文章编号: 1001-0920(2019)07-1514-07 DOI: 10.13195/j.kzyjc.2017.1677

基于MapReduce模型带任务分割的平行机调度优化

黄基诞[1†], 郑斐峰[1], 徐寅峰[1], 刘 明[2]

(1. 东华大学 旭日工商管理学院，上海 200051；2. 同济大学 经济与管理学院，上海 200092)

摘 要: 研究一类基于MapReduce模型的两阶段平行机调度问题. 该模型中的每个工件包含Map和Reduce两道工序,前一工序的任务可以划分并同步加工,而后一工序不可划分,结合工件的到达时间、交货时间等约束,以最大完工时间和总延迟时间的加权和作为优化目标构建混合整数规划模型,设计采用差分变异策略和逐维角度扰动机制的改进鲸鱼优化算法求解模型. 数值仿真实验结果表明,所设计的算法相对于经典的鲸鱼优化算法、粒子群算法的求解效果有显著的提升,验证了模型和所设计算法的有效性.

关键词: 平行机调度；MapReduce；鲸鱼优化算法；并行处理；混合整数规划；任务分割

中图分类号: TP18 **文献标志码:** A

Parallel machine scheduling with splitting jobs in MapReduce system

HUANG Ji-dan[1†], *ZHENG Fei-feng*[1], *XU Yin-feng*[1], *LIU Ming*[2]

(1. Glorious Sun School of Business and Management，Donghua University，Shanghai 200051，China；2. School of Economics and Management，Tongji University，Shanghai 200092，China)

Abstract: Based on the MapReduce model, a two-phase parallel machine scheduling problem is studied. In the model, each job consists of two operations named Map and Reduce. The Map operation can be split and processed simultaneously, while the Reduce shall be processed on a single machine. Considering the arrival time, and due date of each job, we establish a mixed integer linear programming (MILP) model, aiming at minimizing the weighted makespan and total tardiness. An improved whale optimization algorithm (IWOA) is proposed, which uses differential perturbation and dimension-by-dimension Levy perturbation to obtain a near-optimal solution. The numerical results show that the IWOA outperforms both the particle swarm optimization and the whale optimization algorithms for the considered problem.

Keywords: parallel machines scheduling; MapReduce; whale optimization algorithm; parallel processing; mixed integer programming; job splitting

0 引 言

MapReduce调度模型是在大数据时代背景下由Google提出的用于数据分析的分布式计算框架[1]. MapReduce工件由Map与Reduce两部分组成,当一个工件被释放后,系统将工件自动分成两种类型的任务: Map工序的任务和Reduce工序的任务. 加工排序模型表述如下: 1) 每个工件包含Map工序任务和Reduce工序任务; 2) 只有当它的Map工序完成后才可以启动加工该工件的Reduce工序; 3) Map工序的任务可以分割且并行加工,即每个Map工序的任务可以分割为多个部分并在多台机器上同时加工; 4) Reduce工序只能在一台机器上连续加工. 基于MapReduce模型的生产制造活动在实际中也很常见,比如钢丝绳制造企业,如图1所示,一根粗的钢丝绳由许多细的钢丝绳拧成. 在第一阶段,几根细的钢丝绳可以分别在几台机器上同时生产;在第二阶段,在其中一台机器上将前一阶段完成的细钢丝绳拧成一根粗的绳子. 在生产过程中,必须考虑工件的释放时间和交货时间各不相同,而且订单大小也不相同,这些都是MapReduce模型调度的实际情况.

国内外学者针对MapReduce模型的平行机调度研究在近几年才兴起,大多数文献是借助建立整数规划模型的方法来研究MapReduce模型平行机调度. 文献[2]研究了批处理模式下的MapReduce问题,但是没有考虑到工件释放时间以及加工准备时间;文献[3]提出了求解MapReduce模型的两类算法;文

收稿日期: 2017-12-08；**修回日期:** 2018-03-14.

基金项目: 国家自然科学基金项目(71832001, 71771048, 71571061, 71531011, 71571134, 71428002)；国家社会科学基金项目(17BJY158)；东华大学非线性科学研究所项目；中央高校基本科研业务费专项资金项目.

责任编委: 樊治平.

†通讯作者. E-mail: huangjd@dhu.edu.cn.

图 1　钢丝绳制造

献[4]提出了离线调度优化算法——UAAS算法求解MapReduce问题; 文献[5-6]把MapReduce模型处理成两阶段工作,并分别提出了平衡池算法和传统的John算法. 一些学者研究了任务可分割的平行机调度模型. 文献[7]对于任务可分割的平行机调度模型考虑了任务的释放时间,但没有考虑任务准备时间; 文献[8]研究了带任务分割的平行机调度,并用差分算法求解该模型,但模型中没有考虑任务释放时间和准备时间; 文献[9]研究了可将任务拆分为更小粒度子任务并分批次执行的项目调度问题. 还有一些学者研究了MapReduce问题的在线调度模型. 文献[10]针对加工可中断和不可中断的两种情况,设计了竞争比均为$2-1/m$的最优在线算法; 文献[11]针对两台平行机的加工环境,对于每个工件的Map任务不小于Reduce任务的情形,设计给出了具有最优竞争性能的在线调度算法.

