基于双仲裁机制和田口正交法的猫群优化任务调度算法

张兴明 殷从月* 魏 帅 叶盛钊 吕 平
(国家数字交换系统工程技术研究中心 郑州 450002)

摘 要：针对异构计算系统任务调度过程中通信冲突以及算法运行时间的问题，该文提出一种基于双仲裁机制和田口正交法的猫群优化任务调度算法。首先利用双仲裁机制对任务资源进行管理，动态判决任务的分配，有效避免通信冲突，再将田口正交法应用到猫群优化过程的跟踪模式中，降低算法运行时间，提高解的质量。实验结果表明，该算法运行速度明显高于其他算法至少约10%，算法在处理大量任务时的并行化效果最优，在异构环境中也体现出其相当大的优势。

关键词：异构计算；任务调度；猫群优化；双仲裁；田口正交法
中图分类号：TP39 **文献标识码**：A **文章编号**：1009-5896(2018)10-2521-08
DOI: 10.11999/JEIT180215

Cat Swarm Optimization Task Scheduling Algorithm Based on Double Arbitration Mechanism and Taguchi Orthogonal Method

ZHANG Xingming YIN Congyue WEI Shuai YE Shengzhao LÜ Ping
(*National Digital Switching System Engineering & Technological Research Center*, *Zhengzhou* 450002, *China*)

Abstract: To solve communication conflicts and algorithm running time problem in task scheduling process of heterogeneous computing system, a cat swarm optimization task scheduling algorithm is proposed based on double arbitration mechanism and Taguchi orthogonal method. Firstly, the double arbitration mechanism is used to manage the task resources, and the task assignment is dynamically decided to avoid effectively communication conflicts. Then, the Taguchi orthogonal method is applied to the tracking mode of the cat swarm optimization process to reduce the algorithm running time and improve the quality of the solution. Experimental results show that the algorithm runs at a rate of at least about 10% faster than other algorithms. The algorithm performs best in parallelism when dealing with a large number of tasks and has considerable advantages in heterogeneous environments.

Key words: Heterogeneous computing; Task scheduling; Cat swarm optimization; Double arbitration; Taguchi orthogonal method

1 引言

异构计算系统(Heterogeneous Computing System, HCS)为各种应用程序提供了具备可扩展性的计算资源，它是通过高速网络互连的各种处理器组成[1]。异构计算环境中的任务调度主要是将任务分解成多个具有并行关系的子任务，再将这些任务按顺序分配给处理器，从而优化网络计算性能，实现高效计算[2]。因此，任务调度是异构计算系统性能提升的关键。

异构计算系统中的任务调度问题已经被证明是一个NP完全问题(Non-deterministic Polynomial complete problems)[3]，主要优化目标是减少算法运行时间，保证最优解的可靠性。目前，国内外学者主要采用启发式方法来解决这个问题，可以获得较快的问题求解时间和较为优化的任务调度方案。文献[4,5]将启发式任务调度算法与遗传算法(Genetic Algorithm, GA)相结合，提出了一种具有自适应参数的混合启发式算法(Hybrid Heuristic Genetic Algorithm with Adaptive Parameter, HGAAP)，该算法在收敛速度方面改进了原有的遗传算法，但执行时间仍然在几十分钟左右。文献[6]

收稿日期：2018-03-07；改回日期：2018-07-25；网络出版：2018-08-02
*通信作者： 殷从月 503637088@qq.com
基金项目：国家科技重大专项资助项目(2016ZX01012101)，国家自然科学基金(61572520, 61521003)
Foundation Items: The National Science Technology Major Project (2016ZX01012101), The National Natural Science Foundation of China (61572520, 61521003)

提出了一种保证可行性、类似旅行商问题(Traveling Saleman Problem, TSP)的元启发式任务调度算法(Feasibility Assured TSP-likened Scheduling, FATS)，该算法将调度问题转化为类似于TSP问题的结构图，再与蚁群优化算法(Ant Colony Optimization, ACO)相结合；文献[7,8]则将ACO算法与贪婪算法(greedy algorithm)相结合，算法的运行时间均有所降低，但算法复杂度高，同时最优解的可靠性被忽略。文献[9]将粒子群算法(Particle Swarm Optimization, PSO)与GA算法相结合(Hybrid PSO-GA meta heuristic, HPSO-GA)，PSO算法用于空间搜索，GA算法用于探索比较有前景的空间，可以获得局部最优的结果，但收敛速度慢，算法的运行时间较长。文献[10,11]引入猫群优化算法(Cat Swarm Optimization, CSO)，在搜寻和跟踪模式下进行任务调度，得到解的质量较高，但算法运行时间还有待优化。

上述研究表明，在异构计算系统任务调度问题中，常规启发式算法以及混合启发式算法无法同时满足优化算法执行时间和提高解质量的要求。此外所有算法的前提条件是处理器间的通信并发执行并且与处理器本身无关，一个处理器可以同时接收、处理和发送任务，也可以和其他多个处理器进行通信，通信时的资源冲突问题忽略不计。而在异构计算系统中，时常会发生通信冲突，这对任务调度的结果会产生严重影响[12]。

