为基本单位,通过最优的调度算法将任务合理地分配给多个边缘服务器,以充分利用边缘服务器的资源,在满足终端设备的任务请求的同时,尽可能地减小任务的完成时间,降低整个系统设备的能耗[5]。合适的任务调度方法直接影响任务请求的完成时间、整个边缘系统的计算性能以及用户的服务质量。

国内外学者针对边缘计算的任务调度问题进行相关的研究工作,提出了大量的边缘计算任务调度算法并取得了不错的进展[6-7]。文献[8]提出一个基于基尼系数的贪婪启发式(GCGH)算法,通过智能移动设备分类和基尼系数的计算为智能移动设备分配无线电资源和计算资源。文献[9]使用改进的蚁群算法作为边缘计算任务调度的策略,将负载均衡作为目标。通过改进启发式因子实现待分配任务和可选择的空闲计算资源的匹配,让边缘环境下的计算资源得到充分利用。文献[10]使用遗传算法作为任务调度策略,满足具有不同关键级别的任务对计算时延的需求和解决用户设备计算资源不足的问题,有效减少任务执行的时延和降低终端设备的能耗,同时提高任务执行的成功率。文献[11]使用萤火虫算法将任务分配给最合适的计算节点资源,实现均匀分配负载,减少总体完成时间。文献[12]使用人工蜂群算法将边缘计算的任务调度转化为约束条件下的寻优问题,该算法具有较快的收敛速度,能够实现任务弹性卸载,有效地降低了任务处理时延。文献[13]提出一种基于粒子群优化算法的任务调度算法,该算法针对处理规模较大的任务优化分配,通过将任务进行动态分组后进行任务调度,减少了任务的完成时间,同时平衡了任务负载。文献[14]提出一种基于混合蜘蛛猴优化的任务调度算法,通过调度种子引导混合蜘蛛猴算法,从而达到提高算法的性能,在满足预算和期限约束的同时,优化了任务调度的完成时间以及成本。文献[15]提出一种基于量子启发的二进制混沌群算法来进行多处理器系统中的任务调度,在任务调度过程中保持了负载均衡,提高了计算系统的吞吐量、可扩展性、可靠性、响应时间,提高了整个系统的资源利用率。文献[16]提出了一种基于任务优先级的改进 Min-Min 调度算法。该算法将任务调度时的最小时延和服务器的负载均衡转化为约束条件下的递归性优化问题。算法中任务优先级根据任务处理代价及任务数据量大小计算,然后结合任务截止时间、服务器调度次数,共同决定为不同优先级的用户分配合适的边缘计算资源。该算法能够在尽量满足任务计算时间和任务截止时间的前提下,最大化各个边缘服务器的利用率。文献[17]提出一种异构最早完成时间(HEFT)算法,该算法充分考虑了任务的计算时间和通信传输时间、子任务对父任务影响,在进行分配边缘计算资源时,将每个任务都分配到能使其最早完成的边缘计算资源上,对各个任务的处理时间需求有较好的考虑。文献[18]提出的人工鱼群算法,对初值的要求不高,能够自适应的调整搜索空间,鲁棒性较强,简单易实现,容易与其他方法结合,适用于求解组合优化问题。但人工鱼群算法会存在前期收敛速度快、后期易陷入局部最优、求解精度不高等问题。

上述已有工作中的启发式搜索算法,由于没有对探索空间进行自适应的动态调整,算法存在全局搜索能力低、容易陷入局部最优、过早收敛等问题,在进行边缘计算环境下的任务调度时,会导致任务计算时延较长、任务计算负载不均衡、计算资源利用率不高等问题。针对以上问题,本文提出了一种基于人工鱼群搜索的任务调度算法(AFETSA),进行边缘计算的任务调度,以进一步提高启发式任务调度算法的全局搜索能力,降低任务的计算时延,提高边缘计算资源利用率,提升用户任务请求的服务质量。

2　任务调度模型

2.1　模型设计

边缘计算环境下的任务调度,就是根据不同用户的计算任务特点,和不同的边缘计算资源特点,将不同的计算任务分配到合适的边缘计算资源上进行计算,实现计算任务和边缘计算资源合理的匹配,从而满足边缘计算

环境下不同用户的计算需求,充分利用边缘计算资源,提升边缘系统的计算性能。本质上它是一个组合优化问题[19],该问题可以抽象为:若干终端设备随机生成 n 个独立任务,采取合理的调度优化方法将任务分配给 m 个边缘服务器,使得边缘服务器的资源能够充分利用,让任务完成的时间达到最小。本文任务调度模型主要包括终端设备和边缘服务器端,终端设备将任务通过无线链路传输到边缘服务器中,到达的任务和所有可用的边缘服务器的资源暂存于位于某一边缘服务器任务调度队列和边缘服务器调度队列中,随后通过合适的任务调度策略将任务分配给对应的边缘服务器资源进行计算。具体任务调度模型如图 1 所示。

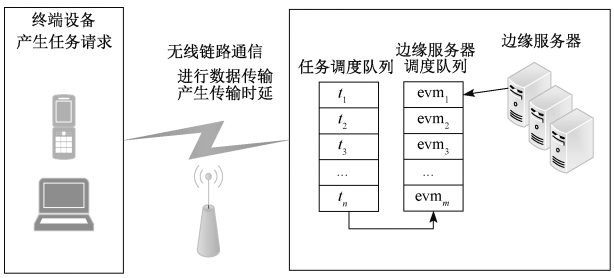


图 1　边缘计算任务调度模型

Fig. 1　Edge computing task scheduling model

模型中任务表示为 $T=\{t_1,t_2,\cdots,t_i,\cdots,t_n\}$,其中 t_i 表示第 i 个任务;边缘服务器表示为:$\mathrm{EVM}=\{\mathrm{evm}_1,\mathrm{evm}_2,\mathrm{evm}_3,\cdots,\mathrm{evm}_j,\cdots,\mathrm{evm}_m\}$,其中 evm_j 表示第 j 个边缘服务器。其中每个边缘服务器的计算能力不同,同时计算任务也呈现出不同的状态。边缘计算任务调度属于离散空间的组合优化问题,任务和边缘服务器映射关系如图 2 所示。

