GM(1,1)流预测研究

刘明

(学号200051；)

主要内容：考虑了安装需要人工参与而加工时无须人工参与的基于MapReduce模型带安装时间平行机调度。基于MapReduce模型的每个工件包含Map和Reduce两道工序，工件在Map工序中可以分割成若干个子任务， 并在效性。

# 1 引言

等待时间受限的调度问题普遍存在于钢铁，玻璃和塑料等行业，它的研究起源于生产过程中的高温连续性，要求工件在相邻两个工序之间保持一定的温度，慢慢在其他行业中开始也引起应用，因此，可将两工序之间的温度要求、缓冲区受限要求等转换为对等待时间的要求，从而形成了等待时间受限的调度问题。Kim等[1]研究了有重叠等待时间的三机流水车间调度问题；构建了混合整数规划模型并提出了问题的下界。Hong等[2]研究了在薄膜晶体管液晶显示(TFT-LCD)阵列工艺中存在等待时间有限的生产调度问题，建立了混合整数规划模型，并用遗传算法进行求解。王柏琳等[3][4]针对等待时间受限的流水车间调度问题，研究了加工机器上的工件序列特征。常晓坤等[5]建立了以最小化期望成本为目标的等待时间受限的混合流水车间，考虑了机器加工时间不确定等环境。An等[6]研究了等待时间受限的2个机器流水车间调度，其中工件在第2个机器具有准备时间的约束。

MapReduce调度模型最早是由Google提出用于大数据时代背景下的数据分析与计算， 是当前应用较为广泛的分布式计算框架。MapReduce工件由Map和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。利用加工排序模型表述如下[9，10]：(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、锚链制造企业， 如图1所示， 一根长的锚链由许多个小的链环连接而成。在第一阶段， 把整个锚链分成几段，许多短的链环段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链环段分别连接成一根长的锚链。在生产过程中， 必须考虑工件在机器间运输时间等因素， 而且工件释放时间也不相同， 这些都是MapReduce模型调度应该考虑的实际情况。

针对MapReduce模型的平行机调度研究在近几年引起国内外学者高度关注， 一些学者研究了MapReduce问题的在线调度算法[9][10]，但是一部分文献借助建立整数规划模型的方法来研究MapReduce模型平行机调度。Chen等[11]和Li等[12]分别研究了考虑搬运工作和批处理模式下的 MapReduce 问题， 但二者都忽略了工件的释放时间。Tang等[13]求解MapReduce模型提出了两类算法，最后验证算法的有效性。Tian等[14]提出了UAAS算法来求解MapReduce问题。Verma等[15][16]把MapReduce模型处理成两阶段工作， 提出了平衡池算法及传统的John算法来求解。

鸡群算法（Chicken Swarm Optimization，CSO）是由Meng 等[17]在2014 年提出的一种基于鸡群等级制度、搜索行为和繁殖更新的随机优化方法。 它模拟了鸡群家族将鸡群分为公鸡、母鸡和小鸡3 个等级，觅食能力依次减少，通过对不同函数的优化测试，证明了该算法的有效性。它具有参数少、全局收敛性强等优点，但是仍然存在一些不足之处， 比如易出现早熟收敛等缺点，然而，对CSO算法的改进和应用拓展研究国内外刚刚起步[18-20]。

综上所述， MapReduce模型调度问题与经典的可拆分平行机调度问题和混合流水车间调度不同：经典的可拆分平行机调度模型中通常假设只有一个工序；而混合流水车间调度问题往往先假设工件不可分割加工；但是，在MapReduce模型调度中的Map部分可以分割且并行加工，且加工工序需要2道。因此， 在算法设计上将会有所不同。

