基于入侵肿瘤生长优化的云计算调度算法

周 静[1)] 董守斌[1)] 唐德玉[2)]

[1)](华南理工大学计算机科学与工程学院 广州 510641)

[2)](广东药科大学医药信息工程学院 广州 510006)

摘 要 随着云计算应用的发展,云计算任务调度的要求越来越复杂.群智能算法能在满足多种约束限制下,实现复杂的云计算任务调度问题,因而得到广泛应用.入侵肿瘤生长优化算法 ITGO(Invasive Tumor Growth Optimization)是一种新型的启发式群智能算法,该算法通过模拟肿瘤的生长和入侵行为,在解空间中搜寻最优解,具有较高的准确性和较快的收敛速度.该文将入侵肿瘤生长优化算法离散化,提出了一种离散化的入侵肿瘤生长调度算法 D-ITGO,通过将云计算任务调度方式的可行解即任务—虚拟机对应关系映射成为肿瘤细胞的坐标,使之可以应用于云计算任务调度问题;并针对云计算调度问题进行优化设计,包括:(1)设计生长细胞到入侵细胞的转换策略,使得更容易和更快地跳出局部最优解;(2)设计死亡细胞到入侵细胞转换策略,以避免浪费资源,并提高搜索效率;(3)调整生长细胞的生长步长,在逼近最优解时放慢生长速度,以避免跳过最优解.该文基于 CloudSim 仿真环境对 D-ITGO 算法以及优化策略进行了实验测试,并且使用非参数假设检验,对实验结果进行了评估和分析.实验结果和分析结果表明,这些策略均能提高收敛速率和搜索效率,其中,生长细胞到入侵细胞的转换策略和死亡细胞到入侵细胞转换策略在一定程度上减少了计算时间,生长细胞的生长步长调整策略能强化 D-ITGO 的搜索效率.同时,D-ITGO 算法比目前应用于云计算任务调度的算法,在云任务执行时间上有 7.1%~11.2%的提升,在调度开销上也有一定的优势.

关键词 云计算;群智能算法;入侵肿瘤生长优化算法;任务调度;时间开销

中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2018.01360

Task Scheduling Algorithm in Cloud Computing Based on Invasive Tumor Growth Optimization

ZHOU Jing[1)] DONG Shou-Bin[1)] TANG De-Yu[2)]

[1)](*Department of Computer Science and Technology, South China University of Technology, Guangzhou* 510641)

[2)](*Department of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou* 510006)

Abstract With the development of cloud computing, there are more and more complex requirements raised by cloud task schedule and then the corresponding schedule methods need to solve more and more problems. Swarm intelligence algorithms are widely used in this field (task schedule in cloud environment) because they can solve these complex problems effectively, and at the same time satisfy a lot of constraints. Invasive Tumor Growth Optimization is a new meta-heuristic algorithm of swarm intelligence. This algorithm can effectively find out the optimal solution(s) in the solution space, and obtain high convergence rate and accuracy by imitating the behavior of tumor cells, including the growth behavior and invasive behavior. In this paper, we proposed a Discrete Invasive Tumor Growth Optimization (D-ITGO), which is to make the original Invasive Tumor Growth Optimization to be discretized, to solve the task scheduling problem in cloud environment. The main schedule strategy of D-ITGO is to make a correspondence between cloud

收稿日期:2017-01-15;在线出版日期:2017-11-21.本课题得到广东省自然科学基金(2015A030308017,2016A030310300)资助.**周 静**,女,1992 年生,博士研究生,主要研究方向为云计算、分布式计算. E-mail: zjingscut@yahoo.com. **董守斌**(通信作者),女,1967 年生,博士,教授,主要研究领域为高性能计算、云计算、信息检索. E-mail: sbdong@scut.edu.cn. **唐德玉**,男,1978 年生,博士,副教授,主要研究方向为智能计算、生物信息技术.

tasks and virtual machines, then map this kind of correspondence into the coordinates of a tumor cell, finally solve the problem by improving some search strategies of original ITGO. We proposed three new strategies for this problem, including: (1) the design of a new transformation strategy, which transforms the growing cells into invasive cells by the criteria of *Life_span*, to make the D-ITGO to be easier and faster to get away from local optima; (2) a re-design of the transformation strategy between the death cells and invasive cells to improve the search efficiency; The new strategy make death cells transform into invasive cells directly, which can maximize the release of memory space, and at the same time save the computational resource; (3) an adjustment of growth step-size of the growing cells, which can slow down the growth of the growing cells when they approach to the optima values, to avoid skipping the optimal solutions. In the experiment, we used CloudSim toolkit for testing, and used the non-parametric test (Wilcoxon Test) to evaluate the effectiveness. Experimental results and analysis showed that these three strategies of D-ITGO can help it to improve the convergence rate and search efficiency. The transformation which transforms the growing cells into invasive cells and the transformation which transforms the death cells into invasive cells can reduce the computation cost. The step-size adjustment which slows down the growth of the growing cells can make the computation cost down to a certain value. The third strategy can enhance the search efficiency of D-ITGO. Experimental results and analysis showed that the new D-ITGO algorithm has 7.1%—11.2% advantage in make-span of tasks and has lower latency of network and lower running time compared with some generally used heuristic algorithms such as genetic algorithm (GA), particle swarm optimization (PSO), artificial colony algorithm (ABC), Cuckoo search algorithm (Cuckoo), bat search algorithm (BA) and cat optimization algorithm (CSO).

Keywords　cloud computing; swarm intelligence; invasive tumor growth optimization; task schedule; makespan

1　引　言

随着云计算技术的发展，云计算的应用越来越多样化，任务和资源的规模均快速增长，对云计算任务调度的要求也越来越高. 云计算中应用和资源的多样性，导致任务约束也有很大差异；云计算的商业运营特点决定了不仅要满足用户的多个目标需求，同时还要保障资源提供者的利益. 因此如何在多种约束下满足复杂的调度需求是目前云计算调度要解决的关键问题.

当前，常用的云计算调度算法主要可以分为两大类：非启发式算法和启发式算法. 非启发式算法将云计算问题视为一般的任务调度问题直接求解；启发式算法通常将云计算问题视为一个优化问题，并使用相应的启发式策略进行求解.

群智能算法是启发式算法中的一个大类，目前已经被广泛应用在云计算任务调度问题上，例如常见的粒子群算法(Particle Swarm Optimization)和蚁群算法(Ant Colony Optimization)等. 由于群智能算法通常能解决更复杂的调度问题，并满足多种多样的条件限制，因而被日益重视.

入侵肿瘤生长优化算法(Invasive Tumor Growth Optimization)是近年提出的一种启发式算法[1]，具有较高的准确率和收敛速度，已经被应用在解决连续优化问题上[2]，文献[2]的实验表明，在多数情况下，该算法能在解空间中找到问题的最优解.

本文基于入侵肿瘤生长优化算法，提出了一种新的云计算任务调度策略. 该策略针对 n 个相互独立的任务，构造了一个 $n\times m$ 维的解空间，并将每一种可行的调度方式映射为解空间中的可行解. 由于入侵肿瘤生长优化算法可以找到当前解空间中的可行解和相对最优解，因此，该调度算法能找到当前条件下相对最优的云计算任务调度策略. 实验表明本文提出的云计算调度策略效果良好.

本文的第 2 节将系统地阐述云计算任务调度的

相关算法;第 3 节将会简单介绍入侵肿瘤生长优化算法,以及详细介绍本文提出的基于入侵肿瘤生长优化算法的云计算调度策略;第 4 节针对该策略进行一系列的实验;第 5 节是总结和展望.

2 相关工作

针对云计算任务调度问题,国内外的许多学者都进行了相应的研究,并提出了许多行之有效的调度策略.这些策略主要可以分为两大类:

(1)使用非启发式算法,进行云计算任务调度.非启发式算法是任务调度中最常见的一类算法,通常出现在 PC 端或集群的任务调度上.这类算法不使用任何启发式策略,没有任何迭代过程,使用相对简单的策略进行任务调度的工作,通常将云计算任务调度问题视为一般的系统调度问题,并直接求解.例如,Max-min 算法是一类常见的非启发式算法,原理是将具有最大完成时间的任务映射到当前空闲的虚拟机上,Gao 等人[3]改进了 Max-min 算法,并将其应用在云计算环境中.该算法直接将云计算环境中的任务调度问题视为一般的调度问题,并使用 Max-min 算法直接求解.又例如,Min-min 算法是将当前具有最短完成时间的任务映射到空闲的虚拟机上,调度策略的原理与 Max-min 算法相反,但执行过程相对简单.Chen 等人[4]改进了该算法并将其应用在了云计算环境中.因此,诸如 Max-min、Min-min 的算法,都可以直接应用在云计算环境中,并直接求解云计算调度问题.这一类常见的调度算法还包括 FCFS(First Come First Serve)以及轮转法等.

上述算法是对一些常见的非启发式任务调度算法的改进.除此之外,一些国内外学者提出了新的非启发式算法,同样不使用任何启发式策略,直接求解云计算任务调度问题.例如,Ergu 等人[5]将等待调度的任务通过成对比较矩阵技术和给定可用资源和用户偏好的层次分析法(Analytic Hierarchy Process)进行排序,并根据任务的序列和等级来分配计算资源.当各个任务中的权重相互冲突时,进一步使用诱导偏差矩阵来识别不一致的元素并提高一致性比率.

非启发式算法可以在更短的时间(此处指算法的运行时间)内找到任务调度策略,但该策略的云任务执行时间(Makespan)不一定是最短的.因此,非启发式算法适合进行在线调度.

