基于进化自适应蝙蝠算法的异构多核处理器任务调度

冯爽，江波，徐宏

（中国电子科技集团公司第三十二研究所，上海 201808）

摘　要： 异构多核处理器的任务调度问题已经被证明是一个 NP 完全问题。为满足复杂应用的计算需求，提高异构多核处理器的任务调度效率，提出一种基于进化自适应蝙蝠算法（EABA）的异构多核处理器任务调度算法。首先，对任务调度问题进行描述，并建立相应的数学模型；接着，设计任务分配编码方案和适应度函数，将所提算法映射到离散空间，使其能够适用于离散的异构多核处理器任务调度问题的研究；然后，为避免算法过早陷入局部最优，引入衰减脉冲策略和进化自适应变换策略；最后，设计仿真实验，将所提算法与蝙蝠算法（BA）、改进粒子群算法（IPSO）、人工鱼群算法（AFSA）、改进鲸鱼优化算法（IWOA）进行比较。实验结果表明，在中等规模任务（40～70 个）和大规模任务（80～100 个）下，EABA 算法的最优调度长度与次优算法相比分别缩短了 12.86%和 13.67%，算法平均执行时间分别减少了 14.51%和 13.50%。

关键词： 异构多核；任务调度；蝙蝠算法；进化自适应；有向无环图

中图分类号： TP391　　**文献标志码：** A　　**DOI：** 10.19678/j.issn.1000-3428.0068440

Task Scheduling for Heterogeneous Multi-Core Processors Based on Evolutionary Adaptive Bat Algorithm

FENG Shuang, JIANG Bo, XU Hong

(The 32nd Research Institute of China Electronics Technology Group Corporation, Shanghai 201808, China)

【Abstract】 The task scheduling problem for heterogeneous multi-core processors has been proven to be NP-complete. To meet the computational demands of complex applications and enhance the efficiency of task scheduling in heterogeneous multi-core processors, an Evolutionary Adaptive Bat Algorithm (EABA)-based task scheduling algorithm for heterogeneous multi-core processors is proposed. First, the task scheduling problem is described, and a corresponding mathematical model is established. Subsequently, a task allocation encoding scheme and a fitness function are designed to map the proposed algorithm into a discrete space, making it suitable for studying discrete task scheduling problems in heterogeneous multi-core processors. Subsequently, to prevent the algorithm from prematurely converging to the local optima, a decaying pulse strategy and an evolutionary adaptive transformation strategy are introduced. Finally, simulation experiments are designed to compare the proposed algorithm with the Bat Algorithm (BA), Improved Particle Swarm Optimization (IPSO) algorithm, Artificial Fish Swarm Algorithm (AFSA), and Improved Whale Optimization Algorithm (IWOA). The experimental results demonstrate that, under medium-scale tasks (40 to 70 tasks) and large-scale tasks (80 to 100 tasks), the optimal scheduling length of the EABA is shortened by 12.86% and 13.67%, respectively, compared with that of the suboptimal algorithm, with average execution time reductions of 14.51% and 13.50%, respectively.

【Key words】 heterogeneous multi-core; task scheduling; Bat Algorithm (BA); evolutionary adaptive; Directed Acyclic Graph (DAG)

0　引言

随着人工智能、大数据挖掘和物联网的迅速崛起，当今社会各行业、各领域已经开始向信息化、数字化和智能化的发展模式转变，传统的单一结构处理器已经难以满足复杂应用的多样化需求，因此，异构多核处理器成为未来处理器架构行业发展的主流。但在异构多核环境下，任务调度问题面临新的挑战，如何有效且合理地分配异构多核处理器中的任务，以充分发挥其性能优势，满足各种应用需求，已成为亟需研究的关键问题。任务调度问题已被证明是一个 NP 完全问题[1]，这意味着在多项式时间内找到最优解几乎不可能。目前，许多专家和学者仍在持续关注异构多核环境下的任务调度问题，该问题在未来很长时间内仍然会是异构多核技术领域的研究热点。

目前，在任务调度领域，针对异构多核处理器的研究主要集中在智能算法上，具体表现在启发式算

收稿日期： 2023-09-22　　**修回日期：** 2024-01-24

通信作者 E-mail： 786428943@qq.com

法[2]、群智能算法[3-5]、混合智能算法[6]等的应用。文献[7]提出了基于布谷鸟搜索的多处理器任务调度算法，文献[8]提出了一种异构多核处理器间任务调度的改进混合粒子群算法，文献[9]提出了基于蚁群优化算法的异构多核线程调度方法，文献[10]提出了基于正交自适应鲸鱼优化的云计算任务调度算法，文献[11]提出了基于麻雀搜索算法的异构多核处理器任务调度算法。

虽然粒子群算法、遗传算法等群智能算法通过不断搜索邻域来寻找最优解，希望在确定的迭代次数内找到更优解，但是这些算法均存在一定的缺陷[12]。遗传算法实现难度大，并且很容易陷入局部最优；蚁群算法由于需要存储和处理大量的路径信息和信息素信息，因此往往需要较长的时间才能收敛；粒子群算法高度依赖参数的选择，如粒子质量、惯性权重、加速度因子等，这在一定程度上限制了其收敛速度和搜索能力。

