一种改进樽海鞘群算法及其多目标云工作流调度应用

李 果[1] 陈 信[2] 吴迎来[3]*

[1](安徽新闻出版职业技术学院 安徽 合肥 230601)

[2](杭州电子科技大学计算机学院 浙江 杭州 310018)

[3](杭州电子科技大学创业学院 浙江 杭州 310018)

摘 要 为了优化云工作流应用的调度效率与代价,提出基于改进樽海鞘群算法的工作流调度策略。建立截止时间与预算约束的多目标优化模型,利用樽海鞘觅食的位置变化对工作流调度进行编解码,设计融合执行跨度与执行代价的权重适应度函数。为了增强樽海鞘群的寻优性能,引入基于疯狂算子的领导者更新模式,通过疯狂变量,减少领导者更新的停滞早熟现象;引入遗传算子的追随者更新模式,利用个体交叉和变异使樽海鞘群具有更均衡的搜索全局性和个体多样性,避免陷入局部最优。结果证明,改进樽海鞘群算法可以有效提升计算精度和收敛速度;应用于工作流调度求解后,其调度解收敛性更好,调度解集空间分布更加一致。

关键词 云计算 工作流调度 樽海鞘群算法 疯狂算子 遗传算子 多目标优化

中图分类号 TP393 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2023.12.039

AN IMPROVED SALP SWARM ALGORITHM AND IT'S APPLICATION ON MULTI-OBJECTIVE CLOUD WORKFLOW SCHEDULING

Li Guo[1] Chen Xin[2] Wu Yinglai[3]*

[1](*Anhui Vocational College of Press and Pubilishing*, *Hefei 230601*, *Anhui*, *China*)

[2](*School of Computer*, *Hangzhou Dianzi University*, *Hangzhou 310018*, *Zhejiang*, *China*)

[3](*Institute of Innovation and Entrepreneurship*, *Hangzhou Dianzi University*, *Hangzhou 310018*, *Zhejiang*, *China*)

Abstract For optimizing scheduling efficiency and cost of cloud workflow applications, we propose a workflow scheduling strategy based on improved salp swarm algorithm. A multi-objective scheduling optimization model meeting the deadline and budget was built. We used the location changes of salps foraging to code and decode the workflow scheduling. We designed a weighted fitness function intergrating the execution makespan and the execution cost. In order to enhance the performance of salp swarm, the algorithm introduced a leader update model based on crazy operator. By this crazy variable, we could reduce stagnation precocious phenomenon in leader update. This algorithm also introduced a follower update model based on genetic operators, which could use the individual crossover and mutation to make salp group having a more balanced global search and individual diversity. Experiments prove that the improved algorithm can promote the accuracy and convergence rate. When solving the workflow schedule problem, the new algorithm has a faster convergence rate and more uniform space distribution.

Keywords Cloud computing Workflow scheduling Salp swarm algorithm Cray operator Genetic operator Multi-objective optimization

0 引 言

工作流调度多用于科学计算领域,在天文学、遗传学和生物信息学等领域得到广泛应用。工作流任务由大量串行和并行任务组成,计算量更大、结构更加复杂、对执行环境要求更高。工作流任务调度的本质是需要将相互依存的若干子任务调度到服务资源上执

收稿日期:2020-09-09。安徽省教育厅自然科学重点研究项目(KJ2017A870);安徽省教育厅省级融媒体采编教学创新团队项目(2019cxtd101)。**李果**,副教授,主研领域:计算机图形学,智能优化算法。**陈信**,讲师。**吴迎来**,实验师。

行,调度目标是在满足用户定义的相关服务质量约束的前提下,以预定义的目标函数完成所有子任务的执行[1]。传统的计算模式中,工作流调度只注重于调度效率,即以优化时间为目标,多以任务截止时间为限制[2],而没有考虑服务利用的代价以及任务提交用户本身的费用约束问题。云环境不同于传统的分布式计算环境,其工作流调度需要考虑的因素更多也更复杂[3]。首先,用户对于服务质量的需求更多样,通常会同步考虑调度效率与调度代价;其次,云资源提供具有异质、动态和弹性特征,导致任务执行更受限制,执行期限与执行预算约束需要在资源选择上同步予以考虑。云计算资源的处理能力及利用代价各有不同,执行性能越强,相对利用代价也越高,不同方法的调度结果对应的执行时间和代价也不同。综上,云工作流调度需同步考虑截止期限和预算约束,确保双重约束下的工作流任务顺利执行,这样更加符合云环境下工作流调度的场景。

然而,调度时间和调度代价之间的同步优化问题本身是个冲突问题。执行时间的降低、调度效率的提升,需要将任务调度至具备更强计算性能的云资源上执行。然而,资源性能的提升则代表着相应执行代价的增加,单纯提升调度效率可能导致工作流执行代价无法满足预算约束。反过来,单纯降低执行代价,可将任务调度至廉价资源上执行,而代价低代表着性能也越低,此时生成的调度方案则可能无法满足截止时间约束。本文提出一种基于改进樽海鞘群算法的云工作流调度算法,实现多约束条件下的多目标调度优化。算法分别利用疯狂算子和遗传算子改进了樽海鞘群领导者和追随者的寻优性能,最终求解的工作流调度解收敛性更好,调度解集空间分布更加一致,性能优于对比算法。

1 相关研究

文献[4]基于遗传算法设计了期限与预算约束的云工作流调度算法,仅单一地对执行代价或时间进行了优化。文献[5]设计了一阶段算法 IC-PCP 和两阶段算法 IC-PCPD2,可以较低的时间复杂度得到满足期限约束的最小代价调度解。文献[6]设计了混合云环境中的期限约束最小化代价调度算法,通过引入子期限进行资源重调度,实现代价最小。文献[7]则以包任务的形式实现了期限约束下的代价最小化调度,也仅可适用于独立非关联式的任务调度。一些工作以多目标优化为基础进行了研究。文献[8]为了实现执行效率与执行代价的同步优化,设计了基于遗传算法的多目标最优化工作流调度算法 MOEGA,但未考虑任何服务质量约束。文献[9]引用非占优排序设计了混合多目标粒子群优化算法 HPSO,实现了调度时长、代价和能耗的优化。文献[10]基于帕累托占优的调度算法,实现了效率与代价的优化,但算法没有引入预算约束。文献[11]为了实现能耗与时间的均衡调度,提出基于模糊逻辑的调度算法。但考虑的是虚拟机调度,不是云工作流调度。文献[12]利用模糊聚类方法对云资源分组,再以极小极大思想建立工作流调度模型,实现了执行时间和代价的降低。问题在于算法没有优化资源实例选择,其出现约束违例情况较多。文献[13]利用蚁群算法求解不同 QoS 偏好下的工作流调度,利用可变的启发式规则,优化了执行时间和能耗。但算法没有考虑约束问题,调度方案在实际环境中不一定可行。文献[14]设计了双约束 BHEFT 算法,但考虑的环境资源相对固定,若有资源预留情况,会导致部分用户无法按期完成任务,调度成功率不高。综合考虑以上研究,提出一种基于改进樽海鞘群算法的工作流调度策略,在截止时间和预算的双重约束下,通过调度任务选择和最优资源实例选择机制进行均衡的多目标调度任务。