鲸鱼优化算法[12](Whale optimization algorithm, WOA)是Seyedali Mirjalili于2016年提出的一种新的元启发式种群智能算法. 该算法主要通过对鲸鱼狩猎时不断逼近猎物的捕食过程进行数学模拟,找到待求解问题的最优解. 关于WOA算法的应用研究在国内外刚刚起步[13-14],虽然具有参数少、全局收敛性强等优点,但目前还没有发现国内学者将该算法用于平行机调度. 本文将尝试该算法的应用拓展,把它应用于平行机调度,并将该算法与经典的调度算法进行比较.

综上所述,MapReduce模型调度问题与经典平行机或流水线调度问题不同: 经典调度模型中通常假设每个工件在各工序中不可分割加工,而MapReduce调度中的Map部分可以分割且并行加工. 因此,在算法设计上将会有所不同.

1　问题描述及数学模型

本文是在文献[11]的基础上加以改进. 给定M台相同的平行机处理 N个加工工件,并根据实际情况考虑每个工件i的释放时间r_i和交货时间d_i. 每个工件i包含Map和Reduce两道工序,其中Map工序包含n_i个子任务(即Map部分最多可以分割为n_i个子任务),它们可在多台机器上同时加工. 每个子任务加工时间长度为p_i. Reduce工序必须在Map工序完成后才可以启动,而且只能在其中一台机器上加工. 要求设计一个调度方案使得最大完工时间和总延期时间的加权和最小.

1.1　模型假设

问题基于以下基本假设:

1) 工件的Map和Reduce工序的加工时间长度已知;

2) 每台机器在任意时刻只能加工Map工序的一个子任务或Reduce工序;

3) 机器加工任何工序均不可中断;

4) 每个工件的Map工序所有子任务完成之后才能启动该工件的Reduce工序;

5) 每个工件的Reduce工序只能在一台机器上完成.

1.2　参数符号

本文使用的符号如下:

i表示工件编号;

ϕ表示机器的集合;

m表示机器的编号, $m \in \phi, \phi = \{1, 2, \cdots, M\}$;

I表示工件的集合;

N表示最后一个任务;

$i, j \in I, I = \{1, 2, \cdots, N\}$;

H表示Map工序子任务或Reduce工序在机器上的加工位置集合,如果所有的Map工序和Reduce工序都在一台机器上加工,则位置最多为$2N$,即$H = \{1, 2, \cdots, 2N\}$;

h表示Map工序子任务或Reduce工序在机器上的加工位置, $h = 1, 2, \cdots, 2N, h \in H$;

T_i^R表示工件i的Reduce工序加工时间;

λ表示权重系数, $0 \leqslant \lambda \leqslant 1$;

L表示一个足够大的正数.

1.3　决策变量

下面介绍决策变量:

$x_{h,i,m} \in \{0, 1\}$: 如果工件i的Map工序子任务在机器m上的第h个位置加工,则$x_{h,i,m} = 1$,否则为0;

$z_{i,m} \in \{0, 1\}$: 如果工件i的Map工序在机器m上加工,则$z_{i,m} = 1$,否则为0;

$R_{i,h,m} \in \{0, 1\}$: 如果工件i的Reduce工序在机器m上的第h个位置执行,则$R_{i,h,m} = 1$,否则为0;

$\delta_{i,m} \in \{0, 1\}$: 如果工件$i$的Reduce工序在机器

m上加工,则$\delta_{i,m}=1$,否则为0;

$y_{h,i,m}$表示工件i的Map工序在机器m上第h个位置的加工量或子任务数量;

$Q_{i,m}$表示工件i的Map工序在机器m上加工的子任务数量;

$\widetilde{S}_{i,m}$表示机器m对工件i的Map工序的子任务加工开始时间;

$\widetilde{C}_{i,m}$表示机器m对工件i的Map工序的子任务加工完成时间;

C_i^M表示工件i的Map工序的完成时间;

$S_{i,m}^R$表示机器m对工件i加工Reduce工序的开始时间;

$C_{i,m}^R$表示机器m对工件i加工Reduce工序的完成时间;

$C_{\max}$表示最大完工时间,即最后一个工件的Reduce工序完成时间.

1.4 目标函数

目标函数是最小化工件的最大完工时间与总延期时间的加权和,即

$$\min\Big\{\lambda C_{\max}+(1-\lambda)\sum_{i\in I}\sum_{m\in\phi}\max\{0,(C_{i,m}^R-d_i)\delta_{i,m}\}\Big\}. \tag{1}$$

当$\lambda=0$时,退化为最小化总延期时间;当$\lambda=1$时,退化为最小化最大完工时间.