受蝙蝠在搜索目标时的行为的启发，研究人员设计出一种优化算法——蝙蝠算法。在该算法中，不仅考虑了蝙蝠个体的位置和速度，还考虑了其脉冲率、频率和脉冲响度，这里的脉冲率指蝙蝠个体发出脉冲的概率。在搜索目标的整个过程中，蝙蝠种群通过不断调节脉冲响度以及脉冲响度的变化快慢，改变搜索节奏，从而更新自己的位置和速度，直到找到更优解。在实际应用中，蝙蝠算法也确实表现出了出色的性能。本文基于标准蝙蝠算法的优化原理，设计一种任务分配的编码方案，并针对调度目标设计相应的适应度函数。在寻找最佳任务调度方案时，通过引入 2 种策略来避免算法陷入局部最优。

1 异构多核处理器任务调度研究

1.1 异构多核系统模型

异构多核系统模型由异构多核处理器架构模型(Processor Model)、任务调度模型(Task Model)、调度算法(Task Scheduling Algorithm)3 个部分组成[13-14]，各个模型之间的关系如图 1 所示。异构多核处理器架构模型将异构多核环境抽象成简化的数学模型，反映了处理器之间的连接方式、拓扑结构以及通信计算能力等信息；任务调度模型抽象成有向无环图(DAG)，通过节点以及节点之间的有向路径反映任务间的依赖关系和通信开销；调度算法在异构多核处理器架构模型和任务调度模型之间建立映射关系，按照约束规则统一进行任务分配和任务调度，形成任务调度序列结果，为系统执行任务提供合理的解决方案。

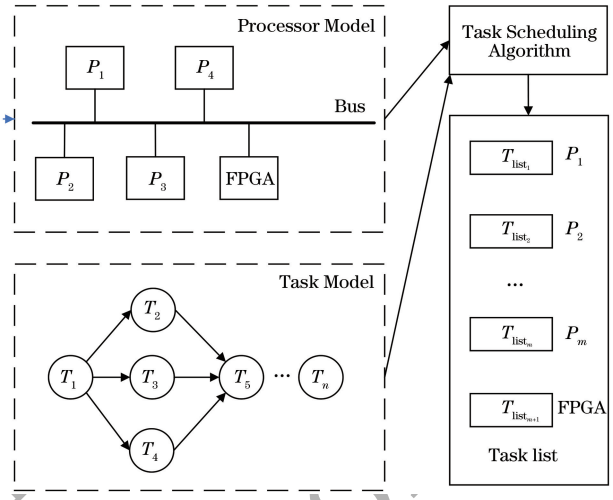

图 1 异构多核系统模型

Fig. 1 Heterogeneous multi-core system model

1.2 任务调度模型

在异构多核环境下，由于不同处理器在制造工艺、功耗控制、架构设计等方面存在差异，即便是针对同一任务，它们的处理计算能力也不尽相同。而对于不同任务来说，它们本身的属性已经决定了更适合在何种处理器上执行，处理器的选择在一定程度上决定了任务的最终完成时间。进一步地，通信模式、通信数据量、处理器之间的连接方式也是影响任务间通信开销的重要因素。

针对任务调度问题，本文将任务及任务间的约束关系抽象成 DAG 的形式。一般地，设一个异构多核系统模型的处理器数和任务数分别为 m 和 n，其中，m 个互为异构的处理器用集合 P 来描述，具体表示为 $P=\{P_1,P_2,\cdots,P_m\}$，n 个任务节点组成的集合表示为 $T=\{T_1,T_2,\cdots,T_n\}$，第 i 个任务表示为 $T_i(1\leqslant i\leqslant n)$。在 DAG 中，若 2 个任务节点之间存在一条带权有向边，那么这 2 个任务之间存在依赖关系，即它们必须按照先后顺序严格执行。每一条带权有向边上的权值代表前驱任务和后继任务之间的通信开销，本文把 DAG 中的通信开销集合描述为 $C=\{\cdots,C_{i,j},\cdots\}$。由于存在依赖关系的不同任务在同一处理器上执行时不需要数据传输，本文约定，在同一处理器上执行的有依赖关系的任务之间的通信开销为零，而在不同处理器上执行的存在依赖关系的任务，由于存在数据传输，因此通信开销不可忽略。每个任务在处理器上的实际执行时间称作计算开销，这个集合可以描述为 $W=\{\cdots,W_{T_i,P_j},\cdots\}$，其中，$W_{T_i,P_j}$ 表示任务 T_i 在处理器 P_j 上的实际执行时间，即计算开销。下面描述任务调度的属性：在 DAG 中，用 T_{entry} 表示入口任务节点，