2 系统模型

云任务类型通常为有向无环图模型,表示为 $G(T,E)$,T 为任务顶集合,表示为 $T=\{t_1,t_2,\cdots,t_n\}$;E 为有向边集,边 $(t_i,t_j)\in E$ 代表 t_i 与 t_j 间的执行顺序。若存在边 (t_i,t_j),则表示 t_i 为 t_j 的父任务,t_j 为 t_i 的子任务。图 1 为由 8 个任务组成的工作流,t_1 为入口,t_8 为出口。父任务完成后将数据发送给子任务,子任务接收数据后开始执行。

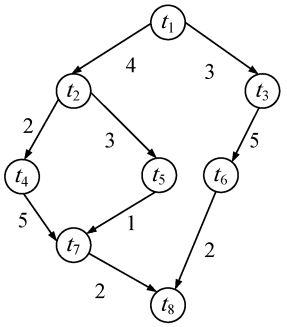

图 1 工作流应用示例

将云计算资源表示为 $R=\{r_1,r_2,\cdots,r_m\}$。所有资源为全连通拓扑结构,资源处理能力与执行代价都相同。图 2 为四个资源的拓扑。以每秒执行的百万指令数 MIPS 表示一个资源 $r_j\in R$ 的处理能力 C_j。资源实例的利用代价选择 Amazon EC2 的代价模型,用户按资

源使用的时间周期数付费,若资源利用不足一个时间周期单位,仍按单个时间周期支付。

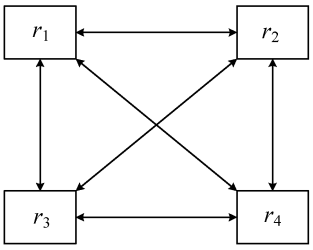

图 2　资源拓扑结构

以百万指令数 MI 定义单个任务的大小,表示为 S_i,即任务 t_i 的大小。则工作流任务 t_i 在资源 r_j 上的执行时间 $ET(t_i,r_j)$ 可表示为任务大小除以资源处理能力:

$$ET(t_i,r_j)=S_i/C_j \tag{1}$$

工作流任务 t_i 在资源 r_j 上的执行代价 $EC(t_i,r_j)$ 表示为:

$$EC(t_i,r_j)=\mu_j\times ET(t_i,r_j) \tag{2}$$

式中:μ_j 表示资源 r_j 的使用单价。

实验中仅考虑调度于不同资源实例上的两个依赖任务间的数据传输时间为 c_t。令 $EST(t_i,r_j)$ 为 t_i 在资源 r_j 上的最早开始时间,$EFT(t_i,r_j)$ 为 t_i 在 r_j 上的最早完成时间。对于 t_{entry} 有:

$$EST(t_{\text{entry}},r_j)=Avail(r_j) \tag{3}$$

式中:$Avail(r_j)$ 为资源 r_j 的可用资源量。

对于除了入口任务的工作流 DAG 中的其他任务,其 EST 和 EFT 计算为:

$$EST(t_i,r_j)=\max\{Avail(r_j),\max_{t_p\in pred(t_i)}\{AFT(t_i)+ct\}\} \tag{4}$$

$$EFT(t_i,r_j)=ET(t_i,r_j)+EST(t_i,r_j) \tag{5}$$

式中:$pred(t_i)$ 为 t_i 的父任务集;$Avail(r_j)$ 为 r_j 的就绪时间;$AST(t_i,r_j)$ 为 t_i 在 r_j 上的实际开始时间,$AFT(t_i,r_j)$ 为 t_i 在 r_j 上的实际完成时间。工作流应用的执行跨度 M_{akespan} 等于 t_{exit} 的实际完成时间,定义为:

$$M_{\text{akespan}}=AFT(t_{\text{exit}}) \tag{6}$$

工作流总体执行代价为所有任务执行代价之和,表示为:

$$E_{\text{C}}=\sum_{i=1}^{n}EC(t_i,r_j) \tag{7}$$

该系统模型的特征在于:以有向无环图表示的工作流模型可以明确工作流结构中所有任务间的偏序关系,即前驱与后继关系,以此约束任务间的执行次序。而工作流结构中的任务入口与出口在有向无环图中也可明确定义。基于有向无环图模型,工作流调度算法设计对于任务调度选择也更加便利。此外,全连通拓扑结构的云资源模型使得工作流任务映射至相应分配资源后,任务间的数据输出可以在云资源间实现无缝传输,从而更好地解决工作流调度与相应资源的映射问题。

3　工作流调度的优化目标

给定集合 T 和 R,调度目标即为产生调度解集 S,在满足给定的截止时间 D_{eadine} 和预算约束 B_{udget} 的约束条件下,使任务的总体执行跨度和总执行代价达到最小。因此,工作流调度的多目标优化问题可形式化为:

$$\min(M_{\text{akespan}},E_C) \tag{8}$$

$$\text{s. t. } M_{\text{akespan}}<D_{\text{eadline}}\text{ 和 }E_{\text{C}}<B_{\text{udget}}$$

式(8)的多目标优化模型的构建依据在于:对于实际的云计算环境而言,工作流调度复杂于常规分布式计算环境。用户的 QoS 需求呈现多样化,会同步考虑调度效率与调度代价。同时,云资源提供具有异质、动态和弹性特征,从用户角度出发,执行期限与执行预算约束在资源选择上需要同步考虑。云计算资源的处理能力及利用代价各有不同,调度算法对任务和资源间进行不同匹配时会得出完全不同的调度效率与调度代价。因此,此时的工作流调度将是一种多约束多目标优化问题,这更加符合云环境工作流调度应用场景。

4　樽海鞘群算法

樽海鞘群算法 SSA 是 2017 年才首次提出的最新群体智能算法[15]。樽海鞘是一种生活于海洋的生物,身体为透明水桶型,与水母相似。樽海鞘种群个体可分领导者和追随者。群首为领导者,它在寻找食物源过程中拥有最优判断,引导整个种群移动。除领导者外的樽海鞘均为追随者。追随者相互跟随,相互直接或间接领导。食物源即为种群搜索目标。在每次迭代中,樽海鞘会按适应性排序,每个个体都紧随前一个体移动,而非所有个体都只向着当前最优个体移动,这极大降低了搜索过程陷入局部最优的概率,因此其性能优于粒子群算法、遗传算法、灰狼算法等。此外,SSA 的控制参数相对更少,大大减少了算法对控制参数的依赖。以下对 SSA 的寻优过程作详细描述:

SSA 中,樽海鞘种群由 N 个维度为 n 的个体构成,个体位置矢量 $\boldsymbol{X}$ 相当于 n 维搜索空间,n 为决策变量个数。因此,樽海鞘种群由以下矩阵构成:

$$\boldsymbol{X}=\begin{bmatrix}x_{1,1} & x_{1,2} & \cdots & x_{1,n}\\ x_{2,1} & x_{2,2} & \cdots & x_{2,n}\\ \vdots & \vdots & & \vdots\\ x_{N,1} & x_{N,2} & \cdots & x_{N,n}\end{bmatrix} \tag{9}$$

