改进 DPSO 算法求解仿真任务调度问题

郑 伟 王 磊 曹建蜀
(电子科技大学电子科学技术研究院，四川成都 611731)

摘 要：针对离散粒子群算法在求解雷达分布式仿真系统中的仿真任务调度时，由于其易陷入局部最优的缺陷导致算法受初始种群的影响较大且结果稳定低的问题，提出基于信息素变异策略的改进离散粒子群算法。文中分析了离散粒子群算法容易陷入局部最优的原因，引入基于信息素的变异策略，充分利用种群中所有粒子的寻优经验信息来累计信息素，以信息素的分布和效率矩阵为依据对基本离散粒子群算法每次迭代后得到的粒子进行变异操作。仿真结果表明，改进算法有效地避免了算法陷于局部最优的问题，且结果的稳定性比基本离散粒子群算法更好，调度跨度和负载平衡度相比离散粒子群算法，蚁群算法，Max-Min 算法和 Min-Min 算法都有明显的改善。

关键词：雷达分布式仿真；任务调度；离散粒子群算法；信息素；变异策略
中图分类号：TP338 **文献标识码**：A **文章编号**：1003-0530(2015)04-0474-09

Apply Modified Discrete Particle Swarm Optimization Algorithm to Solve Simulation Task Scheduling

ZHENG Wei WANG Lei CAO Jian-shu
(Research Institute of Electronic Science and Technology, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China)

Abstract: Aiming at the problems that the algorithm was greatly influenced by initial population and the result was not stable when solving the simulation task scheduling in distributed radar simulation system, due to the defects that the discrete particle swarm optimization (DPSO) algorithm was easy to encounter local optimization, a modified DPSO algorithm based on the mutation strategy of pheromone was proposed. The mutation strategy of pheromone was first introduced in this paper after analysis the cause of high probability to encounter local optimization. Then the optimization empirical information of all particles in swarm was fully used to accumulate pheromone. Finally, mutation operation was conducted on particles obtained from iteration of DPSO according to the distribution of pheromone and efficiency matrix. Simulation results show the effectiveness to avoid the problem of local optimization and the better stability of results than DPSO. Meanwhile, the scheduling span and load balance are both improved greatly compared with DPSO, ACO, Max-Min and Min-Min.

Key words: radar distributed simulation; task scheduling; discrete particle swarm optimization; pheromone; mutation strategy

1 引言

在雷达分布式仿真系统中，仿真任务和处理节点具有较大的异构性，仿真任务的调度算法将影响仿真任务的完成时间和分布式仿真系统的负载平衡，从而影响仿真系统的性能。因此，任务调度成为雷达分布式仿真中的研究热点问题。任务调度问题已被证明是 NP 完全问题，多采用启发式算法求解。其中离散粒子群(Discrete Particle Swarm Optimization, DPSO)算法[1]在求解任务调度问题上具有良好的有效性。

Kennedy 和 Eberhart 于 1997 提出二进制版本的

收稿日期：2014-09-22；修回日期：2015-01-09

离散粒子群算法[2],使得粒子群算法在离散优化问题中得到广泛的应用,如整数规划问题、n 皇后问题、TSP 问题、Job-Shop 问题以及任务调度问题等。离散粒子群算法具有概念简明、搜索速度快、计算效率高、易于实现以及可并行处理等优点,但也存在容易陷入局部最优[3]的问题,具体表现为早熟收敛和种群多样性快速降低,针对这一问题,目前研究学者提出了多种改进方法。文献[4]采用自适应的模糊惯性权重,通过控制粒子"飞行"的速度来控制影响算法收敛速度,克服了 DPSO 的早熟收敛问题。文献[5]引入混沌理论,通过 Logistic 系统方程产生初始种群,使得初始粒子具有比较明显的差异且分布均匀,从而提高了种群的多样性降低了算法陷入局部最优的概率。文献[6]提出多种群的离散粒子群算法,利用 DPSO 的并行性,采用多个种群独立搜索共享全局最优粒子信息的方法,间接提高整个种群的多样性,从而降低算法陷入局部最优的可能性,提高解的质量。文献[7]从保持粒子多样性入手,引入鲶鱼效应和交叉变异操作,从避免了早熟收敛问题获得更好的全局寻优能力。以上这些改进均是从算法的早熟收敛和种群多样性下降快这两个表象入手。文献[8]提出根据所有个体最优值建立概率模型,新的粒子依据此概率模型抽样产生,使得粒子每一次更新时都向全部个体的历史最优信息学习而不是仅向自身个体历史最优学习,从而降低算法陷入局部最优的可能,该算法是从增加了粒子间共享的信息量出发,但仍是以种群中的"精英"粒子引导所有粒子的寻优过程。

通过分析表明 DPSO 算法易陷入局部最优的根本原因是其"精英吸引"策略,以此为基础,提出一种基于信息素变异策略的改进离散粒子群算法:以信息素的形式保存所有粒子的寻优经验信息,并以变异的方式反映这些信息对粒子寻优的影响,并兼顾 DPSO 的"精英吸引"策略,从而保持较快的收敛速度却又避免陷入局部最优。通过仿真实验对比,改进算法有效地避免了算法陷于局部最优,提高了算法解的质量和稳定性。