用 T_{exit} 表示出口任务节点，用 $Pred(T_i)$ 表示任务 T_i 的前驱节点集合，用 $Succ(T_i)$ 表示任务 T_i 的后继节点集合。当 $Pred(T_i)=\varnothing$ 时，该任务节点为入口任务节点；当 $Succ(T_i)=\varnothing$ 时，该任务节点为出口任务节点。

如图 2 所示，给出一个由 7 个任务节点组成的 DAG，$T_1 \sim T_7$ 表示任务，节点 T_1 为入口任务节点，节点 T_7 为出口任务节点，有向边的权值为任务间的通信开销，任务 T_3 与任务 T_6 间的通信开销为 15。如表 1 所示，给出任务在不同处理器上执行的计算开销。

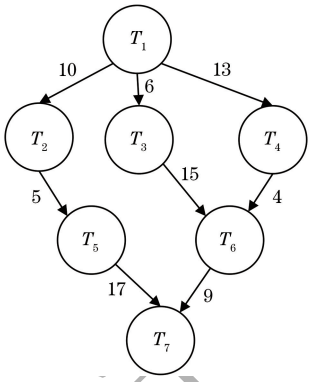

图 2 任务 DAG 示意图

Fig. 2 Schematic diagram of task DAG

表 1 任务在不同处理器上的执行时间

Table 1 Execution time of tasks on different processors

处理器	任务节点						
	T_1	T_2	T_3	T_4	T_5	T_6	T_7
P_1	20	16	8	9	15	12	11
P_2	23	26	13	12	17	6	7

综上，本文将异构多核处理器的任务调度问题描述为：在任务间存在依赖关系的前提下，将 n 个任务分配到 m 个异构处理器上，目标是求得任务调度序列的最优解，保证在执行过程中，最后一个任务的最晚完成时间最小。为此，本文作出以下基本假设：1）所有处理器都能正常工作，在任务调度过程中不会出现故障；2）在同一时刻，一个处理器只能执行一个任务，任务不允许被多个处理器同时执行；3）任务间严格遵守依赖约束，即每个任务总是在完成其全部的前驱任务后才被执行；4）不考虑突发紧急任务的切换或插入，即任务之间的优先级是事先确定的，不存在调度过程中任务优先级突然变更的情况[15-17]。

结合前文所述，出口任务开始执行时表示所有的直接前驱任务节点已执行完毕，出口任务的实际完成时间就是整个任务调度所花费的时间，用 $eft(T_i)$ 表示。若任务存在前驱任务节点，则任务完成的全部时间为任务 T_i 的直接前驱任务 T_j 的计算开销与它们之间通信开销和的最大值，计算公式可以表示为：

$$eft(T_i)=\begin{cases}C_{T_i,P_j}, & T_i=T_{entry}\\ \max\limits_{T_j\in Pred(T_i)}(eft(T_i,P_m))+C_{i,j}, & P_j\in P\end{cases} \tag{1}$$

式中：$\max\limits_{T_j\in Pred(T_i)}(eft(T_i,P_m))$ 表示任务 T_i 和其前驱任务 T_j 之间的通信结束时间。在满足基本假设的条件下，根据任务优先级，将任务合理安排到各个处理器上进行调度，使任务的完成时间尽可能短，调度长度最短的方案即为所求解。

2 进化自适应蝙蝠算法

2.1 算法描述

蝙蝠算法是一种受蝙蝠在搜索目标时的行为启发而设计的群智能算法[18]，它通过对每个蝙蝠个体的速度、位置、脉冲率、脉冲响度等参数进行不断迭代更新来优化最终解。蝙蝠通过回声定位的行为来搜索并探测目标，从而完成觅食或避障的行为。蝙蝠在搜索目标时先发射具有一定响度的脉冲信号，当脉冲信号遇到障碍物或猎物时会以一定的速度反射回来，蝙蝠会根据反射信号的响度、时延等信息来判断障碍物或猎物与自己当前位置的距离、方向等信息。蝙蝠算法的主要思想是将蝙蝠个体看作优化问题解空间中的智能搜索体，根据具体数学问题设计合理的适应度函数，以求解每只蝙蝠个体的适应度值，结合适应度值的含义和数值大小确定其位置的优劣。整个搜索过程是一个持续迭代优化的过程，在过程中每只蝙蝠个体的信息会发生更新，每次迭代结束前会寻得一个问题解，若问题解更优，则替换已经求得的前面较差的解，否则不作改变[19-20]。

在优化问题的研究上，蝙蝠算法已经表现出收敛速度快、寻优效果好等特点，也得到了广泛应用。然而，由于搜索策略的不足，标准蝙蝠算法很容易陷入局部最优，目前针对搜索策略的研究也相对较少。本文以搜索策略为切入点，提出一种进化自适应蝙蝠算法（EABA）。该算法通过引入衰减脉冲策略，在迭代的过程中逐渐减小脉冲率和脉冲响度，增加算法搜索的广度，并减少算法陷入局部最优的风险。另外，本文还结合了当前蝙蝠的进化自适应信息，来引导蝙蝠种群的搜索方向，增加搜索目标的多样性。上述 2 种策略都在蝙蝠算法的基础上进行改进，旨