对于 SSA 而言,追踪食物源是所有樽海鞘个体的目标。因此,领导者的位置更新方式为:

$$x_{1,j}=\begin{cases}F_j+c_1\times[(u_{b_{j,\max}}-l_{b_{j,\min}})\times c_2+l_{b_{j,\min}}] & c_3\geqslant 0.5\\ F_j-c_1\times[(u_{b_{j,\max}}-l_{b_{j,\min}})\times c_2+l_{b_{j,\min}}] & c_3<0.5\end{cases} \tag{10}$$

式中:$x_{1,j}$表示作为领导者的第一只樽海鞘 j 维位置;F_j 表示食物源在 j 维位置;$u_{b_{j,\max}}$表示搜索维度 j 上的最大值;$u_{b_{j,\min}}$表示搜索维度 j 上的最小值;c_2、c_3 为[0,1]内随机数;c_1 为收敛因子,用于控制樽海鞘个体的局部开发和全局勘探能力。式(10)表明,领导者的位置更新主要由食物源的位置决定。收敛因子 c_1 定义为:

$$c_1=2e^{-\left(\frac{4t}{T_{\max}}\right)^2} \tag{11}$$

式中:t 为当前迭代;$T_{\max}$为最大迭代。

追随者的位置根据牛顿运动定理进行更新,表示为:

$$x_{i,j}=\frac{1}{2}a\Delta t^2+v_0\Delta t \tag{12}$$

式中:$x_{i,j}$为追随者 j 维位置,$i\geqslant 2$;Δt 为时间;v_0 初始速率;加速度 $a=(v_{\text{final}}-v_0)/\Delta t$,$v_{\text{final}}=(x_{i-1,j}-x_{i,j})/\Delta t$,$x_{i-1,j}$为樽海鞘个体 $i-1$ 的 j 维位置。时间是迭代数之差,$\Delta t=1$。而 $v_0=0$,因此,追随者更新为:

$$x_{i,j}=\frac{1}{2}(x_{i,j}+x_{i-1,j}) \tag{13}$$

式(13)表明:追随者 i 第 j 维位置更新等于上一迭代中该樽海鞘对应的位置与追随者樽海鞘 $i-1$ 中第 j 维位置之和的一半。

5 基于疯狂算子和遗传算子的樽海鞘群工作流调度算法

5.1 樽海鞘位置编码与调度解码

一个樽海鞘个体的位置即代表一种工作流任务调度可行解。令樽海鞘个体 h 的位置为 n 维矢量,定义为:

$$\boldsymbol{X}_h=[x_{h,1},x_{h,2},\cdots,x_{h,n}] \tag{14}$$

式中:位置矢量中元素 $x_{h,k}$表示樽海鞘 h 在维度 k 上的位置,n 表示任务总量,$h=1,2,\cdots,N$,N 为樽海鞘种群规模。

樽海鞘个体位置的编码及相应调度解的解码步骤为:首先,在初始情况下,在个体位置的每个维度上生成[0,1]间均匀分布的随机数,将该随机数序列作为初始个体矢量位置;然后,对个体位置的离散数值编码作降序排列;最后,根据生成的降序排列,解码相应调度资源,具体方式为:令 n 为工作流任务数量,m 为可用资源数量,$m<<n$,令 $z=\text{INT}(n/m)$,INT 表示取整运算,以 z 表示资源分隔,即以 z 个任务一组按序将任务调度至相应序号的资源上。

以图 3 为例,工作流任务数 $n=10$,资源实例数 $m=3$。序列 1 代表樽海鞘个体位置 $X=(0.39,0.35,0.75,0.89,0.19,0.52,0.26,0.58,0.79,0.64)$,序列 2 代表对位置作降序排列后的相应序列,序列 3 为调度资源。由于 $z=\text{INT}(10/3)=3$,所以排序 1、2、3 的资源为 r_1,排序 4、5、6 的资源为 r_2,依此类推。无法整除的排序 10 对应的任务随机分配调度资源。因此,该樽海鞘个体位置所解码出的任务调度解为:t_1、t_2、$t_7\rightarrow r_3$,t_3、t_4、$t_9\rightarrow r_1$,t_5、t_6、t_8、$t_{10}\rightarrow r_2$。

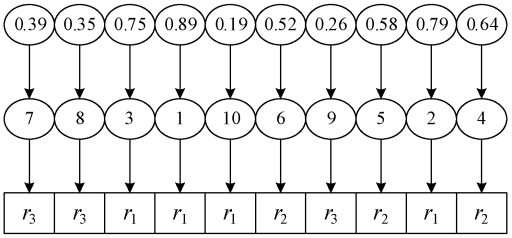

图 3 个体位置的编码及调度解的解码

5.2 基于疯狂算子的领导者位置更新

SSA 中,种群的食物源位置决定了樽海鞘个体的移动方向,会引导樽海鞘种群向最优个体移动。当领导者陷入局部最优时,种群搜索会出现停顿。由于樽海鞘在向着食物源移动过程中,食物源位置存在变化的可能,这种意外行为对于樽海鞘的食物源的追踪是较为合理的。综合考虑,算法引入疯狂算子建立领导者位置更新模式,通过疯狂变量,减小传统领导者位置更新可能导致领导者停滞早熟现象。具体做法是:在领导者位置更新方式中,引入疯狂算子对食物源位置进行扰动,确保一定疯狂概率下维持樽海鞘种群的多样性,避免搜索过程陷入早熟收敛。引入疯狂算子后,领导者位置更新为:

$$x_{1,j}=\begin{cases}F_j+P(c_4)\times\text{sign}(c_4)\times x_{\text{crazy}}+\\ c_1\times[(u_{b_{j,\max}}-l_{b_{j,\min}})\times c_2+l_{b_{j,\min}}] & c_3\geqslant 0.5\\ F_j+P(c_4)\times\text{sign}(c_4)\times x_{\text{crazy}}-\\ c_1\times[(u_{b_{j,\max}}-l_{b_{j,\min}})\times c_2+l_{b_{j,\min}}] & c_3<0.5\end{cases} \tag{15}$$

式中:c_4 为[0,1]内的随机数;x_{crayz}为极小值。$P(c_4)$和 $\text{sign}(c_4)$分别定义为:

$$P(c_4)=\begin{cases}1 & c_4<P_{\text{crazy}}\\ 0 & \text{其他}\end{cases} \tag{16}$$

$$\text{sign}(c_4)=\begin{cases}-1 & c_4\geqslant 0.5\\ 1 & \text{其他}\end{cases} \tag{17}$$