# 2 问题描述及数学模型

本文在文献[3][10]的基础上，结合MapReduce模型与生产实际情况建立模型。给定台同类机处理个加工工件，考虑每个工件的释放时间，每个工件包含Map和Reduce两道工序，其中Map工序包含个子任务(即Map部分可以最多可以分割为个子任务)， 各Map 子任务之间没有严格的先后顺序关系， 拆分的Map子任务可在不同的机器上同时加工。每个工件的Map工序子任务标准加工时间长度为和Reduce工序标准长度加工时间长度。每个机器的Map工序的加工速度为。Reduce工序必须在Map工序上所有子任务完成后才可以启动， 而且只能在其中一台机器上加工且任务不能分割。等待时间受限的车间调度问题要求任一工件在相邻工序间的等待时间不能超过一定的时间上限。即要求任一工件或工件中的子任务在Map工序和Reduce工序的等待时间不超过一定的时间上限，且等待时间越小越好；因此设计一个调度方案使所有工件的逗留时间和最小。这个模型主要决策的内容有（i）Map部分分割的各个子任务的大小；（ii）Map部分分割成的子任务的个数；（iii）Map部分分割的子任务如何在机器上的分配；（iv）Reduce工序在机器上的分配。

# 2.1任务分割的优势分析

通过一个例子来说明任务分割的优势。例如有2台机器处理3个工件(按释放时间排序)，3个工件的基础信息为：,，，。为了便于简单扼要的说明问题，设2台机器处理时间都一样为。可以发现，未进行分割的任务调度（见图2），最后完成时间为33；而进行任务分割的调度（见图3），最后完成时间（不一定是最优）为26；很明显，最后完成时间节约了7；节约了为21.21%。因此任务分割可以有效的提高生产效率。如何进行有效的Map任务分割和机器上的任务分配是本文研究重点。

 