针对以上问题，本文提出一种基于双仲裁机制和田口正交法的猫群优化任务调度算法(Cat Swarm Optimization task scheduling algorithm based on Double arbitration mechanism and Taguchi orthogonal method, DTCSO)，这里主要分为两个步骤：任务映射和任务执行。任务映射主要是利用双仲裁机制对任务资源进行管理，并根据时间和各个处理器的情况来动态判决任务的分配，将各个任务映射到处理器上；任务执行主要是将田口正交法应用到CSO方法的跟踪模式过程中，田口正交法可以在最短的时间内通过最少的实验次数获得最佳的组合方案，CSO算法包括全局和局部搜索两个过程，比较适用于扩展性较强的异构计算系统。实验结果表明，双仲裁机制可以有效避免通信冲突，引入田口正交法后，计算精度更高，算法的运行时间降低很多，最优解的可靠性增加，解质量进一步提高。

2 双仲裁机制

2.1 通信冲突

宏观数据流模型是异构计算系统中的经典任务调度模型[13]，在该模型中处理器一次只能处理一个任务，因此该模型没有考虑各任务之间、处理器之间通信时的资源冲突问题，选择将其忽略不计。而在异构计算系统中，如果任务在通信时通道被占用，则无法在预定的时间执行子任务，这会影响调度结果的准确性。双向单一端口模型[14]与上述经典模型很相似，但处理器的通信端口只有一个，即处理器只能接收和发送一个任务，这样就必须考虑通信冲突的问题。图1是用有向无环图(Directed Acyclic Graph, DAG)描述的任务示例图，假定图中所有节点和通信成本的权重为1，把任务T_0, T_1和T_2分配给处理器P_0，任务T_3, T_4, T_5和T_6分配给剩余的4个处理器。在经典模型中，第0时间段P_0执行T_0，第1时间段P_0执行T_1并将T_0的结果发送给其他处理器，第2时间段所有处理器执行剩余任务，总完成时间为3。在双向单一端口模型中，第0时间段P_0执行T_0，按顺序发送数据到最后一个处理器需要4个时间单位，该处理器执行任务还需要1个时间单位，总完成时间为6。因此父任务T_0到各子任务以及各子任务之间的通信成了完成任务调度的关键。

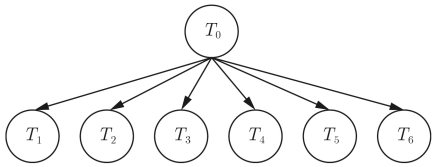


图 1 任务示例图

2.2 双仲裁模式

根据上述分析，采用双向单一端口模型并对常规任务图进行扩展，如图2所示。

定义1 $T_m=\{H_{(m,0)},H_{(m,1)},\cdots,H_{(m,n)}\}$表示由子任务$H_{(m,n)}$组成的任务模块，$H_{(m,n)}$可用二元组$\{t_{H_{(m,n)}},t_{e,H_{(m,n)}}\}$来描述，其中$t_{H_{(m,n)}}$为$H_{(m,n)}$开始映射的时间，$t_{e,H_{(m,n)}}$为$H_{(m,n)}$映射所需的时间。任务周期$\Delta t$表示任务映射的最小时间单位，对实际任务映射时间$t$进行规整，即$t=n\cdot\Delta t(n\in Z)$。因各任务持续映射时间基本相同，可将$t_{e,H_{(m,n)}}$统一设置

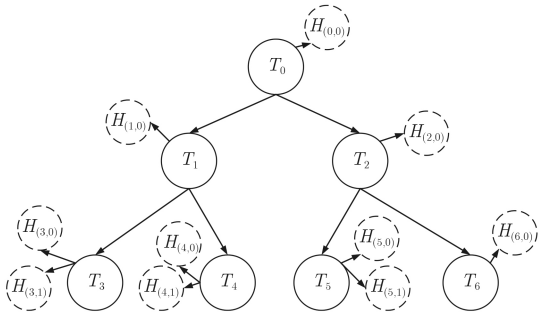


图 2 扩展的任务示例图

为Δt，即$t_{e,H_{(m,n)}}=\Delta t$。图3是由图2所作的任务映射时序图，横轴表示时间，纵轴表示所占处理器的位置。根据时序图，可以采用双仲裁的方式进行判决。

定义2　双仲裁分别指时间仲裁(Time Arbitration, TA)和处理器位置仲裁(Processor Location Arbitration, PLA), TA和PLA均在$n\cdot\Delta t$处记录发生的事件，实现对任务的映射。TA记录任务映射的时间变化情况，当在$n\cdot\Delta t$处有任务开始映射，并占用处理器时，该任务在时序图横轴方向产生的新的时间点就会被记录下来；当有任务退出，并释放所占处理器时，就删除该任务在时序图横轴方向产生的时间记录。PLA记录任务映射所占处理器的位置，任务的位置由处理器空闲状态与任务映射顺序确定，当有任务开始映射时，PLA就记录当前任务所在处理器的位置；当有任务退出时，该任务的位置记录就会被删除。具体算法过程如表1所示。TA与PLA的配合使用能详细记录任务进出处理器的时间和位置，判断任务通道是否畅通，从而有效避免实际执行过程中通信冲突的情况，提高任务调度的效率。