(2)使用启发式算法,进行云计算任务调度.启发式算法是另一类常用的云计算任务调度算法,这类算法通过一定的先验知识和优化策略,对云计算中的任务调度问题进行求解.与非启发式算法相比,启发式算法的典型特征为:将原始问题转化为一个可优化的问题,并通过对该优化问题进行求解,从而找到原问题的近似最优解.常用的启发式算法包括遗传算法(Genetic Algorithm)、粒子群算法(Particle Swarm Optimization)、蚁群算法(Ant Colony Optimization)、人工蜂群算法(Artificial Bee Colony)等等.群智能算法是启发式算法中的一个大类,通常使用一个种群(Swarm)进行问题的求解和优化.目前许多群智能算法都被应用在云计算任务调度问题上.例如上文提到的粒子群算法,就是一种典型的群智能算法.与群智能算法相对,非群体智能的启发式算法主要使用个体进行搜索,例如爬山搜索算法(Hill-climbing).启发式的云计算调度算法通常可以分为以下几类:

① 传统的启发式算法包括遗传算法、粒子群算法、蚁群算法等.这一类启发式算法已经被广泛应用在优化问题上.因此,如果将云计算调度问题表述为一个优化问题,即能用这类算法直接求解.例如,Zhao 等人[6]提出了一种基于遗传算法的优化方法,并将其应用在云计算任务调度上.该方法通过遗传算法的选择、交叉、变异等操作,对云计算环境中的独立可分任务进行调度,实验表明该算法运行良好.Kaur 等人[7]提出了一种基于遗传算法的元启发式方法,该方法通过最小化运行时间和最小化任务开销,完成云计算中的任务调度工作.上述两种算法都是基于遗传算法的云计算任务调度方法,具有典型的进化计算特征,包括遗传算法中的选择、交叉、变异,并且都对求解的问题进行了有针对性的改进和优化.除遗传算法外,其他群智能算法例如粒子群算法、蚁群算法等,也都得到了广泛的应用.例如,Adil 等人[8]提出了一种基于粒子群算法的云计算调度方法,该方法使用粒子群算法来决定任务运行方式,从而对云计算任务调度问题进行求解.该策略基于粒子群算法的内生机制,实验结果表明,该算法运行良好.Tawfeek 等人[9]提出了一种基于蚁群算法的任务调度方法,该算法将蚁群算法布设在云计算环境中,并为蚁群算法的控制参数设计了一种自适应策略,来提高蚁群算法的性能,并实现一个任务集的完成时间最小化.该算法可直接应用在云计算任务调度问题上;

② 基于传统的启发式算法改进的云计算任务调

度算法，如基于爬山搜索算法、遗传算法、粒子群算法、蚁群算法等改进的云计算任务调度算法. 与第 1 类算法相比，第 2 类算法在云任务的执行时间、算法的运行时间以及其他的评价指标上，明显比第 1 类算法要好，而且通常能满足更严格的条件限制，或者在某个细分领域达到更好的效果. 例如，Mondal 等人[10]使用了随机爬山搜索算法[11]解决云计算中的负载均衡问题，该算法以云计算任务调度中的负载均衡为调度目标，使用随机爬山搜索算法来处理生成的作业(job)，并随时更新分配表(Allocation Table)，以达到负载均衡的目标. Shojafar 等人[12]提出了一种混合优化策略 FUGE，该算法将模糊理论和遗传算法结合起来，将等待调度的任务和相应的虚拟机资源做模糊处理，作为算法的输入，并使用遗传算法中的交叉、变异等操作，执行调度的过程. 该算法基于运行时间和用户开销，解决了云计算中的任务调度问题，同时实现了负载均衡. Xu 等人[13]提出，将 Berger 模型引入粒子群算法中，并使用 Berger 模型判定云计算中资源分配结果的公平性，同时通过动态调整粒子群算法的参数，实现云计算任务的调度. Zhao 等人[14]提出了一种基于粒子群算法，并通过用户服务质量(Quality of Service，QoS)引导的任务调度方法，该算法的调度目标是在完成时间和成本之间进行权衡，根据当前的演化状态，动态调整粒子群算法的评估参数. 该算法可以避免过早收敛并更有效地在解空间中进行检索. Nishant 等人[15]提出了一种基于蚁群优化算法，并在节点之间进行工作负载分配的云计算任务调度算法. 该算法的主要目标是实现节点间的负载均衡. 算法的运行方式为：种群中的个体只保留并更新一个单独的结果集，而非更新它们自己所保留的结果集(在常用的蚁群算法中，每个个体会保留自己的结果集并不断更新，直到找到一个最优的解决方案). 该算法保证了高峰时间内的顺利运行. Li 等人[16]提出了一种基于负载平衡蚁群优化(LBACO)算法的云计算任务调度策略. 该策略的目标是平衡整个系统的负载，同时尽量减少给定任务集的最大完成时间. 实验结果表明，该算法优于 FCFS 算法和 ACO 算法. Yao 等人[17]提出了一种改进的人工蜂群算法，并将其应用在云计算任务调度上. 该算法使用一种“替换请求”的策略，提高了系统的吞吐量，并通过更换请求的方式，使节点的运行更加顺畅. 实验结果表明，该算法在可扩展性方面表现良好，解决了当出现负载恶化时的负载均衡问题. Shu 等人[18]改进了克隆选择算法，并提出了一种基于时间开销和能耗效率的新的克隆选择算法，将其应用在云计算调度问题上. 该算法在响应时间和任务集的最大完成时间两个方面，表现出了显著的改进和提高，显示出高潜力的数据中心能耗效率，并能有效地满足用户的服务水平协议；

③ 与其他算法相结合的云计算调度方法. 这一类算法通常将启发式算法与非启发式算法相结合，或者将两种或以上的启发式算法相结合，以形成一种新的调度方法，并应用在云计算调度问题上. 例如，Kumar 等人[19]通过将遗传算法和 Max-min，Min-min 算法结合的方式，使用 Max-min 算法和 Min-min 算法的结果初始化整个种群，并通过遗传算法的交叉因子和变异因子，使整个种群达到进化和提高的效果. 该算法同时具备遗传算法、Max-min 算法和 Min-min 算法的优点，在云计算任务调度的具体应用中表现良好. Kruekaew 等人[20]将人工蜂群算法和最长作业优先(Longest Job First)算法相结合的方式，提出了一种新的 ABC_LJF 算法，以适应动态变化的云计算环境，减少最大任务完成时间(Makespan)和数据处理时间(Data Process Time)，并实现负载均衡. 实验表明该算法优于其他的 5 种算法. Nirmala 等人[21]将鲶鱼粒子群优化算法应用在云计算任务调度中，该算法可以选择出具有最小完成时间和运行开销的调度策略. 实验证明该算法运行良好. 这类算法是近年来出现的一种较为新型的算法，目前国内外的研究相对较少；

④ 新兴的启发式算法. 随着仿生学研究的不断进步，越来越多的群体智能行为被应用在算法设计上，例如模仿布谷鸟行为的布谷鸟搜索算法(Cuckoo Search)、模仿蝙蝠行为的蝙蝠搜索算法(Bat Algorithm)、模仿猫群行为的猫群优化算法(Cat Swarm Optimization)等等. 这一类算法并未被广泛应用，但由于这类算法基本都是基于群体智能的算法，目前已经有许多学者尝试将这类算法应用在云计算任务调度问题上. 例如，Navimipour 等人[22]提出了一种基于布谷鸟搜索(Cuckoo Search Algorithm)的算法，进行云计算任务调度. 该算法的搜索原理完全基于布谷鸟算法，并取得了良好效果. Raghavan 等人[23]使用了一种蝙蝠算法(Bat Algorithm)来解决云计算问题. 该算法借助二进制蝙蝠算法(Binary Bat Algorithm)进行云环境中的工作流调度，旨在将任务映射到相应的资源上，同时使工作流的总资源开销达到最小. Bilgaiyan 等人[24]提出了一种基于猫群优化(Cat Swarm Optimization)的启发式调度

算法,将应用任务分配到可用的资源上.该算法同时考虑了两个相关资源之间的数据传输成本,以及不同资源上任务的执行成本,作者使用一个假设的工作流证明了算法的有效性.这些启发式算法的共同特点是,设计一个合适的适应度函数来描述云计算调度的问题,并使用相应的启发式算法去解决这个问题.

与非启发式算法相比,启发式算法的一个明显特征是:需要多次迭代才能找到最优解.因此,在算法的运行时间上,启发式算法相对于非启发式算法,没有明显的时间优势;但找到最优的调度策略之后,该最优策略的云任务执行时间更具优势.因此,非启发式算法能在更短的时间内找到(相对)最优解,但启发式算法找到的最优解更好.

根据"没有免费午餐"原理[25],没有任何一个算法能完全满足云计算任务调度中的所有条件限制,并且能在所有的应用场景中找到最优解.因此,只存在"某个条件限制下某个算法的效果是最好的",例如,在用户服务质量(QoS)引导的条件下,算法[14]的效果是最好的.

入侵肿瘤生长优化算法是 2015 年提出的一种新的启发式算法,该算法通过模拟肿瘤细胞的生长、入侵、休眠、死亡 4 种行为,并模拟肿瘤细胞周围的微环境,在解空间内进行求解.该算法的求解模式为:通过细胞周围营养液浓度的变化,诱导肿瘤细胞向营养液浓度高的地方游走,在解空间中找到问题的可行解和最优解.当陷入局部最优解时,该算法通过 Levy 飞行的方式,跳出局部最优.目前,入侵肿瘤生长优化算法已经被应用在若干个问题上.例如数据聚类问题[1]和连续优化问题[2]等.