2　雷达仿真任务调度问题模型

对于有 n 个独立的仿真任务和 m 个仿真处理节点的雷达分布式仿真系统,任务调度的实质就是将这 n 个仿真任务以合理的方式调度到 m 个处理节点上执行,目的是最小化调度跨度(Scheduling MakeSpan),即使得 n 个仿真任务总的完成时间最短。因此,任务调度就可以描述为在一定的约束条件下任务集合 $\mathbf{T}$ 到处理节点集合 $\mathbf{P}$ 上的映射 $f: T_i \rightarrow P_j$。其中:任务集合 $\mathbf{T}=\{T_1, T_2, \cdots, T_n\}$ 来表示仿真任务,T_i 表示仿真任务 i 的计算量,且假设仿真任务不可再分解,都是非周期、不可抢占的,一个任务只能在一个处理节点上执行;处理节点集合 $\mathbf{P}=\{P_1, P_2, \cdots, P_m\}$ 来表示仿真处理节点,P_j 表示处理节点 j 的处理速度,且假设同一时刻同一个处理节点上只能有一个任务执行。

映射的结果用一个 $n\times m$ 的调度矩阵 $\boldsymbol{X}$ 表示映射结果,矩阵 $\boldsymbol{X}$ 的元素 $x_{ij}\in\{0,1\}$,若任务 T_i 被分配到处理节点 P_j 上执行,则 $x_{ij}=1$,否则 $x_{ij}=0$,调度矩阵应满足每一行的元素之和为 1,以保证为每个仿真任务都分配了且只分配了一个仿真处理节点。

n 个仿真任务在 m 个仿真处理节点上的估计执行时间(Estimated Time to Complete,ETC)用一个 $n\times m$ 的矩阵 ETC 表示,其元素 t_{ij} 表示任务 T_i 在仿真处理节点 P_j 上的预测处理时间。

仿真处理节点 j 的处理时间 CT_j 是调度到其上所有任务的处理时间之和,即:

$$\mathrm{CT}_j=\sum_{i=1}^{n} x_{ij}t_{ij} \tag{1}$$

调度长度是指所有仿真处理节点处理时间的最大值。则仿真任务调度问题的数学模型为:

$$\min \quad \mathrm{MakeSpan}=\max_{j=1}^{m}(\mathrm{CT}_j) \tag{2}$$

$$\text{s.t.}\begin{cases}\mathrm{CT}_j=\sum\limits_{i=1}^{n} x_{ij}t_{ij}\\ x_{ij}\in\{0,1\}\\ \sum\limits_{j=1}^{m} x_{ij}=1\\ i\in\{1,2,\cdots,n\},\ j\in\{1,2,\cdots,m\}\end{cases}$$

3　基本离散粒子群算法求解任务调度问题

基本离散粒子群算法求解任务调度问题关键是要根据任务调度问题的研究内容定义粒子编码、位置和速度的更新公式,以及适应度函数。

(1)粒子编码

粒子编码分为粒子的位置和速度两部分。每一个粒子都代表一个任务调度问题的可行解,因此粒子 k 的位置定义为 $n\times m$ 的调度矩阵 $\boldsymbol{X}$:

$$\boldsymbol{X}_k=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \tag{3}$$

$$\text{s.t.}\quad x_{ij}\in\{0,1\},\sum_{j=1}^{m}x_{ij}=1,i\in\{1,2,\cdots,n\}$$

DPSO 中粒子速度不再是连续空间所指的速度,而是用来计算粒子位置取值为 1 的概率值,则粒子 k 的速度用矩阵表示如下:

$$\boldsymbol{V}_k=\begin{bmatrix} \nu_{11} & \nu_{12} & \cdots & \nu_{1m} \\ \nu_{21} & \nu_{22} & \cdots & \nu_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \nu_{n1} & \nu_{n2} & \cdots & \nu_{nm} \end{bmatrix} \tag{4}$$

其中,$\nu_{ij}^k\in[-\nu_{\max},\nu_{\max}]$,$i\in\{1,2,\cdots,n\}$,$j\in\{1,2,\cdots,m\}$,$\nu_{\max}$ 为粒子速度的最大值,速度通过 sigmoid 函数转换成[0,1]上的概率值。

$$\text{sig}(\nu_{ij}^k)=\frac{1}{1+\exp(-\nu_{ij}^k)} \tag{5}$$

(2)粒子更新公式

粒子速度 V_k 的更新规则为:

$$\begin{aligned}\nu_{ij}(t+1)=g(&w\cdot\nu_{ij}(t)+c_1\cdot r_1(\text{Pb}_{ij}(t)-x_{ij}(t))+\\&c_2\cdot r_2(\text{Gb}_{ij}(t)-x_{ij}(t)))\end{aligned} \tag{6}$$