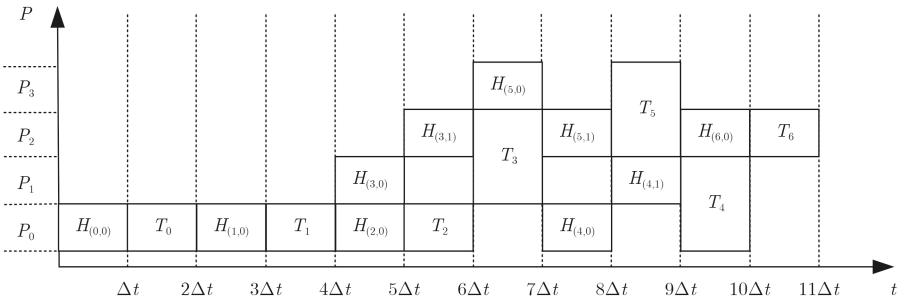


图 3 任务映射时序图

表 1　双仲裁映射判决算法

算法1 双仲裁映射判决算法

输入：TA, PLA, t, T_m

输出：有效的任务映射结果DAM(T)

(1) **if** (($T_m\cdot t_a\le t\le T_m\cdot t_s$) and ($T_m\cdot s=$ IDLE ‖ WAIT)) then enQueue (Q, T_m)

(2)　Select T_m from Q where $T_m\cdot p$ is the highest

(3)　**if** (PLA=NULL) then

(4)　temp P·taskid $\leftarrow T_n\cdot$ taskid;
temp P·pos$\leftarrow[(0,0),(T_n\cdot w,T_n\cdot h)]$;

(5)　$T_n\cdot s\leftarrow$ EXE; $TA\ [i{+}1][0]\leftarrow T_n\cdot w$; $TA\ [i{+}1][1]\leftarrow T_n\cdot h$;

(6)　insert node temp$P$ into PLA; **else**

(7)　DAM(T_j, d); **end if**

(8) **end if**

3　田口-猫群优化

3.1　田口正交法

田口方法是由Taguchi博士所提出的一种工程方法[15]，它主要是用最少的实验次数获得最优的制程变量参数组合条件，以此在最短的时间内筛选出最佳方案[16]。田口方法组合设计变量，用矩阵形式表示正交阵列，每列表示所考虑因子的值，使用的是两级正交阵列[17]。两级正交阵列如式(1)所示，其中$n=2^k$, k为大于1的正整数，$n-1$表示两级正交阵列的列数。举例两组解，每组均有7个参数，为了找到最佳组合值，设计如表2所示的$L_8(2^7)$正交阵列。表2中ABCDEFG是7个两级参数，L_8表示要进行8次实验来研究两级参数。该正交阵列中的元素表示各参数值根据因子来决定被运用到的实验组，0值表示该参数值被运用于第1组实验，1值表示被运用于第2组实验。假设表2中的矩阵被用于任务调度，正交阵列中的水平数值表示任务实例，因子0和1表示处理器的位置，每个处理器计算任务执行过程均有预估的时间，则表3中$n\times m$矩阵表示的是包含任务$T=\{T_0,T_1,T_2,T_3,T_4,T_5,T_6\}$和8个处理器$P=\{P_0,P_1,P_2,P_3,P_4,P_5,P_6\}$的预估计算时间。引入田口正交法的目的是将其结合到CSO方法的跟踪模式中，保证每个变量参数的最优组合和最佳水平，提高猫群优化算法的性能，最大限度地减少任务调度的总完成时间。

$$L=L_n\left(2^{n-1}\right) \tag{1}$$

表 2　$L_8(2^7)$正交阵列

实验序列号	考虑的因子						
	A	B	C	D	E	F	G
1	1	0	0	1	1	0	0
2	1	1	1	0	0	0	0
3	0	0	1	0	1	1	0
4	0	1	0	0	1	0	1
5	1	1	1	1	1	1	1
6	1	0	0	0	0	1	1
7	0	0	1	1	0	0	1
8	0	1	0	1	0	1	0

表 3 预估计算时间矩阵

	T_0	T_1	T_2	T_3	T_4	T_5	T_6
P_0	T_0/P_0	T_1/P_0	T_2/P_0	T_3/P_0	T_4/P_0	T_5/P_0	T_6/P_0
P_1	T_0/P_1	T_1/P_1	T_2/P_1	T_3/P_1	T_4/P_1	T_5/P_1	T_6/P_1
P_2	T_0/P_2	T_1/P_2	T_2/P_2	T_3/P_2	T_4/P_2	T_5/P_2	T_6/P_2
P_3	T_0/P_3	T_1/P_3	T_2/P_3	T_3/P_3	T_4/P_3	T_5/P_3	T_6/P_3
P_4	T_0/P_4	T_1/P_4	T_2/P_4	T_3/P_4	T_4/P_4	T_5/P_4	T_6/P_4
P_5	T_0/P_5	T_1/P_5	T_2/P_5	T_3/P_5	T_4/P_5	T_5/P_5	T_6/P_5
P_6	T_0/P_6	T_1/P_6	T_2/P_6	T_3/P_6	T_4/P_6	T_5/P_6	T_6/P_6
P_7	T_0/P_7	T_1/P_7	T_2/P_7	T_3/P_7	T_4/P_7	T_5/P_7	T_6/P_7