**图2 未进行任务分割的调度图 图3 进行任务分割的较优调度**

**Fig.2 Scheduling without Map task split Fig.3 scheduling with Map task split**

# 2.1 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。

6) 每台机器对所有工件都有各自恒定的加工速度。

7) 机器在加工过程中允许出现空闲。

8) 同一个工件里的同一种任务在同一台机器上生产，不许隔开；即任务中间不许有其他工件的生产任务。

9) 同一工件相邻两道工序之间等待时间是确定的。

10)同一工件从工件释放到工件开始加工时间之间的等待时间是确定的。

# 2.3 参数符号

本文其他所使用的符号如下：

：表示工件编号。

：为机器的集合，为机器的数量。表示机器的编号，，。

：表示工件的集合，为最后一个任务；，。

：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，那么位置最多为，即 。

：表示Map工序子任务或Reduce工序在机器上的加工位置，，。

：表示工件的Map工序可分的子任务个数。

：表示工件中的每个Map子任务的标准长度（标准加工时间）。

：表示机器的加工Map工序的速度。不失一般性假设。

：表示工件在机器和机器之间的运输时间。 其中同一机器间的运输时间为0，即。

：表示工件的Reduce工序标准长度（标准加工时间）。

：工件的释放时间。

：如果工件的Map工序子任务在机器上加工，那么单位任务加工时间。

：表示如果工件的Reduce工序在机器上加工，那么加工时间。

：等待时间的上限。

：是一个足够大的正数。

# 2.4 决策变量

以下是决策变量：

：如果工件的Map 工序子任务在机器上的第个位置加工，则；否则为0。

：工件的Map工序在机器上的第个位置加工量或子任务数量。

：如果工件的Map工序在机器上加工，则；否则为0。

：工件的Map工序在机器上加工的子任务数量。

：如果工件的Reduce 工序在机器上的第个位置执行，则；否则为0。

：如果工件的Reduce工序在机器上加工，则；否则为0。

：机器对工件的Map工序的子任务加工开始时间。

：机器对工件的Map 工序的子任务加工完成时间。

：工件的Map工序的完成时间。

：机器对工件加工Reduce工序的开始加工时间。

：机器对工件加工Reduce工序的完成时间。

：工件加工Reduce工序的开始加工时间。

：最大完工时间，即最后一个工件的Reduce工序完成时间。

# 2.5 目标函数

本文所考虑的问题模型的调度目标是最小化所有工件逗留时间和：

； （1）

实际生产问题中存在许多MapReduce模型的特性和约束，本文约束用数学进行描述如下：

# 处理时间约束：

 （2）

 （3）

 （4）

 （5）

 （6）

 （7）

 （8）

 （9）

 （10）

 （11）

 （12）

 （13）

 （14）

 （15）

 （16）

 （17）

(2)式表示最大完工时间不小于每个工件的Reduce工序完成时间。(3)式表示工件的Map工序完成时间必须不早于该工序的每个分割子任务完成时间。(4)式表示工件的Reduce工序开始时间不得早于Map工序的完成时间。(5)式表示工件的Reduce工序在平行机上加工的开始时间不小于其他机器上Map完成时间加上机器间运输时间。(6)式表示工件的Reduce工序在平行机上加工的完成时间和开始时间的关系。(7)式表示工件的Map工序在平行机上分配的任务开始加工时间必须大于等于工件释放时间。(8)式表示工件的Map工序在各平行机上分配的任务量、加工开始时间与加工结束时间之间的关系。(9)、(10)式表示工件的Map工序在平行机的位置上开始加工时间，必须大于等于前一个位置上加工的Map或Reduce的完成时间。(11)、(12)式表示工件的Reduce工序在平行机的位置上开始加工时间，必须大于等于前一个位置上加工的Map或Reduce的完成时间。(13)式表示工件的Reduce工序开始时间就是机器上的Reduce工序开始。(14)式表示工件从工件释放时间到Map工序开始加工之间的等待时间到不要超过等待时间上限。(15)式表示工件在两工序之间的等待时间不要超过等待时间上限。(16)如果工件的Map工序子任务在同类机上加工，那么子任务的加工时间计算方式。(17)如果工件的Reduce工序任务在同类机上加工，那么他的加工时间计算方式。

# 任务分配约束：

 （18）

 （19）

 （20）

 （21）

 （22）

 （23）

 （24）

 （25）

 （26）

 （27）

 （28）

 （29）

 （30）

 （31）

 （32）

(18)、(19)式表示工件的Map工序在某机器上加工与否及加工工作量的相互关系。(20)-(22)式表示工件的Map工序在一台平行机上的一个位置上只加工一次。(23)式表示工件的Map工序在机器上某位置的加工量与该机器上分配的工作量一致。(24)式表示工件的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。(25)-(28)式表示工件的Reduce工序只能在某一台平行机上加工，并且只加工一次。(29)式表示在平行机的位置安排Map工序任务量的上下限约束。(30)式表示在平行机的位置上只能安排某一工件的一个任务。(31)式表示在平行机上安排任务的位置是连续的。(32)式表示决策变量的取值范围。

# 3 鸡群优化算法（CSO）

Meng等[17]受到鸡群等级制度和鸡群觅食行为的启发，将整个鸡群分为中包含若干子群，每个子群都由雄鸡、雌鸡与小鸡三个类群组成；其中雌鸡有着最多的数目;适应度值最好的若干个体为雄鸡粒子，适应度值最差的若干个体为小鸡粒子，剩下的个体为雌鸡粒子。3个群体分别按照自己的方式在解空间中进行搜索，最后综合比较3个群体的适应度值，找出全局最优粒子和全局最优值。

1)对于公鸡，适应度值较好的公鸡相比适应度值较差的公鸡，能够在更大的范围内寻找食物，公鸡的位置更新公式如下：

 （33）

 （34）

式中，为均值为0，标准差为的一个高斯分布，为一个很小的常数，为所有公鸡中除去第个体外的任一个体。

2)母鸡的位置更新公式如下：

 （35）

 （36）

 （37）

式中，，表示学习因子；，为[0，1]之间均匀分布的随机数，为第只母鸡自身所在群中的公鸡，为整个鸡群中公鸡和母鸡中随机选取的任意个体，且。

3)小鸡的位置更新公式如下：

 （38）

式中，为第只小鸡对应的母鸡，是服从[0，2]的均匀分布的随机数，表述跟随系数，即小鸡跟随母鸡寻找食物，步长相对公鸡和母鸡更大。标准的鸡群优化算法流程见文献[17]。鸡群算法的数学模型构造上可以看出，它是粒子群算法和差分算法的结合体。但是由于基本鸡群算法缺乏变异机制，种群难以保持多样性，个体容易被局部极值吸引而导致早熟收敛，其次鸡群算法难以直接求解离散型的调度问题，为此本文对鸡群算法加以改进，便于扩大搜索范围和应用于MapReduce模型的调度问题。