粒子位置 X_k 的更新规则为:

$$x_{ij}(t+1)=\begin{cases}1 & R(0,1)\leqslant\text{sig}(\nu_{ij}(t+1))\\0 & \text{others}\end{cases}$$

$$\boldsymbol{X}_k(t+1)=\text{h}(\boldsymbol{X}_k(t+1)) \tag{7}$$

其中,$i\in\{1,2,\cdots,n\}$,$j\in\{1,2,\cdots,m\}$,t 为粒子群进化代数,$\nu_{ij}(t)$ 为粒子第 t 次迭代后的速度,$x_{ij}(t)$ 为粒子第 t 次迭代后的位置,$w\in(0,1)$ 为惯性权重,c_1,c_2 为学习因子,r_1,r_2 和 $R(0,1)$ 均为[0,1]上的随机数,$\mathbf{Pb}(t)$ 为粒子 k 前 t 次迭代搜索到的个体最优位置,$\mathbf{Gb}(t)$ 为种群前 t 次迭代搜索到的全局最优位置。

函数 $\text{g}(\nu_{ij}^k)$ 的作用是将速度限制在$[-\nu_{\max},\nu_{\max}]$范围内。

$$g(\nu_{ij}^k)=\begin{cases}\nu_{\max} & \nu_{ij}^k>\nu_{\max}\\\nu_{ij}^k & -\nu_{\max}\leqslant\nu_{ij}^k\leqslant\nu_{\max}\\-\nu_{\max} & \nu_{ij}^k<-\nu_{\max}\end{cases} \tag{8}$$

函数 $\text{h}(\boldsymbol{X}_k(t))$ 的作用是检查更新后的粒子位置是否满足行和为 1 的约束条件,若不满足则将该行清零,然后选择速度最大的位置将其值设为 1,若满足条件则不进行任何操作。

(3)适应度函数

仿真任务调度的目的是使调度长度最小,因此适应度函数选择为求粒子对应的调度长度。

$$\text{fitness}(\boldsymbol{X}_k)=\max_{j=1}^{m}(\text{CT}_j) \tag{9}$$

基本 DPSO 算法的流程描述如下:

步骤 1 初始化:随机初始化种群初始粒子的位置和速度,粒子 k 的个体最优位置为 X_k 自身,全局最优位置为所有粒子中适应度值最小的粒子的位置。

步骤 2 对于每个粒子,根据粒子位置和速度更新公式计算其新的位置和速度。

步骤 3 计算每个粒子的适应度函数值,若其值小于个体最优粒子的适应度值,则更新个体最优位置,若其值小于全局最优粒子的适应度值,则更新全局最优位置。

步骤 4 若达到最大迭代次数,则转到步骤 5,否则转到步骤 2 继续执行。

步骤 5 取最后的全局最优位置对应的调度方案为最优方案,以此进行仿真任务调度。

4 基于信息素变异策略的改进离散粒子群算法

DPSO 求解仿真任务调度问题,具有搜索速度快、搜索范围大、容易实现等优点,但是 DPSO 算法也存在容易陷入局部最优、算法结果稳定性差受初始种群影响较大等缺点。

DPSO 算法中,粒子位置和速度的更新是关键。从粒子速度更新公式可以看出,它模拟了生物习惯的作用(惯性项$w\nu_{ij}(t)$)、个体认知能力(个体认知项 $c_1r_1(\text{Pb}_{ij}(t)-x_{ij}(t))$)以及社会价值取向(社会认知项 $c_2r_2(\text{Gb}_{ij}(t)-x_{ij}(t))$)。DPSO 通过这三项的整体变化来指导粒子的寻优过程,但是基本 DPSO 中仅模拟了个体的认知能力和社会精英的认知能力,而缺少普通个体之间的信息共享、协作与竞争关系的模拟。DPSO 算法采用的是“精英吸引”策略,通过个体最优值和全局最优值这些“精英”引导所有

粒子的寻优过程,当这些“精英”粒子得不到及时更新时算法就陷入局部最优;而且基本 DPSO 没有考虑任务调度到处理速度快的节点上执行时间短等先验信息,初始粒子质量的优劣对算法寻优结果影响较大,寻优结果具有较大不确定性,稳定性较差。

因此,借鉴蚁群算法[9](Ant Colony Optimization, ACO)的思想,引入基于信息素的变异策略,通过共享种群中所有粒子的寻优经验信息实现粒子的寻优过程,提出改进的离散粒子群算法(Modified Discrete Particle Swarm Optimization, MDPSO)。

4.1　离散粒子群与蚁群算法的融合

蚁群算法是一种新的启发式算法,它是基于蚂蚁的行为。当蚂蚁在寻找食物时,每只蚂蚁都会在其经过的路径上释放一些信息素,那么在较短路径上的信息素就会很快增加,每条路径上的信息素的数量,会反映出其他蚂蚁寻找食物时选择该路径的概率。最终,所有蚂蚁将选择最短的路径。蚁群算法应用到任务调度问题中,蚂蚁的一次探路过程就是寻找一个可行调度方案的过程,信息素反映的是任务被分配到各处理节点上的可能性。

将蚁群算法应用于离散粒子群算法中进行融合改进:(1)每一个粒子看成一只蚂蚁,粒子的一次“飞行”看成是蚂蚁的一次“探路”过程;(2)粒子一次更新得到的新的位置,看作蚂蚁一次探路找到的新的“路径”,粒子位置中一个元素 x_{ij} 看作是蚂蚁所走“路径”上的一个“路口”,若粒子位置上某个元素为 1,则表示蚂蚁会经过该“路口”,就会在该“路口”留下信息素,则其余蚂蚁选择路径时选择经过信息素浓度高的“路口”的概率较大。