处理时间约束为

$$C_{\max}\geqslant C_{i,m}^R,\ \forall i\in I,m\in\phi; \tag{2}$$

$$C_i^M\geqslant\widetilde{C}_{i,m},\ \forall i\in I,\ \forall m\in\phi; \tag{3}$$

$$S_{i,m}^R\geqslant C_i^M,\ \forall i\in I,\ \forall m\in\phi; \tag{4}$$

$$S_{i,m}^R+T_i^R\leqslant C_{i,m}^R+L\cdot(1-\delta_{i,m}),\ \forall i\in I,m\in\phi; \tag{5}$$

$$\widetilde{S}_{i,m}+L\cdot(1-z_{i,m})\geqslant r_i,\ \forall i\in I,m\in\phi; \tag{6}$$

$$\widetilde{S}_{i,m}+p_iQ_{i,m}\leqslant L(1-z_{i,m})+\widetilde{C}_{i,m},\ \forall i\in I,m\in\phi. \tag{7}$$

其中:式(2)表示最大完工时间不小于每个工件的Reduce工序完成时间;式(3)表示工件i的Map工序完成时间必须不早于该工序的每个分割子任务的完成时间;式(4)表示工件i的Reduce工序开始时间不得早于Map工序的完成时间;式(5)表示工件i的Reduce工序在平行机m上加工的完成时间与开始时间的关系;式(6)表示工件i的Map工序在各平行机m上分配的任务开始加工时间必须大于等于工件释放时间;式(7)表示工件i的Map工序在各平行机m上分配的任务量、加工开始时间与加工结束时间之间的关系.

任务分配约束为

$$L\cdot z_{i,m}\geqslant Q_{i,m},\ \forall i\in I,m\in\phi; \tag{8}$$

$$Q_{i,m}\geqslant z_{i,m},\ \forall i\in I,m\in\phi; \tag{9}$$

$$\sum_{i\in I}x_{h,i,m}\leqslant 1,\ h=1,2,\cdots,2N,\forall m\in\phi; \tag{10}$$

$$\sum_{h\in H}x_{h,i,m}\leqslant 1,\ \forall i\in I,\forall m\in\phi; \tag{11}$$

$$\sum_{h\in H}x_{h,i,m}=z_{i,m},\ \forall i\in I; \tag{12}$$

$$\sum_{h\in H}y_{h,i,m}=Q_{i,m},\ \forall i\in I; \tag{13}$$

$$\sum_{m\in\phi}Q_{i,m}=n_i,\ \forall i\in I; \tag{14}$$

$$\sum_{h\in H}R_{i,h,m}=\delta_{i,m},\ \forall i\in I; \tag{15}$$

$$\sum_{i\in N}R_{i,h,m}\leqslant 1,\ \forall h\in H,\forall m\in\phi; \tag{16}$$

$$\sum_{h\in H}R_{i,h,m}\leqslant 1,\ \forall i\in I,\forall m\in\phi; \tag{17}$$

$$\sum_{m\in\phi}\delta_{i,m}=1,\ \forall i\in I; \tag{18}$$

$$x_{h,i,m}\leqslant y_{h,i,m}\leqslant n_ix_{h,i,m},\ \forall i\in I,h\in H,m\in\phi; \tag{19}$$

$$\sum_{i\in I}(x_{h,i,m}+R_{i,h,m})\geqslant\sum_{i\in I}(x_{h+1,i,m}+R_{i,h+1,m}),\ \forall h=1,2,\cdots,2N-1,\forall m\in\phi; \tag{20}$$

$$S_{i,m}^R,C_i^M,C_{i,m}^R,\widetilde{C}_{i,m}>0,\widetilde{S}_{i,m}\geqslant 0,\ y_{h,i,m},Q_{i,m}\in Z^+,\ \forall i\in I,m\in\phi. \tag{21}$$

其中:式(8)和(9)表示工件i的Map工序在某机器上加工与否及加工工作量的相互关系;式(10)~(12)表示工件i的Map工序在一台平行机的一个位置上只加工一次;式(13)表示工件i的Map工序在m机器上某位置的加工量与该机器上分配的工作量一致;式(14)表示工件i的Map工序工作量等于该工件分配给所有机器的Map工序子任务工作量之和;式(15)~(18)表示工件i的Reduce工序只能在某一台平行机上加工,并且只加工一次;式(19)表示在平行机m的位置h上安排Map工序任务量的上下限约束;式(20)表示在平行机m上安排任务的位置是连续的;式(21)表示决策变量的取值范围.