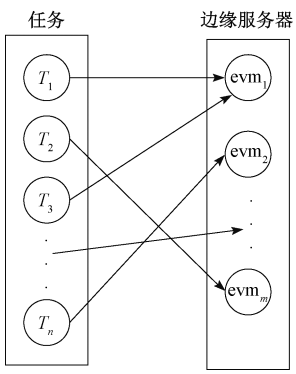


图 2　映射关系图

Fig. 2　Mapping diagram

将这种映射关系用矩阵表示为:

$$\boldsymbol{S}=\begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1m} \\ S_{21} & S_{22} & \cdots & S_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ S_{n1} & S_{n2} & \cdots & S_{nm} \end{bmatrix} \tag{1}$$

其中,S_{ij} 表示第 i 个任务是否分配给第 j 个边缘服务器。当 $S_{ij}=0$ 时,任务 i 没有分配给第 j 个边缘服务器;当 $S_{ij}=1$ 时,任务 i 分配给第 j 个边缘服务器。

2.2　模型求解

结合整个边缘计算任务调度执行过程的特点,对以上任务调度模型求解过程如下。

1)模型求解过程中的约束条件

(1)任务之间是相互独立且每个任务只能分配到一个服务器上,即:

$$\sum_{j=1}^{n} S_{ij}=1 \tag{2}$$

(2)调度到边缘服务器 evm_j 上的任务所需带宽之和不能大于边缘服务器 evm_j 的带宽,即:

$$\sum_{i=1}^{n} S_{ij}b_{ij} \leqslant B_j \tag{3}$$

其中,b_{ij} 为任务 i 分配到边缘服务器 evm_j 执行时所需的带宽,B_j 为 evm_j 的通信带宽。

(3)为了确保每个边缘服务器有足够的存储空间存储所分配到的任务的数据,计算任务所需存储空间和边缘服务器可用的存储空间之间满足如下约束关系:

$$\sum_{i=1}^{n} S_{ij}m_i \leqslant M_j \tag{4}$$

其中,m_i 为第 i 个任务所需的内存大小,M_j 为边缘服务器 evm_j 的可用内存空间。

2)目标函数

在边缘计算任务调度中,任务的完成时间由任务执行时间 ETc_{ij} 和传输时间 ETtr_{ij} 两部分构成。任务执行时间与边缘服务器的计算能力有关,任务 t_i 在边缘服务器 evm_j 处的执行时间可以表示为:

$$\mathrm{ETc}_{ij}=\frac{t_i}{E_j} \tag{5}$$

其中,t_i 表示任务的大小,E_j 表示边缘服务器 j 的计算能力。

任务的传输时间由任务的大小和传输速率决定,任务 T_i 传输到边缘服务器 evm_j 的时间可以表示为:

$$\mathrm{ETtr}_{ij}=\frac{t_i}{r_{ij}} \tag{6}$$

其中,r_{ij} 表示任务到边缘服务器的传输速率,具体计算公式如下:

$$r_{ij}=b_{ij}\times\log\left(1+\frac{h\times p}{\tau}\right) \tag{7}$$

其中,b_{ij} 为通信带宽,τ 为噪声功率,h 为信道功率增

益，p 为传输功率。

因此，任务在边缘服务器总的完成时间为：

$$\mathrm{ET}_{ij} = S_{ij}(\mathrm{ETc}_{ij} + \mathrm{ETtr}_{ij}) \tag{8}$$

在边缘计算中，任务调度优化的目标主要是尽量降低任务的执行时延和边缘设备的能源消耗。为了方便验证，本文在假设各边缘服务器任务传输时间固定的前提下，只用任务的完成时间来度量任务调度的执行效果，任务调度主要目标是任务在各个边缘服务器上的完成时间达到最小，达到最高的执行效率，因此，本文目标函数定义为：

$$F = \min\left[\max_{1\leqslant j\leqslant m}\sum_{i=1}^{n}(S_{ij}\times \mathrm{ET}_{ij})\right] \tag{9}$$

3 任务调度算法设计

3.1 人工鱼群算法分析

人工鱼群算法(artificial fish swarm algorithm, AFSA)是一种智能优化算法，该算法通过鱼类寻觅食物的行为去寻找算法的最优解，根据在水域中食物较多的地方鱼类比较密集这一特点，构造所需要的人工鱼，模仿鱼群的觅食、聚群、追尾等行为，通过不断迭代，求解算法的最优解。在算法中每条人工鱼对应一个最优解，优化问题的解空间对应鱼类生存的水域，最终利用鱼群个体之间的各种行为来实现全局寻优[18]。

人工鱼群行为算法操作主要包括觅食行为、聚群行为、追尾行为、随机行为。各种行为具体操作描述如下。

1)觅食行为

每个个体鱼在其视野范围内不断搜寻食物，若寻找到了则向食物所在方向游去，若没寻找到，则随机游动。设人工鱼的当前状态为 X_i，在它的视野范围内随机选择一个状态 X_j，如式(10)所示。如果对应状态的人工鱼的食物浓度(适应度函数值) $Y_i<Y_j$，则向该状态移动一步，如式(11)所示；否则，再次随机选择状态 X_j，判断是否可以向前移动；当尝试次数达到最大尝试次数后，仍不满足移动条件，则人工鱼随机移动一步，如式(12)所示。

$$X_j = X_i + \mathrm{rand}() \times \mathrm{Visual} \tag{10}$$

$$X_{i1\,\mathrm{next}} = X_i + \frac{X_j - X_i}{\| X_j - X_i \|} \times \mathrm{Step} \times \mathrm{Rand}() \tag{11}$$

$$X_{i1\,\mathrm{next}} = X_i + \mathrm{rand}() \times \mathrm{Step} \tag{12}$$

2)聚群行为

聚群行为能够让鱼群进行分散式搜索，找到食物后向着其方向游去，同时可以躲避和抵御外敌。设人工鱼的当前状态为 X_i，人工鱼视野范围的中心位置为 X_c，探索其视野范围内的人工鱼的个数为 n_i，n 表示人工鱼总数。假如符合条件 $(Y_c/n_i)>\delta Y_i$，则向 X_c 的方向移动一步，说明在 X_c 的位置上不太拥挤并且具有较高的食物浓度，其中 δ 表示拥挤度因子，如式(13)所示，否则，执行觅食行为。

$$X_{i1\,\mathrm{next}} = X_i + \frac{X_c - X_i}{\| X_c - X_i \|} \times \mathrm{Step} \times \mathrm{Rand}() \tag{13}$$