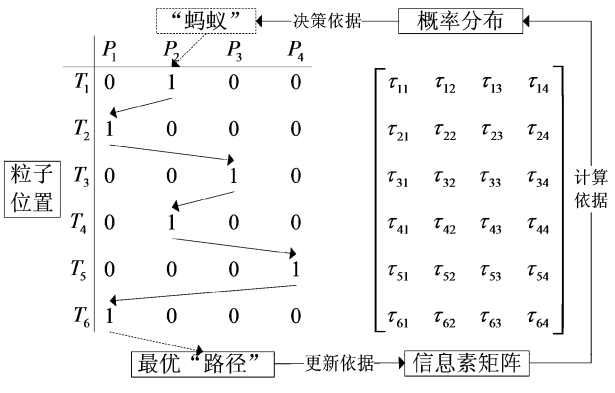


图 1　DPSO 与 ACO 算法的融合示意图

Fig. 1　Schematic of DPSO's integration with ACO

离散粒子群与蚁群算法的融合示意图如图 1 所示,将粒子位置中值为 1 的元素连接起来(图中实线箭头标注)即是一条“路径”,“路径”上的每个元素即是一个“路口”,粒子每一次迭代后,都在其取值为 1 的位置留下信息素,得到的信息素矩阵用来计算粒子位置中各元素取 1 的概率,此概率分布矩阵将影响粒子下一次的“飞行”寻优过程。

因此,本文引入基于信息素的变异策略,通过所有粒子的寻优经验信息积累信息素得到信息素矩阵,根据信息素矩阵和 **ETC** 矩阵得到粒子位置的概率分布矩阵,对根据更新公式得到的粒子位置按照概率分布矩阵进行变异,这样粒子间共享的就是种群中所有粒子的经验信息,可以避免算法陷入局部最优,同时考虑了先验信息 **ETC** 矩阵,从而减弱了算法对初始种群的依赖性,提高了结果的稳定性。

4.2　改进的离散粒子群算法实现

根据 4.1 中的分析,改进算法是在基本 DPSO 算法的基础上引入基于信息素的变异策略,因此改进算法的流程图如图 2 所示,其中信息素的更新、概率分布的计算以及变异操作描述如下。

(1)信息素更新公式

$$\boldsymbol{\tau}(t)=(1-\rho)\,\tau(t-1)+\Delta\tau(t),(t=1,2,\cdots,\mathrm{Iter}) \tag{10}$$

其中,$\boldsymbol{\tau}(t)$ 表示第 t 次迭代后种群中累计的信息素矩阵,$\boldsymbol{\tau}(0)$ 为依据初始粒子位置积累的信息素矩阵,$\rho\in(0,1)$ 为信息素挥发因子,$\Delta\boldsymbol{\tau}(t)$ 为第 t 次迭代中累计的信息素增量矩阵,Iter 为最大迭代次数。信息素增量矩阵的计算公式为:

$$\Delta\boldsymbol{\tau}(t)=\sum_{k=1}^{N}\left(\frac{Q}{\mathrm{fitness}(\boldsymbol{X}_k(t))}\times\boldsymbol{X}_k(t)\right) \tag{11}$$

其中,Q 表示信息素强度,N 为种群规模,$\boldsymbol{X}_k(t)$ 为粒子 k 第 t 迭代更新后的位置。

(2)概率分布矩阵的计算

概率分布矩阵 **Pd** 是一个 $n\times m$ 的矩阵,表示粒子位置中各元素取 1 的概率,其元素 Pd_{ij} 的计算公式为:

$$\mathrm{Pd}_{ij}(t)=\frac{(\tau_{ij}(t-1))^{\alpha}\times(E_{ij})^{\beta}}{\sum_{j=1}^{m}(\tau_{ij}(t-1))^{\alpha}\times(E_{ij})^{\beta}} \tag{12}$$

其中,$i\in\{1,2,\cdots,n\}$,$j\in\{1,2,\cdots,m\}$,$\mathrm{Pd}_{ij}(t)$ 为第 t 次迭代时对粒子位置进行变异时位置 x_{ij} 取 1 的概率,$\tau_{ij}(t-1)$ 为第 $t-1$ 次迭代后信息素矩阵的元素;

E_{ij}为启发式信息,且 $\boldsymbol{E}=1/\mathrm{ETC}$,$\boldsymbol{E}$ 表示任务调度到处理速度快的节点上的概率大;α 表示信息素的重要程度,β 表示启发式信息的重要程度。由公式可以看出 **Pd** 满足条件$\sum_{j=1}^{m}\mathrm{Pd}_{ij}=1$,因此 **Pd** 的每一行满足概率分布函数的完备性,可以以此作为概率分布生成满足要求的随机数。