3.2 并行猫群优化算法

猫群算法是群体智能算法中新增加的一种群优化技术[18]，它将猫的行为分为两种模式，一种是猫在环顾四周时懒散的状态，即搜寻模式；另一种是在跟踪动态目标时的状态，即跟踪模式[19]。通过调整混合比率(Mixed Ratio, MR)，可以控制搜寻和跟踪模式中移动个体的比例，其中MR $\in$ [0,1]。随着猫越来越接近最优解(即猎物)，猫的位置和速度会不断更新，直到所有的猫达到最好的解决方案。

3.2.1 搜寻模式 搜寻模式是CSO方法中的全局搜索过程，它主要包含4个基本要素：搜寻记忆池(Seeking Memory Pool, SMP)，给定变化域(Seeking Range of selected Dimension, SRD)，改变维度位数(Counts of Dimension to Change, CDC)，自身位置判断(Self-Position Considering, SPC)。SMP定义每只猫搜寻记忆的大小，猫会根据适应度的大小来搜寻最好的位置点；SRD设定了所选维度的变异率；CDC表示每只猫将会变异的维度的个数；SPC代表一个布尔变量，决定猫当前所处的位置是否可以作为将要移动过去的候选位置之一，SMP值不受SPC值的影响。这些要素在搜寻模式中都起着重要作用。

3.2.2 并行跟踪模式 并行跟踪模式是CSO方法中的局部搜索过程。在常规CSO方法中，全局搜索和局部搜索是针对每次迭代独立进行的，则速度和位置更新也是独立完成的，算法时间长。此外，若MR = 2%，则进入搜寻和跟踪模式的猫的数量分别占80%和20%，表示进行全局搜索的猫的数量总是会超过局部搜索的猫的数量，CSO方法在局部搜索中的变异操作必然会影响整体性能，导致可能找不到近似最优解。为了更好地解决这些问题，将田口正交法与CSO方法中的局部搜索过程相结合，用少量实验来执行大量任务，减少算法时间消耗。

主要包括以下几个步骤：首先使用迭代式(2)计算并更新第k只猫的速度，其中$x_{k,d}$表示猫的位置，$x_{b,d}$表示第k只猫所在的实验组前一次迭代适应度值最好的位置，$v_{k,d}(t)$表示两个候选速度集，$x_{g,d}$和$x_{l,d}$分别表示猫全局搜索和局部搜索的最佳位置，t是迭代次数，c_1是常数加速度，r_1是在[0, 1]范围内均匀分布的随机数，d是解空间的维数，将候选速度和田口正交阵列相结合而成的速度集如式(3)所示。接着利用式(4)检查更新的速度是否在最大速度范围内，如果超出该范围，则被限制到最大速度。最后用式(5)更新第k只猫的位置，并计算每只猫的适应度值，根据生成的速度将这些值求和，比较选择出最终的速度。具体的田口猫群优化算法过程如表4所示。

$$v_{k,d}(t)=\begin{cases} v_{s_{1k,d}}(t)=v_{k,d}(t-1)+c_1\cdot r_1\cdot[x_{g,d}(t-1) \\ \qquad -x_{k,d}(t-1)],\ d=1,2,\cdots,M \\ v_{s_{2k,d}}(t)=v_{k,d}(t-1)+c_1\cdot r_1\cdot[x_{l,d}(t-1) \\ \qquad -x_{k,d}(t-1)],\ d=1,2,\cdots,M \end{cases} \tag{2}$$

$$v_{k,d}(t)=\begin{cases} v_{s_{1k,d}}(t),\ \text{正交阵列中的元素为0} \\ v_{s_{2k,d}}(t),\ \text{其他} \end{cases} \tag{3}$$

$$v_{k,d}=\begin{cases} v_{\max}, & v_{k,d}+v_{k,d}(t)>v_{\max} \\ v_{k,d}+v_{k,d}(t), & \text{其他} \end{cases} \tag{4}$$

$$x_{k,d}(t)=x_{k,d}(t-1)+v_{k,d}(t) \tag{5}$$

4 DTCSO算法

4.1 算法流程图

如图4，在田口-猫群优化后，双仲裁模块对初步优化的结果进行判决，判断是否存在通道冲突的情

表 4 田口-猫群优化算法

算法2 田口-猫群优化算法
输入：猫的初始位置，初始速度，MR, SMP, SRD, CDC, SPC
输出：最优的任务调度结果
(1) **if** (seeking flag← 1) //按照搜寻模式进行
(2) 生成第k只猫的y (y=SMP) 份副本Z_{qd} ($1\le q\le Y$, $1\le d\le D$)；//D是解空间的维数
(3) 每当CDC对Z_{qd}进行加或减的变异操作就随机改变猫的维度，确定被改变的猫的适应度；
(4) 从Y中随机选取候选值后发现并替换猫的最佳位置； **else**
(5) 选取两级田口正交阵列$L_n(2^{n-1})$,$\forall n\ge N+1$; //N表示任务数量
(6) 根据式(2)生成两组速度并根据任务数量N确定任务调度的维数；
(7) 根据正交阵列$L_n(2^{n-1})$计算n次实验的适应度值；
(8) **end if** 选取当前最优个体x_l和最佳位置x_p；
(9) **if** ($x_l>x_g$) $x_l=x_g$; $x_p=G_b$; //当前最佳位置即全局最佳位置
(10) 根据式(4)，式(5)计算并更新当前速度和位置；
(11) **if** (达到最终条件) 输出最佳的任务调度顺序；
(12) **else** 重新计算每只猫的适应度值； **end if**
(13) **end if**