2 鲸鱼优化算法(WOA)

鲸鱼优化算法(WOA)[12]是模仿座头鲸的狩猎行为而提出的一种新型启发式优化算法. 在WOA算法中,每只座头鲸的位置代表一个可行解. Mirjalili[12]提出了狩猎行为的数学模型,包括两种行为:随机搜

索、包围捕食和攻击猎物.

1) 随机搜索: 鲸鱼搜索猎物采用随机个体位置 X_{rand} 寻找食物,具体过程如下[12-13]:

$$X_{t+1} = X_{\text{rand}} - A \cdot D, \tag{22}$$

$$D = |C \cdot X_{\text{rand}} - X_t|. \tag{23}$$

其中: X_t 是第 t 代个体位置更新后的坐标; D 是当前个体 X 与随机个体 X_{rand} 的距离; C 是区间 $[0,2]$ 上的随机数,用以控制 X_{rand} 距离 X 远近程度的影响. A, C 定义[12-13]如下:

$$A = 2a \cdot r - a, \tag{24}$$

$$C = 2 \cdot r, \tag{25}$$

$$a = 2 - 2 \cdot \frac{t}{T_{\max}}. \tag{26}$$

其中: r 是 $(0,1)$ 中的随机向量, a 的值从2到0 线性下降, t 是当前的迭代次数, $T_{\max}$ 是最大迭代次数.

2) 包围捕食和攻击猎物: 因为鲸鱼在猎杀猎物时,不仅以螺旋形状游向猎物,还要收缩包围圈,所以在收缩环绕机制与螺旋模型之间以50 % 的概率来更新鲸鱼位置,公式[12-13]如下:

$$X_{t+1} = X_{\text{best}} - A \cdot |C \cdot X_{\text{best}} - X_t|,\ p < 0.5; \tag{27}$$

$$X_{t+1} = X_{\text{best}} + D_p \cdot \mathrm{e}^{bl} \cdot \cos(2\pi l),\ p \geqslant 0.5. \tag{28}$$

其中: $D_p = |X_{\text{best}} - X_t|$ 是个体 X 与最优个体 X_{best} 的距离; b 定义为螺旋线轨迹常数; l 是区间 $[-1,1]$ 上的随机数,当 $l = -1$ 时,人工鲸鱼距离食物最近,当 $l = 1$ 时,人工鲸鱼距离食物最远;参数 A 是区间 $[-2,2]$ 之间的随机向量,当 $|A| \leqslant 1$ 时,鲸鱼个体会向着当前位置最优的鲸鱼靠近,当 $|A| > 1$ 时,迫使鲸鱼偏离猎物,借此搜索其他更合适的猎物,以增强算法的全局搜索能力. 允许人工鲸鱼捕捉分布在搜索空间的任意食物,所以要保证人工鲸鱼在寻优时的全局搜索与局部开发,减少寻优盲点. 经典的鲸鱼优化算法流程见文献[12-13]. 但是,鲸鱼优化算法存在一些不足之处,比如易早熟收敛,为此本文设计一种差分变异策略方法对它进行改进以克服上述缺点.

3　改进的鲸鱼优化算法(IWOA)

3.1　编码方式

在改进的鲸鱼优化算法的平行机调度中,运用实数编码的位置向量表示机器调度序列. 假设有 N 个工件,按工件的释放时间 r_i 排序(如果释放时间一样,则按Reduce工序从大到小排序). 为了编程方便,本文采用至多 $N \cdot (1+M)$ 维实数位置向量表示 N 个工件和 M 台机器的平行机调度序列. 向量中的每一维实数的取值范围是 $[1, M+1)$, M 表示机器数量. 编码分成两个部分: 前面 N 位的整数部分表示将 N 个工件的Map工序分割成几个部分;后面 $N \cdot M$ 的实数数字中,整数部分代表Map工序分割部分所分配的机器编号,小数部分表示Map工序分割部分的大小.

针对随机产生的 $N \cdot (1+M)$ 维实数,先分析前 N 个数 $(x_1, x_2, \cdots, x_i, \cdots, x_N)$(其中 $x_i \in [1, M+1)$), $\lfloor x_i \rfloor$ 代表每个工件的Map工序将要被分配的机器数量,比如第 i 个工件由数量 $\lfloor x_i \rfloor$ 台机器执行,其中 $\lfloor \cdot \rfloor$ 表示向下取整. 接下来分析 $\sum_{i=1}^{N} \lfloor x_i \rfloor$ 维数值.

3.2　解码方式

用一个例子说明编/解码的过程. 例如有两台机器处理3个工件(按释放时间排序): $(p_1, p_2, p_3) = (3,7,5)$, $(n_1, n_2, n_3) = (3,2,4)$, $(r_1, r_2, r_3) = (0,3,5)$, $(T_1^R, T_2^R, T_3^R) = (3,2,4)$. 编码信息如表1所示.

