****

硕士学位论文

|  |
| --- |
| 基于协同候鸟前以算法的全自动免疫分析 |
| 检测设备的分批优化调度问题研究 |

|  |  |
| --- | --- |
| 作者姓名 |  |
| 学科专业 | 控制理论与控制工程 |
| 指导教师 |  |
| 所在学院 | 自动化科学与工程学院 |
| 论文提交日期 | 2019年4月 |

一、绪论

# 1.1 课题背景和意义

# 1.2 免疫分析检测设备调度问题的国内外研究现状

# 1.3 论文主要研究内容及其章节安排

二、免疫分析检测设备分批优化调度问题分析及建模

# 2.1 免疫分析检测设备相关知识

## 2.1.1 免疫检验基本原理

## 2.1.2 免疫检验设备基本组成

## 2.1.3 免疫检验流程

# 2.2 分批调度问题特点分析

## 2.2.1 问题描述

讲免疫分析检测设备为什么要分批

分批调度是Reiter提出的一种提高车间生产效率的技术[1]。在一些实际的生产环境中，工件加工往往会呈现批量性。经典的调度一般是把一整批同类的工件视为一个整体，以整个批次为单位放入生产系统，整个批次里所有工件的上一个工序加工完成之后才能进行批次的下一个工序，导致较长的等待时间。而且考虑到实际生产系统中往往会有功能相似、可以并行加工的机器，如果把批量较大的批次划分为子批，以子批为单位进入生产系统，让子批在不同的机器内并行加工，既能充分发挥并行机器的生产能力，又能减少工件的等待时间。

由于流水车间调度问题（Flow Shop Scheduling Problem，FSSP）比较简单，分批调度最早被广泛地应用于该问题中[2][3][4]。后来分批调度也逐渐地被应用于更复杂的生产环境中，例如并行机加工问题（Parallel Machine Scheduling Problem，PMSP）[5][6]和作业车间调度问题[7][8]，带装配的作业车间调度问题（Assembly job shop problem，AJSP）。柔性作业车间分批调度问题是比作业车间更复杂的问题，具（Felxible Job Shop Scheduling Problem，FJSP）有更复杂的约束，已经有少部分学者开始研究它的分批调度问题了，但是相关研究还是相对比较少[9][10]。

## 2.2.2 问题特点分析

说清楚分批有什么好处：减少等待时间，不用等全批加工完就能让一部分进行下一个工序了（overlap），节省准备操作时间，更好地利用柔性机器并行加工，参考Benefits of lot splitting in job-shop scheduling

说清楚为什么不能把每个工件视为一个子批：问题规模大，寻优困难，准备操作时间多

# 2.3 柔性作业车间分批调度问题

## 2.3.1 柔性作业车间分批调度问题背景

## 2.3.2 柔性作业车间分批调度问题分类

# 2.4 免疫检测设备的柔性作业车间分批调度模型

根据前文对免疫分析检测设备的分析，把其分批优化调度问题归为一类准备操作可分离的柔性作业车间分批调度问题（Flexible Job Shop Scheduling Problem with Lot Spliting，LSFJSP），它属于柔性作业车间调度问题的一种拓展，是一个比较复杂的问题，同时具有作业车间、柔性、分批的诸多属性和约束。

有个批次的工件待加工，一个批次内有若干个类型相同的工件，要使用M台机器完成所有的加工，每种工件都有不同的加工工序。每个工序都有不同的可选加工机器，不同机器的加工时间不一样。由于批次的工件量比较大，可以把每个批次都划分为多个数量不等的子批，以子批为单位按照工序顺序进入不同机器进行加工。分批调度问题的任务是，找到最优的子批划分方案，即每个批次分成多少个子批，每个子批包含多少个工件，还要找到最优的调度方案，即每个子批在机器上的加工顺序，从而使所有批次的完工时间最小化。

该问题还有以下的属性：

（1）以子批为加工单位。一个子批里所有工件的上一个工序都完成后，才能进入下一台机器，以批为单位开始下一个工序。一个子批在某个工序的加工时间等于该子批所有工件的该工序加工时间之和；

（2）考虑准备操作。机器在加工一类工件之前还需要进行准备操作。机器连续加工多个同类工件之前只需要进行一次准备操作，如果机器连续加工同一个批次内的不同子批，那么只需要在同批的第一个子批加工前进行一次准备操作；