本文基于入侵肿瘤生长优化算法,提出了一种新的云计算任务调度方法,通过将云计算任务调度方式映射到相应的解空间,同时调整入侵细胞和生长细胞的转换方式,并通过阈值控制各类细胞之间的转化,找到相应的最优解.实验结果表明,该算法的表现良好.

3 算法描述

3.1 ITGO 算法

入侵肿瘤生长优化方法(Invasive Tumor Growth Optimization)是我们之前提出的一种基于群智能的启发式优化算法[1],该算法的原理是:通过模拟肿瘤细胞的生长、入侵、休眠、死亡 4 种行为,以及模拟肿瘤细胞周围的微环境,进行相关问题的计算和求解.在入侵肿瘤生长优化方法中,营养液浓度代表细胞的适应度值,细胞将朝着营养液浓度高的方向移动.入侵细胞位于整个种群的最外层,其次是生长细胞、休眠细胞和死亡细胞.死亡细胞基本上处于营养液浓度极低的状态,不执行任何搜索任务.这 4 类细胞在解空间中的求解状态,具体表现为:

(1)入侵细胞.入侵细胞是整个肿瘤细胞(即整个种群)中最为活跃的一部分,可以在周围的营养液中快速游走,并向整个种群之外的空间移动.由于入侵细胞有极强的搜索能力,在整个种群中具有最大的搜索步长,并能通过 Levy 飞行跳出局部最优解,因此,入侵细胞在整个种群中,承担了部分的外延搜索任务,以及全部的关于"跳出局部最优解"的任务.当细胞种群陷入局部最优解时,唯一的方法就是通过入侵细胞的 Levy 飞行,找到营养液浓度更高的位置,随后引导整个种群向该位置移动,直到跳出局部最优解:

$$Icell_{i,j}(t+1)=Icell_{i,j}(t)+\alpha\cdot Levy(s) \tag{1}$$

$$\alpha=rand\cdot\left(\frac{t}{T}\right) \tag{2}$$

$$Levy(s)\sim|s|^{-1-v},\ 1<v\leq 2 \tag{3}$$

$$s=\frac{u}{|v|^{1/\omega}} \tag{4}$$

$$u\sim N(0,\sigma_u^2),v\sim N(0,\sigma_v^2) \tag{5}$$

$$\sigma_v=\left\{\frac{\Gamma(1+\omega\sin(\pi\omega/2)}{\Gamma[(1+\omega/2]\omega 2^{(\omega-1)/2}}\right\}^{1/\omega} \tag{6}$$

$$\sigma_u=1 \tag{7}$$

式(4)~(7)为蒙特卡罗方法的步长选择[2],参数 ω 的取值详见表 1.在式(1)中,$Icell_{i,j}(t+1)$代表当前的入侵细胞,t 为当前迭代次数,α 是步长控制参数,$Levy(s)$是 Levy 飞行的步长分布.式(2)为步长控制参数 α 的计算方法,其中 t 代表当前迭代次数,T 代表算法最大迭代次数.式(3)是 Levy 飞行的步长分布计算.ITGO 算法[2]使用蒙特卡罗方法计算其步长;

(2)生长细胞.生长细胞位于整个肿瘤细胞的第 2 层,其外层为入侵细胞,里层为休眠细胞.生长细胞承担了绝大部分的搜索工作,主要负责在当前肿瘤细胞种群的搜索范围内,找到更好的解.因此,在整个细胞种群中,生长细胞的生长周期是最长的.但是,当种群陷入局部最优解时,生长细胞只能依靠入侵细胞的引导,才能跳出局部最优.生长细胞有一定的概率生成入侵细胞.当周围的营养液浓度过低

时，部分生长细胞将会转变为休眠细胞；

(3)休眠细胞. 休眠细胞位于整个肿瘤细胞的第 3 层，其外层为生长细胞，里层为死亡细胞. 休眠细胞的直观意义为“即将死亡的细胞”，即当周围的营养液浓度过低时，部分细胞由于营养成分不足而进入休眠状态，转变为休眠细胞，其搜索能力显著降低，直到周围的营养液浓度再次发生变化(如果升高，那么有一定几率重新转变为生长细胞；如果过低，那么有一定几率转变为死亡细胞). 休眠细胞只和周围的细胞有微小的互动. 在整个种群的搜索范围内，休眠细胞基本不执行搜索策略，或很缓慢地执行搜索策略；

(4)死亡细胞. 死亡细胞是肿瘤细胞群体中，由于营养成分不足而死亡的细胞. 在解空间范围内，死亡细胞不执行任何检索策略，不占据任何计算资源，到达一定的细胞周期之后，它们所占据的计算空间会被直接释放. 死亡细胞一般由休眠细胞衰退形成，其转变过程不可逆.

在本文的新算法中，生长细胞、休眠细胞和死亡细胞的初始赋值、坐标计算和转化方式已经改变，将在 3.2 节中阐述.

ITGO 算法的算法流程表述如下：首先初始化整个种群，然后根据不同的比例，分别初始化生长、入侵、休眠和死亡 4 类细胞. 随后通过细胞周围营养液浓度的变化，使整个种群在解空间中，朝着营养液浓度高的方向移动，直到找到可行解和最优解. 在肿瘤细胞生长过程中，每一种细胞都分别执行相应的搜索策略. 文献[2]中详细说明了 ITGO 算法相关的运算流程、公式及参数.

3.2　ITGO 的离散化算法 D-ITGO

初始的 ITGO 算法是基于连续优化问题的算法，只能在连续的解空间内搜索. 但是，云计算任务调度问题属于离散优化问题，因此，初始的 ITGO 算法不能直接应用在该问题上. 在使用之前，需要对 ITGO 算法的解空间进行离散化.

在原始的解空间中，种群中的细胞可以出现在有理数范围内的任意一点. 为了实现解空间的离散化，首先需要对细胞的坐标进行限制. 即

$$cell.center = round(cell.center) \tag{8}$$

本文使用方程(8)，对肿瘤细胞中的每一个细胞的坐标进行限制. 在方程(8)中，$cell.center$ 代表细胞的坐标，包括生长细胞、入侵细胞、休眠细胞和死亡细胞的坐标. $round()$ 将截断细胞坐标中的小数位，仅保留其整数位. 由于在细胞初始化之前，算法将对解空间的范围进行限制(见表 1 参数中的 $upper\_bound$ 和 $lower\_bound$). 因此，在算法离散化之后，细胞的坐标会被限制在正整数范围内，且不大于虚拟机的总数. 因此，种群中的每个细胞在解空间中的位置将是离散的点，而不是整个连续的解空间. 每个细胞也只能在离散的坐标上进行搜索和比较.

为了进一步提升搜索效率，最大限度地找到最优解，本文对 ITGO 算法提出了以下改进：

(1)生长细胞与入侵细胞的转换效率

ITGO 算法[2]根据每个细胞的养分浓度(适应度的值)进行排序，养分浓度高的细胞被判定为生长细胞($Pcell$，占细胞总数的 20%)，中等养分浓度的细胞被判定为休眠细胞($Qcell$，占肿瘤所有细胞的 60%)，养分浓度低的细胞被判定为死亡细胞($Dcell$，占肿瘤总数的 20%). 为了生长，$Pcell$，$Qcell$ 和 $Dcell$ 使用不同的方式获得足够的养分[2]. 同时，生长细胞和死亡细胞都有一定概率转化为入侵细胞[2]，入侵细胞执行 Levy 飞行，尝试引导整个种群在解空间内移动，并跳出局部最优值. 在下一轮迭代之前，ITGO 使用同样的比例进行细胞之间的转换.

但是，在实际应用过程中，不一定需要 40% 的细胞完全转换为入侵细胞. 在良好的搜索条件下，生长细胞远离局部最优解，评价函数的变化速度非常快，不一定需要入侵细胞进行飞行和引导，而应该尽可能多地提高生长细胞数量，增加搜索效率. 在这种情况下，入侵细胞的概率应该远小于 40%. 而在恶劣的检索条件中，几乎所有的生长细胞都集中在若干个局部最优解附近，评价函数的变化速度非常慢，在这种条件下，应该尽可能多地增加入侵细胞数量，以便尽快跳出局部最优解. 在这种情况下，入侵细胞的比例应该尽可能大，应该大于 40%.

因此，D-ITGO 算法将采取以下策略，调整入侵细胞的比例：在一个单独的生长细胞周期内，如果一个外沿的生长细胞没有找到更好的解，那么它将执行一次变异，直接变成入侵细胞，跳出当前的检索区域，尝试在整个解空间内进行搜索，不受原始的 ITGO 算法条件限制：

$$Pcell_{i,j}(t+1) = \begin{cases} Icell_{\text{new}}(t+1), & fit_{sn(last)} - fit_{sn(1)} < 0.01 \\ Pcell_{i,j}(t) + rand() \times (Icell_{i,j}(t) - Pcell_{i,j}(t)), & \text{其他} \end{cases} \tag{9}$$

$Pcell_{i,j}(t+1)$ 代表当前生长细胞，$Icell_{\text{new}}(t+1)$ 代表当前新生成的入侵细胞，$Icell_{i,j}(t)$ 代表引导 $Pcell_{\text{new}}(t)$ 的入侵细胞. $fit_{sn(i)}$ 代表当前细胞周期执

行到第 i 代的适应度值.