在提升算法的寻优性能。

为了模拟蝙蝠在自然环境中的觅食和避障行为，本文作出如下假设：蝙蝠采用回声定位法探测周围环境，搜索目标，能够辨别周围环境中的猎物和障碍物并计算出相应的距离。

设第 i 只蝙蝠所在的位置为 x_i，飞行速度为 v_i，频率为 f_i，蝙蝠种群最大迭代次数为 $T_{\max}$。蝙蝠探测到目标的距离信息后，能根据信息动态调节自身发射的超声波脉冲响度 A_i、频率 f_i 以及脉冲率 $r_i \in [0,1]$。

基于上述假设，蝙蝠群体在 D 维空间中觅食或避障时，速度和位置的更新公式分别如下：

$$v_i^{t+1} = v_i^t + (x_i^t - x^*)A_i^t \tag{2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{3}$$

式中：v_i^{t+1} 和 v_i^t 表示在种群分别迭代到第 $t+1$ 代和第 t 代时第 i 只蝙蝠的速度；x_i^{t+1} 和 x_i^t 表示在种群分别迭代到第 $t+1$ 代和第 t 代时第 i 只蝙蝠的位置；x^* 表示当前最佳位置；A_i^t 表示在种群迭代到第 t 代时第 i 只蝙蝠的脉冲响度。

蝙蝠在搜索目标的过程中，需要不断地调整脉冲响度和脉冲率来准确掌握目标的实时位置。基于此，本文引入了 2 种策略来实现标准蝙蝠算法的改进，使其能够自适应地找到更优解。

2.2 衰减脉冲策略

衰减脉冲策略的核心思想是随着迭代过程的进行，逐渐减小脉冲率和脉冲响度的值。在优化的早期，设置一个较高的脉冲率值，有助于蝙蝠个体探索搜索空间。随着搜索的进行，脉冲率逐渐下降，使得蝙蝠个体更趋向于局部范围的搜索，从而有助于在搜索目标的过程中找到更优解。这种策略可以使算法在开始时更加“激进”，实现大面积探测搜索空间的目的。随着迭代次数的增加，逐渐减小搜索步幅，以便准确地寻找到更优解。脉冲率衰减公式为：

$$r^{t+1} = \max\left(r_{\min}, r_0\left(1 - \frac{t+1}{T_{\max}}\right)\right) \tag{4}$$

式中：r^{t+1} 是第 $t+1$ 代的脉冲率；r_0 是初始脉冲率；$r_{\min}$ 是最小脉冲率；$T_{\max}$ 是种群最大迭代次数。

脉冲响度衰减公式如下：

$$A^{t+1} = \frac{A^t}{1+\alpha t} \tag{5}$$

式中：A^t 和 A^{t+1} 分别是第 t 代和第 $t+1$ 代的脉冲响度；$\alpha \in [0,1]$ 是响度衰减率。

2.3 进化自适应变换策略

进化自适应变换策略的核心思想是通过利用当前蝙蝠的适应度值信息来引导搜索的方向。在 EABA 算法中，引入适应度值信息的比较，提出一种进化自适应变换策略，这里的自适应变换指搜索模式的切换。在每次进化（迭代）时，首先将所有蝙蝠个体按照适应度值从大到小的顺序进行排列，确定蝙蝠的适应度值序列。一般来说，全局最优位置会出现在适应度值较优的个体附近。本文设置一个进化自适应因子 T_r，对于每一代的蝙蝠个体，采取混合搜索模式，即一部分蝙蝠个体采用全局搜索模式，另一部分蝙蝠个体采用局部搜索模式。具体地，对于某只蝙蝠个体来说，若其排名高于 $T_r T_{\max}$，则采用全局搜索模式，否则采用局部搜索模式。

群智能算法的寻优特点是迭代前期往往需要大面积、长步数的全局搜索来确定较优位置，随着迭代次数的增加，搜索步长逐渐减小，专注于对较优位置的进一步优化。对于蝙蝠算法而言，在迭代初期，大多数蝙蝠个体进行大面积全局搜索，而迭代后期则需要通过扰动进行局部寻优。因此，T_r 应随着进化代数的增加而递增，其计算公式如下：

$$T_r = T_{r,\min} + (T_{r,\max} - T_{r,\min})\frac{t-1}{T_{\max}-1} \tag{6}$$

式中：$T_{r,\min}$ 和 $T_{r,\max}$ 分别是进化自适应因子 T_r 的最小值和最大值；t 表示当前迭代次数。

3 基于 EABA 的异构多核处理器任务调度

3.1 编码方案

异构多核处理器任务调度问题是一个典型的组合优化问题，在解决此类问题时，往往要考虑离散性、目标函数、约束条件等特点[21-23]。考虑到启发式算法的特性，本文采用实数编码策略，利用矢量信息表示每只蝙蝠个体，解的具体维数取决于问题本身。因此，在蝙蝠算法求解任务调度问题之前，必须确立一种合适的编码机制，以将蝙蝠个体的位置信息转化为具有确定拓扑结构的任务调度序列。针对异构多核处理器任务调度的需求，本文采用了一种基于任务优先级的编码方案，首先对任务 T_i 赋予 1～n 之间不同的随机整数，这个数值越大，表示任务 T_i 的优先级越高。这个编码过程实际上是一个从蝙蝠位置向量 $\boldsymbol{X}_i = [X_{i,1}, X_{i,2}, \cdots, X_{i,d}]$ 到任务优先级序列 $\pi = (p_1, p_2, \cdots, p_d)$ 的转化过程。一般地，实数编码可以保证解的可行性。表 2 给出了 7 个任务的任务分配编码方案。