（3）准备操作时间与工件类型有关。不同种类的工件有不同的准备操作时间；

（4）准备操作可与工件分离。在工件到达机器之前，只要机器空闲了，就可以提前进行准备操作，不需要等工件到达之后才进行准备操作；

（5）准备操作时间与工序加工时间分开考虑。工序加工时间不包含准备操作时间；

（6）一致分批。子批的划分在所有工序加工过程中是不变的；

（6）不允许抢占。一个子批在某台机器未完成加工的时候，其他子批不允许抢占用该机器加工；

（7）同一时刻每台机器最多只能加工一个工件；

（8）同一时刻每个子批最多只能进入一台机器进行加工；

（8）保证工序约束。每类工件的工序顺序都是固定的，每个子批完成上一个工序的加工之后，才能开始下一个工序；

（9）所有批次的工件都在零时刻释放，即所有工件的第一个工序在零时刻都进入待加工状态；

（10）机器容量限制。每台机器可容纳的子批量是有上限的，因此在划分子批的时候，每个子批的子批量不可以超过机器的容量限制；

# 2.5 柔性作业车间调度问题求解方法

按照是否等量，解决子批划分的方法可以分为两类。第一类是等量分批，每个批次都按照等量的原则被划分为若干个子批，每个批次内所有子批的子批量相等。第二类是不等量分批，不需要按照等量的原则划分子批，每个批次内的各个子批批量可以不相等。许多研究采用了等量分批，因为它是最直接、最简单、最容易实现的方法，它只需要确定每个批次的子批数就可以了，不需要额外确定每个子批的子批量，因此它所需要的计算也是最少的[11][12]。但是等量分批不能同时优化子批数和子批量，只能做很粗糙的子批划分，不能根据机器特点灵活分批，得到的分批方案可能比不等量分批差。同时，由于不同机器的性能不一样，不同机器在加工批量相同的同类子批的加工时间不一样，可能会导致机床的空闲时间增加[13]。虽然不等量分批能更灵活地分批，但是搜索不等量分批的最优解比搜索等量分批的最优解难度大，因为其搜索域更大更复杂。只要配合寻优能力较强的求解算法，不等量分批就能得到更优、更精细的分批方案。因此本文采用不等量分批。

按照分批是否一致，解决子批划分的方法分为两类。第一类是不一致分批，同一个批次在不同的工序中有不同的子批划分方案，例如子批数可以不同，子批量也可以不同。第二类是一致分批，同一个批次在所有工序中都使用相同的子批划分方案，即在所有工序的加工过程中，所有子批都保持不变。不一致分批是灵活性最高的子批划分方法，但是在实际生产过程中，它需要比较复杂的操作以及管理手段，对生产系统有较高的要求，较少被应用，因此本文不采用不一致分批。

三、候鸟迁移算法理论分析

# 3.1 基本候鸟迁移算法

候鸟迁移算法（Migrating Bird Optimization, MBO）是2012年【Duman】提出的一种优化算法。这种算法模拟候鸟迁移的群体性行为。鸟类在飞行的时候，由于翅膀的结构特点，左右翅尖会产生一定的漩涡气流，可以给后面的鸟提供抬升力，从而节省跟随鸟的能量。候鸟群体经常使用V字型的队伍来飞行，即由一个领头鸟来带领整个鸟群，领头鸟的左右翅膀分别跟着一列跟随鸟，形状跟字母“V”十分相似。Duman认为，候鸟使用V字队形飞行可以节省能量，还可以防止鸟之间的碰撞，并互相保持视觉上的联系。在V字形队伍中，领头鸟需要最多的能量，在左右翼的跟随鸟可以从它们前面那只鸟获得一定的抬升力。比起单独飞行，这种飞行方式可以节省大多数鸟的能量。由于领头鸟需要消耗很多能量，所以领头鸟会定期更换。

MBO是一种基于邻域搜索的算法。从领头鸟，到后面每一只跟随鸟，它们都会通过搜索邻域来更新自身。同时，利益机制还会让前鸟帮助它的跟随鸟进化。具体来说，前鸟会把自己未使用的优秀邻域解分享给它的跟随鸟。鸟群会保持同一个V字型来更新迭代多次，直到领头鸟累了需要替换，原来V字型才会改变。领头鸟从最前面的位置退到队伍末尾，鸟群的一轮飞行至此结束。左右翼第一只跟随鸟的其中一只会成为新的领头鸟，鸟群的新一轮飞行从此开始，按照新的V字型更新迭代多次。

以下是MBO算法的一些参数：

：鸟群里鸟的数量

：每只鸟邻域候选解的个数

：前鸟给其跟随鸟分享的邻域解的个数

：一轮飞行中迭代的次数

：飞行的总轮数

以下是描述MBO算法所需要的一些变量：

：鸟群中第只鸟

：领头鸟

：左翼跟随鸟中的第只鸟

：右翼跟随鸟中的第只鸟

：的候选集，里面包含的个候选解

MBO算法的具体流程为：

step1：随机初始化。随机生成个鸟个体，随机选择一直鸟为领头鸟，其余鸟为跟随鸟，并随机将它们组织为V字型，令，；

step2：更新领头鸟。对邻域搜索，得到个邻域解放入候选解集中，如果最优的个体比更优，则用替代；未被使用的邻域解之中最优的个放入左右翼第一只跟随鸟的候选集和中；

step3：更新左跟随鸟；

step3.1：令

step3.2：如果，则需要把的前鸟未使用的个最优邻域解放入候选集；如果，那么进行step3.3;

step3.3：对进行邻域搜索，得到的个邻域解放入候选集；

step3.4：如果候选集中最优的个体比更优，那么用替换；

step3.5：令候选集，如果，则令，否则进行step4；

step4：更新右跟随鸟；

step4.1：令

step4.2：如果，则需要把的前鸟未使用的个最优邻域解放入候选集；如果，那么进行step4.3;