公式的第 1 列是指,当某个生长细胞到达细胞周期的末端,即经过一定次数的搜索行为后,如果该细胞的适应度值与它在细胞周期的初始时刻的适应度差值小于 0.01,即判定该生长细胞陷入局部最优,直接分化出一个入侵细胞,执行 Levy 飞行,以跳出局部最优解.公式的第 2 列是指在其他条件下,程序判定当前的搜索状况良好,该生长细胞不需要分化生成额外的入侵细胞,继续执行原有的搜索策略,与原先的 ITGO 算法策略相同.

(2)死亡细胞的消亡和生长细胞的分裂

在初始的 ITGO 算法中[2],死亡细胞在种群中占据 20%的份额,同时,只有在一个死亡细胞的细胞周期结束之后,才会在原来的空间里生成一个新的入侵细胞.这个策略虽然有效利用了闲置资源,但因为死亡细胞在整个细胞周期内,基本不执行任何搜索任务,因此造成了一定的计算资源浪费.在 D-ITGO 算法中,死亡细胞将执行以下的变异策略:在初始化时刻,死亡细胞的数量为 0(其生物学逻辑为:在肿瘤细胞的初生时刻,周围的营养液浓度较高,因此死亡细胞的数量为 0).当休眠细胞的营养液浓度(适应度值)过低时,部分休眠细胞将直接转变为死亡细胞.但是,当休眠细胞被标记为死亡细胞时,该死亡细胞所占据的计算资源将直接被释放,其细胞周期为 0,同时整个种群随机选择一个生长细胞,分裂并分化出一个新的入侵细胞.即

$$\begin{cases} Dcell_{i,j}(t+1)=lastN\%(Qcells) \\ Release(Dcell_{i,j}(t+1)) \\ Icell_{\text{new}}(t+1)=Pcell_{i,j}(t+1)+\alpha\cdot Levy(s) \end{cases} \tag{10}$$

其中,$Qcells$ 指当前的所有休眠细胞,$lastN\%$ 为死亡细胞生成概率.$Dcell_{i,j}(t+1)$是指 $t+1$ 时刻生成的死亡细胞,$Release()$的功能是释放该细胞所占据的计算资源.$Icell_{\text{new}}(t+1)$是指新生成的入侵细胞,$Pcell_{i,j}(t+1)$是指 $t+1$ 时刻的生长细胞,α 是 ITGO 算法中的 Levy 飞行控制参数,取值为 $rand\times(t/T)$.其中 $rand$ 为随机数,t 为当前迭代次数,T 为总迭代次数[2].

式(10)的第 1 行是指周围营养液浓度排在末位的部分休眠细胞将死亡.第 2 行是指当死亡细胞生成时,死亡细胞所占据的资源将被直接释放.第 3 行是指生长细胞将随机分化为一个入侵细胞,并执行 Levy 飞行.执行该策略后,算法将会直接释放死亡细胞所占据的计算资源,并刺激细胞群体生成一部分的入侵细胞.由于死亡细胞产生的条件为:部分休眠细胞的营养液浓度(适应度值)过低,即搜索效率过低.因此,该策略可以在减小计算开销的同时,一定程度上提高搜索效率.

(3)步长调优

通常在种群搜索过程中,越接近最优解,所需要的步长越小.如果步长过长,很可能会"跳过"最优解,导致无法精确定位到某一个值.因此,本文针对 ITGO 算法的细胞搜索步长进行如下调整:

$$step=\frac{T}{10\times t}step \tag{11}$$

即在 ITGO 算法[2]的步长公式后,添加一个调整系数.该系数可以通过迭代次数调整步长范围.越接近最优解时,步长将会变得越小.公式中 T 为最大迭代次数,t 为当前迭代次数.

3.3 云计算调度算法

云计算任务调度问题是一个 NP-hard 问题,在搜索之前,需要将相关问题映射到一个 $n\times m$ 维的解空间,并使用离散化的 D-ITGO 算法,在解空间中搜索最优解.由于待调度的任务下标和虚拟机下标都是离散的变量,因此相应的解在解空间中,也是离散的点.因此,D-ITGO 算法需要在离散的解空间中搜索最优解.在算法执行之前,需要对整个细胞种群和数据进行初步的预处理.

预处理包括两个部分:(1)调度策略的编码;(2)适应度函数的构造.预处理完成后,需要根据上文提到的优化策略,对 D-ITGO 算法的运算过程进行进一步处理,以提高搜索效率.

3.3.1 种群初始化

在 D-ITGO 算法中,调度策略的编码直接包含在种群初始化的过程中.如上文所述,式(8)直接将种群细胞的坐标映射到了一个离散化的解空间里,所有的细胞坐标都是一个正整数,且不超过虚拟机的最大下标值.因此,细胞坐标本身即包含了一种映射方式,将云任务下标(即坐标的位置)与虚拟机下标(即坐标的值)一一对应.

因此,D-ITGO 算法中,每一个细胞包含一种调度策略,细胞的坐标代表调度策略的映射方式,细胞的适应度值代表调度策略的云任务执行时间(Makespan).所有可行的调度策略构成了整个算法的解空间,每个细胞都将在解空间中寻找最优解.

因此,给定 n 个相互独立的任务,m 个相互独立的虚拟机,则细胞的坐标为

$$\begin{cases} Tumor_i.center=(x_1,x_2,x_3,\cdots,x_n),j\in[1,n] \\ x_j\in[1,m] \end{cases} \tag{12}$$

其中，$Tumor.center$ 代表细胞的坐标值，$Tumor_i$ 代表第 i 个细胞，x_j 为每个任务所对应的虚拟机下标. m 是虚拟机的总数，也是虚拟机的最大下标值，即细胞坐标的最大上限. 在种群生长和搜索过程中，每个细胞都将在解空间中寻找最优的调度策略，直到找到最优解或达到最大迭代次数.

在 D-ITGO 算法中，云计算任务调度方式的可行解即任务——虚拟机对应关系映射成为肿瘤细胞的坐标，例如：假设某个细胞的坐标为(2,1,1,2,2,1,2)，则映射方式为：第 1、4、5、7 个任务被映射到第 2 台虚拟机上，第 2、3、6 个任务被映射到第 1 台虚拟机上. 任务总数为 7，虚拟机总数为 2.

在初始化过程中，每个细胞将会被赋予一个初始值，在后续的计算过程中，每个细胞根据 ITGO 算法的搜索规则不断调整该初始值，在解空间内找到一个合适的解(映射方式). 种群初始化算法表述如算法 1.

算法 1.　种群初始化.

输入：云计算任务列表 Cloudlets，虚拟机列表 Vms

输出：初始化的种群 Tumors

1. $n \leftarrow$ size(Cloudlets)；/*n 是云计算任务规模*/
2. $m \leftarrow$ size(Vms)；/*m 是虚拟机的总数*/
3. $Dmin \leftarrow 1$；/*解空间下限*/
4. $Dmax \leftarrow m$；/*解空间上限*/
5. FOREACH tumor $i \in$ Tumors
6. $i.center \leftarrow rand([Dmin\ Dmax],1,n)$；/*坐标初始化*/
7. $i.center \leftarrow round(i.center)$；/*离散化*/
8. $i.fitness \leftarrow compute\_fitness(i)$；/*适应度函数计算*/
9. ENDFOR

3.3.2　适应度函数的构造

由于 D-ITGO 算法是一个典型的启发式算法，因此，在进行云计算任务调度的问题求解之前，需要将该问题表述为一个优化问题，然后使用 D-ITGO 算法中的相应搜索策略求解. 最优化问题通常以适应度函数的形式表现. 因此，在 D-ITGO 算法中，我们使用以下的适应度函数：

$$Timecost = Exec\_time + Latency \qquad (13)$$

作为云计算任务调度问题的适应度函数. 在该函数中，$Timecost$ 代表在一个策略中(每个细胞都包含一个可行的策略)所有任务的执行时间开销，即该算法的适应度函数值. 因此，在该模型中，$Timecost$ 是指一个策略中的云任务执行时间，即等同于 Makespan. $Timecost$ 分为两个部分：$Exec\_time$ 和 $Latency$. 其中，$Exec\_time$ 指一个云任务在一台虚拟机上运行的时间集合，该集合表现为一个运行时间矩阵，可以通过下标索引，$Latency$ 指代时间延迟. $Timecost$ 是指云计算任务执行完毕所需的时间开销，延迟即在运算过程中的所有延迟. 第 4 节将会详细介绍在模拟过程中这两个部分的计算方法.

由于在整个云计算环境中，即在整个集群中，虚拟机通常同时运行，因此某个策略的云任务执行时间开销，在数值上等于集群中最大的虚拟机运行时间. 即单个细胞的适应度值等于该策略中最大的虚拟机运行时间开销. 适应度函数计算表述如算法 2.

算法 2.　适应度计算.

输入：种群 Tumors，任务执行时间 $Exec\_time$，延迟 $Latency$

输出：带有适应度的种群 Tumors

1. FOREACH tumor i in Tumors /*所有细胞*/
2. FOREACH vm j in tumor i /*每个细胞中的所有虚拟机*/
3. $index \leftarrow find(j == \text{Tumor}(i).center)$；/*找出在策略 i 中，虚拟机 j 上运行的所有任务下标*/
4. $totalExec\_time(j) \leftarrow sum(Exec\_time\ (index))$；/*计算当前虚拟机 j 上运行的所有任务的运行时间的总和*/
5. $totalLatency(j) \leftarrow sum(Latency(index))$；/*计算与当前虚拟机 j 上有关的所有 $Latency$ 的总和*/
6. $timecost(j) \leftarrow totalExec\_time(j) + totalLatency(j)$；/*当前虚拟机 j 的运行时间*/
7. ENDFOR
8. $\text{Tumor}(i).fitness = \max(timecost)$；/*整个策略的完成时间，等于虚拟机运行时间中的最大值，即细胞的适应度值*/
9. ENDFOR

3.3.3　交叉变异

D-ITGO 算法的交叉变异策略与 ITGO 算法相同，但在每一次的步长计算时，需要通过方程(11)来调整相应的步长.