表 2 任务分配编码方案

Table 2 Coding scheme for task allocation

任务节点	优先权
T_1	5
T_2	4
T_3	1
T_4	7
T_5	2
T_6	6
T_7	3

表 3 任务分配解码方案

Table 3 Decoding scheme for task allocation

任务节点	处理器
T_1	P_2
T_2	P_2
T_3	P_1
T_4	P_2
T_5	P_1
T_6	P_1
T_7	P_2

3.2 解码方案

解码方案是编码方案的逆过程，解码方案实际上完成了从搜索空间回到问题空间的转换。设计解码方案的关键是设计适应度函数，本研究关注的是任务调度问题，因此，选择所有任务完成的最终时间（即调度长度）作为衡量适应度函数的标准。对于蝙蝠种群来说，初始状态时每只蝙蝠个体都会获得一个随机位置信息，每轮迭代结束后，蝙蝠个体的位置信息得到更新，通过适应度函数计算出适应度值，根据适应度值的大小来确定蝙蝠的最佳位置，适应度值决定了每只蝙蝠个体位置的优劣度。调度长度的结果取决于任务调度序列结果，一旦确定了任务调度序列，调度长度很容易就能计算得出。进一步地，在确定任务调度序列之前，需要设计基于任务优先权的解码方案。通过逆向分析，调度长度的计算最终抽象成确定任务间的依赖关系和任务的优先级。本文根据任务间的依赖关系和任务的优先级来确定任务调度序列，从而确保任务调度的顺序性。在满足任务约束的条件下，处理器对于任务的调度严格按照优先级顺序来工作，在任务选择处理器阶段，为了得到每个任务的分配结果，本文主要通过对任务优先权值与处理器数目进行取余操作。目标函数公式为：

$$f(x)=\max\{\mathrm{eft}(T_i)\},\ i=1,2,\cdots,n \tag{7}$$

式中：$\mathrm{eft}(T_i)$ 表示任务 T_i 的最早执行完成时间，具体计算公式见式(1)。

任务调度长度计算实际上就是问题的编解码过程。任务的调度要严格按照任务优先级顺序以及依赖关系进行，每个任务在被调度时，它的前驱任务一定已经全部执行完成；在满足任务间约束依赖的条件下，高优先级任务总是先于低优先级任务执行。结合以上分析，可以得到任务调度结果。表 2 中的编码方案对应的解码方案如表 3 所示。

3.3 EABA 异构多核处理器任务调度算法流程

如图 3 所示，基于进化自适应蝙蝠算法的异构多核处理器任务调度算法的具体步骤如下：

步骤 1 将 DAG 和计算开销作为输入，根据优先级规则确定调度序列。

步骤 2 初始化 EABA 参数，包括蝙蝠种群规模 M、最大迭代次数 $T_{\max}$、初始脉冲强度 A_0、响度衰减率 α、进化自适应因子的最大值 $T_{r,\max}$ 和最小值 $T_{r,\min}$ 等。

步骤 3 计算每只蝙蝠个体的适应度值，将每只蝙蝠个体按照适应度值从大到小的顺序排列，记录每只蝙蝠个体的排名。计算进化自适应因子 T_r，确定每只蝙蝠个体的搜索模式。对每只蝙蝠个体，根据其当前位置和速度以及与其他蝙蝠的交互，更新其位置和速度。

步骤 4 计算每只蝙蝠个体的适应度值，根据蝙蝠的适应度值的大小，更新全局最优蝙蝠的位置和适应度值。

步骤 5 更新蝙蝠的脉冲响度和脉冲率，使其随着迭代次数的增加逐渐减小，使蝙蝠的探索范围逐渐缩小，探索能力逐渐减弱。

步骤 6 重复步骤 3～步骤 5，不断寻找更高质量的解，直到迭代次数达到最大，算法结束。

在异构多核处理器任务调度中，输入有向无环图 DAG 和计算开销，DAG 表示了各个任务之间的依赖关系以及通信开销。基于 DAG，根据确定的优先级规则来初始化任务的调度序列（步骤 1）。对于 EABA 参数的初始化，蝙蝠种群的规模和最大迭代次数取决于任务数和处理器数，根据经验，种群规模一般设置为任务数的 10～15 倍；初始脉冲强度和响度衰减率这 2 个参数是控制蝙蝠行为的核心特性，进化自适应因子的最大值和最小值会对蝙蝠的搜索模式产生影响（步骤 2）。接着，动态地模拟蝙蝠在搜索空间中的飞行行为，计算每只蝙蝠个体的适应度值，并根据这些值对蝙蝠进行排序。适应度值反