step3.3：对进行邻域搜索，得到的个邻域解放入候选集；

step3.4：如果候选集中最优的个体比更优，那么用替换；

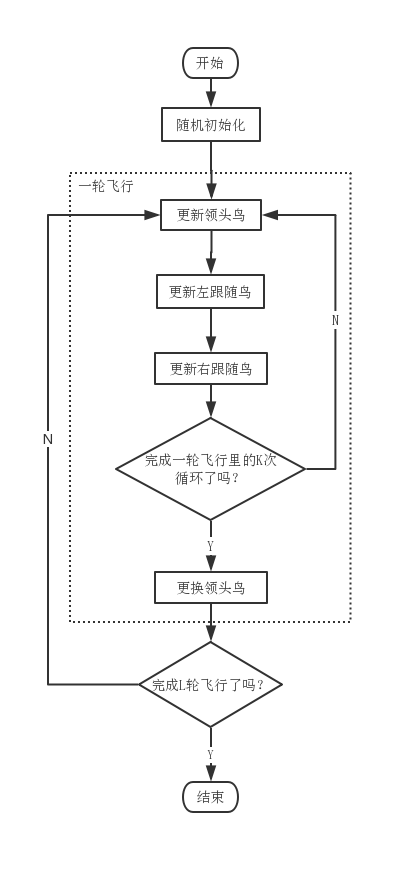
step3.5：令候选集，如果，则令，否则进行step5；

step5：如果，那么，回到step2，否则进行step6；

step6：领头鸟更换，对比左右翼的第一只跟随鸟，如果比更优，那么领头鸟退到左翼的队伍末尾，成为新的领头鸟，否则领头鸟退到右翼的队伍末尾，成为新的领头鸟；

step7：如果，那么进行新一轮的飞行，，，回到step2，否则结束，输出寻优结果。

基本MBO算法的流程图如下所示：



# 3.2 候鸟迁移算法研究现状

# 3.3 候鸟迁移算法性能分析

## 3.3.1 多样性损失分析

对于基于种群的智能进化算法来说，“种群收敛”代表种群的大多数个体进化到一定程度之后趋于一致。种群收敛并不是越快越好，它需要兼顾算法全局搜索和局部搜索的平衡。如果种群收敛得太快，在寻优的前期就严重损失了多样性，在全局搜索还未足够的时候过早地专注于局部搜索了，那么就算种群收敛了，也只能收敛到一个比较差的解。通常这称为“早熟收敛”，或者“停滞”。种群收敛快的算法通常很容易陷入局部最优。出现这种情况是由于多样性损失严重，种群失去了继续寻优的活力。

下面详细分析基本MBO算法中加剧多样性损失的一些地方。

3.2.1.1 基于邻域共享的利益机制

在基本MBO算法中，为了体现前鸟的翅尖对跟随鸟提供抬升力的这种利益机制，前鸟会把自己未使用的优秀邻域分享给其跟随鸟，即进行邻域共享。它的好处是，当某只跟随鸟不能通过自身邻域搜索来提升自己，此时它还可以通过前鸟分享给它的优秀邻域解来提升自己。通过邻域共享，比较差的个体可以在前鸟的协助下，通过一次或者多次迭代快速进化为比较优秀的个体，因此邻域共享有助于种群快速收敛。

基本MBO算法的邻域共享在加快种群收敛的过程中，同时也在大大减少种群的多样性，容易导致早熟收敛，陷入局部最优。前鸟的邻域解替换了当前跟随鸟之后，对于前鸟邻域的挖掘是有利的，有更多个体能集中在前鸟的邻域附近搜索。但是当前跟随鸟所在的邻域却失去了被挖掘的机会，随着迭代不断进行，越来越多有潜力的区域被忽略了，种群中越来越多的个体被局部最优个体替代，种群的多样性损失越来越严重，种群越来越难以跳出局部最优。

3.2.1.2 种群异步更新

在基本MBO算法中，整个种群所有鸟的更新是异步更新。异步更新指的是，对于V字型的左翼队列或者右翼队列，都是从第一个跟随鸟开始，按照队形顺序，前面一只鸟生成候选集并更新自己之后，后面一只鸟才能根据已更新的前鸟来生成自己的候选集并更新，按顺序直到排末尾的跟随鸟更新，整个种群的更新才完成。

因为异步更新是在前鸟更新之后才去更新的，所以在一次种群的更新过程，前鸟的优秀邻域分享给第一只跟随鸟并更新之后，第一只跟随鸟落入了前鸟的邻域范围，当更新后的第一只跟随鸟把自己的邻域个体分享给第二只跟随鸟并更新之后，第二只跟随鸟也很有可能落入前鸟的邻域范围。使用异步更新模式的邻域共享不仅可以让第一只跟随鸟进入前鸟的邻域，还有可能让后面多只跟随鸟都落入前鸟的邻域，特别是当前鸟都优于后面的多只跟随鸟的时候，这种情况时常发生。在整个种群的一次更新过程中，左翼或者右翼队伍可能会有连续几个个体被其某只前鸟的邻域个体替换，这表示在一次迭代的过程中，异步更新加剧了种群多样性的损失。虽然异步更新加快了种群的收敛速度，但是却让算法更快陷入局部最优。

（考虑画图示意）

## 3.3.2 群体寻优效率分析

基于群体的进化计算方法的最大的特点是，可以通过个体之间互相协作达到共同进化的效果。个体间协作的效率对算法寻优效率有着很重要的影响。协作的方式一般是把个体按照某种拓扑结构组织起来，然后按照规则互相传播有效消息，互相学习优秀编码。不同的拓扑结构对群体进化有不同影响。高效的拓扑结构可以充分发挥群体协作的潜力，有助于算法的高效搜索。丰富多样的拓扑结构可以让群体有多种进化方向，让种群保持全局寻优的活力。