# 4 改进的鸡群优化算法（ICSO）

# 4.1 编码方式

由于基本鸡群算法适用于连续型函数优化，而基于MapReduce模型的调度问题则属于离散型组合优化问题，因此运用鸡群算法求解MapReduce模型问题时，首先要明确的是解空间和问题空间之间的对应映射关系，即如何对算法编码进行合理设计，使其适用于MapReduce模型调度问题的求解。在改进鸡群优化算法的平行机调度中，运用实数编码的位置向量来表示机器调度序列。假设有个工件，按工件的释放时间排序（如果释放时间一样，按Reduce工序从大到小排序）。为了编程方便，本文采用至多维实数位置向量表示个工件和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，+1)，表示机器数量。编码分成三个部分，前面位的整数部分表示个工件的Map工序分割成几个部分;后面的实数数字：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。

针对随机产生维实数，先分析前个数（其中），代表每个工件的Map工序将要被分配的机器数量，比如第个工件由数量台机器来执行。其中表示向下取整，接下来分析维数值。

# 4.2 解码方式

用一个例子说明编/解码的过程。例如有2台机器处理3个工件，机器各自的加工速度为,；其中3个工件的基础信息见表1(按释放时间排序)：,，，。

**表1 3个工件2台机器的编码信息**

**Table1 Code information for 3 jobs 2 machines.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | … | 10 | 11 | 12 |
| 位置 | 2.338 | 1.426 | 2.153 | 1.841 | 2.422 | 2.131 | 1.114 | 2.229 | … | 1.015 | 2.318 | 1.479 |
| 解码 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 2 | … | 1 | 2 | 1 |

解码：

1）先取前位，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分别列出第1、3个工件的Map工序分割成2个部分，第2个工件Map工序只在1台机器上运行。

2）然后分析接下来的2+1+2=5位数字。例如，由于第1个工件的Map工序分配给2台机器，进而观察后续2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和第2台机器，分割的子任务数量看小数部分：，这里[]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。

3）最后看后位，取整得到的机器号表示个工件的Reduce工序分配给哪一台机器。这里，机器1来执行工件1的Reduce工序。同理可以推算出其他工件的Reduce工序机器分配，同时要注意必须在Map工序结束后才可以启动。解码后的具体调度方案见如图4：



**图4 2台平行机3个工件的调度方案(,)**

**Fig.4 Scheduling scheme for 3 jobs of 2 machines,**

# 4.3 单纯形扰动策略

为提高鸡群优化算法的搜索性能，本文引入单纯形法策略进行扰动，在一次迭代完成之后，利用单纯形法搜索策略，选择某一个的小鸡位置进行优化。找出最优点、次优点以及最差点，通过反射、压缩、扩张等操作更新最差点，形成一个新的多面体。它是一种局域的搜索方法。假设较差小鸡的位置为最优位置和次优位置，为最优位置和次优位置的中心。

1)反射操作：，为反射点，反射系数，通常取1。

2)扩张操作：，为扩张点，扩张系数，通常取2。

3)压缩操作：，为压缩点，压缩系数，通常取0.5。

4)收缩操作：，为收缩点，收缩系数与压缩系数相同。

本文的单纯形扰动策略的步骤如下：选取任意一个较差小鸡的位置为，分别计算出，，，，比较这些目标函数值，选取单纯形中最优的值和当前最优函数值比较，如果单纯形中的更优，则替换。