3)追尾行为

在当前范围内，如果发现拥有更多食物的位置后，人工鱼会向食物更多的地方游去，设人工鱼的当前状态为 X_i，探索其视野范围内的人工鱼的个数为 n_f，其邻域内状态最优的邻居为 X_{best}，如果满足 $(y_{\mathrm{best}}/n_f)>\delta Y_i$，则向 X_{best} 的方向移动一步，说明 X_c 的位置不太拥挤并且食物浓度要比其他地方高，如式(14)所示，否则执行觅食行为。

$$X_{i1\,\mathrm{next}} = X_i + \frac{X_{\mathrm{best}} - X_i}{\| X_{\mathrm{best}} - X_i \|} \times \mathrm{Step} \times \mathrm{Rand}() \tag{14}$$

4)随机行为

当人工鱼没有发生其他行为时，人工鱼在一定范围之内可以随意游动，游动过程中随机地在所在视野范围内选择一个状态，然后向所选的方向进行移动，它通常作为觅食行为的一个缺省行为，如式(12)所示。

3.2 视野及步长的调整

基于人工鱼群搜索行为进行设计边缘计算任务调度算法时，控制参数会影响算法的优劣，从而影响边缘计算任务的执行速度。文献[20]提出人工鱼群算法中的视野和步长对算法的收敛速度和求解精度有着较大的影响，视野参数决定了人工鱼的搜索范围，在比较大的视野下，鱼群比较利于全局搜索，能够提升寻优速度，但很容易跳过最优解。在比较小的视野下，鱼群能够提高局部搜索的能力，但收敛较慢且容易陷入局部最优。步长参数决定了算法的优化精度，步长越大则前期收敛速度越快，同时也会出现震荡现象而大大影响求解精度；步长越小则收敛速度越慢，求解精度越高。为了解决以上问题，需要根据实际应用的特点对人工鱼群算法的视野范围和步长参数进行改进。

1)视野调整方法

在人工鱼群算法运行前期，需要较大的视野去进行全局搜索，提高收敛速度。搜索后期，为了提高算法的精度以及局部搜索能力，需要较小的视野。本文通过引入非线性递减函数来动态调整视野参数，采用高斯正态分布函数作为非线性递减函数，函数曲线如图 3 所示，由于视野不能下降太快，当 $\mu=0$、$\sigma^2=5$ 时函数曲线下降较为平缓且下降幅度较小，适合用来调整人工鱼视野。高斯正态分布函数的具体形式为：

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{15}$$

通过使用式(16)对视野进行调整：

$$\mathrm{Visual}_{t+1} = \mathrm{Visual}_t \times \theta \times f\left(4\times\frac{g}{G};0,5\right) \tag{16}$$

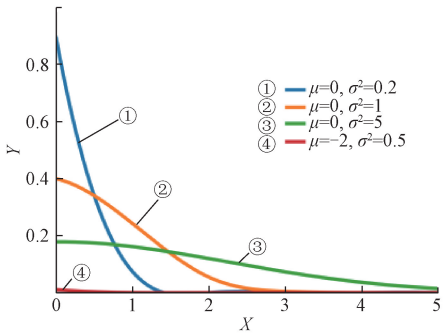


图 3　高斯正态分布函数

Fig. 3　The Gaussian normal distribution function

其中 Visual_{t+1} 为下一更新后的视野，Visual_t 为当前视野范围，由于当 $\mu=0$、$\sigma^2=5$ 时，下降比较平缓且较缓和，但是初始值较低，会导致初试视野下降迅速，因此引入幅度因子 θ，本文 θ 取 5.605，g 表示当前迭代次数，G 表示最大迭代次数。

2）步长改进方法

采用固定的步长，算法前期能够提升收敛速度，但在后期固定的步长会使得鱼群无法聚集到最优值。为了协调算法寻优速度与解精度的之间的平衡，对步长也进行了改进。在算法运行前期，拥有较大的步长让其搜索范围增大，在算法运行后期，随着算法的迭代步长逐渐减小，提高算法的局部搜索能力和寻优精度。本文使用式（17）随视野范围变化对步长动态调整。

$$\text{Step}_{t+1}=\text{Step}_t\times\theta\times f\left(4\times\frac{g}{G};0,5\right) \tag{17}$$

其中，Step_{t+1} 表示更新后的步长大小，Step_t 为当前步长大小，幅度因子 θ 取 5.605。

此外，拥挤度因子 δ 决定着觅食和随机行为是否突出，当 δ 越小，觅食和随机行为越突出，人工鱼避免陷入局部极值的能力也越强[20]。因此，为了防止算法陷入局部极值，采用指数式的衰减策略更新拥挤度因子：