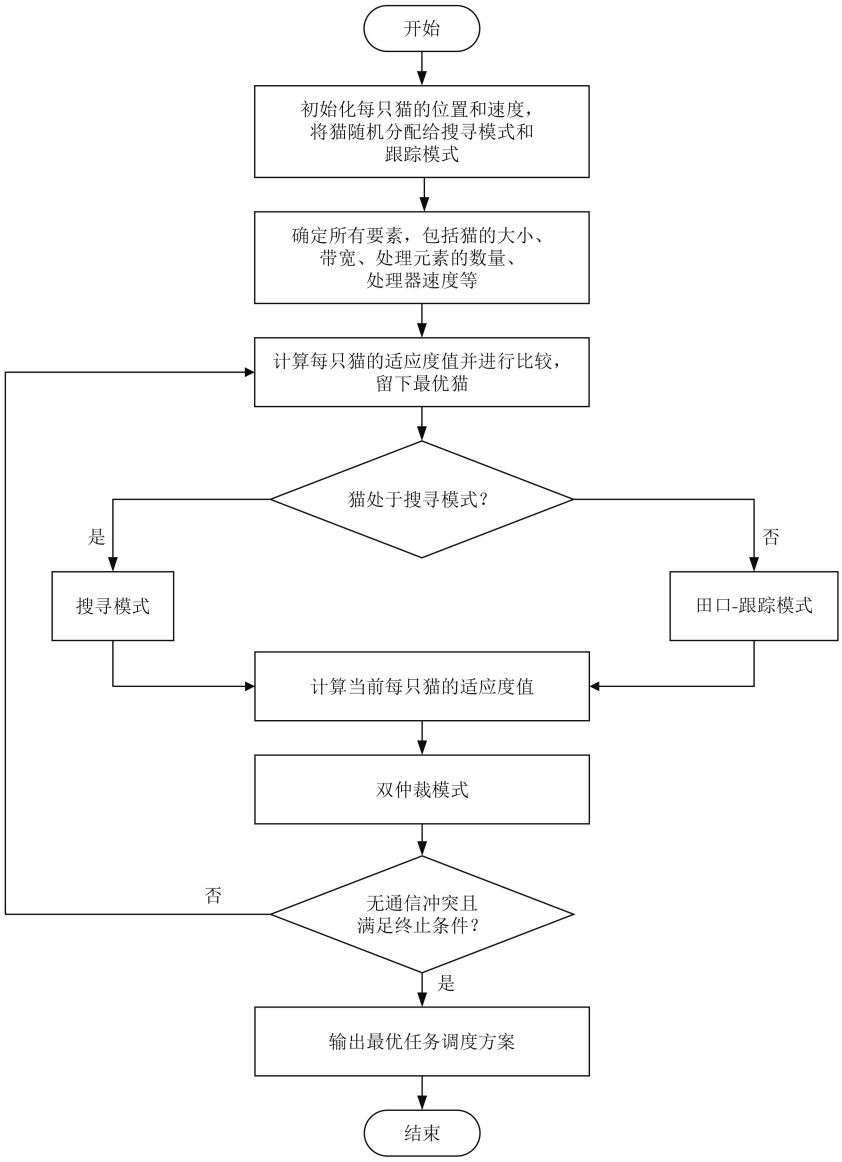


图 4 DTCSO算法流程图

况，若存在则重新开始猫群优化，若不存在且满足猫群优化算法的终止条件，则输出最优任务调度方案。

4.2 算法复杂度分析

在双仲裁机制中，时间复杂度为$O(n+e)$，n为任务数量，e为边数。而在复杂的任务图中，e与$O(n^2)$成正比，所以时间复杂度为$O(n^2)$。在田口-猫群优化过程中，每只猫都要进行搜寻或追踪，然后将任务按顺序分配给处理器，这个过程时间复杂度为$O(mn)$，m为处理器数量。因此该算法总时间复杂度为$O(mn+n^2)$。

5　实验与结果分析

实验在Matlab R2013a平台上进行，采用DAG生成程序DAGGEN[20]生成随机拓扑结构的任务图，将DTCSO算法与HGAAP算法[4,5]、ACO-GA算法[7,8]、HPSO-GA算法[9] 和常规CSO算法[10,11]进行比较。

5.1 实验性能评价指标

5.1.1 对比指标　调度下界比(Schedule Length Ratio, SLR)[21]：分子是调度算法总完成时间，分母是在关键路径上总最短执行时间，用于衡量调度时间长度。加速比Speedup(S)：分子是将任务分配给处理器后按顺序执行的最短执行时间，用于衡量并行系统并行化效果。算法优良百分比：本文算法比其他算法更好、相同和更差的百分比。

$$\mathrm{SLR}=\frac{\mathrm{mp}}{\sum\limits_{v_i\in CP_{\min}}\min\limits_{m_t\in M}\{w_i,t\}} \tag{6}$$

$$\mathrm{S}=\frac{\min\limits_{m_t\in M}\left\{\sum\limits_{v_i\in V}w_{i,t}\right\}}{\mathrm{mp}} \tag{7}$$