# 4.4 改进鸡群算法流程

综上所述，数值仿真MapReduce模型的改进鸡群算法流程可归结如下：

**Step1**初始化鸡群、算法基本参数：NP为种群个数，鸡群分组G，母鸡数量HN，公鸡数量RN，小鸡数量CN，问题维数D和最大迭代次数等参数等；

**Step**2计算鸡群的目标函数值，得到当前的最优函数值，保存好当前最好位置*pbest*和鸡群全局最好位置*gbest*，t=1。

**Step3**如果（即当前为第一代），排序适应度值，建立鸡群等级制度，将鸡群分成数个子群并确定母鸡粒子和小鸡粒子的对应母子关系，其中，表示开始更新等级制度、支配关系和母子关系的代数；否则进行各种个体的位置更新；

**Step4**使用公式(32)- (37)分别更新公鸡、母鸡和小鸡的位置，并分别计算子群个体更新后的适应度值。

**Step5**对改进后的鸡群与当前质量最优的鸡进行比较，如果质量更优，则更新鸡群的个体当前最好位置和鸡群全局最好位置；

**Step6**对小鸡群单纯形扰动机制，扰动后与当前质量最优的公鸡继续进行比较；若较好，则改变当前最优值，并用较好的鸡群替代较差的鸡群；

**Step7**若未达到最大迭代次数，则返回步骤3~步骤7，否则，输出全局最优位置；

# 5 问题的一个松弛下界

为了评价算法解的质量，首先对问题解的下界进行研究。

**定理1**由于Map工序的子任务可在多台机器上加工，相应的机器数量比较难以确定，故松弛二个条件。不失一般性，先假设，假设工件一下达，目前机器都是空闲的，该问题的下界为：

 （39）

证明：假设工件一下达，目前机器都是空闲的。将该工件的Map工序加工时间平摊到台机器，假设机器的加工速度最大，那么工件的Map部分加工时间为，那么Reduce工序的从下达到开始加工这个时间间隔至少为。考虑到Reduce工序的也有加工速度问题，那么Reduce工序的加工时间至少为，不考虑运输时间，那么这是一个工件的最少流程时间。显然表达式是最优解的一个下界。

# 6 数值实验

为有效衡量改进算法寻优性能，进行数据仿真实验。算法采用MATLAB2015a编程，实验运行环境为：CPU2。1GHz，内存4GB，Windows7操作系统(64位)，为验证算法寻优性能，分小数据和大数据两部分进行试验。小数据的时候，混合数学规划模型可用IBMCPLEX12。5软件进行精确求解；但是大数据量的时候，由于无法用CPLEX进行求解。本文CPLEX设置为3小时为上限，如果3小时跑不出来，就停止。用传统的算法如粒子群算法GA和标准鸡群算法求解并进行比较。

# 6.1 评判标准

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来衡量

 （40）

其中，是算法中计算获得的目标函数值，其中。是CPLEX计算出来的目标函数值。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，由CPLEX计算获得。本文CPLEX设置为3小时为上限，如果3小时跑不出来，就停止。

# 6.2 实验参数

为验证本文提出的数学模型和改进鸡群算法的性能，本文随机产生测试算例，并与基本鸡群算法（CSO）和粒子群算法(PSO)进行比较；算例规模见如下表2。算法的参数设置如表3所示。在三种算法中，种群数量都设定为*NP*=100，最大搜索次数均为。PSO算法中的c1和c2为粒子群的学习因子，惯性权重。

**表2 算例的参数取值范围**

**Table 2 The range of parameters for instance.**

|  |  |  |
| --- | --- | --- |
| 参数 | 描述 | 分类取值范围 |
| N | 工件数量 | 10,20,30,40,50;100,150,200 |
| M | 机器数量 | 2,4;4,8 |
|   | 机器速度 | U[1,2] |
|   | Map子任务标准长度 | U[1,10] |
|   | 释放时间 | U[1,1000],U[1,3000] |
|   | Map子任务的数量 | U[1,10]取整 |
|   | 允许最大等待时间 | 60 |
|   | Reduce的标准长度 | U[1-10] |
|   | 机器间的运输时间 | U[1-8] |

**表3 算法的的参数设置**

**Table 3 Parameter values for each algorithm.**

|  |  |
| --- | --- |
| 算法 | 分类取值范围 |
| PSO | c1 = c2 = 1.93, |
| CSO | G=10,RN=20%*NP*,HN=60%*NP*CN=NP-RN-HN, |
| ICSO | 参数见CSO |