$$\delta_{t+1}=\alpha\times\delta_t\quad\alpha\in(0,1) \tag{18}$$

其中，α 为衰减因子，衰减因子大小可以影响人工鱼的觅食能力，为了使人工鱼获得较好的觅食能力，根据文献[21]中的经验值，将 α 设置为 0.6。

3.3　禁忌搜索机制的融合

禁忌搜索算法是一种按照人类记忆功能寻找东西的启发式算法[22]，即当在寻找某个东西时，对已经搜寻过的地方短时间内不会进行再次寻找，而是去没有搜寻过的地方进行寻找，若仍然没有找到，才会去搜寻过的地方再次进行搜索。为了避免算法陷入局部最优解，禁忌搜索算法通过建立禁忌表，记录最近若干次迭代过程中求解到的最优值。在当前迭代过程中，记录在禁忌表中的最优值不再被继续搜索，从而避免算法重新访问已经访问过的解，使得算法摆脱局部最优，当禁忌对象满足一定禁忌长度之后，将被释放出来重新寻优，直到满足终止条件[23]。因为边缘计算的任务调度是离散的。随着任务数量和边缘服务器数量增加，边缘计算任务调度的计算量也会增加，使得基于原始人工鱼群搜索的边缘任务调度算法很容易陷入局部最优。为避免人工鱼在局部极值点或者附近来回震荡，防止人工鱼为已经分配的任务再次进行边缘服务器选择，将禁忌搜索算法与人工鱼群算法融合。通过在人工鱼群搜索中融入禁忌搜索机制，可以防止在寻优过程中对局部最优值重复搜索，人工鱼在每次移动之前都要检测目标位置是否在禁忌表中，提高了人工鱼群寻优的能力，提高任务调度的服务质量，实现对边缘设备合理的分配，减少任务完成时延。

具体禁忌搜索机制如下。

1）首先通过建立禁忌表，在禁忌表中放入人工鱼每次寻得的结果，将人工鱼每次寻优的结果作为禁忌搜索的初始值放入禁忌表中，并作为当前最优解 X_{best} 和当前解 X_{now}，将它们对应的目标函数值作为当前解的函数值 Y_{now} 和特设准则的解 best so far。

2）在禁忌区域内生成邻域解，将其中最优的作为候选解 $X_{\text{candidate}}$，对应的函数值为 $Y_{\text{candidate}}$，如果 $Y_{\text{candidate}}>Y_{\text{now}}$，就把 $Y_{\text{candidate}}$ 当做下一次迭代的当前解 X_{now}，更新禁忌表；

3）如果 $Y_{\text{candidate}}<Y_{\text{now}}$，并且 $Y_{\text{candidate}}<\text{best so far}$，将 $X_{\text{candidate}}$ 作为目前最优解 X_{best}，并将其加入禁忌表；

4）若 $Y_{\text{candidate}}<Y_{\text{now}}$，但 $Y_{\text{candidate}}>\text{best so far}$，判断 $X_{\text{candidate}}$ 是否在禁忌表中，若不在，则向目标值移动，并将其加入禁忌表，并更新禁忌表；若在，则根据人工鱼群的行为去重新选择新的人工鱼的位置。

3.4　基于人工鱼群搜索的任务调度算法

本文算法在人工鱼群算法的基础上改进了步长和视野范围，并且引入了禁忌搜索机制，对于跳出局部最优和全局搜索能力进行了优化，从而实现更合理的边缘任务调度。

对于 n 个任务、m 个边缘服务器进行任务调度时，采用直接编码方式[24]对人工鱼进行编码，人工鱼的位置状态编码如式（19）所示：

$$X_i=(X_{i1},X_{i2},X_{i3},\cdots,X_{in}) \tag{19}$$

其中，$i=1,2,3,\cdots$，FishNumber，X_i 表示第 i 条人工鱼的位置，FishNumber 为人工鱼总数，x_{ij} 为 1～m 的整数，表示第 j 个任务在第 x_{ij} 个边缘服务器上面执行。例如：有 10 个待执行的任务，有 5 个边缘服务器，需要将 10 个任务合理的分配到 5 个边缘服务器上去，如果某条人工鱼编码为（1,2,5,2,3,5,3,4,1,4），则表示第 1、9 个任务分配到第 1 个边缘服务器上执行，第 2、4 个任务分配到第 2 个边缘服务器上执行，第 5、7 个任务分配到第 3 个边缘服务器上执行，第 8、10 个任务分配到第 4 个边缘服

务器上执行,第 3、6 个任务分配到第 5 个边缘服务器上执行。在本文算法中人工鱼当前所在位置的食物浓度 Y 为当前分配任务的预计完成时间。

1)算法处理流程

算法的整个处理流程如图 4 所示。

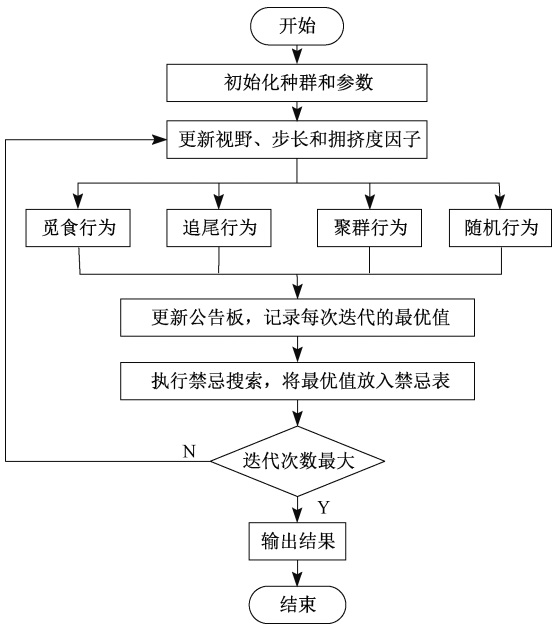


图 4 AFETSA 算法流程

Fig. 4 Flow chart of the AFETSA algorithm

2)算法实现

算法的具体实现过程如下:

(1)初始化参数:包括设置迭代次数,鱼群规模,人工鱼视野,步长,拥挤度因子,拥挤度因子衰减系数,最大尝试次数,设置任务长度和虚拟机的计算能力。

(2)对每条人工鱼进行随机解码操作,随后计算人工鱼当前状态下的任务执行时间(目标函数值),然后将最短的目标函数最优值记入公告板;

(3)人工鱼执行觅食、聚群、追尾、随机 4 种行为,对人工鱼的各个行为进行评价,选择最短的任务执行时间更新公告板,确定目标位置。

(4)将公告板的结果放入禁忌表,如果禁忌表中存在目标位置,但该位置不是当前发现的最优位置,返回 3),否则人工鱼向该位置移动;