MBO就属于这种基于群体的进化计算方法。下面分析MBO算法中影响群体寻优效率的一些地方。

3.3.2.1 轮替式的领头鸟更换

在基本MBO中，领头鸟的替换是轮替式的，跟随鸟按照V字队形的前后顺序，轮流替换旧的领头鸟，成为新的领头鸟。在领头鸟疲劳了之后，它会退到V字型队伍的队尾，此时只有紧跟领头鸟的两只跟随鸟有机会成为新的领头鸟。无论排在队伍中间和队伍后面的鸟有多优秀，无论它们是否比新的领头鸟优秀，它们在本轮领头鸟替换过程中是没有机会成为领头鸟的。假如种群中有只鸟，那么排在中间的优秀个体最多要等待轮才能成为领头鸟，而且平均需要再平均轮才能再次成为领头鸟，每轮飞行需要迭代次，所以等待时间总计平均需要次迭代。这种轮替式的领头鸟更换机制并没有让优秀个体在种群中发挥带领作用，影响种群的寻优效率。

3.3.2.2 过于固定的协作拓扑

在3.3.3.1所分析的轮替制下，领头鸟更替之前和更替之后，几乎所有鸟在V字型队伍里的顺序是没有变化的。假如种群中有只鸟，对于除了领头鸟个左右翼第一只跟随鸟以外的其他鸟来说，它们各自的前鸟，平均每轮飞行之后才可能变化一次，每轮飞行需要迭代次，这就导致了平均次迭代过程中，一只鸟的前鸟都是固定不变的。鸟群队伍的排序比较固定和单一，每只鸟的眼光都只能长期局限于某一只前鸟，鸟个体之间协作的拓扑结构比较固定，种群长期只能按照同一个方向去进化，没有充分发挥个体间协作的潜力，不利于种群的高效寻优。（考虑插图示意拓扑怎么固定）

## 3.3.3 局部搜索能力分析

MBO是一种基于邻域搜索的进化计算方法，它的局部搜索能力非常强。对于跟随鸟来说，每一次迭代每个个体都会进行次的邻域搜索，产生个邻域解，每一个个体都会通过自身或者前鸟的邻域解来进化，因此每一只鸟对于自身邻域的探索是比较充足的。对于领头鸟来说，每次迭代都会进行次邻域搜索，产生个邻域解，比跟随鸟产生的邻域解更多，因此领头鸟能得到更充分的邻域探索。总的来说，每一只鸟所在的邻域都可以得到充分的局部搜索。在算法前期，种群多样性还比较高的时候，这种能力有助于同时挖掘多个区域的潜力，在算法后期，种群开始收敛的时候，这种能力有助于深入挖掘目标区域的最优解。

在局部搜索能力方面，相比于以下几种经典群体进化计算方法，MBO与它们的区别在于：

（1）PSO的搜索步长是根据个体在搜索域之间的距离决定的，在搜索中后期种群逐渐收敛的时候，由于个体趋于相近，个体之间的距离非常小，导致搜索步长变得非常小，此时局部搜索的效率可能比较低。而MBO的搜索步长不会因为中后期收敛而变得很小，因此它能保持高效的局部搜索；

（2）ABC主要是跟随蜂在做局部搜索，但是跟随蜂会根据蜜源适应度的大小来选择蜜源，它会偏向于选择适应度高的蜜源。因此适应度低的蜜源很可能得不到局部搜索，这些蜜源得不到有效的局部搜索之后，很快就会被丢弃。而MBO会给每只跟随鸟相同的局部搜索机会，给予充分的局部搜索；

（3）TLBO的教师阶段是最优个体引领整个种群进化，相当于所有个体朝着最优个体的方向做局部搜索。当最优个体多代不变化的时候，整个种群多次迭代都朝着同一个方向去搜索，容易忽略其他方向的潜力。而MBO虽然也有领头鸟带领种群进化，但是领头鸟会定期更换，而且MBO的邻域搜索不会限制个体的进化方向，所以能挖掘更多区域的潜力。

# 3.4 应用于分批调度问题的优势分析

根据后面第四章的设计，本文使用一个矩阵来代表分批方案，其中矩阵的每一个向量称为分批向量，分别代表每个批次的具体分批方案。每一个分批向量都有批量总数约束，如果对两条分批向量进行交叉寻优，那么很大概率会破坏约束，产生不可行的子代，使交叉无效。如果对两条分批向量进行加减运算，得到的和或者差是无意义的，而且也很有可能破坏约束，得到不可行的结果，使运算无效。因此，对分批向量的搜索不能依靠交叉，也不能依靠加减运算，只能依靠基于变异的邻域搜索。

由于本文的分批调度问题是不等量分批，所以分批子问题的复杂度非常高，不仅要确定每一个批次的子批的数量，还需要确定每一个子批的子批量。例如工件数为10的批次可以有35种不同的分批方式，工件数为15的批次可以有110种不同的分批方式，工件数为20的批次则可以有434种不同的分批方式。因此该问题对算法的邻域搜索能力要求非常高，分批向量的寻优是分批调度问题中的一个难点。

而根据3.3.3所分析，MBO是一种基于邻域搜索的群体智能进化算法，它刚好契合了分批调度寻优的特点，它的邻域搜索为分批向量的寻优提供了强有力的支持。邻域搜索在不破坏约束的前提下，能对分批向量进行有效的搜索。因此本文使用MBO算法来解决分批调度问题。