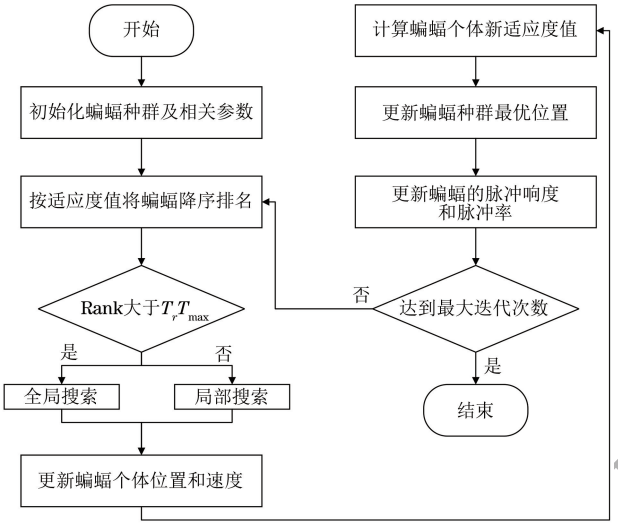

图 3 EABA 算法流程

Fig. 3 Procedure of the EABA algorithm

映了每只蝙蝠当前位置(即任务分配方案)的效果好坏。基于适应度值和进化自适应因子,确定每只蝙蝠的搜索模式。根据当前位置、速度和交互情况,更新每只蝙蝠的位置和速度(步骤 3)。再次计算每只蝙蝠的适应度值,并用来更新全局最优蝙蝠的位置和适应度值,确保算法始终保持对当前最好解的追踪,即使在探索新的可能解时也不会丢失已经找到的好解(步骤 4)。随着算法迭代的进行,蝙蝠的脉冲响度和脉冲率逐渐减小,这意味着随着时间的推移,蝙蝠的探索范围和探索能力逐渐减弱,从而使算法逐渐从广泛探索转向更集中地优化当前最优解(步骤 5)。重复执行步骤 3～步骤 5,不断寻找和优化最优解。每次迭代都可能改进当前的任务调度结果。当达到预设的最大迭代次数时,算法结束,此时,全局最优蝙蝠的位置代表了算法找到的最佳任务调度方案(步骤 6)。

4 实验仿真与分析

本文实验仿真使用 Matlab 2021a 完成,实验的输入为任务数、处理器数、任务 DAG、计算开销、通信开销。为了验证基于 EABA 任务调度算法的有效性,将本文所提方法与 BA[19]、AFSA[24]、IWOA[25]、IPSO[8]等算法进行比较。设置种群规模为 100,最大迭代次数为 200。

本实验中 EABA 算法的仿真参数设置如下:初始脉冲率取 0.9,最小脉冲率取 0.1,响度衰减率取 0.1,进化自适应因子的最小值和最大值分别取 0.3 和 0.7。

针对 10 个随机任务在 4 个处理器中的任务调度问题,图 4 给出了 5 种不同算法的迭代进化曲线。由图 4 可知,与其他 4 种算法相比,EABA 算法找到更优解的迭代次数相对较少。在第 26 代后,标准蝙蝠算法 BA 陷入了局部最优,适应度值保持不变。在第 153 代后,AFSA 算法陷入局部最优。在第 2 代后,IWOA 算法陷入局部最优。在第 16 代以后,IPSO 算法陷入局部最优。而 EABA 通过引入衰减脉冲策略和进化自适应变换策略,成功避免了算法过早陷入局部最优的情况,找到了更优解,这也进一步证明了本文算法的优势。

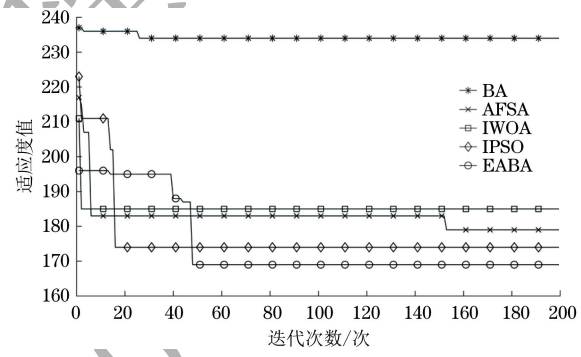

图 4 任务数为 10 时不同算法的迭代进化曲线

Fig. 4 Iterative evolution curves of different algorithms when the number of tasks is 10

在实验过程中,为了不失一般性,本文在不同任务量下均进行 15 次实验,并基于实验结果的平均值进行分析。图 5 展示了在不同任务规模下各个算法在处理器上的最优调度长度结果。由图 5 可知:在任务数量少于 40 的情况下,EABA 算法相较于其他 4 种算法在最优调度长度上略显优势;随着任务规模的增加,各算法得到的最优调度长度结果差距逐渐增大,其中 EABA 算法的优势更加明显,在中等规模任务(40～70 个)和大规模任务(80～100 个)下,本文算法最优调度长度比次优算法分别缩短了 12.86%和 13.67%,这也进一步证实了本文 EABA 算法能够寻得更高质量的解以及能够发挥出良好的稳定性。