表 1　3个任务两台机器的编码信息

	1	2	3	4	5	6	7	8	···
位置	2.331	1.421	2.156	1.841	2.422	2.131	1.337	2.112	···
解码	2	1	2	1	2	2	1	2	···

1) 先取前 N 位,计算出每个工件Map工序被分配的机器数量. 这里有3个工件,因而取前3位;表1中,第1和第3个工件的Map工序分割成两个部分,第2个工件Map工序只在1台机器上运行.

2) 下面分析接下来的 $2+1+2=5$ 位数字. 例如,由于第1个工件的Map工序分配给了两台机器,观察后续两位(见表1)1.841和2.422,取整即可得出,Map工序分割的子任务分别安排在第1台机器和第2台机器上,分割的子任务数量看小数部分: $\left\lceil n_1 \times \dfrac{0.841}{0.841+0.422} \right\rceil = 2$,这里 $\lceil \cdot \rceil$ 表示取整,可以得出机器1分配到的Map工序子任务数为2,从而可以推算出机器2分配的Map工序子任务数为 $n_1 - 2 = 1$.

3) 确定工件1的Reduce工序分配依赖于哪一台机器最后完成该工件的Map子任务,由该机器来完成Reduce工序. 这里,机器1最后完成工件1的Map工序的子任务,故由机器1来执行工件1的Reduce工序. 同理可推算出其他工件的Map工序子任务的分割和机器分配. 解码后的具体调度方案如图2所示.

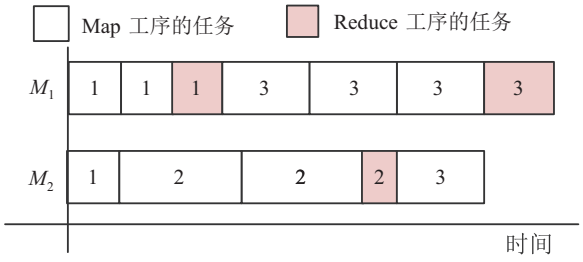

图 2　两台平行机3个工件的调度方案

3.3 差分变异策略

差分进化算法最初是由Storn等[15]提出的,它具有强大的探索能力,可以使新解朝着最优的方向搜索,优点是收敛速度快. 为了引导群体快速朝最优方向移动,本文在差分进化算法的基础上提出了差分变异策略,如下式所示:

$$X_i^{t+1} = X_{\text{best}}^t + F \cdot (X_j^t - X_k^t). \tag{29}$$

其中: X_{best}^t 是 t 代最优个体; j,k 是不同于 i 的两个互异的随机整数; F 是缩放系数,是服从(0,1)均匀分布的随机数.

3.4 逐维角度扰动机制

在WOA算法中,虽然游走觅食有利于提高种群的多样性以及加强算法的全局搜索能力,但随着迭代次数的增加,也容易陷入局部最优. 对于多维目标函数而言,对解采用整体更新评价策略将影响算法收敛速度和解的质量;随着搜索的深入,解中部分维度的退化,直接导致整体解的恶化,而忽视了其他维度的改良. 比如多维目标函数 $f(X) = (x_1-1)^2 + (x_2-3)^2 + (x_3-5)^2$,第 t 代的第 i 个解 $X_{t,i} = (0.7, 2.5, 4.6)$,此时,目标函数值 $f(X_{t,i}) = 0.50$. 经过迭代后, $X_{t,i}$ 更新为 $X_{t+1,i} = (1,1,5)$,计算目标函数值 $f(X_{t+1,i}) = 4$. 比较目标函数 $f(X_{t+1,i}) = 4 > f(X_{t,i})$ 发现未能改进,算法将放弃当前的新解;但是,比较全局最优解 $X_{\text{opt}} = (1,3,5)$ 发现,第1和第3维度已经进化到全局最优,由于第2维度变差,使得整体解质量变差,从而导致解被抛弃,这增加了函数的进化次数,减缓了收敛速度. 若保持第1、第3维取值不变,而调整第2维的值,则可以改善结果.

逐维随机扰动机制能够有效地改善上述问题,从而获得更高质量的解并提高算法的收敛速度,如鲸鱼位置 $X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,d})$,其第 k 维分量需要执行动态随机角度扰动运算,角度扰动公式为

$$x'_{i,k} = x_{i,k} + \gamma \cdot \cos\theta_k. \tag{30}$$

变异维数 k 采取轮盘赌方式进行选择确定. 其中: $\gamma = \min\{x_{i,k} - a_k, b_k - x_{i,k}\}$, a_k, b_k 分别表示当前第 k 维分量的下界与上界; $\theta_k = 2\pi \times N(0,1)$, $N(0,1)$ 是服从均值为0,方差为1的标准高斯分布的随机数. 逐维扰动的引入可以扩大算法的搜索区域,但对所有原解都进行逐维扰动,比较浪费时间,所以引入一个扰动概率 p_c(本文选取0.3),当随机数 $p < p_c$ 时,则执行逐维扰动.