3.3.4　算法描述

综上所述，D-ITGO 算法的运算过程描述如下：首先初始化整个种群，并根据营养液浓度的不同，分别调整 4 类不同细胞的数量，随后通过细胞周围营养液浓度的变化，让细胞群在解空间中朝营养液浓度高的方向移动，直到整个种群逼近最优解或近似最优解.

在单个细胞周期内，算法仅执行搜索操作.

细胞周期 Sn 的定义如下[2]：每一类细胞都有自己的细胞周期，在单独的细胞周期内，只有一类细胞能生长和移动. 当所有细胞的细胞周期结束之后，

一次迭代完成.因此,细胞周期的意义是局部搜索(local search).

算法在初始化时,需要将适应度排在前 $TopN\%$ 的细胞作为生长细胞,其余作为休眠细胞.在 D-ITGO 算法中,入侵细胞由生长细胞动态生成,死亡细胞由休眠细胞动态生成,因此这两种细胞在初始化时,数量为 0.但在原始的 ITGO 算法中[2],入侵细胞和死亡细胞的数量均不为 0.同时,在 D-ITGO 算法中,生长细胞的细胞周期长度定义为 $Life\_span$,而死亡细胞由适应度排行最末尾的 $LastN\%$ 的休眠细胞形成,其生成规则如策略 3 所示.同时,种群中的入侵细胞生成由策略 1 和(或)策略 2 决定.上述 3 个参数的取值见表 1.算法用伪代码表述如下.

算法 3. D-ITGO 算法.

输入:初始化的种群 Tumors

输出:最优云计算调度策略 Tumor_best

```
1. sort(Tumors);
2. ITumors←TopN%_of_Tumors; /*生长细胞*/
3. PTumors←else_of_Tumors; /*休眠细胞*/
4. WHILE t<T
5.   /*首轮迭代时,入侵细胞种群为空*/
6.   IF num(ITumors)>0 /*如果入侵细胞种群不为空*/
7.     FOREACH i∈ITumors /*入侵细胞的细胞周期内*/
8.       ITumors(i)←ITumors(i)+α·Levy(λ); /*Levy 飞行*/
9.     ENDFOR
10.  ENDIF
11.  FOREACH i∈PTumors
12.    WHILE 1≤Sn≤Life_span /*生长细胞的细胞周期内*/
13.      update PTumor.center /*更新生长细胞的坐标*/
14.      Sn←Sn+1;
15.      IF Sn==Life_span
16.        produce ITumor by equation(2) /*当搜索环境恶劣时,边沿的生长细胞将变异成为入侵细胞*/
17.      ENDIF
18.    ENDWHILE
19.  ENDFOR
20.  FOREACH i∈QTumors /*休眠细胞的细胞周期内*/
21.    update QTumor.center /*更新休眠细胞的坐标*/
22.  ENDFOR
23.  DTumors←LastN%_of_QTumor; /*得到营养最少的休眠细胞将死亡*/
24.  FOREACH i∈DTumors /*方程(10)*/
25.    Release(i);/*死亡细胞消失*/
26.    select k; /*随机选择一个生长细胞*/
27.    tumor←PTumors(k)+α·Levy(λ)/*生成入侵细胞*/
28.    ITumors←ITumors+tumor/*添加到相应种群中*/
29.  ENDFOR
30.  t←t+1;
31. ENDWHILE
32. sort(Tumors);
33. Tumor_best←Tumors(1);
```

3.3.5 复杂度分析

在 D-ITGO 算法中,给定 m 个虚拟机和 n 个任务,同时给定细胞种群规模 pop_size,则算法所需的空间资源为:存储细胞坐标和适应度值的空间 $pop\_size\times(n+1)$,以及存储其他变量、常量所需的空间 C,因此,算法的空间复杂度为

$$S(n)=O(pop\_size\times n+C) \tag{14}$$

在 D-ITGO 算法中,总共有 pop_size 个细胞需要执行搜索工作(由于死亡细胞所占据的资源直接被释放,并由一个生长细胞直接分化出入侵细胞,因此,细胞种群的数量是恒定的).

在搜索过程中,总共需要执行 $n\times\log(m)$ 次操作,即搜索任务的规模与云任务和虚拟机的数量正相关.因此,D-ITGO 算法的时间复杂度表示为

$$T(n)=O(pop\_size\times n\times\log(m)+C) \tag{15}$$

4 实验和分析

4.1 实验环境和实验数据

本文将使用 CloudSim3.0 模拟云计算环境,并使用 300 个随机生成、相互独立的云任务(Cloudlet),同时使用 20 个相互独立的虚拟机,进行任务调度的初步模拟实验.本文使用 8 个额外的对比算法:遗传算法(GA)、粒子群算法(PSO)、人工蜂群算法(ABC)、基于模糊策略的遗传算法(FUGE)、鲶鱼粒子群优化算法(CPSO)、布谷鸟搜索算法(Cuckoo)、蝙蝠搜索算法(BA)、猫群优化算法(CSO),与本文提出的 D-ITGO 算法进行对比.

为了进一步说明算法的有效性,本文在初步的模拟实验之外,还将使用 200~400 个不等的云计算任务(虚拟机数量不变),在同等实验条件下,证明

D-ITGO 算法在不同任务规模下的有效性.

4.2　算法参数配置

本文的算法参数配置，主要分为 3 个部分：

(1) 由于 FUGE 算法[12] 和 CPSO 算法[21] 已经直接被应用在云计算任务调度问题上，因此在本文的模拟实验中，FUGE 算法的参数设置将采用文献[12]的参数，CPSO 的算法参数将采用文献[21]的参数；

(2) 由于遗传算法(GA)、粒子群算法(PSO)、人工蜂群算法(ABC)、布谷鸟搜索算法(Cuckoo)、蝙蝠搜索算法(BA)、猫群优化算法(CSO)都是基于连续优化问题设计的算法，其原始参数不能直接应用在云计算任务调度问题上. 因此，本文将分别采用这 6 种算法在云计算调度问题上的参数设置. 例如，文献[26]使用了遗传算法来解决云计算调度问题，但没有对遗传算法进行实质上的改动，因此，本文将使用文献[26]中遗传算法的交叉率和变异率，作为本实验中遗传算法的参数. 同理，文献[27]使用了粒子群算法解决云计算资源调度问题，但没有对粒子群算法做出实质性的改动，因此，本文将使用文献[27]中粒子群算法的相关参数，作为实验中粒子群算法的参数. 文献[20]使用人工蜂群算法进行云计算任务调度，但没有对人工蜂群算法做出改动，因此，本文将使用文献[20]中人工蜂群算法的相关参数，作为本实验中人工蜂群算法的参数. 文献[22]使用了布谷鸟搜索算法解决云计算资源调度问题，但没有对布谷鸟搜索算法做出实质性的改动，因此，本文将使用文献[22]中的相关参数，作为实验中布谷鸟搜索算法的相关参数. 文献[23]使用蝙蝠搜索算法来解决云计算调度问题，但没有对蝙蝠算法做出改动，因此，本文将使用文献[23]的相关参数值，作为本实验中蝙蝠算法的实验参数. 文献[24]使用猫群优化算法来解决云计算调度问题，但没有对猫群算法做出改动，因此，本文将使用文献[24]中的相关参数，作为本实验中猫群优化算法的相关参数. 这些参数都经过实验验证，可以在云计算调度问题中得到良好的结果，因此，本文使用的参数具有一定的现实意义；

(3) 由于 ITGO 算法是基于连续优化问题设置的算法，其原始参数不能直接应用在离散优化问题上. 因此，本文除了使用方程(8)，对算法进行离散化，并使用方程(11)，对算法步长进行调整外，其余参数均采用随机数的形式，并使用多次实验得到的粗略平均值，作为参数的取值或取值范围. 相关参数的取值或取值范围如表 1 所示.

表 1　D-ITGO 算法的参数取值

参数名称	取值或取值范围	释义
pop_size	30	种群规模
n	300	任务数
m	20	虚拟机个数
$upper\_bound$	m	解空间范围上限
$lower\_bound$	1	解空间范围下限
ω	[0.1,3]	Levy 飞行的参数
β	[0.8,1.4]	步长参数
$Life\_span$	7	细胞的生命周期
$TopN$	30	生长细胞的占比
$LastN$	10	死亡细胞的占比

在算法中，种群规模被设置为 30，任务数目为 300，虚拟机规模为 20. 解空间的范围上限等于虚拟机的总数(即最大下标)，下限为虚拟机的最小下标 1. 算法中的参数 ω，在本次实验中取随机数，随机数的取值范围为[0.1,3]，该范围由多次实验的结果得出. 参数 β 在本次实验中取随机数，随机数的取值范围为[0.8,1.4]，由多次实验的结果得出. 细胞生长周期 $Life\_span$ 设为 7，同样是多次随机实验得到的粗略平均值，数值太小无法体现出细胞周期的作用，数值太大则会减慢运行速度. 初始化时刻，生长细胞的占比 $TopN\%$ 为 30%，是多次随机实验得到的粗略平均值，能使整个算法表现相对良好. 死亡细胞的占比 $LastN\%$ 设为 10%，是多次实验得到的粗略平均值，数值太大会导致策略 2 的不稳定，数值太小会导致算法臃肿.