# 6.3 实验分析

对于上述实例用ICSO、CSO和PSO三种算法进行求解，其中小数据量的时候还用上Cplex求解，并对实验结果做参照对比。针对机器数量和工件规模不同，本文进行了16组实验，每个实验在同一算法下运行5次，并求得5次运行结果的平均最优值；根据算式（40）计算出每个算法的评判指标。具体运行结果数据列于表4。但对于图表而言，由于篇幅有限，本文就选取了100个工件4个机器和200个工件4个机器的情形，每个情形下随机选取1个运行结果，具体见图5和图6。



**图5 4台机器100个工件的算法收敛曲线图**

**Fig. 5 Algorithm convergence curve for 100 jobs of 4 machines**



**图6 4台机器200个工件的算法收敛曲线图**

**Fig.6 Algorithm convergence curve for 200 jobs of 4 machines**

从图5，图6分别在N=200，M=5和N=200，M=8的两种情况下描绘了各类算法收敛曲线的分布；从图中可以看出，当达到一定的迭代次数的时候，优化目标无法再改进。另一方面，PSO和CSO算法有着类似的收敛速度，而ICSO的收敛性能优于其他二个算法。

**表4 各算法的性能指标比较**

**Table 4 The performance indexes of each algorithm are compared.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 规模 |  | LB | CPLEX | PSO | CSO | ICSO |
| 目标值 | GAP(%) | 目标值 | GAP(%) | 目标值 | GAP(%) |
| 10/2 | 60 | 262.61 | 275.12 | 278.26 | 1.14 | 277.95 | 1.03 | 268.280 | 0.61 |
| 10/4 | 60 | 160.67 | 168.76 | 170.55 | 1.07 | 170.34 | 0.94 | 169.76 | 0.59 |
| 20/2 | 60 | 498.15 | 518.38 | 528.54 | 1.96 | 527.89 | 1.83 | 523.41 | 0.97 |
| 20/4 | 60 | 306.91 | 320.24 | 326.25 | 1.88 | 325.97 | 1.79 | 323.18 | 0.92 |
| 30/2 | 60 | 641.35 | - | 681.18 | 6.21 | 680.73 | 6.14 | 673.94 | 5.08 |
| 30/4 | 60 | 436.77 | - | 464.76 | 6.41 | 463.99 | 6.23 | 459.74 | 5.26 |
| 40/2 | 60 | 1016.49 | - | 1080.62 | 6.27 | 1079.93 | 6.20 | 1069.074 | 5.13 |
| 40/4 | 60 | 612.64 | - | 653.01 | 6.59 | 651.84 | 6.40 | 645.05 | 5.29 |
| 50/2 | 60 | 1326.47 | - | 1409.77 | 6.28 | 1408.58 | 6.19 | 1393.59 | 5.06 |
| 50/4 | 60 | 753.82 | - | 802.74 | 6.57 | 801.61 | 6.42 | 792.34 | 5.19 |
| 100/4 | 60 | 1653.72 | - | 1765.34 | 6.75 | 1763.20 | 6.62 | 1738.23 | 5.11 |
| 100/8 | 60 | 1025.48 | - | 1096.13 | 6.89 | 1094.91 | 6.77 | 1079.11 | 5.23 |
| 200/4 | 60 | 2912.58 | - | 3110.93 | 6.81 | 3108.59 | 6.73 | 3064.03 | 5.20 |
| 200/8 | 60 | 1897.81 | - | 2029.89 | 6.96 | 2027.62 | 6.84 | 1998.01 | 5.28 |
| 300/4 | 60 | 4412.36 | - | 4726.95 | 7.13 | 4721.64 | 7.01 | 4647.09 | 5.32 |
| 300/8 | 60 | 3207.53 | - | 3439.11 | 7.22 | 3436.55 | 7.14 | 3382.34 | 5.45 |

通过不同规模下的实验，很好地验证了改进鸡群算法(ICSO)的有效性。该算法在所有实验的结果都超过GA和基本鸡群算法(CSO)，说明其运行的性能较好，算法改进效果和解的质量令人满意。