5.1.2 与随机生成的DAG图有关的指标 任务数量N；处理器数量Q；密度den：图中节点相互依赖程度，密度越大依赖性越强；并行度fat：值越大，DAG图并行度越高；相似度reg：各级任务数量相似度，值越大相似度越高；跳数jump：通信时所跨越的级数；通信和计算时间比(Communication to Computation Ratio, CCR)[22]：任务间平均通信时间和平均执行时间比值；异构因子β：处理器执行的时间范围，值越大异构性越高，处理器间计算时间越长。

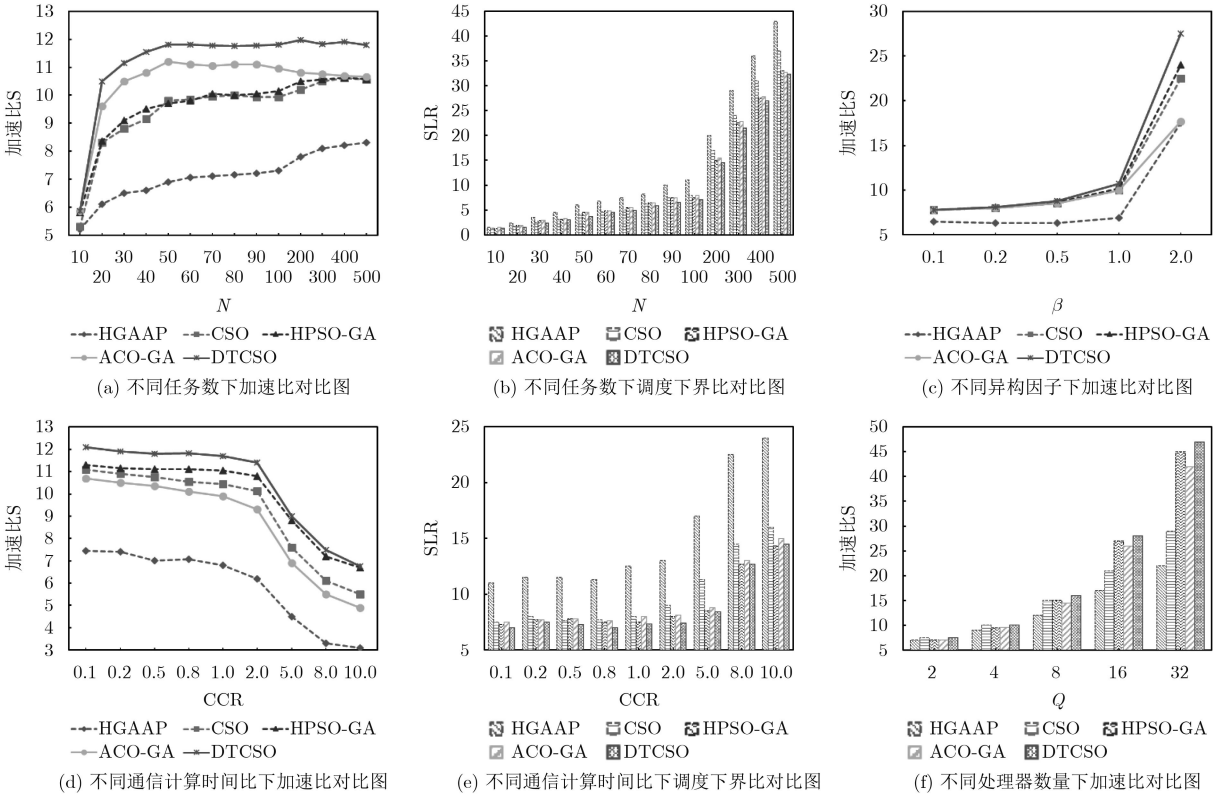
5.1.3 实验参数设置 表5中组合参数的设置会产生215000组DAG图，对于具有相同参数设置的DAG会随机生成25个具有不同通信和计算时间的随机图，即总共生成5375000个DAG图。

5.2 实验结果分析

5.2.1 不同算法加速比S和SLR的对比 图5(a)中DTCSO总体性能高于其他算法，当N=10时，DTCSO将加速比提高约13%；随着N增加，DTCSO的优势越明显，当N=40时，DTCSO加速比相对要高57%；当N>50时，DTCSO趋于收敛，但仍然优于其他算法，其他算法增加趋势缓慢甚至出现性能下降的情况。图5(b)也表明DTCSO在处理大量任务时性能最优。图5(c)表明DTCSO在异构环境中的有效性，β=2时，DTCSO性能比ACO-GA, HGAAP, CSO, HPSO-GA分别高57%, 59%, 46%, 21%，高度异构性有利于发挥DTCSO的优势。图5(d)中随着CCR的增加，加速比降低，表明通信和计算时间比越小越有利于任务调度，DTCSO的表现仍然最优，当CCR=2时，DTCSO比HPSO-GA, CSO, ACO-GA, HGAAP分别高6%, 8%, 17%, 63%，由图5(e)也能看出DTCSO的显著优势。图5(f)表明处理器数量的增加有利于提高加速比，同样DTCSO