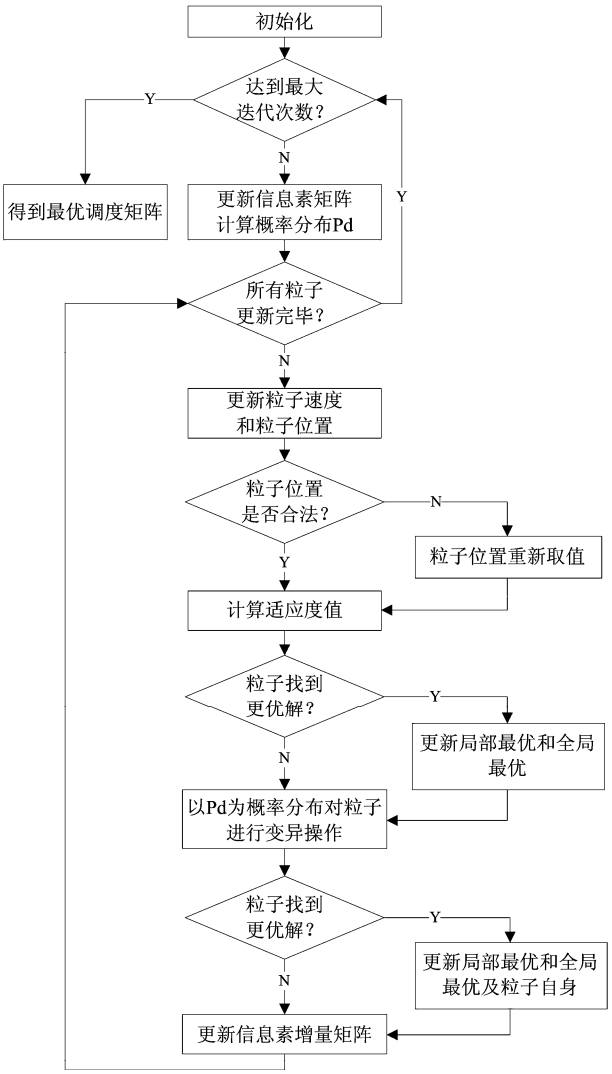


图 2 改进算法 MDPSO 的流程图

Fig. 2 The flow chart of MDPSO

(3)变异规则

粒子位置 $\boldsymbol{X}(t)$ 中的每一行表示一个任务的分配情况,**Pd**(t) 中的每一行表示一个任务分配到各个处理节点上的概率,因此对粒子 $X(t)$ 进行变异得到的新的粒子位置 $\boldsymbol{X}'(t)$ 的规则为:

$$\boldsymbol{X}'(t)=\mathrm{zeros}(n,m) \tag{13}$$

$$r=\mathrm{randsrc}([\mathrm{Pd}_{i1}(t),\mathrm{Pd}_{i2}(t),\cdots,\mathrm{Pd}_{im}(t)]) \tag{14}$$

$$X'_{ir}(t)=1 \tag{15}$$

首先将 $\boldsymbol{X}'(t)$ 中元素全部设置为0,然后以$[\mathrm{Pd}_{i1}(t),\mathrm{Pd}_{i2}(t),\cdots,\mathrm{Pd}_{im}(t)]$为概率分布生成$\{1,2,\cdots,m\}$上的一个随机数 r,r 取 m 的概率为 $\mathrm{Pd}_{im}(t)$,最后设置 $\boldsymbol{X}'(t)$ 的第 i 行的第 r 列的元素为 1。

若对粒子 k 变异后的粒子 $\boldsymbol{X}'_k(t)$,则重新计算 $\boldsymbol{X}'_k(t)$ 的适应度函数值,并对粒子自身、个体最优粒子和全局最优粒子进行更新操作,更新规则如下:

```
if(fitness(X'_k(t))<fitness(Pb_k(t)))
{
    X_k(t)=X'_k(t);//更新粒子自身
    Pb_k(t)=X'_k(t);//更新个体最优位置
    //更新个体最优适应度值
    PbFitness_k(t)=fitness(X'_k(t));
    if(fitness(X'_k(t))<fitness(Gb(t)));
    {
        Gb(t)=X'_k(t);//更新全局最优位置
        //更新全局最优适应度值
        GbFitness(t)=fitness(X'_k(t));
    }
}
```

5 仿真实验及性能分析

为了验证文中提出的基于信息素变异策略的改进离散粒子群算法的性能,进行以下仿真实验分析比较改进算法与基本 DPSO 算法结果的稳定性和算法收敛性,并将改进算法(MDPSO)的结果与随机搜索算法 DPSO 和 ACO 算法,以及经典调度算法 Max-Min 和 Min-Min[10] 算法的完成时间和系统负载平衡度进行了比较分析。