4.3　3 个改进策略的实验对比

在上文中，D-ITGO 算法使用了 4 个策略，对 ITGO 算法进行改进. 在这 4 个策略中，策略 1(即方程(8))是对细胞坐标的离散化，其目标是将 D-ITGO 算法的细胞坐标映射到离散化的解空间，对算法的收敛速度和计算效率没有影响，因此，只有第 2、3、4 个策略才能影响算法的有效性. 因此，本文将针对策略 2、策略 3、策略 4 分别进行实验，以分别验证 3 个策略的有效性，并将 3 个策略合并，验证其整体的有效性.

原始的 ITGO 算法是基于连续优化问题的算法，不能直接应用在云计算调度问题上，因此，本文首先将 ITGO 算法离散化，并将其定义为 D-ITGO-0. D-ITGO-0 不采取任何改进策略，除了离散化步骤之外，均与原始的 ITGO 算法相同；因此，该算法可以作为本实验中的初始对比算法. 同时，本文把单独使用策略 2(修改生长细胞与入侵细胞的转换效率)的算法定义为 D-ITGO-1，单独使用策略 3(调整死亡细胞的消亡和生长细胞的分裂)的算法定义为 D-ITGO-2，单独使用策略 4(调整步长)的算法定

义为 D-ITGO-3,同时使用 3 个策略的算法定义为 D-ITGO-4,在同一实验条件下进行比较:

本文使用 300 个独立的云任务,在 CloudSim 环境中进行任务调度的模拟实验. 时间延迟用 CloudSim 中的网络时延矩阵的形式表示. 5 个算法分别运行 30 次,其收敛曲线如图 1 所示. 同时,本文将 5 个算法找到的最优策略的云任务的执行时间(Timecost/Makespan)的平均值记录在表 2 上,同时进行对比. 表 2 的第 1 列是指算法名称,第 2 列是指找到的最优调度策略的云任务执行时间的均值,第 3 列是算法找到的最优策略相对于初始算法找到的最优策略的效率提升比例,即 D-ITGO-1 找到的最优策略相对于 D-ITGO-0 找到的平均最优策略的效率提升比例,D-ITGO-2 找到的最优策略相对于 D-ITGO-0 找到的最优策略的效率提升比例等等. 算法运行的结果采用非参数假设检验威尔科克森符号秩检验(Wilcoxon Test)进行统计分析,置信度 α 设为 0.05,并将结果记录为表 3. 表 3 的第 1 列是统计分析的算法对,第 2 列是算法统计分析的结果. N 是指本实验中的云计算任务个数,$N=300$ 即云计算任务个数为 300. 5 个算法运行的时间开销如表 4 所示. 表 4 的第 1 列是算法名称,第 2 列是对应的时间开销. 由于离散化步骤不会改变原始的 ITGO 算法的收敛速度和搜索效率,因此,D-ITGO-0 的收敛速度,可以认为与原始的 ITGO 算法相同.

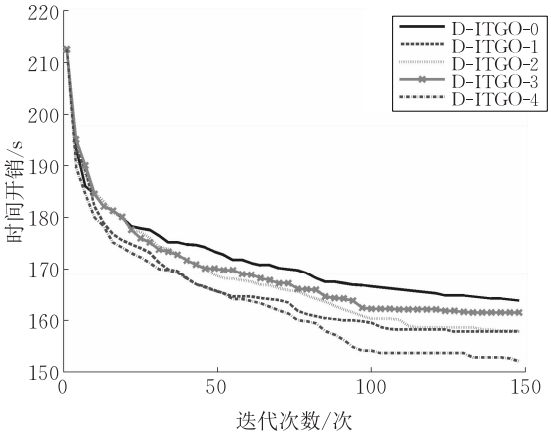

图 1 使用不同策略的 D-ITGO 算法收敛曲线图

表 2 使用不同策略的算法找到的最优调度策略

算法名称	最佳策略的云任务执行时间/s	平均最优策略的效率提升比例/%
D-ITGO-0	164.0012	—
D-ITGO-1	157.7106	3.8
D-ITGO-2	157.8791	3.7
D-ITGO-3	161.0260	1.8
D-ITGO-4	151.8869	7.4

表 3 3 种改进策略的假设检验分析($\alpha=0.05$)

算法对	$N=300$
D-ITGO-0 vs. D-ITGO-1	6.2026e-04
D-ITGO-0 vs. D-ITGO-2	1.8916e-04
D-ITGO-0 vs. D-ITGO-3	0.0091
D-ITGO-0 vs. D-ITGO-4	3.5200e-07

表 4 使用不同策略的算法运行时间开销

算法名称	算法运行的时间开销/s
D-ITGO-0	1.6901
D-ITGO-1	1.2689
D-ITGO-2	1.1386
D-ITGO-3	1.6907
D-ITGO-4	1.0092

从图 1 中可以看出,D-ITGO-1、D-ITGO-2、D-ITGO-3、D-ITGO-4 的收敛曲线均比 D-ITGO-0 要好,且收敛速度均比 D-ITGO-0 要快,因此可以得出结论:3 个策略均对 ITGO 算法的搜索效率有明显的提高,而且找到的最优调度策略均比原始算法要好. D-ITGO-4 的结果最好,收敛速度最快,因此 3 个策略共同使用的效果是最好的. 曲线 D-ITGO-1 和 D-ITGO-2 的结果相似,且均比 D-ITGO-3 的结果要好,因此可以得出结论:策略 2(修改生长细胞与入侵细胞的转换效率)和策略 3(调整死亡细胞的消亡和生长细胞的分裂)的效果比策略 4(调整步长)要好. 策略 2 的效果和策略 3 的效果相似. 但是,由于 D-ITGO-4 的结果远远好于其他 3 条曲线,因此可以得出结论:这 3 个策略之间,存在相互促进和相互提高的效果.

表 2 可以直观地比较各个算法的结果,以及各个算法的效率提升比例. D-ITGO-4 找到的平均最优调度策略,比初始算法 D-ITGO-0 找到的平均最优调度策略,有大约 7.4% 的效率提升,因此算法 D-ITGO-4 明显比初始算法要好. 其余算法均可以此类推.

表 3 表明,在经过 Wilcoxon Test 的统计分析之后,4 个算法的结果远小于置信值 0.05,因此可以认为这 4 个算法有显著的不同. 其中,D-ITGO-4 的显著性差异最为明显,因此可以认为 D-ITGO-4 的结果最好. D-ITGO-2 和 D-ITGO-3 的结果较好,而 D-ITGO-1 的结果在 4 个策略中最差,但依然比原始的 D-ITGO-0 算法要好.

同时,图 1 的收敛曲线均值从侧面证明了表 3 的结论. D-ITGO-4 的最终结果是最好的.

从表 4 中可以看出,使用策略 2 和策略 3 的改进算法,在运行时间均有较大幅度的缩短,因为策略 2 和策略 3 减少了许多不必要的步骤. 策略 4 的运行

时间和原算法基本相同，因为策略 4 改进了步长计算参数，但不改变整个细胞种群的结构，也不改变细胞种群的搜索方式.

为了更进一步地比较各个改进策略在不同任务规模上的差异，本文在基本的模拟实验之外，分别使用 200、400 个随机生成、相互独立的云任务，在上述实验条件下，经过 30 次运行，记录它们找到的最终结果，并使用 Wilcoxon Test 进行统计分析（置信度 $\alpha=0.05$）. 统计分析如表 5 所示. 表 5 的第 1 列是比较的算法对，第 2 列和第 3 列分别是不同任务规模下，不同算法的运行数据的非参数检验统计结果.

表 5　3 种改进策略的假设检验分析（$\alpha=0.05$）

算法对	$N=200$	$N=400$
D-ITGO-0 vs. D-ITGO-1	8.3213e-04	5.1508e-04
D-ITGO-0 vs. D-ITGO-2	2.1021e-04	1.0192e-04
D-ITGO-0 vs. D-ITGO-3	0.0097	0.0089
D-ITGO-0 vs. D-ITGO-4	7.1828e-07	2.2931e-07

表 5 的结果表明，3 个改进策略在不同的任务规模下，均表现良好. 另外，使用 3 个策略的算法 D-ITGO-4 在不同的任务规模下，表现均为最佳. 在不同的任务规模下，策略 2 和策略 3 的表现均优于策略 4，因此可以证明，这 3 个策略的结果具有普遍的意义.