式中:P_{crazy}表示疯狂概率。

樽海鞘初始种群的个体位置通过随机方式产生,即个体位置属于区间[0,1]内的随机值。而根据式

(15)的基于疯狂算子的领导者位置更新方式并不能保证位置更新仍然保持在[0,1]区间内,这样无法进一步进行图 3 所示的调度解解码。因此,如果领导者樽海鞘的新的更新位置中任一维度不在[0,1]区间内,需要对位置进行转换。引入以下转换函数进行位置转换:

$$x_{1,j}=\begin{cases}|x_{1,j}| & x_{1,j}<0\\ x_{1,j} & 0\leqslant x_{1,j}\leqslant 1\\ x_{1,j}-1 & x_{1,j}>1\end{cases} \tag{18}$$

5.3　基于遗传算子的追随者位置更新

对于传统的 SSA 而言,所有追随者位置更新均只考虑与其相邻的个体,为了兼顾随机性和寻优性能,引入遗传算子对一半追随者位置进行扰动,保证搜索全局性和个体多样性,从而改善搜索过程易于陷入局部最优的不足。具体地,引入遗传交叉算子和变异算子对追随者位置进行更新。

1)遗传交叉。为了改进追随者个体的全局搜索能力,在进化迭代过程中引入单个体交叉和双个体交叉两种交叉算子。

单个体交叉。单个体交叉随机选择父个体 P 的位置的两个维度进行交换,然后重新对所有维度位置进行排序,并重新解码调度解,得到子代个体 C。若子代个体 C 的适应度小于父代个体 P,则保留 C 至下一代种群;否则,依然保留原父代个体 P 在种群中。图 4 为一个单个体交叉示例,随机选择位置维度第 4 维和第 8 维,互换维度位置取值后,重新按维度进行降序排列,再依据解码规则即可得到新的工作流调度解。

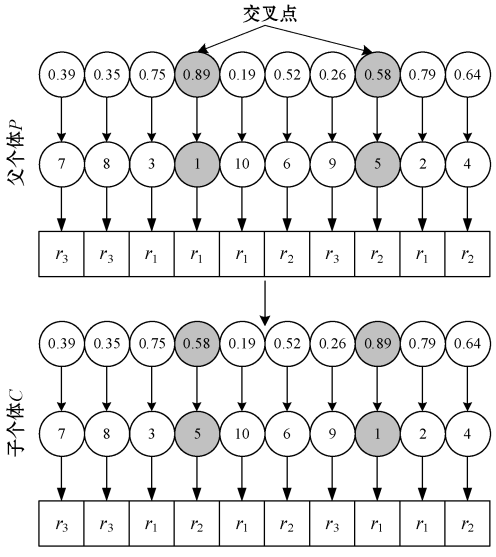

图 4　单个体交叉算子

单个体遗传交叉规则:随机生成范围(0,1)间的数值 r_{and},若 $r_{\text{and}}<p_{\text{rob}_{\text{single-crossover}}}$,$p_{\text{rob}_{\text{single-crossover}}}$ 为交叉概率,则对选择的个体执行单个体交叉;否则,不执行单个体交叉操作。

双个体间交叉。对于两个选定的父个体 P_1 和 P_2,在相同维度位置选定交叉点,交叉点将个体划分为左右两部分,互换两个父个体的右半部分(也可为左半部分),得到两个子个体 C_1 和 C_2。重新计算位置维度的降序排列,并重新解码为工作流调度解。对于两个新的子个体 C_1 和 C_2,比较两个父个体和两个子个体的适应度,保留适应度较小的两个个体至下一代种群。图 5 为双个体交叉示例,随机交叉点选定为维度位置 5 与 6 之间,互换两个父个体的右侧阴影部分位置,得到两个新的子个体,重新计算位置维度的降序排列,即可解码出新的工作流调度解。

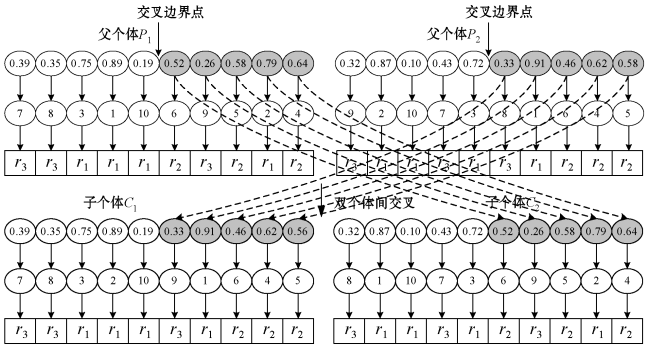

图 5　双个体交叉算子

双个体遗传交叉规则:随机生成范围(0,1)间的数值 r_{and},若 $r_{\text{and}}<p_{\text{rob}_{\text{double-crossover}}}$,$p_{\text{rob}_{\text{double-crossover}}}$ 为交叉概率,执行双个体交叉;否则不执行。

2)遗传变异。变异操作的过程为:对于选定的变异父个体 P,随机选择两个待变异的个体位置维度,改变其位置取值。重新计算维度的降序排列,并重新解码为工作流调度解,得到一个子个体 C。若子个体的适应度 C 小于父个体适应度,则将子个体 C 保留至下一代种群中;否则,依然保留父个体 P 在种群中。图 6 为个体变异示例,随机选择的变异点为维度位置 4 和 8,重新计算维度位置的降序排列后,即可解码出新的调度解。

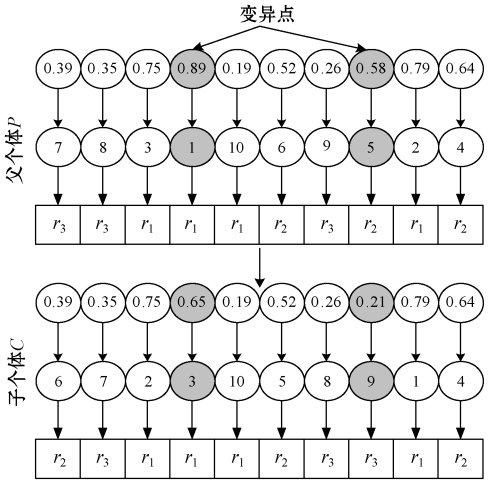

图 6　个体变异算子

个体遗传变异规则：随机生成范围(0,1)间的数值 r_{and}，若 $r_{\text{and}} < p_{\text{rob}_{\text{mutation}}}$，$p_{\text{rob}_{\text{mutation}}}$ 为变异概率，则对选择的个体执行个体变异操作；否则，不执行个体变异。

遗传交叉和变异操作可以通过提升个体多样性，保证算法的全局勘探能力，避免局部最优解。

5.4 适应度函数

适应度函数用于评估樽海鞘个体位置所代表的工作流调度解的质量。根据执行效率与执行代价的同步优化目标，将适应度函数定义权值形式：

$$f_{itness} = \alpha \times \frac{M_{\text{akespan}}}{D_{\text{eadline}}} + \beta \times \frac{E_{\text{C}}}{B_{\text{udget}}} \tag{19}$$

式中：α 和 β 分别代表时间权重和代价权重，用于描述用户对不同目标优化间的偏好，α、$\beta \in [0,1]$，且 $\alpha + \beta = 1$。由此可知，适应度函数值越小，表明樽海鞘个体位置所代表的工作流调度解的质量更优。