(5)更新视野范围、步长及拥挤度因子,同时迭代次数加 1;视野更新如式(16)所示,步长更新如式(17)所示,拥挤度因子更新如式(18)所示,

(6)当前迭代次数小于算法的最大迭代次数时,返回步骤(3),如果算法达到最大迭代次数,停止运行并输出最后结果,即获得的最优值。输出寻得的最优调度方式及其对应任务执行时间。

算法伪代码实现如下:

```
算法 1:基于人工鱼群搜索的任务调度算法
输入:Gmax, FishNumber, visual step, δ, trynumber
      被调度的任务 T
      可用的边缘服务器 EVM
输出:最优值 Y
开始
初始化 FishNumber, visual, step, δ, trynumber
While (gen<Gmax)
  for i=1:FishNumber do
  Switch (bulletin(i))
    case (Yc/ni)>δ·Yi:
      执行聚群行为操作
      swarm(i):
        通过式(13)移动,并计算适应度值
    case (ybest/nf)>δ·Yi:
      执行追尾行为操作
      follow(i):
        通过式(14)移动,并计算适应度值
    default:
      执行觅食行为操作
      prey(i):
        if Yi<Yj
          通过式(11)移动,并计算适应度值
        else
          按式(12)执行随机搜索操作
  end Swtich
  获得当前最佳位置 Xij
  获得当前的最优适应度值 Ybest
  在公告板中设置禁忌搜索的初始值为最佳位置 best Xij
    if  best Xij  在禁忌表中;
      执行随机搜索操作;
    else
      朝最佳位置移动, i.e Xi=best Xij
      更新禁忌表
    end for
  更新最优的人工鱼;
  根据式(16)更新视野 visual;
  根据式(17)更新步长 step;
  根据式(18)更新拥挤度因子 δ;
  更新适应度值;
  更新迭代次数;
end while
获得最优的任务调度策略
获得最优值的适应度值
结束
```

4 实验验证

4.1 实验仿真环境及参数设置

为了验证本文提出的基于人工鱼群搜索的边缘计算

任务调度方法的正确性和可靠性，采用开源的 CloudSim[25] 仿真平台进行仿真对比实验，在边缘服务器数量以及任务数量改变的情况下，本文算法 AFETSA 分别和 AFSA、ACO 和 PSO 这 3 种调度策略从算法的执行时间、算法稳定性和负载均衡 3 个方面进行对比。

实验运行环境为 Windows10 64 位操作系统，Intel Core i5-5200U CPU，8 GB 内存。

在实验中人工鱼群算法的基本参数设置如表 1 所示。

表 1　人工鱼群算法的基本参数表

Table 1　The basic parameters of the artificial fish algorithm

参数	AFETSA
鱼群规模	10
人工鱼视野范围 visual	2
移动步长 step	2
拥挤度因子 δ	1
衰减因子 α	0.6
最大迭代次数 Gmax	20
最大尝试次数 try_number	5

由于在边缘计算中，各个边缘服务器的处理能力是不同的，为了验证方便，设置了 10 种不同计算能力的边缘服务器。本实验使用的边缘服务器的计算能力数据设置如图 5 所示。为了方便进行对比试验，除边缘服务器的计算能力设置不同外，边缘服务器其他性能参数都设置相同。在本实验中任务的大小为 500～10 000 的随机整数。

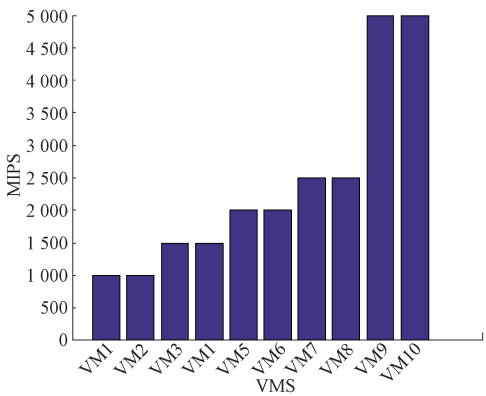


图 5　边缘服务器的计算能力

Fig. 5　Computing power of the edge server

4.2　算法计算性能对比

由于任务长度是随机生成的，因此为了更好的达到对比效果，实验取 10 次的平均值去进行对比。本文算法 AFETSA 分别和 AFSA、ACO 和 PSO 这 3 种调度算法从执行时间进行对比。