4.4　3 种策略的有效性分析

从 4.3 节的结论中可以看出，D-ITGO 算法的 3 种针对搜索效率和计算速率进行优化的策略中，策略 2（修改生长细胞与入侵细胞的转换效率）和策略 3（调整死亡细胞的消亡和生长细胞的分裂）的表现最好，策略 4（调整步长）同样有效，但效果不如策略 2 和策略 3. 这 3 个策略相互叠加的效果，远远超出了分别使用 3 个策略的效果. 由于整个算法大量使用随机数（见参数列表），且计算结果为 30 次运行的平均值，因此，从概率上看，5 个实验结果的可信度较高. 下面将对这 3 个优化策略生效的原因进行分析：

（1）单独使用策略 4，不会对整个种群的结构和搜索方式有任何改进，仅仅从步长的角度，使算法运行更快、更有效地逼近最优解，其结果是，当算法陷入局部最优时，唯一能跳出局部最优解的方法，是原始的 ITGO 算法所使用的 Levy 飞行. 由于原始的 ITGO 算法在 Levy 飞行的过程中，具有较大的随机性（可以从参数列表中看到，Levy 飞行使用了较多的随机数），因此跳出局部最优解的优势并不明显. 总体而言，策略 4 对整个种群的贡献相对较小，无法彻底解决陷入“当算法陷入局部最优解”的问题，尚有一定的补充改进空间. 所以，单独使用策略 4 的结果，远小于单独使用策略 2 和策略 3；

（2）策略 2 和策略 3. 策略 2 和策略 3 的改进思路，均基于入侵细胞的生成方式. 在入侵肿瘤生长优化算法的整个种群中，入侵细胞承担了部分的搜索工作，以及所有的关于“跳出局部最优解”的工作. 因此，策略 2（见式（9））中，使用了一个“适应度函数差值小于 0.01”的条件判定，当某个细胞本身不属于全局最优解，且该细胞在自己的细胞周期内，搜索效率变得严重低下时，算法将判定该细胞陷入局部最优解，并通过细胞分化（生成一个新的入侵细胞，使用原有的 ITGO 算法中的 Levy 飞行）来跳出局部最优. 该条件判定是跳出局部最优解的基石. 因此，在 4.3 节的实验结果中，策略 2 的效果远远大于策略 4. 此外，在入侵肿瘤生长优化算法的细胞分类中，休眠细胞是搜索效率极慢的一类细胞，而死亡细胞不执行任何搜索策略，因此为了释放计算空间，提高搜索效率，本文使用策略 3（即式（10））直接释放死亡细胞所占据的空间，并直接生成搜索效率最高的入侵细胞. 因此，在 4.3 节的实验结果中，策略 3 能极大地提高算法的搜索效率. 而在细胞初始化时，为了最大程度地削减计算开销，入侵细胞和死亡细胞所占据的空间为 0，同样节省了一定的计算时间.

由于策略 2 可以更有效地跳出局部最优解，策略 3 可以极大地提升搜索效率，策略 4 可以在逼近最优解时，一定程度上提高搜索效率，因此在 4.3 节的实验结果中，D-ITGO-1 和 D-ITGO-2 的效果是最好的，D-ITGO-3 的效果稍次于前两者，但不执行任何改进策略的 D-ITGO-0 算法效果好. 3 个策略可以在一定程度上相互促进、相互提高，因此，使用 3 个改进策略的效果（D-ITGO-4）是最好的.

4.5　D-ITGO 算法和其他算法的对比

为了进一步验证 D-ITGO 算法的运行效果，本文使用 8 个额外的对比算法：遗传算法（GA）、粒子群算法（PSO）、人工蜂群算法（ABC）、基于模糊策略的遗传算法（FUGE）、鲶鱼粒子群优化算法（CPSO）、布谷鸟搜索算法（Cuckoo）、蝙蝠搜索算法（BA）、猫群优化算法（CSO），与本文提出的 D-ITGO 算法进行对比. D-ITGO 算法将同时使用上文中提到的 3 个对比策略，即 D-ITGO-4，并使用表 1 中的参数进行运算. 其余 8 个算法的参数配置如 4.2 节所示. 在本实验中，D-ITGO-4 在图标中均表示为 D-ITGO.

同时，本文使用 300 个独立的云任务，在 CloudSim

环境中进行任务调度的模拟实验.实验分别记录9个算法的30次运行结果,并记录其平均收敛速度,将结果绘制成图2.此外,本文将9个算法找到的最优策略的云任务执行时间(Makespan)的平均值记录在表6上,同时进行对比.表6的第1列是指算法名称,第2列是指找到的最优调度策略的云任务执行时间均值,第3列是指D-ITGO算法找到的最优策略相对于其他算法找到的最优策略的效率提升比例.另外,本文将30次运行的数据采用非参数假设检验Wilcoxon Test进行统计分析(置信度$\alpha=0.05$),统计其显著性差异,并将结果记录为表7.表7的第1列是进行比较的算法对,第2列是不同算法的运行数据的非参数检验统计结果.表8记录了9个算法在30次运行中的平均运行时间.表格的第1列为算法名称,第2列为实验记录.

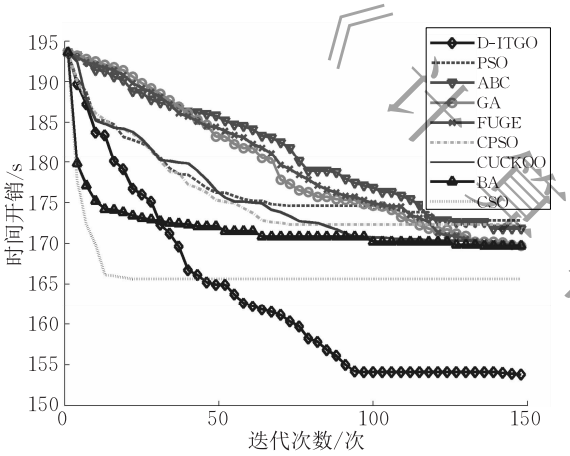

图2 D-ITGO和其他算法的收敛曲线图

表6 9种算法找到的最优调度策略

算法名称	最佳策略的云任务执行时间/s	平均最优策略的效率提升比例/%
D-ITGO	153.7818	—
PSO	173.2202	11.2
ABC	171.2074	9.0
GA	169.4800	9.3
CUCKOO	172.8963	11.1
FUGE	168.3210	8.6
CPSO	169.0298	9.0
BA	169.4708	9.3
CSO	166.0908	7.1

表7 9种算法的假设检验分析($\alpha=0.05$)

算法对	$N=300$
D-ITGO vs. PSO	7.1185e-09
D-ITGO vs. ABC	8.4847e-09
D-ITGO vs. GA	7.2208e-06
D-ITGO vs. CUCKOO	1.8731e-07
D-ITGO vs. FUGE	1.6743e-05
D-ITGO vs. CPSO	1.2540e-08
D-ITGO vs. BA	1.4298e-05
D-ITGO vs. CSO	2.2539e-04

表8 9种算法的运行时间开销

算法名称	算法运行的时间开销
D-ITGO	1.0119
PSO	0.6061
ABC	1.8219
GA	1.2510
CUCKOO	5.3477
FUGE	1.2184
CPSO	0.6224
BA	2.6339
CSO	9.8896

从图2中可以看出,D-ITGO算法的收敛曲线比其他8种算法的收敛曲线要好.收敛曲线的末端表明,在同样的迭代次数下,D-ITGO算法找到的平均最优解比其他算法找到的平均最优解更好.此外,从图2的收敛曲线交点中可以看出,在大约30次迭代之前,D-ITGO算法的收敛速率略小于蝙蝠搜索算法(BA)和猫群优化算法(CSO),但远大于遗传算法(GA)、粒子群算法(PSO)、人工蜂群算法(ABC)和布谷鸟搜索算法(Cuckoo).在30次迭代之后,D-ITGO的收敛曲线远远超出了蝙蝠算法(BA)和猫群优化算法(CSO).因此可以得出结论:在大部分的情况下,D-ITGO的效果远比其他算法更好.

从表6中可以看出,D-ITGO算法找到的最优调度策略,在均值上比其他算法找到的最优调度策略,有7.1%~11.2%不等的时间优势.

从表7中可以看出,D-ITGO算法找到的最优调度策略,与其他算法找到的最优调度策略,均有显著性差异.其中,与ABC算法和PSO算法的显著性差异最大,与CSO的显著性差异最小.因此可以得出结论:D-ITGO算法比其他算法要好.

从表8的运行时间开销中可以看出,虽然蝙蝠算法(BA)和猫群优化算法(CSO)在前30次迭代中的收敛速率更好,但由于时间开销太大,因此从整体性能上看,D-ITGO算法优于蝙蝠算法(BA)和猫群优化算法(CSO).尤其是猫群优化算法(CSO),其时间开销在9种算法中是最高的,远远超出了适用范围.

此外,从表8中可以看出,虽然D-ITGO算法的运行开销比粒子群算法(PSO)和鲶鱼粒子群算法(CPSO)要大,但比其他的启发式算法要小,因此D-ITGO算法在运行时间开销上,同样具有一定的优势.同时结合图2可以得出结论:D-ITGO算法的收敛速度以及最终的实验结果,远远优于粒子群算法和鲶鱼粒子群算法,因此就总体效果而言,D-ITGO算法优于上述两种算法.

另外，D-ITGO 算法比其他 3 种算法：人工蜂群算法（ABC）、遗传算法（GA）和基于模糊理论的遗传算法（FUGE），在最终结果和时间开销上，均有一定的优势.

为了更进一步地比较各个算法在不同任务规模上的差异，我们分别使用 200、400 个随机生成、相互独立的云任务，在上述实验条件下，经过 30 次运行，将最终结果进行记录并使用 Wilcoxon Test 进行统计分析（置信度 $\alpha=0.05$）. 统计分析如表 9 所示. 表 9 的第 1 列是算法对，第 2 列和第 3 列分别是不同任务规模下，不同算法的运行数据的非参数检验统计结果.