3.5 改进鲸鱼优化算法(IWOA)流程

综上所述,求解MapReduce问题的改进鲸鱼优化算法流程可归结如下.

Step 1: 种群初始化. 设定种群规模大小为NP,最大迭代次数为 $T_{\max}$.

Step 2: 选择最优位置. 计算NP个体适应度值的大小,选择适应度值最优的个体位置作为最优位置.

Step 3: 迭代计算. 依据随机搜索、包围捕食、攻击猎物,用式(22)～(28)进行迭代寻优.

Step 4: 差分变异. 基于当前最优解进行差分变异: 如果求得适应度更好的新解,则替换当前最优解;若不是适应度更好的解,则随机产生概率 p,若 $p < p_c$,则进行逐维角度扰动,扰动后,如果求得适应度更好的新解,则替换当前最优解.

Step 5: 若达到终止条件,则输出最优个体,即算法找到的最优解;否则,返回Step 3.

4 仿真实验

为验证算法寻优性能,下面分为小数据和大数据两部分进行实验. 算法采用Matlab 2014a编程,实验运行环境为: CPU 2.1 GHz,内存4 GB, Windows 7操作系统(64位), IBM CPLEX 12.5.

4.1 评判指标

为了说明改进算法的有效性,采用相对指标进行算法性能的度量. 各类算法所求得的解的质量可用RPD来衡量,其计算公式[16]为

$$\text{RPD}(\%) = \frac{f_{\text{alg}} - f_{\text{best}}}{f_{\text{best}}} \times 100\%. \tag{31}$$

其中: f_{alg} 是算法alg的目标函数值, f_{best} 是各类算法中最好的目标函数值. 当数据量大到一定程度时,CPLEX在有限的时间内无法求解. 对于小数据规模实验, f_{best} 由CPLEX计算获得.

4.2 实验参数

本文采用文献[16]的方法生成随机测试算例,算例规模和参数见表2. 各算法的参数设置见表3.

表 2 算例的参数分类和取值范围

参数	分类取值范围
加工规模 N	$10, 20, 30, 40, 50; 100, 200$
机器数量 M	$3, 5$
每个子任务长度 p_i	$U[1, 10]$
释放时间 r_i	$U[1, 100], U[1, 1\,000]$
Map工序子任务的数量 n_i	$U[1, 10]$ 取整
交货期 d_i	$r_i + n_i p_i + U[10, 50]$
Reduce长度 T_i^R	U[1～10]
λ	$0, 0.1, 0.2, \cdots, 0.9, 1.0$

表 3 参数设置

算法名称	参数设置
PSO	NP=100, $T = 200, c_1 = c_2 = 1.93$
WOA	NP=100, $T = 200, \tilde{r} = 0.5, a = 2$
IWOA	NP=100, $T = 200, \tilde{r} = 0.5, p_c = 0.3, a = 2$

4.3 实验分析

对于上述实例用IWOA、WOA和PSO三种算法进行求解,并对实验结果进行参照对比. 由于篇幅有限,分别随机选取两组: $N = 200, M = 5, \lambda = 0.8$和$N = 100, M = 5, \lambda = 0.8$.

图3和图4分别在$N = 100$和$N = 200$的两种情况下描绘了算法的收敛曲线. 从图3和图4可以看出: 当达到一定的迭代次数时,优化目标无法再改进; 另一方面, WOA和PSO算法有着类似的收敛速度,而IWOA的收敛性能明显优于WOA和PSO.

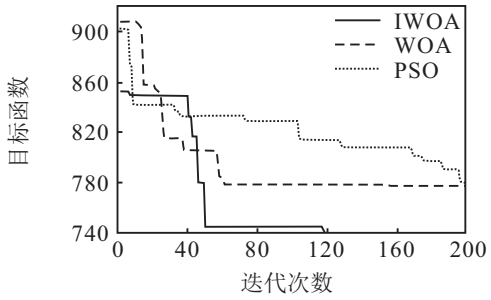

图 3　5台机器100个工件的算法收敛曲线

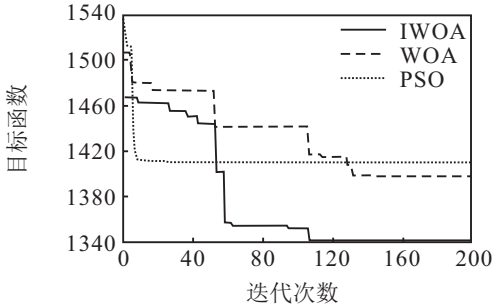

图 4　5台机器200个工件的算法收敛曲线

针对机器数量和工件规模不同,进行14组实验,每个实验在同一算法下运行5次. 由式(31)计算出每个算法的评判指标RPD,并求其RPD的平均值. 由于篇幅有限,对于RPD的区间分布图,只取小数据量的实例(即$N = 10, 20, 30, 40$),如图5和图6所示,其中$\lambda = 1$. 从图5和图6可以看出,IWOA算法下各个算例的RPD值比较小,波动区间小,稳定性比其他算法(PSO和WOA)好. 所有实例的运行结果对比见表4(测试规模为: 工件数量/机器台数),限于篇幅,表中只列出$\lambda = 0, 1$两种情况.