本方算法和其他算法的对应的调度方案的时间对比如图 6 所示。

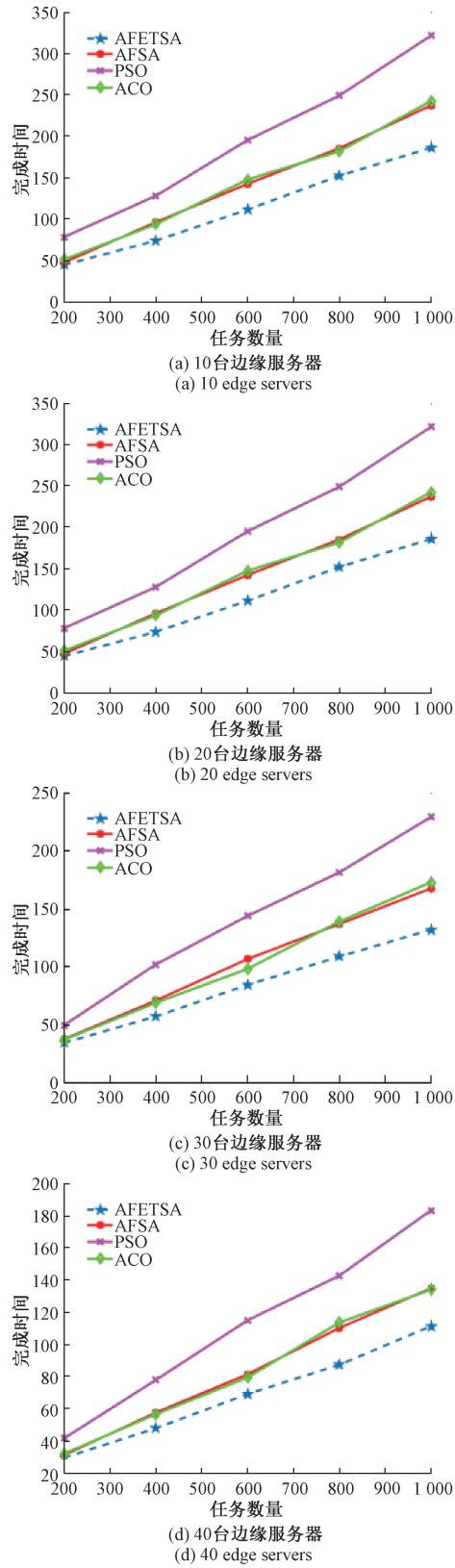


图 6　任务调度时间对比

Fig. 6　The task scheduling time comparison

在图6(a)中,边缘服务器为10台,当任务数量为200时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了27.1%、22.4%和53.6%;当任务数量为400时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了24.3%、20.7%和46.4%;当任务数量为600时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了29.1%、18.5%和47.6%;当任务数量为800时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了27%、21.6%和46%;当任务数量为1 000时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了23.5%、18.5%和40.5%。

在图6(b)中,边缘服务器为20台,当任务数量为200时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了5.7%、12.2%和42.7%;当任务数量为400时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了23.2%、21.2%和42.2%;当任务数量为600时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了21.7%、24.5%和42.8%;当任务数量为800时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了17.9%、16.2%和38.8%;当任务数量为1 000时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了21.5%、23.23%和42%。

在图6(c)中,边缘服务器为30台,当任务数量为200时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了8.5%、7.3%和30.6%;当任务数量为400时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了19.4%、17.1%和44.3%;当任务数量为600时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了20.9%、14%和40%;当任务数量为800时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了27%、21.6%和46%;当任务数量为1 000时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了21.3%、23.64%和42%。

在图6(d)中,边缘服务器为40台,当任务数量为200时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了5.2%、7.7%和28.9%;当任务数量为400时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了16.9%、15.4%和38.5%;当任务数量为600时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了15.1%、12.9%和39.8%;当任务数量为800时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了20.7%、23.1%和38.7%;当任务数量为1 000时,本文算法比AFSA、ACO和PSO算法在时间上分别降低了17.4%、16.9%和39.2%。

从上述图中可以看出,当采用固定数量的边缘服务器时,对于不同任务数进行任务调度时,AFETSA算法的完成时间均小于AFSA、ACO和PSO算法。因此AFETSA算法对于边缘计算任务调度具有更好的效果。由于上述对比图为了避免实验的偶然性,均取10次实验的平均值去进行对比实验,因此对于各个算法运行多次的数据结果也进行了对比,以比较各个算法的稳定性。

4.3　算法稳定性对比

在本实验中,由于任务的大小是随机生成的整数常数,因此各个算法在每次进行任务调度时的任务大小都是不同的,所以要求算法具有较好的稳定性,对于不同的初始化情况,算法具有较好的收敛性。为了更好地验证算法的稳定性,固定边缘服务器数量为10,任务数量为200。各个算法的运行次数为10次,算法的稳定性对比如图7所示,通过对比各算法多次运行结果,可以看出ACO算法对于边缘计算任务调度是的寻优能力不太好,算法的变化起伏比其他算法都要大。而AFETSA算法的变化较平缓,稳定性较好,具有较强的寻优能力。

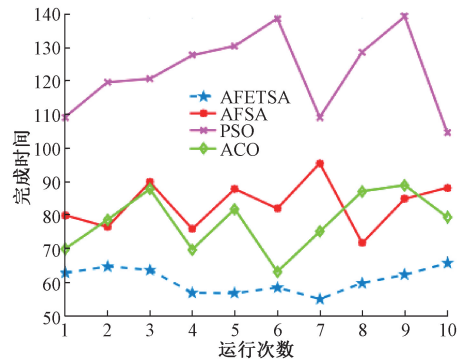


图7　任务数为200时算法稳定性对比

Fig. 7　Algorithms stability comparison when the number of tasks is 200

为了进一步对比算法稳定性,再一次增加任务数量为600,边缘服务器数量保持不变,迭代次数为20的结果如图8所示,可以看出AFETSA的算法稳定性更好。

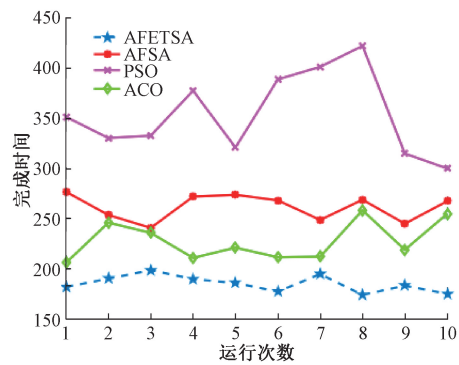


图8　任务数为600时算法稳定性对比

Fig. 8　Algorithms stability comparison when the number of tasks is 600

4.4　负载均衡对比

对于完成时间和算法稳定性的对比，都可以看出 AFETSA 算法要优于其他算法。接下来将负载均衡也作为评判指标进行对比。使用不平衡程度[26]去衡量虚拟机之间的负载均衡。具体不平衡程度由式(20)计算：

$$D_i = \frac{T_{\max} + T_{\min}}{T_{\text{avg}}} \tag{20}$$

其中，$T_{\max}$ 是边缘服务器的最大执行时间，$T_{\min}$ 是边缘服务器的最小执行时间，T_{avg} 是边缘服务器的平均执行时间，D_i 为算法的平均不平衡程度。每个算法的任务数之间的平均不平衡程度(D_i)如图 9 所示。