表 5 DTCSO算法参数设置

参数	值
猫的数量	100
最大迭代次数	1000
权重w	0.5
c_1	1
r_1	[0,1]
MR	0.5
SMP	3
N	{10,20,30,40,50,60,70,80,90,100,200,300,400,500}
den	{0.2,0.5,0.8}
fat	{0.1,0.4,0.8}
reg	{0.2,0.5,0.8}
jump	{1,2,4}
CCR	{0.1,0.2,0.5,0.8,1.0,2.0,5.0,8.0,10.0}
β	{0.1,0.2,0.5,1.0,2.0}
Q	{2,4,8,16,32}



(a) 不同任务数下加速比对比图 (b) 不同任务数下调度下界比对比图 (c) 不同异构因子下加速比对比图

(d) 不同通信计算时间比下加速比对比图 (e) 不同通信计算时间比下调度下界比对比图 (f) 不同处理器数量下加速比对比图

图 5 算法性能对比图

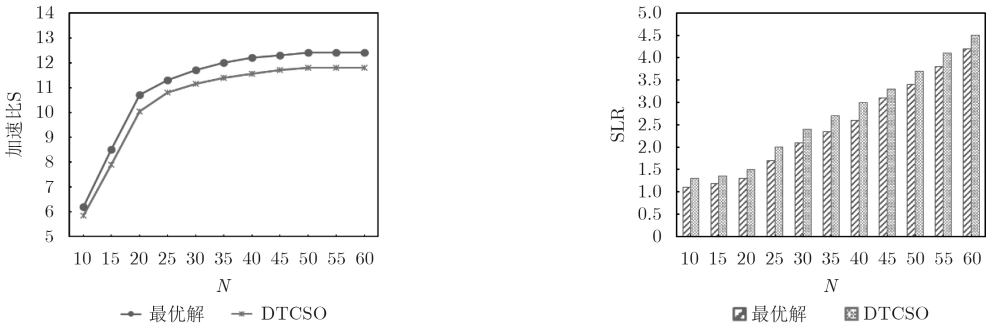
性能最佳。

图6是以任务数量N为例，取N={10,15,20,25,30,35,40,45,50,55,60}，对DTCSO算法实验结果和最优解进行对比。最优解是在同等任务数量下，将各个参数调节至最优，使结果达到最理想状态。由图中可以看出，虽然DTCSO算法与最优解仍有差距，但差距不是很大，且曲线走势大致相同。

5.2.2 算法优良百分比　表6是根据5375000个DAG图统计出的比较结果，由表6中第1行可以看出在绝大部分情况下，DTCSO要比其他算法获得的调度结果更好，这与图5所示内容一致。

6　结束语

本文针对异构计算系统中的任务调度问题，提出了一种基于双仲裁机制和田口正交法的猫群优化任务调度算法DTCSO，并验证其可行性和有效性。DTCSO算法利用双仲裁机制对任务资源进行管理，根据时间和处理器的情况来动态判决任务的分配，避免通信冲突，再将田口正交法应用到猫群优化方法的跟踪模式中，提高计算精度。仿真结果表明，DTCSO有效降低算法运行时间，增加最优解的可靠性，解质量进一步提高。



(a) 不同任务数下加速比对比图　　(b) 不同任务数下调度下界比对比图

图 6 DTCSO算法与最优解对比图

表 6　调度算法优良比较(%)

		HGAAP	CSO	HPSO-GA	ACO-GA
DTCSO算法	更好	100	98.6	93.62	86.7
	相同	0	0	0.08	0.6
	更差	0	1.4	6.3	12.7

参 考 文 献

[1] KHOKHAR A A, PRASANNA V K, SHAABAN M E, *et al*. Heterogeneous computing: Challenges and opportunities[J]. *Computer*, 1993, 26(6): 18–27. doi: 10.1109/2.214439.

[2] 朱晓敏, 贺川, 王建江,等. 异构计算系统中弹性节能调度策略研究[J]. 计算机学报, 2012, 35(6): 1313–1326. doi: 10.3724/SP.J.1016.2012.01313.
ZHU Xiaomin, HE Chuan, WANG Jianjiang, *et al*. An elastic energy-aware scheduling strategy for heterogeneous computing systems[J]. *Journal of Computer*, 2012, 35(6): 1313–1326. doi: 10.3724/SP.J.1016.2012.01313.

[3] MACHOVEC D, PASRICHA S, MACIELEWSKI A A, *et al*. Preemptive resource management for dynamically arriving tasks in an oversubscribed heterogeneous computing system[C]. IEEE, Parallel and Distributed Processing Symposium Workshops, Lake Buena Vista, USA, 2017: 54–64. doi: 10.1109/IPDPSW.2017.158.

[4] DAOUD M I and KHARMA N. A hybrid heuristic-genetic algorithm for task scheduling in heterogeneous processor networks[J]. *Journal of Parallel & Distributed Computing*, 2011, 71(11): 1518–1531. doi: 10.1016/j.jpdc.2011.05.005.