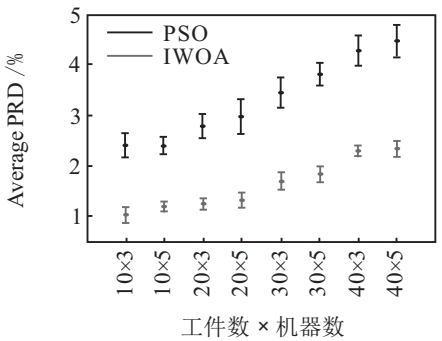

图 5　PSO与IWOA的平均RPD置信区间比较

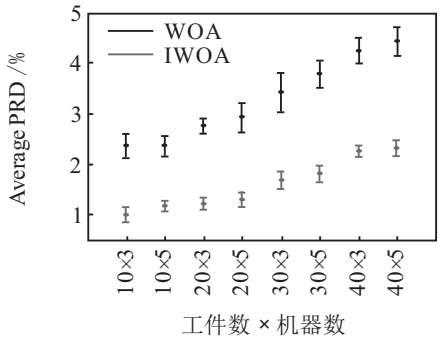

图 6　WOA与IWOA的平均RPD置信区间比较

表 4　各算法的性能指标比较

规模N/M	优化算法	评价指标$\lambda = 1$		评价指标$\lambda = 0$	
		Avg.RPD	t/s	Avg.RPD	t/s
10/3	PSO	2.39	111.12	2.37	102.38
	WOA	2.34	103.76	2.23	108.91
	IWOA	1.01	135.54	0.96	129.78
	CPLEX	0.00	1 181.32	0.00	1 192.03
10/5	PSO	2.33	123.94	2.31	131.22
	WOA	2.38	127.98	2.42	133.87
	IWOA	1.18	147.90	1.13	149.28
	CPLEX	0.00	2 171.89	0.00	2 166.08
20/3	PSO	2.77	287.12	2.53	266.93
	WOA	2.71	289.09	2.61	281.75
	IWOA	1.24	292.19	1.23	292.66
	CPLEX	0.00	3 671.81	0.00	3 664.17
20/5	PSO	2.91	326.54	2.95	329.08
	WOA	2.84	344.83	2.92	341.65
	IWOA	1.31	350.97	1.38	358.03
	CPLEX	0.00	5 465.18	0.00	5 405.01
30/3	PSO	3.47	440.05	3.43	447.86
	WOA	3.45	450.77	3.51	455.93
	IWOA	1.69	466.62	1.76	468.88
	CPLEX	0.00	7 612.11	0.00	7 673.67
30/5	PSO	3.57	493.82	3.80	491.27
	WOA	3.69	503.28	3.78	501.24
	IWOA	1.82	510.13	1.88	502.49
	CPLEX	0.00	10 612.76	0.00	10 631.44
40/3	PSO	4.24	578.83	4.27	573.06
	WOA	4.29	582.23	4.26	581.10
	IWOA	2.28	589.09	2.33	588.56
	CPLEX	0.00	14 887.92	0.00	14 801.12
40/5	PSO	4.41	591.23	4.45	593.37
	WOA	4.27	605.53	4.43	601.11
	IWOA	2.33	611.92	2.38	613.37
	CPLEX	0.00	18 652.25	0.00	18 452.74
50/3	PSO	3.42	680.57	3.34	686.49
	WOA	3.37	694.32	3.31	692.87
	IWOA	0.23	708.89	0.29	701.18
	CPLEX	-	-	-	-
50/5	PSO	3.33	721.12	3.37	723.75
	WOA	3.31	731.21	3.35	729.08
	IWOA	0.20	749.16	0.25	744.29
	CPLEX	-	-	-	-
100/3	PSO	3.92	1 544.29	3.96	1539.82
	WOA	3.98	1 559.61	3.99	1 548.34
	IWOA	0.27	1 596.34	0.28	1 598.02
100/5	PSO	4.12	1 653.86	4.15	1 658.19
	WOA	3.84	1 679.02	4.12	1 677.35
	IWOA	0.20	1 694.67	0.21	1 695.97
200/3	PSO	4.25	2 787.36	4.33	2 777.05
	WOA	4.28	2 793.87	4.31	2798.13
	IWOA	0.30	2805.39	0.31	2 801.72
200/5	PSO	4.42	3 356.38	4.49	3 332.88
	WOA	4.48	3 393.27	4.46	3 387.23
	IWOA	0.30	3 401.45	0.33	3 411.86

不同规模下的实验很好地验证了改进鲸鱼算法的有效性. 该算法在所有实验的RPD均未超过5 %, 说明其运行的稳定性较好,而且解的质量令人满意.