# 7 结束语

工件的部件在相邻加工阶段之间的等待时间具有一定的上限，是工业制造中常见的一类约束；本文探讨了具有任务释放时间、机器间工件具有运输时间约束下工序间等待时间受限的MapReduce模型同类机调度，建立了混合整数规划模型，分析了问题的松弛下界和目标函数之间的关系。最后，为了克服鸡群算法易早熟收敛等缺点，本文设计改进了鸡群算法(ICSO)，采用单纯形改进策略，数值实验表明所改进的算法的有效性能。下一步可以研究具有恶化效应或Shuffle时间的基于MapReduce模型的调度。

# 参考文献

[1] Kim H J, Lee J H. Three-machine flow shop scheduling with overlapping waiting time constraints[J]. Computers & Operations Research, 2019, 101:93-102.

[2] Hong T Y, Chien C F, Wang H K, et al. A two-phase decoding genetic algorithm for TFT-LCD array photolithography stage scheduling problem with constrained waiting time [J]. Computers & Industrial Engineering, 2018, 125(2):200-211.

[3] 王柏琳, 李铁克. 等待时间受限的流水车间调度问题的若干性质[J]. 系统工程理论与实践, 2012, 32(6):1346-1352.

[4] 王柏琳, 李铁克. 启发式算法求解等待时间受限的两阶段流水车间调度问题[J]. 管理工程学报, 2014, 28(2):182-190.

[5] 常晓坤, 董明. 不确定环境下等待时间受限的混合流水车间调度问题研究[J]. 工业工程与管理, 2016, 21(5):35-41.

[6] An Y J, Kim Y D, Choi S W. Minimizing makespan in a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times [J]. Computers & Operations Research, 2016, 71:127--136.

[7] Alan J. Soper, Vitaly A, Strusevich. Schedules with a single preemption on uniform parallel machines [J]. Discrete Applied Mathematics, https://doi.org/10.1016/j.dam.2018.03.007.

[8] Lin S W, Yian K C. Uniform parallel-machine scheduling for minimizing total resource consumption with a bounded makespan [J]. IEEE Access, 2017, P(99):1181-1193.

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

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

[11] Chen F, Kodialam M S, Lakshman T V, et al. Joint scheduling of processing and Shuffle phases in MapReduce systems[C]. International conference on computer communications , 2012: 1143--1151.

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

[13] Tang S J, Busung L,Bingsheng H. Dynamic Job Ordering and Slot Configurations for MapReduce Workloads [J]. IEEE Transactions on Services Computing, 2016, 9(1): 4--17.

[14] Tian W, Luo G, Tian L, et al. On Dynamic Job Ordering and Slot Configurations for Minimizing the Makespan Of Multiple MapReduce Jobs [J]. arXiv: Data Structures and Algorithms, 2016.

[15] Verma A, Cherkasova L, Campbell R H, et al. Two Sides of a Coin: Optimizing the Schedule of MapReduce Jobs to Minimize Their Makespan and Improve Cluster Performance[C]. Modeling, analysis, and simulation on computer and telecommunication systems, 2012: 11--18.

[16] Verma A, Cherkasova L, Campbell R H, et al. Orchestrating an Ensemble of MapReduce Jobs for Minimizing Their Makespan [J]. IEEE Transactions on Dependable and Secure Computing, 2013, 10(5): 314--327.

[17] Meng X, Liu Y, Gao X, et al. A New Bio-inspired Algorithm: Chicken Swarm Optimization [M]. Advances in Swarm Intelligence. Germany: Springer International Publishing, 2014: 86--94

[18] 许世鹏, 吴定会, 孔飞等. 基于改进鸡群算法的柔性作业车间调度问题求解[J]. 系统仿真学报, 2017, 29(7):1497--1505.

[19] Dengke Xing,Jianfei Liu,et al. Mitigation of crosstalk based on CSO-ICA in free space orbital angular momentum multiplexing systems[J]. Optics Communications 2018, 423(2): 200--206.

[20] 黄霞, 叶春明, 郑军. 混合改进搜索策略的鸡群优化算法[J]. 计算机工程与应用, 2018(7)：178--181.