从上述图中可以看出，对于不同任务数和不同服务器数，AFETSA 算法的不平衡度均小于 AFSA 算法、ACO 算法和 PSO 算法，因此本文算法在边缘计算的任务调度中能实现更好的系统负载平衡。

5　结　论

本文针对边缘计算环境下的任务调度问题，在深入研究边缘计算任务调度特点和人工鱼群搜索算法特征的基础上，通过引入非线性递减函数动态调整人工鱼参数以及融合禁忌探索机制，设计了基于人工鱼群搜索的边缘计算任务调度方法。在 CloudSim3.0 仿真平台上的仿真实验结果分析表明，本文提出的边缘计算任务调度算法，在计算负载和边缘计算资源变化的情况下，任务的执行时间、负载均衡度、稳定性方面和已有的调度算法相比都有明显的提升。使用本文调度方法对边缘计算任务进行调度，可充分利用边缘服务器的计算资源，提高边缘计算任务的计算性能，解决任务调度不均导致的时延和负载不均衡问题。由于边缘计算任务调度优化时，不仅需要在任务调度时延方面进行提升，还需要考虑包括成本，能耗等因素，后续研究要进一步考虑以上成本、能耗、传输等因素情况下的多目标任务调度优化问题，全面提升边缘计算环境下的任务调度性能。

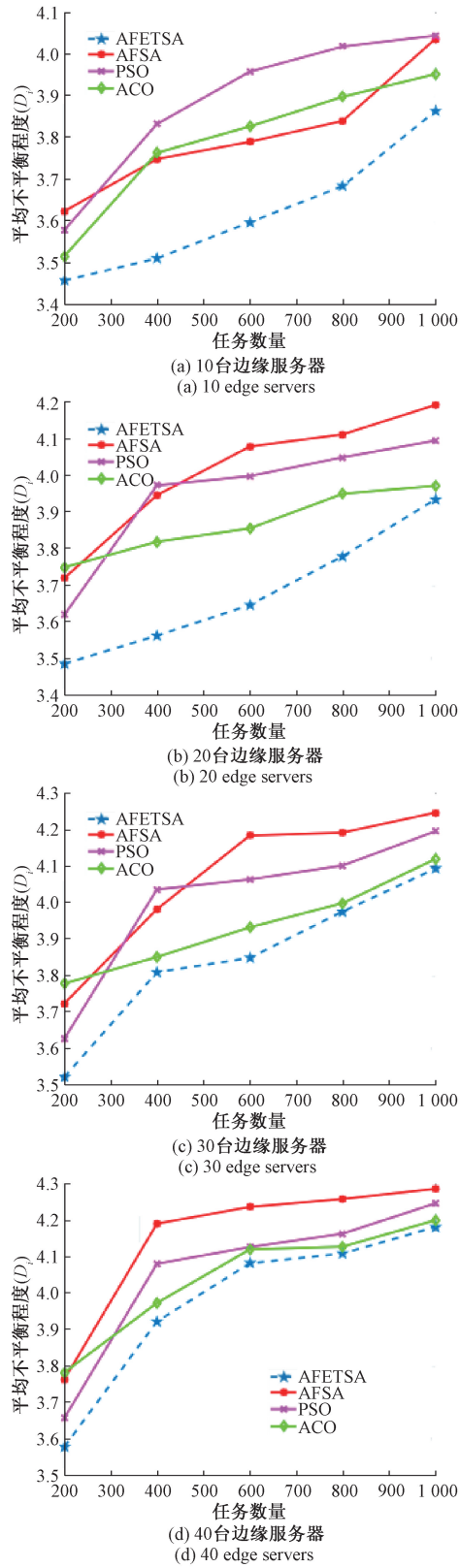


图 9　任务调度不平衡度对比

Fig. 9　Task scheduling imbalance degree comparison

参考文献

[1] 刘通，方璐，高洪皓. 边缘计算中任务卸载研究综述[J]. 计算机科学，2021，48(1)：11-15.
LIU T, FANG L, GAO H H. Review of task unloading studies in edge computing [J]. Computer Science, 2021, 48 (1): 11-15.

[2] YOUSEFPOUR A, FUNG C, NGUYEN T, et al. All one needs to know about fog computing and related edge computing paradigms: A complete survey[J]. Journal of Systems Architecture (S1383-7621), 2019, 98(1): 289-330.

[3] ZHANG Y, CHEN X, CHEN Y, et al. Cost efficient scheduling for delay-sensitive tasks in edge computing

system[C]. 2018 IEEE International Conference on Services Computing (SCC), 2018, DOI: 10.1109/SCC.2018.00017.

[4] SHI W, SUN H, CAO J, et al. Edge computing-an emerging computing model for the internet of everything era[J]. Journal of Computer Research and Development, 2017,54(5): 907-924.

[5] 王硕,郑爱云,刘怀煜.应用于智能制造的边缘计算任务调度算法研究[J].制造业自动化,2020,42(12):98-105.
WANG SH, ZHENG AI Y, LIU H Y. Research on the edge computing task scheduling algorithm applied to intelligent manufacturing [J]. Manufacturing Automation, 2020,42 (12): 98-105.

[6] 刘炎培,朱运静,宾艳茹,等.边缘环境下计算密集型任务调度研究综述[J].计算机工程与应用,2022,58(20):28-42.
LIU Y P, ZHU Y J, BIN Y R, et al. Review of computationally intensive task scheduling studies in the edge environment [J]. Computer Engineering and Application,2022,58(20):28-42.

[7] 王凌,吴楚格,范文慧.边缘计算资源分配与任务调度优化综述[J].系统仿真学报,2021,33(3):509-520.
WANG L, WU CH G, FAN W H. Review of edge computing resource allocation and task scheduling optimization [J]. Journal of Systems Simulation, 2021, 33(3): 509-520.

[8] ZHAO P, TIAN H, QIN C, et al. Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing[J]. IEEE Access, 2017, 5: 11255-11268.