[5] DING Shan, WU Jinhui, XIE Guoqi, *et al*. A hybrid heuristic-genetic algorithm with adaptive parameters for static task scheduling in heterogeneous computing system[C]. The 14th IEEE International Conference on Embedded Software And Systems, Sydney, Australia, 2017: 761–766. doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.310.

[6] MORTEZA M and HADI S S. An efficient ACO-based algorithm for scheduling tasks onto dynamically reconfigurable hardware using TSP-likened construction graph[J]. *Applied Intelligence*, 2016, 45(3): 695–712. doi: 10.1007/s10489-016-0782-2.

[7] JING Chao. Ant-colony optimization based algorithm for energy-efficient scheduling on dynamically reconfigurable systems[C]. The Ninth IEEE International Conference on Frontier of Computer Science and Technology, Sydney, Australia, 2015: 127–134. doi: 10.1109/FCST.2015.10.

[8] KIANPISHEH S, CHARKARI N M, and KARGAHI M. Ant colony based constrained workflow scheduling for heterogeneous computing systems[J]. *Cluster Computing*, 2016, 19(3): 1–18. doi: 10.1007/s10586-016-0575-8.

[9] KUMAR N and VIDVARTHI D P. A novel hybrid

PSO–GA meta-heuristic for scheduling of DAG with communication on multiprocessor systems[J]. *Engineering with Computers*, 2016, 32(1): 35–47. doi: 10.1007/s00366-015-0396-z.

[10] WANG Hui, WU Zhij, RAHNAMAVAN S, *et al*. Enhancing particle swarm optimization using generalized opposition-based learning[J]. *Information Sciences*, 2011, 181(20): 4699–4714. doi: 10.1016/j.ins.2011.03.016.

[11] KUMAR Y and SAHOO G. An improved cat swarm optimization algorithm based on opposition-based learning and Cauchy operator for clustering[J]. *Journal of Information Processing Systems*, 2017, 13(4): 1000–1013. doi: 10.3745/JIPS.02.0022.

[12] JIANG Yunlian, SUN Guangzhong, WU Wentao, *et al*. Efficient communication contention aware scheduling in heterogeneous system[J]. *Journal of University of Science & Technology of China*, 2006, 8(8): 875–881. doi: 10.3969/j.issn.0253-2778.2006.08.015.

[13] BEAUMONT O, BOUDET V, and ROBERT Y. A realistic model and an efficient heuristic for scheduling with heterogeneous processors[C]. Proceeding 16th International Parallel and Distributed Processing Symposium, Ft.Lauderdale, USA, 2002: 1–14. doi: 10.1109/IPDPS.2002.1015663.

[14] WANG Yan, LI Kenli, and LI Keqin. Partition scheduling on heterogeneous multicore processors for multi-dimensional loops applications[J]. *International Journal of Parallel Programming*, 2016, 45(4): 1–26. doi: 10.1007/s10766-016-0445-2.

[15] LE D V, GO B S, SONG M G, *et al*. Mathematical design of a pulsed power induction coilgun system using the taguchi method[C]. IEEE 21st International Conference on Pulsed Power (PPC), Brighton, UK, 2017: 1–5. doi: 10.1109/PPC.2017.8291193.

[16] TSAI J T, FANG J C, and CHOU J H. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm[J]. *Computers & Operations Research*, 2013, 40(12): 3045–3055. doi: 10.1016/j.cor.2013.06.012.

[17] SAMARAWEERA L, THALAGALA S, GAMAGE P, *et al*. Optimization of green sand casting parameters using taguchi method to improve the surface quality of white cast iron grinding plates—A case study[C]. IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 2017: 1723–1727. doi: 10.1109/IEEM.2017.8290186.

[18] KUMAR B, KALRA M, and SINGH P. Discrete binary cat swarm optimization for scheduling workflow applications in cloud systems[C]. IEEE International Conference on Computational Intelligence & Communication Technology, Ghaziabad, India, 2017: 1–6. doi: 10.1109/CIACT.2017.7977296.

[19] GABI D, ISMAIL A S, ZAINAL A, *et al*. Cloud scalable multi-objective task scheduling algorithm for cloud computing using cat swarm optimization and simulated annealing[C]. International Conference on Information Technology, Amman, Jordan, 2017: 1007–1012. doi: 10.1109/ICITECH.2017.8080065.

[20] SUTER F and HUNOLD D S. A synthetic task graph generator[OL]. https://github.com/frs69wq/daggen.2017.4.

[21] TOPCUOGLU H, HARIRI S, and WU Minyou. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2002, 13(3): 260–274. doi: 10.1109/71.993206.

[22] ARABNELAD H and BARBOSA J G. List scheduling algorithm for heterogeneous systems by an optimistic cost table[J]. *IEEE Transactions on Parallel & Distributed Systems*, 2014, 25(3): 682–694. doi: 10.1109/TPDS.2013.57.

张兴明：男，1963年生，教授，主要研究方向为新型网络体系结构.

殷从月：女，1994年生，硕士生，研究方向为异构计算.

魏 帅：男，1984年生，讲师，主要研究方向为嵌入式计算.

叶盛钊：男，1994年生，硕士生，研究方向为拟态防御.

吕 平：女，1977年生，博士生，研究方向为芯片设计技术.