采用文献[10]中的方法生成仿真实验所需的数据 **ETC** 矩阵:首先,生成基准随机列向量 $\boldsymbol{B}_{n\times1}$,且 $B(i)=x_b^i$,$i\in\{1,2,\cdots,n\}$,其中 $x_b^i\in[1,\phi_b)$,ϕ_b 为 $\boldsymbol{B}$ 中元素取值范围的上界,$\mathbf{ETC}_{n\times m}$ 矩阵通过公式 $\mathbf{ETC}(i,j)=B(i)\times x_r^{i,j}$,$i\in\{1,2,\cdots,n\}$,$j\in\{1,2,\cdots,m\}$ 来构造,其中 $x_r^{i,j}$ 为$[1,\phi_r)$上的随机数,$\mathbf{ETC}_{n\times m}$中元素的取值范围为$[1,\phi_b\times\phi_r)$。对于生成的 $\mathbf{ETC}_{n\times m}$矩阵,任务的异构程度由 ϕ_b 确定,$\phi_b=3000$ 表示任务异构程度高,$\phi_b=100$ 表示任务异构程度低;处理节点性能的异构程度由 ϕ_r 确定,$\phi_r=$

1000 表示处理节点性能的异构程度高,$\phi_r = 10$ 表示处理节点性能的异构程度低。

仿真实验的主要参数设置如下:(a)$n = 20$,$m = 6$,$\phi_b = 3000$,$\phi_r = 1000$;(b)粒子群相关参数:$N = 20$,Iter = 100,$c_1 = 2.0$,$c_2 = 2.0$,$w = 0.6$,$v_{\max} = 4$;(c)蚁群相关参数:Iter = 100,NAnt = 20,$\rho = 0.02$,$\alpha = 0.8$,$\beta = 0.2$,$Q = 200$。

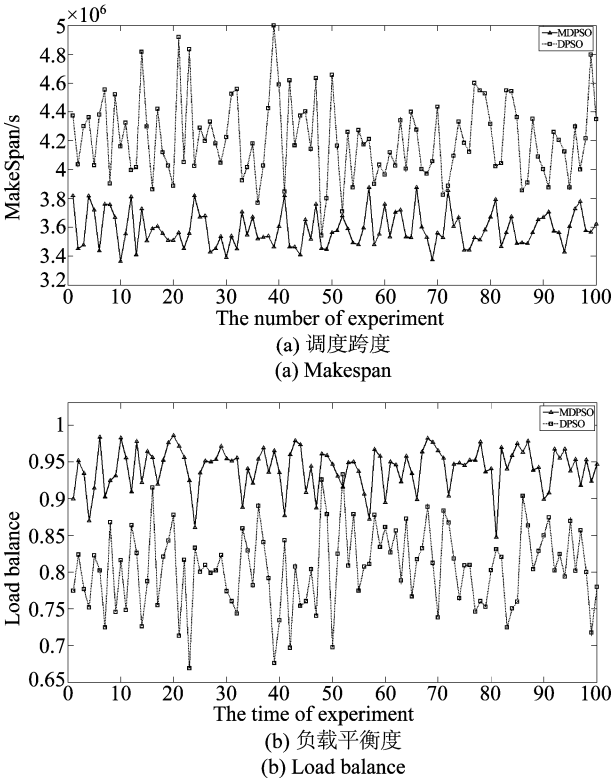
(1)算法结果的稳定性分析

算法结果的稳定性是指在不同的初始种群下得到的结果的差异程度。实验方法:对于相同的 **ETC**,进行 100 次独立重复实验,每次实验生成初始种群后分别采用 MDPSO 和 DPSO 进行处理,记录每次实验得到的任务完成时间和负载平衡度。负载平衡度 bal 定义为

$$\text{bal} = \frac{1}{m}\sum_{j=1}^{m}\frac{\text{CT}_j}{\text{MakeSpan}} \tag{16}$$

其中 m 为资源节点个数,$\text{bal} \in (0,1]$,bal 值越大表示负载平衡度越好。

实验得到调度跨度和负载平衡度随实验次数的变化曲线如图 3 所示。



(a) 调度跨度
(a) Makespan

(b) 负载平衡度
(b) Load balance

图 3　算法结果的稳定性

Fig. 3　Stability of results

对实验数据进行统计分析,得到结果如表 1、2 所示,其中稳定度定义为 $\frac{1}{100}\sum_{i=1}^{100}\frac{\text{data}_i}{\max(\text{data})}$,其中 data_i 为第 i 次实验得到的结果数据,max(data) 为 100 次实验得到结果数据中的最大值。

表 1　调度跨度统计分析结果

Tab. 1　Statistical analysis results of makespan

	DPSO	MDPSO	改进效果
最好结果	3540404.53	3362684.03	5.29%
最差结果	4998652.77	3874893.55	29.00%
平均值	4210110.97	3585735.53	14.83%
稳定性	0.842	0.925	9.87%

表 2　负载平衡度统计分析结果

Tab. 2　Statistical analysis results of load balance

	DPSO	MDPSO	改进效果
最好结果	0.932	0.986	5.79%
最差结果	0.669	0.847	21.17%
平均值	0.806	0.941	16.75%
稳定性	0.864	0.955	10.53%

由图 3 看出,对于不同的初始种群,MDPSO 的调度跨度和负载平衡度的波动范围比 DPSO 的波动范围要小,由表 1 可知,MDPSO 得到的调度跨度的稳定性比 DPSO 提高了 9.87%,由表 2 可知,MDPSO 得到的负载平衡度的稳定性比 DPSO 提高了 10.53%,说明本文提出的改进离散粒子群算法的结果相比基本 DPSO 的结果有更好的稳定性,算法结果受初始种群的影响更小。