[9] 张云飞,高岭,丁彩玲,等.边缘计算环境下改进蚁群算法的任务调度算法[J].计算机技术与发展,2021,31(9):86-91.
ZHANG Y F, GAO L, DING C L, et al. Task scheduling algorithm for improved ant colony algorithm in an edge computing environment [J]. Computer Technology and Development, 2021,31 (9): 86-91.

[10] 邓添,沈艳,史奎锐.基于遗传算法的移动边缘计算混合关键任务卸载[J].信息与电脑(理论版),2021,33(11): 26-29.
DENG T, SHEN Y, SHI K R. Genetic algorithm based mobile edge computing hybrid critical task unloading [J]. Information and Computer (Theoretical Edition), 2021, 33 (11): 26-29.

[11] GHASEMI S, KHEYROLAHI A, SHALTOOKI A A. Workflow scheduling in cloud environment using firefly optimization algorithm [J]. International Journal on Informatics Visualization,2019,3(3):237-242.

[12] 王文礼.边缘计算与云协同任务弹性调度方法研究[D].石家庄:河北经贸大学,2021.
WANG W L. Edge computing and cloud collaborative task elastic scheduling method [D]. Shijiazhuang: Hebei University of Economics and Trade, 2021.

[13] MALARVIZHI N, ASWINI J, SASIKALA S, et al. Multi-parameter optimization for load balancing with effective task scheduling and resource sharing [J]. Journal of Ambient Intelligence and Humanized Computing, 2021(9):1-9.

[14] RIZVI N, DHARAVATH R, EDLA D R. Cost and makespan aware workflow scheduling in IaaS clouds using hybrid spider monkey optimization [J]. Simulation Modelling Practice and Theory, 2021, 110(3): 102328.

[15] MISHRA K, PRADHAN R, MAJHI S K. Quantum-inspired binary chaotic salp swarm algorithm (QBCSSA)-based dynamic task scheduling for multiprocessor cloud computing systems[J]. The Journal of Supercomputing, 2021, 77(9): 10377-10423.

[16] 董思岐,吴嘉慧,李海龙.面向优先级用户的移动边缘计算任务调度策略[J].计算机应用研究,2019,37(9):2701-2705.
DONG S Q, WU J H, LI H L. Mobile edge computing task scheduling policy for priority users [J]. Computer Application Research, 2019,37(9):2701-2705.

[17] TOPCUOGLU H, HARIRI S, WU M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Transactions on Parallel and Distributed Systems, 2002, 13 (3): 260-274.

[18] 李晓磊,邵之江,钱积新.一种基于动物自治体的寻优模式:鱼群算法[J].系统工程理论与实践,2002(11):32-38.
LI X L, SHAO ZH J, QIAN J X. An optimization mode based on animal autonomy: Fish group algorithm [J]. System Engineering Theory and Practice, 2002 (11): 32-38.

[19] 黄少荣. 云计算任务调度算法研究[J]. 沈阳师范大学学报:自然科学版, 2015, 33(3):417-422.
HUANG SH R. Research on cloud computing task scheduling algorithm [J]. Journal of Shenyang Normal University: Natural Science Edition, 2015,33 (3): 417-422.

[20] 王联国,施秋红.人工鱼群算法的参数分析[J].计算机工程,2010,36(24):169-171.
WANG L G, SHI Q H. Parametric analysis of the artificial fish algorithm [J]. Computer Engineering,

2010,36 (24): 169-171.

[21] 李孝宇,李涛,李鹏,等. 基于自适应人工鱼群算法的大型光伏电站燃气轮机组的优化配置 [J]. 陕西电力,2013(8):15-20.
LI X Y, LI T, LI P, et al. Optimized configuration of gas turbine units in large photovoltaic power stations based on adaptive artificial fish swarm algorithm [J]. Shaanxi Electric Power, 2013 (8): 15-20.

[22] GLOVER F. Future paths for integer programming and links to artificial intelligence [J]. Computer and Operations Research,1986,13(5) : 533-549.

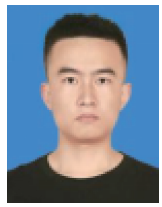
[23] 贺一. 禁忌搜索及其并行化研究[D]. 重庆: 西南大学, 2006.
HE Y. Taboo search and its parallelization study [D]. Chongqing: Southwest University, 2006.

[24] 谭文安,查安民,陈森博. 优化粒子群的云计算任务调度算法[J]. 计算机技术与发展,2016,26(7): 6-10.
TAN W AN, ZHA AN M, CHEN S B. Cloud computing task scheduling algorithm for optimizing particle swarm [J]. Computer Technology and Development, 2016,26 (7): 6-10.

[25] CALHEIROS R N,RANJAN R,BELOGLAZOV A,et al. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms [J]. Software: Practice & Experience,2011,41(1) : 23-50.

[26] PAUL M, SANYAL G. Survey and analysis of optimal scheduling strategies in cloud environment [C]. 2011 World Congress on Information and Communication Technologies. IEEE, 2011: 789-792.

作者简介



杨文杰,2019 年于太原工业学院获得工学学士学位,现为兰州交通大学硕士研究生,主要研究方向为边缘计算研究和智能计算研究。
E-mail: 1357847919@ qq. com

Yang Wenjie received his B. Sc. degree from Taiyuan Institute of Technology in 2019. Now he is a M. Sc. candidate in Lanzhou Jiaotong University. His main research interests include edge computing research and intelligent computing research.

巨涛(通信作者),分别获兰州大学工学学士学位,西安理工大学工学硕士学位,西安交通大学工学博士学位,现为兰州交通大学电子与信息工程学院副教授,主要研究方向为并行计算、高性能计算、机器学习并行优化、边缘计算。
E-mail: jutao@ mail. lzjtu. cn

Ju Tao (Corresponding author), received the B. Sc. degree from Lanzhou University, M. Sc. degree from the Xi′ an University of Science and Technology, and Ph. D. degree from Xi′an Jiaotong University. Currently, he is an associate professor with the School of Electronic and Information Engineering, Lanzhou Jiaotong University, China. His main research interests include parallel computing, high performance computing, machine learning and parallel optimization, and edge computing.