四、针对分批调度问题的约束处理策略

# 4.1 引言

根据2.2的分析，柔性作业车间分批调度问题同时具有作业车间、柔性、分批的属性和约束，约束条件多而且复杂，这对问题求解产生了两方面的问题：

（1）不可行解导致寻优效率低下。由于约束条件比较多，任何约束在搜索过程中都很容易被打破，得到不可行解。对于不可行解，一般的处理方法包括解码时修补，直接丢弃或者是适应度惩罚。如果采用解码时修补的方法，每次解码时都需要执行一次修补算法，增加解码的时间复杂度，同时会大量改变原始解的编码信息，对该解所做的搜索就白白浪费了，搜索效率低下。如果采用直接丢弃，对于这种约束条件严格且数量多的问题，得到不可行解的概率非常高，如果都直接丢弃的话，群体进化难以进行。如果使用适应度惩罚的话，任由不可行解在种群内进化，可能导致整个种群都是不可行解。总的来说，不可行解的出现会导致算法效率的低下，因此需要避免不可行解的出现[14]。

在本章的4.2节，针对分批调度这一复杂问题设计了一套编码解码方案，它既可以有效表达一个分批调度方案，由于编码结构的独特性，还杜绝了不可行解的出现，提高寻优的效率；

（2）可行域太复杂，导致寻优困难。符合约束的解域是可行域，复杂的可行域具有很不规则的非凸边界，而且可能具有多个峰值，搜索的难度非常大。这要通过增加算法的搜索能力来解决。

在本章的4.3节，针对分批调度这一复杂问题设计了多种搜索算子，使用粗粒度和细粒度配合搜索，融入了启发式及其他复合搜索策略，提高算法在复杂可行域搜索的能力。

# 4.2 编码解码方案设计

编码解码方案是分批调度问题和优化算法之间的桥梁。

在分批调度问题中，编码代表一个完整分批调度方案的信息，在基于群体的优化算法里，它代表种群内的一个个体。良好的编码方案能有效地表示分批调度方案的信息，还能简化算法的搜索域，避免不可行解的出现，提高寻优的效率。

解码方案是由一个个体的编码求解目标函数的方法。通过解码方案，能够计算得出每个工件在机器上的加工起始时刻和加工结束时刻，可以计算每一台机器最有一个工件的加工结束时刻，从而得到目标函数值。良好的解码方案可以在原始编码的基础上，加入合适的启发式规则来优化调度方案，有助于提高寻优精度。

## 4.2.1双矩阵编码

柔性作业车间的分批调度问题分解为两个子问题，分批问题（SP1）和调度问题（SP2），一个完整的分批调度方案应该包含这两方面的信息，因此一个个体的编码也必须包含这两方面的信息。

分批编码代表SP1的信息，它要确定每一个批次分别要分多少个子批，即子批数，而且每个子批包含多少个工件，即子批量。调度编码代表SP2的信息，它要确定所有子批应该以怎样的顺序在机器上进行加工。分批方案与调度方案存在着互相影响的关系，因此两个编码的设计不能被割裂开来，而是要兼顾两方面的特点和需求。相关学者在这方面设计了不同的方法，做了不同的试验，下面对不同的方法进行分析。

一部分研究为了简化分批的难度，使用等量分批的编码。只需要确定每个批次的子批数，根据等量分批的原则就能确定每个子批的子批量，而不需要在编码里体现子批量[15][16][11]。由此可以简化编码的设计，减小编码的长度。由于编码长度减小了，寻优难度也有所减小，但是这种编码只能优化子批数，不能优化每个子批的批量，因此寻优的结果可能不太理想，本文不采用等量分批编码。

白俊杰提出了一种基于游标的柔性分批编码，同时配合基于工件的调度编码[13]。让每一位基因的每一位数作为分批的游标，这种编码方法虽然能实现不等量分批，而且可以把分批编码的长度控制为与批次数相同，但是基因表示非常累赘。为了保证基因的可行还需要额外多加其他约束，对于批量大于10的实例，还需要使用多位十进制数表示一个基因，对于批量大于10且不为10的倍数的批次，情况就更为复杂，使用非常不方便。在使用搜索算子的时候，非常容易产生不可行解。白俊杰配合使用了传统的基于工序的编码作为调度编码，在编码里体现每个子批的每个工序的顺序。但是对在本问题中，子批数是不一定，不同分批方案下的调度编码长度是不一样的，个体之间进行交叉会产生不可行解。因此基于游标的分批编码与基于工序的编码组合实用性不强。

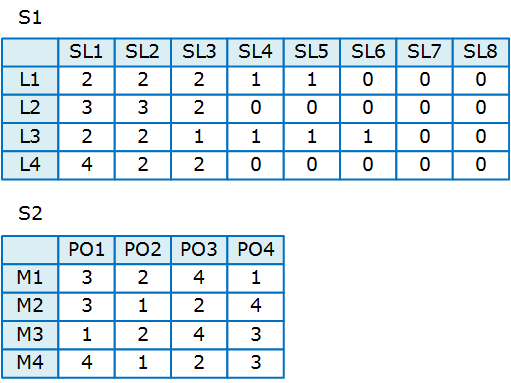
王万良使用了基于分批方案号的编码方法，同时也配合了基于工序的调度编码。首先对每个批次的所有子批划分方案罗列出来并编号，编码分为两部分，用无意义的“0”隔开，第一部分用来指定每个批次的子批划分方案编号，第二部分用来指定每一个子批的大小[17][18]。由于每个批次的子批数不是确定的，该编码第二部分的长度也是不确定的，因此只有编码的第一部分可以进行个体之间的交叉，而对方案编号进行交叉属于间接改变分批方案，难以把握改变的跨度。同时配合使用了基于工序的调度编码，同样是因为不同分批方案下的调度编码长度不一，个体之间交叉会产生不可行解。为了解决这个问题，一些研究规定只有分批方案完全相同的个体才能做调度编码的交叉[15][19]。虽然这能避免进化产生的不可行解，但是可交叉的个体可能很少，特别是在搜索前期，因此严重限制了算法寻优能力。