(2)算法收敛性及种群多样性分析

算法收敛性是指全局最优粒子的适应度值随迭代次数变化的收敛特性。实验方法:对于同一 **ETC** 进行 100 次独立重复实验,记录每次实验每一次迭代得到的全局最优粒子的适应度值,求 100 次实验记录结果的平均值用以分析算法的收敛性,得到收敛性曲线如图 4 所示。

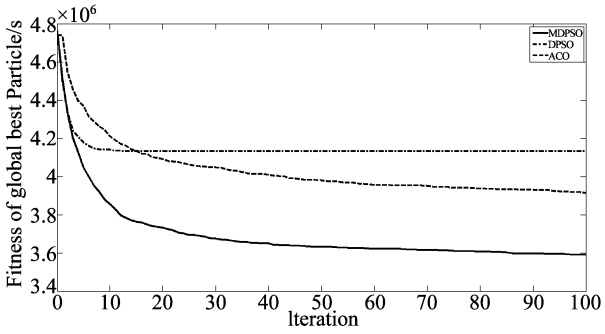


图4 算法的收敛性曲线

Fig. 4 The convergence cure

由图4可以看出,在搜索前期,由于信息素的缺乏,MDPSO以粒子的迭代为主,从而保持了DPSO较快的收敛速度,从而快速找到最优解所在区域;搜索后期,信息素增多,MDPSO的变异操作发挥主要作用,从而继承了ACO的较好的局部搜索能力,可以在局部区域内找到更好的解。可见,本文提出的MDPSO可以有效地避免算法陷入局部最优。

(3)不同异构程度下算法性能分析

在不同的任务异构程度和不同的节点性能异构程度下算法的性能会有所不同。实验方法:分别在四组条件(a)$\phi_b=3000,\phi_r=1000$(记为H-H),(b)$\phi_b=3000,\phi_r=10$(记为H-L),(c)$\phi_b=100,\phi_r=1000$(记为L-H),(d)$\phi_b=100,\phi_r=10$(记为L-L)下,分别用MDPSO、DPSO、ACO、Max-Min算法和Min-Min算法求解仿真任务调度问题,记录各算法找到的最优解对应的调度跨度和负载平衡度进行比较分析。其中,Max-Min和Min-Min算法是经典的调度算法,算法的核心思想都是将每个任务分配给有最早完成时间的处理节点,Max-Min算法优先考虑长任务,Min-Min算法优先考虑短任务;MDPSO、DPSO和ACO都是随机搜索算法,故取50次独立重复实验中最好结果的作为最终结果进行分析。得到的结果如表3和表4所示。

表3 各算法的调度跨度比较/秒

Tab. 3 Comparison of makespan/s

	MDPSO	DPSO	ACO	Max-Min	Min-Min
H-H	3341915.03	3688056.81	3494431.48	3525850.87	4872482.90
H-L	27590.56	29375.82	28614.70	28232.23	40446.25
L-H	95485.56	100776.84	100093.50	102467.98	126848.33
L-L	970.09	1062.61	1061.42	1056.89	1341.01

表4 各算法的负载平衡度比较

Tab. 4 Comparison of load balance

	MDPSO	DPSO	ACO	Max-Min	Min-Min
H-H	0.99255	0.93712	0.94055	0.98229	0.69702
H-L	0.99330	0.92721	0.93609	0.98621	0.66938
L-H	0.98968	0.93713	0.95465	0.97734	0.73935
L-L	0.99156	0.92980	0.94241	0.98517	0.73755

表5 MDPSO相对其他算法的改进效果

Tab. 5 The discrepancy of MDPSO and other algorithms

	调度跨度的相对改善效果				负载平衡度的相对改善效果			
	DPSO	ACO	Max-Min	Min-Min	DPSO	ACO	Max-Min	Min-Min
H-H	9.39%	4.36%	5.22%	31.41%	5.91%	5.53%	1.04%	42.40%
H-L	6.08%	3.58%	2.27%	31.78%	7.13%	6.11%	0.72%	48.39%
L-H	5.25%	4.60%	6.81%	24.72%	5.61%	3.67%	1.26%	33.86%
L-L	8.71%	8.60%	8.21%	27.66%	6.64%	5.22%	0.65%	34.44%

对表 3 和表 4 中的结果分别进行比较分析得到 MDPSO 的改进效果如表 5 所示。由表 5 可以看出:(a)与经典调度算法 Max-Min 和 Min-Min 算法比较,改进算法属于随机搜索算法,它不仅利用了效率矩阵这一先验信息,还利用了寻优过程中的经验信息,故可以找到更优的解;(b)相比于 DPSO 和 ACO 两种随机搜索算法,MDPSO 既采用了“精英策略”,又利用了所有粒子的寻优经验信息,故能改善 DPSO 易陷入局部最优的情况和 ACO 收敛速度慢且易出现“停滞”现象的问题,得到更短的调度跨度和更好的负载平衡度;(c)在 H-H 和 L-L 两种情况下,由于任务在各个节点上的处理节点上的执行时间相差不大,导致“精英”粒子与普通粒子、普通粒子与普通粒子之间的差异度减小,降低了种群多样性,使得 DPSO 更易陷入局部最优,使得 ACO 易出现“停滞”现象,故在此条件下,MDPSO 比单纯采用“精英策略”的 DPSO 和单纯采用信息素机制的 ACO 算法的改善效果更好。由此可见,本文提出的改进算法有效地缩短了调度跨度,提高了负载平衡度,从而提高雷达分布式仿真系统的仿真效率和资源利用率。