5.5 算法执行过程

基于改进樽海鞘群算法的多目标云工作流调度算法执行步骤如下。

步骤 1 参数初始化。对改进樽海鞘算法的参数作初始化：种群规模 N、随机变量 c_1、c_2、c_3 和 c_4、疯狂概率 P_{crazy}、疯狂算子中的极小值 x_{crazy}、位置上限值 $u_{b_{\max}}$ 和位置下限值 $l_{b_{\min}}$、最大迭代数 $T_{\max}$；初始化遗传算子相关参数，包括：单交叉概率 $p_{\text{rob}_{\text{single-crossover}}}$、双交叉概率 $p_{\text{rob}_{\text{double-crossover}}}$ 和变异概率 $p_{\text{rob}_{\text{mutation}}}$；初始化云工作流调度的相关参数，包括：云工作流结构 DAG、资源实例集合 R、截止时间 D_{eadline}、预算 B_{udget}、适应度函数中的时间权重 α 和代价权重 β。

步骤 2 种群初始化。将樽海鞘个体的位置根据式(20)进行随机初始化。

$$x_{i,j} = l_{b_{j,\min}} + c_5 \times (u_{b_{j,\max}} - l_{b_{j,\min}}) \tag{20}$$

式中：$i = 1,2,\cdots,N$，N 表示樽海鞘种群的规模，$j = 1,2,\cdots,n$，n 表示工作流任务总量；c_5 表示区间[0,1]内的随机数；$u_{b_{j,\max}}$ 表示第 j 维空间的上限值；$l_{b_{j,\min}}$ 表示第 j 维空间的下限值。由此，工作流调度解空间是一个 $N \times n$ 的欧氏空间，$x_{1,j}$ 表示领导者第 j 维空间的位置，$x_{i,j}(i>1)$ 表示追随者第 j 维空间的位置。

步骤 3 调度解解码。经过步骤 2 得到樽海鞘个体的位置编码后，根据 5.1 节的调度解码方法将位置编码解码出相应的工作流任务调度解。

步骤 4 适应度计算。根据式(19)计算樽海鞘种群中所有 N 个樽海鞘个体所代表的工作流调度解的适应度。

步骤 5 确定食物源位置。根据个体代表的工作流调度解的适应度，对所有樽海鞘个体作降序排列，将适应度值最小即最优调度解的樽海鞘个体作为当前种群中的食物源位置。

步骤 6 确定领导者和追随者。除食物源之外，将种群中剩余排序的 $N-1$ 个樽海鞘个体中的第一个樽海鞘个体视为领导者个体，将剩下的 $N-2$ 个樽海鞘视为追随者。

步骤 7 领导者位置更新。根据式(15)更新领导者，再根据式(18)实现领导者位置至区间[0,1]的位置转换，并由 5.1 节的方法根据位置编码解码出相应的工作流调度解。

步骤 8 追随者位置更新。将所有 $N-2$ 个樽海鞘追随者个体按适应度降序排列划分为两部分，前半部分的追随者个体根据式(13)更新追随者位置，若位置不在区间[0,1]，则依据式(18)进行位置转换，再由 5.1 节的方法依据位置编码解码出相应工作流调度解；后一半追随者个体则通过遗传算子中的交叉算子和变异算子更新追随者位置，再由 5.1 节的方法依据位置编码解码出相应工作流调度解。

步骤 9 重新计算个体适应度。重新计算位置更新后的个体适应度，选择适应度最小的个体，将其与食物源位置比较，若小于原食物源适应度，则将该个体作为新食物源位置；否则，保留原食物源。同时，根据步骤 7 和步骤 8 中的方法更新领导者个体和追随者个体位置。

步骤 10 若迭代数未达到 $T_{\max}$，则返回步骤 5；否则，按适应度大小的升序对樽海鞘个体进行排列，输出当前种群中的非占优解作为最终工作流调度最优解。

图 7 是算法的完整流程。

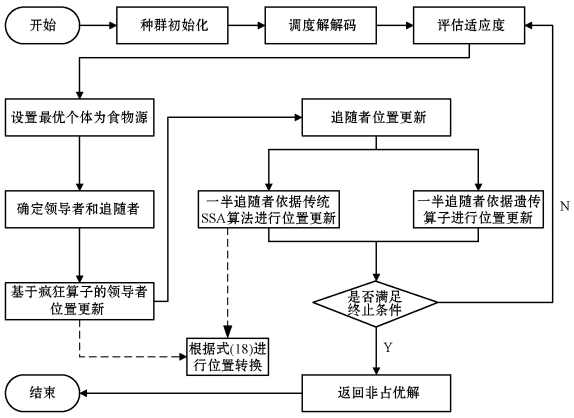

图 7 算法流程

算法时间复杂度分析：令种群规模为 N，最大迭代数为 $T_{\max}$。种群进行初始化之后，对调度解进行解码的时间复杂度为 $O(N)$，计算适应度的时间复杂度为 $O(N)$。确定领导者和追随者后，除食物源外，领导者为 1 个，追随者为 $N-2$ 个，领导者更新的时间复杂度为 $O(1 \times T_{\max})$，追随者更新的时间复杂度为 $O((N-2) \times T_{\max})$。综上，改进算法的时间复杂度即为 $O(N)$。可以看出，算法的实现代价主要与种群规模和算法的

迭代次数相关,在固定最大迭代次数时,问题求解代价与问题规模呈线性关系。

6　模拟实验

为了验证算法性能,分两组实验分别进行性能模拟与评估。将基于疯狂(Crazy)算子和遗传(Genetic)算子的改进樽海鞘群算法 SSA 命名为 CGSSA,将其应用于云工作流调度(Cloud Workflow Scheduling)优化问题时的算法命名为 CWS-CGSSA。第一组实验单独测试改进樽海鞘群算法的性能,利用 CGSSA 在 8 个经典的标准测试函数最优值求解上进行实验。第二组实验配置模拟云计算环境并利用 CWS-CGSSA 进行工作流任务调度。

6.1　CGSSA 评估

1)基准测试函数与环境配置。利用表 1 给出的 8 种基准测试函数作为测试 CGSSA 的目标函数,测试函数由多个单峰值函数和多峰值函数构成,其中:f_1、f_2、f_3 和 f_4 为单峰值函数;f_5、f_6、f_7 和 f_8 为多峰值函数。函数属性包括:特征、维度、最优值和定义域。定义域内的单峰值函数仅存在单个极值,可检测收敛速度和求解精度。多峰值函数则可能含有多个局部最优解或全局最优解,主要作用是检测算法的局部开发和全局勘探能力。由于群体智能算法在求解高低维度不同的测试函数时的性能并不是完全一致,所以选取的 8 个基准函数维度从 2 覆盖至 120,大跨度的函数维度可以实现性能的全面检测。仿真软件环境为 MATLAB 14b,操作系统为 Windows 10,硬件环境为 2.5 GHz CPU,内存 8 GB。CGSSA 的实验参数中,疯狂概率 P_{crazy} 为 0.3,x_{crazy} 为 10^{-4},最大迭代次数为 1 000,樽海鞘群规模为 40。性能对比选择:传统樽海鞘群算法 SSA[15]、粒子群算法 PSO、灰狼优化算法 GWO[16]。GWO 的收敛系数最大值设置为 2。PSO 的惯性权重值为 1,两个学习因子均设置为 1.5。种群规模设置为 40。