Defersha使用一种实数分批编码，配合基于工序的调度编码[9][20]。每个基因都在之间取值，其值占所有基因总和的比例就是该子批数占批次工件总数的比例。基于工序的调度编码都按照最大子批的规格来构造，以此保证调度编码的长度和批次参数相同，使不同分批方案的调度编码都可交叉，保证解可行。但是实数分批编码使用实数搜索域做离散问题的搜索会出现大量冗余，导致搜索难度增大。

Wong提出了一种3D编码方式，其中的分批编码由多个向量组成，第一个向量表示每个批次的子批数，其余向量分别表示每个批次里每个子批的子批量[21]。虽然分批编码里每个向量的长度不固定，但是向量个数是固定的，例如对于有个批次的实例，就一定有个向量来组成分批编码。而且这种编码方式非常直观，把每个子批量都放在编码里，各种搜索算子可以直接改变子批量的大小。同时配合基于偏好顺序的调度编码，在编码里体现机器对于不同批次工件的偏好顺序。Han也使用了基于偏好顺序的编码，但是他让所有机器都使用同一个偏好顺序[16]。对于柔性车间来说，不同机器的加工特点是不相同的，例如它们可加工的工序不相同，可加工的批次不相同，加工时间也不尽相同。因此让所有机器都使用同一个偏好顺序不能让机器发挥其自身优势，甚至会导致较长的等待时间。而Wong让每台机器都有各自的偏好顺序，解决了这一问题。相比于基于工序的编码，基于偏好的调度编码的优点是能避免不可行解。

根据以上分析，为了避免其他编码方式中出现的各种问题，本文借鉴Wong的思路，改进Wong的3D编码方式，提出一种针对分批调度问题的双矩阵编码。

首先设计一种矩阵式分批编码（S1）。矩阵内每一个向量都代表着一个批次所有子批的批量，对于有个批次的实例，如果最大子批数为，那么矩阵大小为。相比起Wong的编码，这种编码呈现固定大小的矩阵形式。对于4批次4机器的问题，如果每个批次工件数为8，那么最大子批数为8，调度编码矩阵大小为，每一行代表每个批次的子批量，如图所示，L1~L4分别代表4个批次。对于子批数不足的批次，在向量中填充“0”以达到的长度。通过统计向量中非“0”的基因个数就能知道该批次的子批数。相比于Wong的编码，本编码不需要额外增加一条向量来记录每个批次的子批数。在本文的编码解码方案里，一个批次向量里面每一个基因代表一个子批量，子批量的顺序是无意义的，向量内基因不同的排列方式都代表同一个分批方案，因此会产生冗余。为了减小冗余，需要对每一个批次向量内的基因进行降序的排序，保证每一个编码的唯一性。使用该编码之后，就可以同时优化子批数和各个子批的批量了。



同时配合使用基于偏好顺序的调度编码（S2）。这种编码是Davis在1985年提出的，在车间调度问题里是一种常用的编码方法[22]。Ponnambalam曾经对作业车间的四种常见调度编码做了对比，包括基于工序的编码、基于工序的编码、基于偏好顺序的编码、基于有限规则的编码，通过实验和分析发现，基于偏好顺序的编码是最高效的编码[23]。这种编码在分批调度问题里面也有被使用[14]。当某台机器处于空闲状态，而且有多个待加工的工件，此时机器就按照编码中的偏好顺序，选择待加工工件中偏好度最前的工件[24][25]。基于偏好度的调度编码并不能直接得到工件加工的顺序，而是需要通过解码才能得到具体可行的调度顺序。另一种常用的基于工序的调度编码虽然能表示工件加工的顺序，似乎比基于偏好顺序的编码更直观和简单，但其实编码里的顺序也并不具体可行的调度顺序，它也是需要通过解码才能得到具体可行的调度顺序。在解码过程中，为了保证作业车间的各种约束，编码中许多基因都需要往后移，需要改变原有顺序（建议找一篇基于工序编码的解码文章）。因此换一种角度来看，它与基于偏好顺序的编码实质上都是表示优先顺序的编码，并不直接代表调度过程中的加工顺序。对于分批调度问题来说，使用基于工序的编码很容易出现不可行解，而使用基于优先顺序的编码可以避免不可行解[24]。根据上面的分析，不可行解的出现会限制算法的能力，会降低搜索的效率，因此本文使用基于偏好顺序的调度编码。如图所示，M1~M4分别代表4台机器，每一行代表一台机器的偏好顺序，例如对M1来说，加工的优先顺序是批次3优于批次2优于批次4优于批次1。

## 4.2.2 基于机器柔性指数的解码方式

双矩阵编码中的S1和S2配合使用能有效表达一个分批调度方案的信息，但是如果想得到每个子批的完工时间，还需要进行解码。对于本问题来说，解码是利用调度编码中的信息，根据确定的分批方案，为每个子批的每个工序指定合适的加工机器，确定每个子批的每个工序的加工其实时间和加工结束时间，即生成一个完整的调度方案，并由此计算目标函数值。解码方式有很多种，同一个编码可以使用不同的解码方式，得到不同的调度方案，从而得到不同的目标函数值[14]。因为在不同的规则下，工序的顺序可以不同，工序的等待时间会有差异，工序的加工机器选择也不一定相同。好的解码方式可以把编码解为更优的调度方案，从而提高寻优的精度。