5 结 论

本文探讨了任务释放时间以及交货时间的MapReduce模型平行机调度,建立了整数规划模型,设计了改进鲸鱼优化算法(IWOA),数值实验验证了所设计的算法的有效性能. 下一步研究可以从以下几个方面展开:一是将所设计的算法推广到更一般的MapReduce调度模型中,比如平行机加工速度不同或者任务带准备时间的情形;二是探讨工件动态释放的情形,并设计竞争算法.

参考文献(References)

[1] Valcarce D, Parapar J, Barreiro A. A MapReduce implementation of posterior probability clustering and relevance models for recommendation[J]. Engineering Applications of Artificial Intelligence, 2018, 75(1): 114-124.

[2] Li X, Jiang T, Ruiz R, et al. Heuristics for periodical batch job scheduling in a MapReduce computing framework[J]. Information Sciences, 2016, 326(1): 119-133.

[3] Tang S J, Busung L,Bingsheng H. Dynamic job ordering and slot configurations for mapReduce workloads[J]. IEEE Trans on Services Computing, 2016, 9(1): 4-17.

[4] Shao Y L, Li C L, Gu J G, et al. Efficient jobs scheduling approach for big data applications[J]. Computers & Industrial Engineering, 2018, 117(2): 249-261.

[5] Shi V Y, Zhang K H, Cui L Z, et al. MapReduce short jobs optimization based on resource reuse[J]. Microprocessors and Microsystems, 2016, 47(1): 178-187.

[6] Verma A, Cherkasova L, Campbell R H, et al. Orchestrating an ensemble of mapReduce jobs for minimizing their makespan[J]. IEEE Trans on Dependable and Secure Computing, 2013, 10(5): 314-327.

[7] Wang C J, Liu C C, Zhang Z H, et al. Minimizing the total completion time for parallel machine scheduling with job splitting and learning[J]. Computers and Industrial Engineering, 2016, 97(1): 170-182.

[8] Wang W L, Wang H Y. Parallel machine scheduling with splitting jobs by a hybrid differential evolution algorithm[J]. Computers & Operations Research, 2013, 40(5): 1196-1206.

[9] 王磊, 聂兰顺, 战德臣, 等, 求解任务可拆分多项目协同调度问题的启发式算法[J]. 控制与决策, 2017, 32(6): 1013-1018.
(Wang L, Nie L S, Zhan D C, et al. Heuristic algorithm for solving multi-project collaborative scheduling problem with activity splitting[J]. Control and Decision, 2017, 32(6): 1013-1018.)

[10] Luo T B, Zhu Y, Wu W L, et al. Online makespan minimization in MapReduce-like systems with complex reduce tasks[J]. Optimization Letters, 2017, 11(2): 271-277.

[11] Huang J D, Zheng F F, Xu Y F, et al. Online MapReduce processing on two identical parallel machines[J]. J of Combinatorial Optimization, 2018, 35(1): 216-223.

[12] Mirjalili S, Lewis A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95(1): 51-67.

[13] 龙文, 蔡绍洪, 焦建军, 等. 求解大规模优化问题的改进鲸鱼优化算法[J]. 系统工程理论与实践, 2017, 37(11): 2983-2994.
(Long W, Cai S H, Jiao J J, et al. Improved whale optimization algorithm for large scale optimization problems[J]. Systems Engineering — Theory & Practice, 2017, 37(11): 2983-2994.)

[14] Wang J Z, Du P, Niu T, et al. A novel hybrid system based on a new proposed algorithm — Multi-objective whale optimization algorithm for wind speed forecasting[J]. Applied Energy, 2017, 208(15): 344-360.

[15] Storn R, Price K. Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces[J]. J of Global Optimization, 1997, 11(4): 341-359.

[16] José Elias C A, Joseph Y T, Leung. An effective iterated greedy algorithm for scheduling unrelated parallel batch machines with non-identical capacities and unequal ready times[J]. Computers & Industrial Engineering, 2017, 105(1): 84-100.

作者简介

黄基诞(1975−), 男, 讲师, 博士, 从事平行机调度的研究, E-mail: huangjd@dhu.edu.cn;

郑斐峰(1976−), 男, 教授, 博士生导师, 从事大数据挖掘、调度优化、医疗调度等研究, E-mail: ffzheng@dhu.edu.cn;

徐寅峰(1962−), 男, 教授, 博士生导师, 从事大数据挖掘、调度优化等研究, E-mail: xyf@dhu.edu.cn;

刘明(1983−), 男, 副教授, 博士生导师, 从事飞机调度、船舶调度等研究, E-mail: mingliu@tongji.edu.cn.

(责任编辑: 闫 妍)