表 1　基准测试函数

函数	维度	特征	定义域	最佳值
f_1 = Schwefel 1.2	10	UN	[-100,100]	0
f_2 = Step	10	US	[-100,100]	0
f_3 = Sphere	60	US	[-100,100]	0
f_4 = Quartic	60	US	[-1.28,1.28]	0
f_5 = Schaffer	2	MN	[-100,100]	0
f_6 = Rastrigin	10	MS	[-5.12,5.12]	0
f_7 = Ackley	60	MN	[-32,32]	0
f_8 = Griewank	120	MN	[-600,600]	0

2)测试结果分析。表 2 统计了实施 40 次独立地模拟实验后四种算法得到的函数最优值、平均值和标准方差等情况的展示。最优值和均值反映算法寻优能力和收敛精度。从单峰值函数测试结果看,CGSSA 可达到 1E-29 的最高求解精度。当搜索维度增加时,算法的寻优精度均有所降低。维度增加导致求解难度呈指数级增长,收敛精度降低是正常的。综合来看,CGSSA 是寻优精度是最优的,且比较稳定,起伏较小。从多峰值函数测试结果看,算法的求解精度相比单峰值函数均有所降低,并且维度的增加也会使得寻优精度的降低。综合两类峰值函数的测试结果看,本文的 CGSSA 在求解单峰值和多峰值函数上均具有更好的精度。

表 2　8 个基准测试函数的测试结果

基准函数	算法	最优值	均值	标准方差
f_1	PSO	3.69E-06	6.48E-04	5.56E-04
	GWO	5.82E-12	4.92E-09	4.83E-09
	SSA	1.23E-18	8.45E-14	6.76E-14
	CGSSA	2.45E-29	3.78E-25	1.12E-25
f_2	PSO	3.13E-03	2.19E-03	1.12E-02
	GWO	4.34E-06	3.76E-06	3.69E-06
	SSA	5.17E-08	4.91E-08	3.74E-08
	CGSSA	6.33E-10	5.23E-10	3.56E-10
f_3	PSO	2.13E-14	7.45E-13	5.91E-14
	GWO	1.96E-17	4.23E-16	5.12E-16
	SSA	8.12E-20	7.34E-20	5.82E-19
	CGSSA	3.56E-28	4.87E-28	3.45E-29
f_4	PSO	5.54E-03	3.89E-03	1.19E-03
	GWO	8.34E-06	9.12E-05	8.34E-06
	SSA	5.81E-07	4.19E-07	1.19E-07
	CGSSA	2.82E-09	2.74E-09	2.13E-10
f_5	PSO	7.23E-09	5.35E-06	3.89E-05
	GWO	1.23E-09	4.28E-09	2.39E-08
	SSA	2.73E-10	3.42E-10	2.73E-09
	CGSSA	0.00E+00	0.00E+00	0.00E+00
f_6	PSO	2.78E+00	8.99E+00	5.34E+00
	GWO	0.00E+00	8.23E-02	5.89E-01
	SSA	3.89E+00	1.67E+01	1.38E+01
	CGSSA	0.00E+00	0.00E+00	0.00E+00
f_7	PSO	2.34E+00	8.23E+00	5.12E+00
	GWO	0.00E+00	7.43E-03	5.45E-02
	SSA	5.09E-01	8.56E-01	1.56E-01
	CGSSA	0.00E+00	0.00E+00	0.00E+00
f_8	PSO	3.89E+00	1.67E+01	1.38E+01
	GWO	0.00E+00	2.67E+01	2.12E+01
	SSA	4.23E-02	4.01E-01	3.91E-01
	CGSSA	0.00E+00	0.00E+00	0.00E+00

6.2 CWS-CGSSA 评估

1）环境搭建与性能指标。在利用 CloudSim 平台上实现云工作流调度算法并测试算法性能。利用 Montage、CyberShake 和 LIGO 工作流进行性能评估。三种工作流在其结构和计算资源需求上具有完全不同的特征表现[3]。Montage 工作流属于天文学领域内的工作流应用，任务模式以 I/O 密集型任务为主要类型，串行任务相对较少，任务对处理器资源能力的要求不高。LIGO 以计算密集型任务为主，任务结构上拥有较多且计算需求较小的串行任务，且对存储空间拥有较大需求。CyberShake 工作流属于地震科学领域应用结构，以数据密集型任务为主，对于存储能力和资源处理能力均有较高需求。引用三种不同领域内的工作流应用模型可以充分测试工作流调度算法自身的适应性和稳定性。在 CloudSim 中构建的云环境配置情况如下：资源处理能力均匀分布于[1 000 MIPS，10 000 MIPS]之间，虚拟机实例利用代价可以参考 Amazon EC2 的虚拟机实例定价机制。

两个约束方面利用灵活可变的非定值设置方式，以考察算法同步满足约束条件的能力。截止时间和预算约束的定义方式分别表示为：

$$D_{\text{eadline}} = L_{B_D} + \theta \times (U_{B_D} - L_{B_D}) \tag{21}$$

$$B_{\text{udget}} = L_{C_B} + \rho \times (U_{C_B} - L_{C_B}) \tag{22}$$

式中：L_{B_D}表示初始种群生成方法中以最早完成时间最小为目标生成的种群个体所得到的任务执行跨度时间；U_{B_D}设置为 3 倍的 L_{B_D}；θ 表示截止时间因子，$\theta \in [0,1]$；L_{C_B}表示初始种群生成方法中以最小执行代价为目标生成的种群个体所得到的任务执行代价；U_{C_B}设置为将任务调度至代价最高的资源上执行时得到的执行代价；ρ 表示预算因子，$\rho \in [0,1]$。

选取指标如下：

世代距离（Generation Distance，GD），可以评估相对边界处的收敛性指标，定义为：

$$G_D = \frac{\sqrt{\sum_{i=1}^{|Q|} d_i^2}}{|Q|} \tag{23}$$

式中：d_i 为 Q 的解与相对边界的最近解间的欧氏距离；Q 为计算 GD 指标时算法得到的边界。

间隔 S_{pacing}。该指标用于评估调度解的多样性，定义为：

$$S_{\text{pacing}} = \sqrt{\frac{1}{Q}\sum_{i=1}^{|Q|}(d_i - d_{\text{avg}})^2} \tag{24}$$