对于作业车间调度来说，特别在使用基于偏好顺序的编码时，解码过程中通常会使用Giffler and Thompson启发式，其思想是，首先选择一台机器，优先选择最早能空闲的机器，然后对该机器可选的待加工工件进行选择，优先选择偏好度高的待加工工件[26][27][25]。但是对于柔性作业车间的分批调度问题来说，需要考虑更多因素，例如属于同一个批次的不同子批，它们的偏好度在同一台机器上是一样的，应该优先加工哪个批次？例如某个子批的某个工序有多个可选机器，其加工时间和准备操作时间各不相同，其空闲时刻也不相同，应该选择哪一台机器来加工？该工序完工时间的最小化是否与所有批次完工时间最小化是否存在矛盾？上述问题能否得到解决关系到解码效果好不好，关系到寻优的精度。为此，本小节提出一种基于机器柔性指数的解码方式，可以在保证问题约束的前提下，引入机器柔性指数，使用针对分批调度问题的规则进行调度。

在柔性作业车间问题里，每一个工件的每一个工序都可以从一台或以上的机器之中选择一台来加工，各台机器之间是有差异的。有些机器能加工的工序多一些，导致会有许多工件同时等待这种机器。而有些机器能加工的工序比较少，导致机器经常处于等待工件的空闲状态。对于机器的这一性质，本节引入柔性指数来定量描述。

sot规则 Comparative evaluation of genetic algorithms for job-shop scheduling

## 4.2.3 实验分析

# 4.3 搜索算子的设计

## 4.3.1 粗细粒度配合搜索

## 4.3.2 粗粒度交叉搜索算子

## 4.3.3 细粒度邻域搜索算子

4.3.3.1 单一邻域

4.3.3.2 启发式邻域

4.3.3.3 细粒度全邻域

4.3.3.4 邻域搜索策略

4.3.3.5 实验分析

五、基于竞争式协同候鸟迁移算法的优化调度

# 5.1 引言

根据3.3对MBO算法性能的分析，如果想提高算法的寻优能力，应该从以下两方面着手：

（1）缓解种群多样性的急剧减少

（2）提高群体协同的效率

本章基于这两个出发点，提出了竞争式协同候鸟迁移算法（Competitive Cooperative Migrating Bird Optimization, CCMBO）。CCMBO改进了基于“邻域共享”的利益机制，改进了异步更新机制，在一定程度上控制了多样性急剧减少的情况。然后提出了减速调整阶段和竞争阶段，改进了“轮替式”的领头鸟替换机制，丰富了鸟群协作的拓扑结构，提高个体之间协作的效率，提高了算法在复杂空间中的寻优能力。

# 5.2 竞争式协同候鸟迁移算法

## 5.2.1 竞争阶段

在竞争阶段，要根据竞争的规则确定领头鸟以及V字型队伍的排序。根据3.3.2.1的分析，基本MBO领头鸟的替换是轮替式，优秀个体在种群中难以发挥带领作用，影响寻优效率。而且鸟个体之间协作的拓扑结构比较固定，没有充分发挥个体间协作的潜力，不利于种群的高效寻优。

为了让优秀的个体在种群中充分发挥个体优势的影响作用，同时为了吩咐鸟群的协同拓扑结构，本章给MBO引入竞争阶段。鸟群在减速飞行调整阶段结束之后，鸟群的队形依然是散乱的，经过竞争阶段之后，鸟群可以按照实力优劣竞争领头鸟的位置，确定下一轮飞行的V字型队形，可以形成与上一轮飞行完全不一样的协同拓扑，丰富鸟个体之间互相协作的关系。

在竞争阶段的过程中，所有鸟凭借自身实力角逐领头鸟的位置，越优秀的鸟越有可能成为新的领头鸟。同时个体之间也会互相竞争队伍中间的位置，越优秀的鸟越有可能排在队伍的前面，越差的鸟越有可能排在队伍后面。经过这样的竞争之后，新形成的V字型队形具有一定程度的优劣梯度，这样可以让更优秀的个体有更多机会发挥引领作用，提高寻优的效率，而不是像基本MBO中轮替式的机会均等，优秀个体可能要等待好久才能成为领头鸟。

在大自然的复杂环境下，偶然因素不可避免，竞争阶段也会受到偶然随机因素的影响，具体表现为，最优秀的个体并不一定能成为下一轮飞行的领头鸟，最差的个体也不一定是排在队伍末尾的那个。竞争阶段的准则是，更优秀的个体排在前面的可能性更大，越优秀的个体成为新领头鸟的可能性越大，而不是按照严格适应度排序来构成队形。这种偶然因素的加入可以防止领头鸟的位置被某一个陷入局部最优的个体长期垄断，提高跳出局部最优的能力。即使不是最优秀的个体，只要足够优秀了，也有机会成为领头鸟，这样一来，除了最优秀的个体所在的区域，其他优秀个体所在的区域的潜力，在不同领头鸟的带领作用下，也能得到挖掘。这种加入偶然因素的排序被称为模糊排序。竞争阶段鸟群竞争形成新的V字队形具体过程如下：

step1：对所有鸟进行模糊排序，得到；

step2：成为新的领头鸟，；

step3：如果为偶数，那么加入左翼队列的末尾，即成为，否则加入右翼队列的末尾，即成为；

step4：如果，那么回到step3，否则新的V字队形构建完成。

## 5.2.2 改进的V字飞行阶段

根据3.3.1.1的分析，基本MBO里面基于“邻域共享”的利益机制会加剧种群多样性的损失，让算法更容易陷入局部最优。这种共享模式还有另一个不足的地方。跟随鸟如果被前鸟的优秀邻域个体替换，那么跟随鸟自身所包含的所有信息都会被丢弃。虽然跟随鸟不如前鸟的邻域个体优秀，但是它的编码里也许会包含有价值的代码片段，这些代码片段也被统统丢弃了。