表 9　9 种算法的假设检验分析（$\alpha=0.05$）

算法对	$N=200$	$N=400$
D-ITGO vs. PSO	8.9817e-09	1.7208e-10
D-ITGO vs. ABC	9.8201e-09	2.8532e-09
D-ITGO vs. GA	1.1012e-05	4.2441e-06
D-ITGO vs. CUCKOO	9.5498e-07	1.0093e-08
D-ITGO vs. FUGE	5.3044e-05	8.9972e-06
D-ITGO vs. CPSO	2.7333e-08	6.0017e-09
D-ITGO vs. BA	4.1821e-05	9.9813e-06
D-ITGO vs. CSO	7.0417e-04	2.0782e-04

表 9 可以直观地比较各个策略在不同任务规模下，找到的最佳调度策略的统计性差异. 表 9 的结果佐证了前文的结论，证明 D-ITGO 算法在不同的任务规模下均表现良好，且该算法在更大的任务规模下，效果更为明显.

综上，D-ITGO 算法在云计算调度策略和平均运行时间上都有一定的优势.

5　结论和展望

云计算任务调度是云计算研究中的一个重要内容，同时也是一个 NP-hard 问题. 目前，国内外的许多学者都针对该问题，提出了相应的解决策略和算法，以完成该问题中的各种调度目标（如截止时间限制，负载均衡等），同时满足各种限制条件（如时间开销、延迟开销、QoS 等等）. 这些算法包括非启发式算法和启发式算法. 入侵肿瘤生长优化算法（Invasive Tumor Growth Optimization）是近年来提出的一种新的启发式算法，同时也是一种群智能算法，该算法通过模拟肿瘤细胞的生长、入侵、休眠、死亡等行为，在函数解空间中寻找最优解，具有较高的搜索速度和精确度. 本文对 ITGO 算法做出了一定的改进和提高，并将其应用在云计算任务调度问题上. 改进策略包括：修改入侵-生长细胞转换策略以及修改死亡-入侵细胞转换策略，以避免计算资源的浪费，并针对 ITGO 算法中的步长进行调优，使之具有更高的搜索速率. 实验结果和分析表明，新算法比目前常用的和最新的启发式算法例如遗传算法（GA）、粒子群算法（PSO）、人工蜂群算法（ABC）、布谷鸟搜索算法（Cuckoo）、蝙蝠搜索算法（BA）、猫群优化算法（CSO），在云任务执行时间 Makespan 和网络延迟 Latency 上均有提升，在调度开销上也有一定的优势.

在未来的工作中，我们将会对提出的 D-TIGO 算法进行更为深入细致的研究，并将其应用在更多的问题上，例如虚拟机迁移（Virtual Machine Consolidation）、云安全（Security）、云计算中的多目标优化（Multi-objective）等等.

参　考　文　献

[1] Tang D, Dong S, He L, et al. Intrusive tumor growth inspired optimization algorithm for data clustering. Neural Computing and Applications, 2016, 27(2): 349-374

[2] Tang D, Dong S, Jiang Y, et al. ITGO: Invasive tumor growth optimization algorithm. Applied Soft Computing, 2015, 36(Supplement C): 670-698

[3] Gao M, Hao L. An improved algorithm based on Max-Min for cloud task scheduling//Qian Zhihong, Cao Lei, Su Weilian, et al, eds. Recent Advances in Computer Science and Information Engineering. Berlin, Germany: Springer, 2012: 217-223

[4] Chen H, Wang F, Helian N, et al. User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing//Proceedings of the Parallel Computing Technologies. Bangalore, India, 2013: 1-8

[5] Ergu D, Kou G, Peng Y, et al. The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. The Journal of Supercomputing, 2013, 64(3): 835-848

[6] Zhang M, Yang Y, Mi Z, et al. An improved genetic-based approach to task scheduling in inter-cloud environment// Proceedings of the IEEE International Conference on Cloud and Big Data Computing. Beijing, China, 2015: 997-1003

[7] Kaur S, Verma A. An efficient approach to genetic algorithm for task scheduling in cloud computing environment. International Journal of Information Technology and Computer Science, 2012, 4(10): 74

[8] Adil S H, Raza K, Ahmed U, et al. Cloud task scheduling using nature inspired meta-heuristic algorithm//Proceedings of the 2015 International Conference on Open Source Systems & Technologies. Lahore, Pakistan, 2015: 158-164

[9] Tawfeek M A, El-Sisi A, Keshk A E, et al. An ant algorithm for cloud task scheduling//Proceedings of the International Workshop on Cloud Computing and Information Security. Hangzhou, China, 2013: 169-172

[10] Mondal B, Dasgupta K, Dutta P. Load balancing in cloud computing using stochastic hill climbing—A soft computing approach. Procedia Technology, 2012, 4(Supplement C): 783-789

[11] Russell S, Norvig P. Artificial Intelligence: A Modern Approach. 3rd Edition. Beijing: Pearson Publication, 2010

[12] Shojafar M, Javanmardi S, Abolfazli S, et al. FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Computing, 2015, 18(2): 829-844

[13] Xu A, Yang Y, Mi Z, et al. Task scheduling algorithm based on PSO in cloud environment//Proceedings of the IEEE International Conference on Cloud and Big Data Computing. Beijing, China, 2015: 1055-1061

[14] Zhao S, Lu X, Li X. Quality of service-based particle swarm optimization scheduling in cloud computing//Proceedings of the 4th International Conference on Computer Engineering and Networks. Shanghai, China, 2015: 235-242

[15] Nishant K, Sharma P, Krishna V, et al. Load balancing of nodes in cloud using ant colony optimization//Proceedings of the 2012 UKSim 14th International Conference on Computer Modelling and Simulation. Cambridge, UK, 2012: 3-8

[16] Li K, Xu G, Zhao G, et al. Cloud task scheduling based on load balancing ant colony optimization//Proceedings of the 2011 6th Annual Chinagrid Conference. Liaoning, China, 2011: 3-9

[17] Yao J, He J. Load balancing strategy of cloud computing based on artificial bee algorithm//Proceedings of the 2012 8th International Conference on Computing Technology and Information Management (ICCM). Seoul, South Korea, 2012: 185-189

[18] Shu W, Wang W, Wang Y. A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. EURASIP Journal on Wireless Communications and Networking, 2014, (1): 64

[19] Kumar P, Verma A. Scheduling using improved genetic algorithm in cloud computing for independent tasks//Proceedings of the International Conference on Advances in Computing, Communications and Informatics. Chennai, India, 2012: 137-142

[20] Kruekaew B, Kimpan W. Virtual machine scheduling management on cloud computing using artificial bee colony//Proceedings of the International MultiConference of Engineers and Computer Scientists. Hong Kong, China, 2014: 18-22

[21] Nirmala S J, Bhanu S M S. Catfish-PSO based scheduling of scientific workflows in IaaS cloud. Computing, 2016, 98(11): 1091-1109

[22] Navimipour N J, Milani F S. Task scheduling in the cloud computing based on the Cuckoo search algorithm. International Journal of Modeling and Optimization, 2015, 5(1): 44

[23] Raghavan S, Sarwesh P, Marimuthu C, et al. Bat algorithm for scheduling workflow applications in cloud//Proceedings of the 2015 International Conference on Electronic Design, Computer Networks & Automated Verification. Shillong, India, 2015: 139-144

[24] Bilgaiyan S, Sagnika S, Das M. Workflow scheduling in cloud computing environment using cat swarm optimization//Proceedings of the 2014 IEEE International Advance Computing Conference. Gurgaon, India, 2014: 680-685

[25] Wolpert D H, Macready W G. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67-82

[26] Mocanu E M, Florea M, Andreica M I, et al. Cloud computing—Task scheduling based on genetic algorithms//Proceedings of the 2012 IEEE International Systems Conference. Vancouver, Canada, 2012: 1-6

[27] Abdi S, Motamedi S A, Sharifian S. Task scheduling using modified PSO algorithm in cloud computing environment//Proceedings of the International Conference on Machine Learning, Electrical and Mechanical Engineering. Dubai, United Arab Emirates, 2014: 8-9

ZHOU Jing, born in 1992, Ph. D. candidate. Her research interests include cloud computing and distributed computing.

DONG Shou-Bin, born in 1967, Ph. D., professor. Her research interests include high performance computing, cloud computing and information retrieval.

TANG De-Yu, born in 1978, Ph. D., associate professor. His research interests include intelligent computing and bioinformatics.

Background

The problem researched by this paper belongs to the inter-discipline of cloud computing and intelligence computing, including the cloud task scheduling problem in cloud computing and the application of intelligent algorithms. These years the

research of cloud task scheduling includes: (1) using non-heuristic algorithms, such as Max-min, Min-min and FCFS to schedule the tasks; and (2) using heuristic algorithms, such as genetic algorithm, particle swarm optimization, and etc., to schedule the tasks. The application of intelligent algorithms includes: (1) applications in mathematics (such as solving continuous optimization problem); and (2) applications in industrial (such as solving the Job Shop Scheduling Problem and logistics scheduling problems); (3) applications in data-mining (such as to train to parameters in Neural Network). This paper combined the task scheduling problem in cloud computing with the application of intelligent algorithm. It means, which used the invasive tumor growth optimization to solve the cloud task scheduling problem, keeps up with the main research direction in the world.

This work belongs to the research subject: Research on Big Data Management of Biological Gene Based on Cloud Computing, supported by the Foundation of the Natural Science Foundation of Guangdong Province (No. 2015A030308017, No. 2016A030310300). In the past few years, we usually used the ant colony algorithm, genetic algorithm and other proposed algorithms to solve cloud task scheduling problem and applied ITGO to other fields such as data-mining and support vector machine. This paper firstly combined the cloud task scheduling problem with the invasive tumor growth optimization and got a good result. This result can solve the key problem of cloud computing of Research on Big Data Management of Biological Gene Based on Cloud Computing.