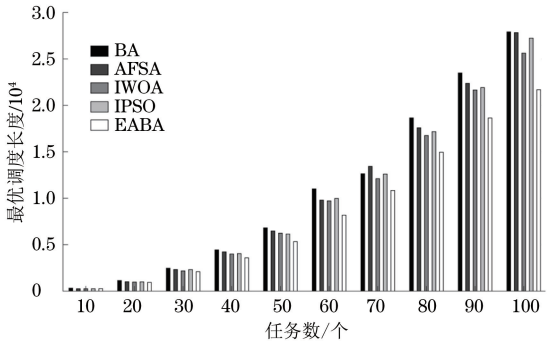

图 5 不同任务规模下的最优调度长度对比

Fig. 5 Comparison of optimal scheduling length under different task sizes

图 6 为不同任务规模下 5 种算法的平均执行时间对比。由图 6 可知，相较于其他算法，EABA 的平均执行时间始终最少，当任务数少于 30 时，EABA 算法平均执行时间与其他算法差别较小，当任务数超过 30 时，随着任务数的增加，EABA 算法的平均执行时间开始与其他算法拉开差距，在中等规模任务（40～70 个）和大规模任务（80～100 个）下，EABA 算法平均执行时间比次优算法分别减少了 14.51%和 13.50%。

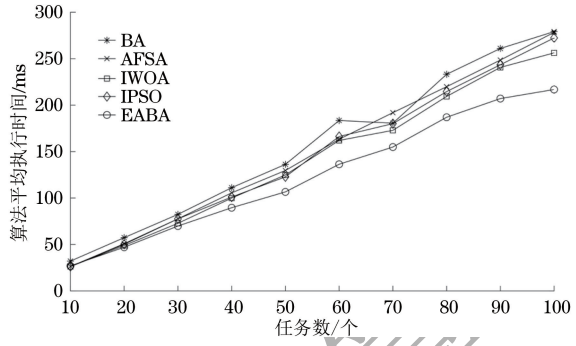

图 6 不同任务规模下的算法平均执行时间对比

Fig.6 Comparison of average execution time of algorithms under different task sizes

通过进化自适应机制，EABA 能够在全局探索（寻找新的可能解）和局部开发（优化已知解）之间实现有效的平衡，能够在探索解空间的同时保持对当前最优解的精细调整。在上述实验结果中，EABA 算法表现出了比 BA、IPSO、AFSA、IWOA 更好的性能，主要归功于衰减脉冲策略和进化自适应变换策略的有效性。综上，在不同的任务规模下，本文所提 EABA 任务调度算法在处理异构多核处理器任务调度问题时可以发挥出更好的性能，寻得质量更优的解。

5 结束语

为满足复杂应用的计算需求，同时提高异构多核处理器的任务调度性能，本文对蝙蝠算法进行改进，提出一种基于进化自适应蝙蝠算法的异构多核处理器任务调度算法。一方面，通过引入衰减脉冲策略，在迭代的过程中逐渐减小脉冲率和脉冲响度，使得算法搜索的广度增加，减少算法陷入局部最优的风险；另一方面，引入进化自适应变换策略，充分利用蝙蝠个体的适应度值信息来引导蝙蝠种群的搜索方向，从而增加搜索目标的多样性，提升算法的寻优性能。为验证 EABA 的有效性，将其与 BA 等 4 种算法进行对比，实验结果表明，EABA 在求解异构多核处理器任务调度问题时能够寻得质量更优的解。在下一步的研究中，将考虑如何提高 EABA 算法的收敛速度，同时在更加复杂的情况（如考虑能耗等因素的影响）下，探究异构多核处理器的任务调度性能。

参考文献

[1] ULLMAN J D. NP-complete scheduling problems [J]. Journal of Computer and System Sciences, 1975, 10(3): 384-393.

[2] SAHOO R M, KUMARI PADHY S. A novel algorithm for priority-based task scheduling on a multiprocessor heterogeneous system [J]. Microprocessors and Microsystems, 2022, 95: 104685.

[3] HAMED A Y, ALKINANI M H. Task scheduling optimization in cloud computing based on genetic algorithms[J]. Computers, Materials & Continua, 2021, 69(3): 3289-3301.

[4] ABUALIGAH L, ALKHRABSHEH M. Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing[J]. The Journal of Supercomputing, 2022, 78(1): 740-765.

[5] SIDDESHA K, JAYARAMAIAH G V. Energy-aware task scheduling approach using DVFS and particle swarm optimization for heterogeneous multicore processors [EB/OL]. [2023-09-10]. https://link.springer.com/chapter/10.1007/978-981-16-1342-5_75.