6　结论

本文采用离散粒子群算法求解雷达仿真分布式系统中的仿真任务调度问题,并在此基础上提出改进的离散粒子群算法。改进算法引入基于信息素的变异策略,将离散粒子群算法与蚁群算法进行融合,以种群中所有粒子的寻优经验信息来积累信息素,通过信息素和先验信息(效率矩阵)计算粒子变异概率分布,以此概率分布对基本 DPSO 得到粒子位置进行变异操作,改变了基本 DPSO 只以个体最优粒子和全局最优粒子引导粒子寻优的“精英吸引”策略,有效避免了算法陷入局部最优。改进算法充分利用种群中所有粒子的寻优经验信息,有效地避免了算法陷入局部最优,提高了算法结果的稳定性,降低了算法对初始种群的依赖性,缩短了仿真任务的完成时间,提高了分布式仿真系统的资源利用率。

文中在求解仿真任务调度问题以最小化任务完成时间(调度跨度)为目标,将任务调度问题抽象为单目标优化问题,而负载平衡度、通信延时等都是影响分布式仿真系统性能的关键问题,在后续研究中将进一步考虑更多的因素和限制条件,将仿真任务调度问题抽象为多目标优化问题进行求解,研究改进算法在多目标任务调度问题中的应用。

参考文献

[1] Salman A, Ahmad I, Al-Madani S. Particle swarm optimization for task assignment problem[J]. Microprocessors and Microsystems, 2002, 26(8): 363-371.

[2] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm[C]// Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on. IEEE, 1997, 5: 4104-4108.

[3] 杨靖雯, 王金龙, 徐以涛. 用于最大似然 2-DOA 估计的裂变粒子群算法[J]. 信号处理, 2010, 26(8): 1181-1186.
Yang Jingwen, Wang Jinlong, Xu Yitao. FPSO algorithm for the ML estimation of 2-DOA[J]. Signal Processing, 2010, 26(8):1181-1186. (in Chinese)

[4] Shi Y, Eberhart R C. Fuzzy adaptive particle swarm optimization[C]// Evolutionary Computation, 2001. Proceedings of the 2001 Congress on. IEEE, 2001, 1: 101-106.

[5] 舒涛. 混沌粒子优化算法在网格任务调度的应用[J]. 计算机仿真, 2012, 29(10): 154-157.
Shu Tao. Application of Chaotic Particle Swarm Optimization in Grid Task Scheduling[J]. Computer Simulation, 2012,29(10): 154-157. (in Chinese)

[6] Nickabadi A, Ebadzadeh M M, Safabakhsh R. A competitive clustering particle swarm optimizer for dynamic optimization problems[J]. Swarm Intelligence, 2012, 6(3): 177-206.

[7] 姜伟, 王宏力, 何星, 等. 并行免疫离散粒子群优化算法求解背包问题[J]. 系统仿真学报, 2014, 26(1): 56-61.
Jiang Wei, Wang Hongli, HE Xing, et al. Parallel bina-

ry particle swarm optimization algorithm based on immunity for solving knapsack problem[J]. Journal of System Simulation, 2014, 26(1): 56-61. (in Chinese)

[8] 周雅兰, 王甲海, 印鉴. 一种基于分布估计的离散粒子群优化算法[J]. 电子学报, 2008, 36(6): 1242-1248.
Zhou Yalan, Wang Jiahai, Yin Jian. A discrete particle swarm optimization algorithm based on estimation of distribution[J]. Acta Electronica Sinica, 2008, 36(6): 1242-1248. (in Chinese)

[9] 赵飞, 吴航, 龚跃. 蚁群算法解决网格环境下任务调度问题的研究[J]. 长春理工大学学报: 自然科学版, 2013, 36(1):97-100.
Zhao Fei, Wu Hang, Gong Yue. Research on solving the problem of task scheduling in grid environment by ant colony algorithm[J]. Journal of Changchun University of Science and Technology: Natural Science Edition, 2013, 36(1):97-100. (in Chinese)

[10] Braun T D, Siegel H J, Beck N, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems [J]. Journal of Parallel and Distributed computing, 2001, 61(6): 810-837.

作者简介

郑 伟 男,1988年生,湖北省蕲春县人,硕士生,电子科技大学电子科学技术研究院,主要研究方向为并行仿真技术。
E-mail: zwxiaowei1988@163.com

王 磊 男,1980年生,助理研究员,电子科技大学电子科学技术研究院,主要研究方向为分布式与并行处理技术。
E-mail:wang_lei@uestc.edu.cn

曹建蜀 男,1970年生,湖南临武县人,副研究员,电子科技大学电子科学研究院,研究方向为机载雷达信号处理、高速实时信号处理。
E-mail:js_cao@163.com