式中：d_{avg}为距离度量 d_i 的平均值；d_i 为当前解与 Q 的最近解间的距离。对于工作流调度解而言，世代距离和间隔两个性能指标越小越好。

选择同类型的两种调度算法进行性能对比：基于进化遗传算法的多目标优化工作流调度算法 MOEGA[8]和基于混合粒子群优化的非占优排序多目标调度算法 HPSO[9]。CWS-CGSSA 的相关参数与 CGSSA 法的实验配置相同，遗传算子中的交叉概率和变异概率分别设置为 0.6 和 0.4。HPSO 中的种群大小为 40，两个学习因子的取值范围分别由 2.5 递减至 0.5 和 0.5 递增至 2.5，粒子惯性权重由 0.9 递减至 0.1。MOEGA 的种群大小为 40，交叉概率为 0.6，变异概率为 0.4。迭代次数均设置为 100。

2）结果分析。第一组实验观察在三种工作流结构中进行双目标优化时得到的 Pareto 非占优解的分布情况，三种工作流结构下的测试结果分别如图 8、图 9 和图 10 所示。得到的结果为算法运行 10 次仿真实验得到的均值。从三幅图所展示的结果可以看出，本文的 CWS-CGSSA 要优于另外两种算法，所求解的大部分工作流调度解均与真实边界 True Front 较为邻近，且各个解之间维持着更加一致的空间分布。混合多目标粒子群优化调度算法 HPSO 又优于基于进化遗传算法的多目标优化工作流调度算法 MOEGA，由于 HPSO 能够利用非占优排序求解满足帕累托最优的调度解集。由于对于拥有多个决策变量的多目标优化问题而言，并不存在对于所有优化目标的绝对单个最优解，此时得到的解应考虑为多个目标最优的潜在解的集合形式，因此，本文对于得到的调度解是以 Pareto 解集的形式予以展示。由于篇幅原因，本文并未对该概念以及相关的解之间的占优关系和最优边界等概念作详细说明。

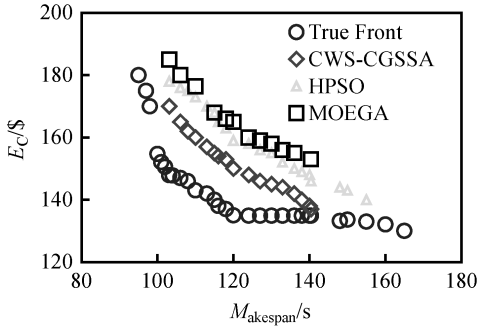

图 8 Montage 工作流的双目标非占优解分布

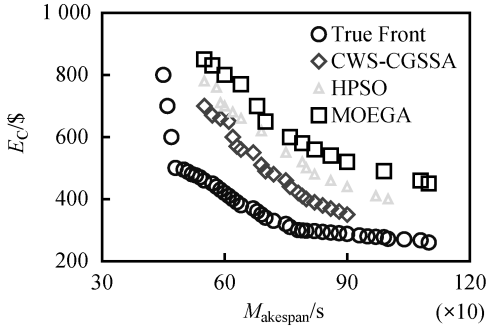

图 9 CyberShake 工作流的双目标非占优解分布

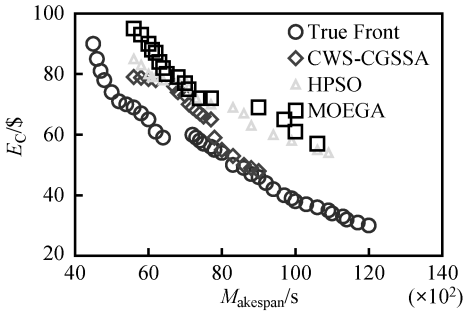

图 10　LIGO 工作流的双目标非占优解分布

第二组实验观察了两个约束因子在不同取值下算法性能指标的表现,本组实验利用 Montage 工作流类型进行单独测试,结果如图 11 和图 12 所示。图形为一横两纵形式,横坐标对应于截止时间因子和预算因子的不同取值,柱状图对应左侧纵坐标,表示世代距离指标得到的均值表现,折线图对应右侧纵坐标,表示间隔 S_{pacing}指标得到的均值表现。可以看出,根据 0.2 的步长逐步增加两种约束因子的取值,算法的世代距离 G_D 和间隔 S_{pacing}的取值也在逐步减小,说明性能在变好,原因可分析为:依据截止时间和预算约束的定义公式,增大约束因子的取值,表明相应的约束条件更加宽松,即截止时间会更长或可用预算会更多,则工作流调度算法在工作流任务与虚拟机实例的选取上可以选择更为合适高效的调度方案,得到的调度方案性能固然也会更好。相对而言,本文设计的 CWS-CGSSA 可以得到更小的世代距离 G_D 和间隔 S_{pacing},这是由于 CWS-CGSSA 可以基于改进的樽海鞘群寻优机制得到非占优的解集合,基于疯狂算子的领导者位置更新模式,通过疯狂变量,减少传统领导者位置更新可能带来的停滞早熟现象。而基于遗传算子的追随者位置更新模式,利用个体交叉和变异使樽海鞘群具有更均衡的全局搜索能力和个体多样性,避免陷入局部最优。HPSO 同样考虑了多目标优化调度,在宽松约束下依然可以得到表现不错的调度解,但在约束变得严格时,其调度方案选择将极大受限,得到帕累托解集的分布不如 CWS-CGSSA 紧致。MOEGA 并未考虑预算约束问题,因此改变预算因子对其调度解的性能不产生直接影响,所以指标上基本表现为平滑直线条。

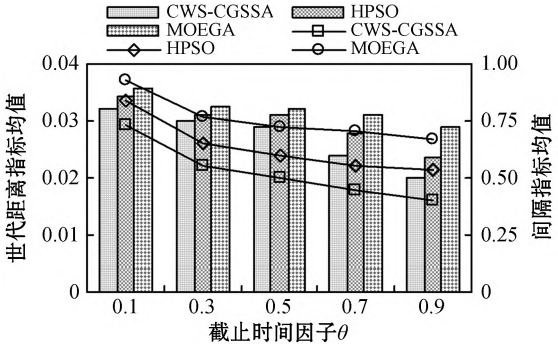

图 11　截止时间因子对世代距离 G_D 和间隔 S_{pacing}的影响

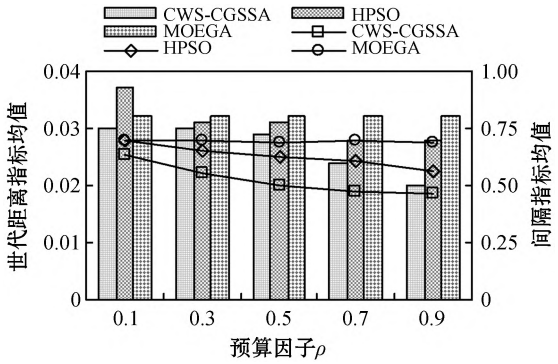

图 12　预算因子对世代距离 G_D 和间隔 S_{pacing}的影响

由于该算法是基于多目标多约束条件的云工作流调度算法,因此无法生成单一目标最优化的调度解。但是,通过调整适应度函数中的时间权重和代价权重,可以分别侧重于优化调度效率和调度代价。在存在执行期限与执行预算的双约束工作流调度环境中,可以生成目标均衡的调度解集合,并以 Pareto 解集的形式予以展示,因此较为适用于用户方多重服务质量需求的云任务调度场景。

7　结　语

提出一种基于改进樽海鞘群算法的工作流调度策略。利用樽海鞘个体在觅食中的位置变化对工作流任务调解进行编解码,设计融合执行跨度与执行代价的权重适应度函数。重点在于:引入基于疯狂算子的领导者位置更新模式,通过疯狂变量,减少传统领导者位置更新可能带来的停滞早熟现象;引入基于遗传算子的追随者位置更新模式,利用个体交叉和变异使樽海鞘群具有更均衡的搜索全局性和个体多样性,改善搜索易于陷入局部最优。实验结果证明,改进樽海鞘群算法 CWS-CGSSA 得到的调度解收敛性更好,调度解集空间分布更加一致,更加符合多目标调度优化需求。

参　考　文　献

[1] Liu L, Zhang M, Lin Y, et al. A survey on workflow management and scheduling in cloud computing[C]//2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing,2015:837 - 846.