[6] KHAN M S A, SANTHOSH R. Task scheduling in cloud computing using hybrid optimization algorithm [J]. Soft Computing, 2022, 26(23): 13069-13079.

[7] 杨辉华，张晓凤，谢谱模，等. 基于布谷鸟搜索的多处理器任务调度算法[J]. 计算机科学，2015，42(1)：86-89.
YANG H H, ZHANG X F, XIE P M, et al. Multiprocessor task scheduling method based on cuckoo search algorithm [J]. Computer Science, 2015, 42(1): 86-89. (in Chinese)

[8] PIROZMAND P, JALALINEJAD H, HOSSEINABADI A A R, et al. An improved particle swarm optimization algorithm for task scheduling in cloud computing[J]. Journal of Ambient Intelligence and Humanized Computing, 2023, 14(4): 4313-4327.

[9] 李静梅，张大虎，吴艳霞，等. 基于蚁群优化算法的异构多核线程调度方法[J]. 计算机工程与设计，2014，35(6)：1946-1950.
LI J M, ZHANG D H, WU Y X, et al. Heterogeneous multi-core thread scheduling method based on ant colony optimization algorithm [J]. Computer Engineering and Design, 2014, 35(6): 1946-1950. (in Chinese)

[10] 张金泉，徐寿伟，李信诚，等. 基于正交自适应鲸鱼优化的云计算任务调度[J]. 计算机应用，2022，42(5)：1516-1523.
ZHANG J Q, XU S W, LI X C, et al. Cloud computing task scheduling based on orthogonal adaptive whale optimization[J]. Journal of Computer Applications, 2022, 42(5): 1516-1523. (in Chinese)

[11] 程小辉，童辉辉，康燕萍. 基于麻雀搜索算法的异构多核处理器任务调度[J]. 计算机应用与软件，2023，40(4)：211-216.
CHENG X H, TONG H H, KANG Y P. Heterogeneous multiprocessor task scheduling based on sparrow search algorithm[J]. Computer Applications and Software, 2023, 40(4): 211-216. (in Chinese)

[12] CLERC M, KENNEDY J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(1): 58-73.

[13] 蒋清源. FPGA 计算资源约束下的异构多核系统任务调度算法设计与实现[D]. 上海：华东师范大学，2022.
JIANG Q Y. Design and implementation of task scheduling

algorithm for heterogeneous multi-core system under the constraint of FPGA computing resources[D]. Shanghai: East China Normal University, 2022. (in Chinese)
[14] 梁秋玲，张向利，张红梅，等. 基于多核处理器的关联任务并行感知调度算法[J]. 计算机工程，2021，47(7)：212-217.
LIANG Q L, ZHANG X L, ZHANG H M, et al. Parallel perceptual scheduling algorithm for related tasks based on multi-core processors[J]. Computer Engineering, 2021, 47(7): 212-217. (in Chinese)
[15] KIM S I, KIM J K. A method to construct task scheduling algorithms for heterogeneous multi-core systems[J]. IEEE Access, 2019, 7: 142640-142651.
[16] PRADHAN R, SATAPATHY S C. Energy-aware cloud task scheduling algorithm in heterogeneous multi-cloud environment[J]. Intelligent Decision Technologies, 2022, 16(2): 279-284.
[17] XIE G Q, ZENG G, LIU L J, et al. Mixed real-time scheduling of multiple DAGs-based applications on heterogeneous multi-core processors[J]. Microprocessors and Microsystems, 2016, 47: 93-103.
[18] YANG X S. A new metaheuristic bat-inspired algorithm[EB/OL]. [2023-09-10]. https://link.springer.com/chapter/10.1007/978-3-642-12538-6_6.
[19] GANDOMI A H, YANG X S. Chaotic bat algorithm[J]. Journal of Computational Science, 2014, 5(2): 224-232.
[20] CAI X J, WANG L, KANG Q, et al. Bat algorithm with Gaussian walk[J]. International Journal of Bio-Inspired Computation, 2014, 6(3): 166-174.
[21] DENG Z, SHEN H, CAO D, et al. Task scheduling on heterogeneous multiprocessor systems through coherent data allocation[J]. Concurrency and Computation: Practice and Experience, 2021, 33(10): e6183.
[22] CHEN J C, HE Y, ZHANG Y, et al. Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems[J]. Journal of Systems Architecture, 2022, 129: 102598.
[23] ELCOCK J, EDWARD N. An efficient ACO-based algorithm for task scheduling in heterogeneous multiprocessing environments[J]. Array, 2023, 17: 100280.
[24] HASAN R A, ALHAYALI R A I, MOHAMMED M A, et al. An improved fish swarm algorithm to assign tasks and cut down on latency in cloud computing[J]. TELKOMNIKA (Telecommunication Computing Electronics and Control), 2022, 20(5): 1103.
[25] CHAKRABORTY S, SAHA A K, CHHABRA A. Improving whale optimization algorithm with elite strategy and its application to engineering-design and cloud task scheduling problems[J]. Cognitive Computation, 2023, 15(5): 1497-1525.

编辑 吴云芳