[2] Garg R, Singh A. Adaptive workflow scheduling in grid computing based on dynamic resource availability[J]. Engineering Science and Technology,2015,18(2):256 - 269.

[3] Alkhanak E, Lee S, Rezaei R, et al. Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues[J]. Journal of Systems and Software,2016, 113(2):1 - 26.

(下转第 331 页)

参 考 文 献

[1] 吴建. 数据加密技术在计算机网络通信安全中的应用分析[J]. 信息通信,2020(6):158 - 159.

[2] Liu Y J, Jiang Z G, Xu X P, et al. Optical image encryption algorithm based on hyper-chaos and public-key cryptography[J]. Optics & Laser Technology,2020,127:106171 - 106180.

[3] Sureshkumar V, Amin R, Obaidat M S, et al. An enhanced mutual authentication and key establishment protocol for TMIS using chaotic map[J]. Journal of Information Security and Applications,2020,53:102539.

[4] Cui J, Wang Y, Zhang J, et al. Full session key agreement scheme based on chaotic map in vehicular ad hoc networks [J]. IEEE Transactions on Vehicular Technology,2020,69(8):8914 - 8924.

[5] 肖成龙,孙颖,林邦姜,等. 基于神经网络与复合离散混沌系统的双重加密方法[J]. 电子与信息学报,2020,42(3):687 - 694.

[6] 禹思敏,吕金虎,李澄清. 混沌密码及其在多媒体保密通信中应用的进展[J]. 电子与信息学报,2016,38(3):735 - 752.

[7] 吴琼琼,马子洋,李启华,等. 高速混沌光通信研究进展[J]. 光通信技术:2021,45(1):22 - 27.

[8] álvarez G, Montoya F, Romera M, et al. Cryptanalysis of an ergodic chaotic cipher[J]. Physics Letters A, 2003, 311(2):172 - 179.

[9] 鲍芳,李军,李旭. 基于高维广义猫映射的图像加密算法[J]. 西安理工大学学报,2012,28(2):193 - 197.

[10] 芮杰,杭后俊. 基于超混沌系统的明文关联图像加密算法[J]. 图学学报,2020,41(6):917 - 921.

[11] 师婷,高丽. 基于混沌映射和 DNA 编码的彩色图像加密算法研究[J]. Journal of Measurement Science and Instrumentation,2021,12(1):68 - 73.

[12] 谢国波,王添. 基于像素置乱和比特替换的混沌图像加密算法[J]. 微电子学与计算机,2016,33(3):80 - 85.

[13] 朱淑芹,王文宏,李俊青. 对像素置乱和比特替换混沌图像算法的破解[J]. 计算机应用,2017,37(S2):44 - 47.

[14] 马在光,丘水生. 基于广义猫映射的一种图像加密系统[J]. 通信学报,2003,24(2):52 - 58.

[15] 韩雪娟,李国东. 动态猫变换和混沌映射的图像加密算法[J]. 计算机工程与设计,2020,41(8):2381 - 2387.

[16] 王晓雷. 基于动态猫映射和 Unix 时间戳的图像加密算法研究[D]. 新乡:河南师范大学,2019.

[17] 孙倩,胡苏. 基于改进 cat 映射与混沌系统的彩色图像快速加密算法[J]. 计算机应用研究,2017,34(1):233 - 237,255.

[18] 王兴元,王宇,王思伟,等. A novel pseudo-random coupled LP spatiotemporal chaos and its application in image encryption[J]. Chinese Physics B,2018,27(11):423 - 433.

[19] 郭媛,敬世伟,周艳艳. 基于相邻像素间比特置乱的图像加密算法[J]. 计算机工程与设计,2020,41(7):1829 - 1835.

[20] 谢国波,邓华军. 二次广义 Cat 映射的混合混沌图像加密算法[J]. 计算机工程与应用,2018,54(15):197 - 202.

(上接第 271 页)

[4] 曹斌,王小统,熊丽荣,等. 时间约束云工作流调度的粒子群搜索方法[J]. 计算机集成制造系统,2016,22(2):372 - 380.

[5] Abrishami S, Naghibzadeh M, Epema D. Deadline-constrained workflow scheduling algorithms for Infrastructure as a service clouds[J]. Future Generation Computer Systems, 2015,29(1):158 - 169.

[6] Chopra N, Singh S. HEFT based workflow scheduling algorithm for cost optimization within deadline in hybrid clouds [C]//2013 4th International Conference on Computing, Communications and Network Technology,2013:1 - 6.

[7] Bossche D, Vanmechelen K, Broeckhove J. Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds[J]. Future Generation Computer Systems,2013,29(4):973 - 985.

[8] 王国豪,李庆华,刘安丰. 多目标最优化工作流调度进化遗传算法[J]. 计算机科学,2018,45(5):31 - 37,48.

[9] 杜艳明,肖建华. 云环境下基于混合多目标粒子群的科学工作流调度算法[J]. 计算机科学,2017,44(8):252 - 259.

[10] Su S, Li J, Huang Q, et al. Cost-efficient task scheduling for executing large programs in the cloud[J]. Parallel Computing,2015,39(4 - 5):177 - 188.

[11] Bényi A, Dombi J D, Kertesz A. Energy-aware VM scheduling in IaaS clouds using pliant logic[C]//4th International Conference on Cloud Computing and Services Science,2014: 519 - 526.

[12] 陈爱国,王玲,任金胜,等. 基于资源分组的多约束云工作流调度算法[J]. 电子科技大学学报,2017,46(3):562 - 568.

[13] 马晓虹,韩强,辛阔. 基于服务质量的云工作流调度优化算法[J]. 计算机工程与设计,2018,23(1):151 - 158,259.

[14] Zheng W, Sakellariou R. Budget-deadline constrained workflow planning for admission control[C]//International Workshop on Grid Economics and Business Models,2015:105 - 119.

[15] Mirjalili S, Gandomi A, Saremi S, et al. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems[J]. Advances in Engineering Software,2017,114:163 - 191.

[16] Yassien E, Masadeh R, Alzaqebah A, et al. Grey wolf optimization applied to the 0/1 knapsack problem[J]. Journal of Computer Applications,2017,169(5):11 - 15.