为了缓解“邻域共享”所带来的多样性急剧减少，也为了适当保留跟随鸟自身的优秀代码片段，本章提出使用“代码片共享”以取代“邻域共享”。具体来说，让跟随鸟与前鸟通过交叉，用二者的代码片重新构成一个新的解。新的解里面带有前鸟的代码片，也带有跟随鸟的代码片，体现“代码片共享”。这个新的解处于前鸟的邻域范围内，同时也处于跟随鸟的邻域范围内，即两个鸟邻域范围的交集。这种方式既能够体现利前鸟对于跟随鸟的利益机制，又能避免跟随鸟的信息被完全丢弃，使种群保持一定的多样性，保持一定活力，同时还避免跟随鸟所在的邻域被完全废弃，保持对解域的全局搜索，防止早熟收敛。

在CCMBO的V字飞行阶段里，每一只跟随鸟首先与前鸟进行次代码片共享，即交叉。每次交叉得到两个子代，选择较优的子代作为代码片共享的结果。次代码片共享之后得到个新的解，放入候选集里面待跟随鸟选择。接下来，与基本MBO相同，领头鸟是把次邻域搜索得到的邻域解放入候选集，然后择优更新，跟随鸟做次邻域搜索，得到的个邻域解也放入候选集里面。此时候选集里就有了个供当前跟随鸟选择的解，如果这个解之中最优解优于当前跟随鸟，那么就用最优解来更新当前跟随鸟。一只跟随鸟更新的具体过程如下，以左翼队列中某一只跟随鸟为例：

step1：令当前鸟的候选集；

step2：令当前鸟与其前鸟交叉，得到两个子代，较优的子代放入候选集；

step3：如果候选集里解个体的数量少于，那么回到step2，否则进行step4；

step4：对当前鸟进行一次邻域搜索，得到一个邻域解，放入候选集；

step5：如果候选集里解个体的数量少于，那么回到setp4，否则进行step6；

step6：找出候选集 最优的解，如果比更优，那么用替换。

值得注意的是，对于种群更新来说，有两种更新方式，异步更新和同步更新。根据3.2.2.2的分析，异步更新会加剧种群多样性的损失，基本MBO属于异步更新。同步更新是指，所有跟随鸟，无论排在队伍中的第几个位置，都是在同一时刻，根据上一次迭代中它的前鸟来生成各自的候选集，然后再使用各自的候选集更新当前跟随鸟。在一次种群的同步更新过程中，前鸟的信息只会分享给紧跟其后的一只跟随鸟，不会对其后多只跟随鸟产生影响，不会加剧种群多样性损失。因此CCMO的种群更新采用同步更新的方式。

V字飞行阶段需要进行次的迭代更新，整个种群采用同步更新一代的具体过程如下：

step1：领头鸟生成候选集，择优更新；

step2：对左翼队列所有鸟， 使用它们各自的前鸟生成各自的候选集

step3：使用各自的候选集对更新；

step4：对右翼队列所有鸟，用它们各自的前鸟生成各自的候选集，

step5：使用各自的候选集对更新。

## 5.2.3 减速调整阶段

当鸟群在领头鸟的带领下加速或者匀速飞行时，鸟群呈现出标准的V字队形。但是当领头鸟更替过程中导致鸟群减速的时候，鸟群的V字队形会被打乱。当领头鸟感到疲劳的时候，它会减慢飞行速度，退到鸟群里面，不再飞在鸟群的最前方。跟随鸟也各自调整自己的速度，由于每只鸟速度不一致而无法保持原来的V字队形。根据这一个鸟类飞行现象，本章提出减速飞行调整阶段，来模拟V字队形被打乱后，鸟个体在散乱队形中自由试探自由飞行的行为。

在这个阶段里，鸟个体两两之间原本比较固定的跟随与被跟随的关系被打破，每只鸟都会不断试探，跟随不同的鸟，调整自己飞行的方向和速度。与V字飞行阶段类似，减速飞行调整阶段也需要几个循环才能完成，循环次数设为。在减速飞行调整阶段的每个循环里，每只鸟都会选择另一只鸟去跟随，同一只鸟在不同的循环里会跟随不同的鸟，以体现鸟个体在调整速度和试探方向的过程中，比较无序的行为特点和散乱的队形结构。在这个阶段，一只鸟跟随另一只鸟所代表的搜索与V字飞行阶段是一样的，都是进行信息共享。只不过减速飞行调整阶段中，跟随与被跟随的关系的动态变化的，在V字飞行阶段，跟随与被跟随的关系是固定不变的。减速调整阶段需要进行代的迭代更新，整个种群每次更新的过程如下：

step1：令；

step2：令鸟个体的候选集，为随机选择另一只鸟，与交叉得到两个子代，选择较好的子代放入候选集；

step3：如果候选集里面个体的数量小于，那么回到step2，否则进行step4；

step4：鸟个体进行一次邻域搜索，把邻域解放入候选集；

step5：如果候选集里面个体的数量小于，那么回到step4，否则进行step6；

step6：如果，那么，否则结束本次迭代。

## 5.2.4 算法流程及复杂度分析

CCMBO主要由四个阶段组成，分别是：

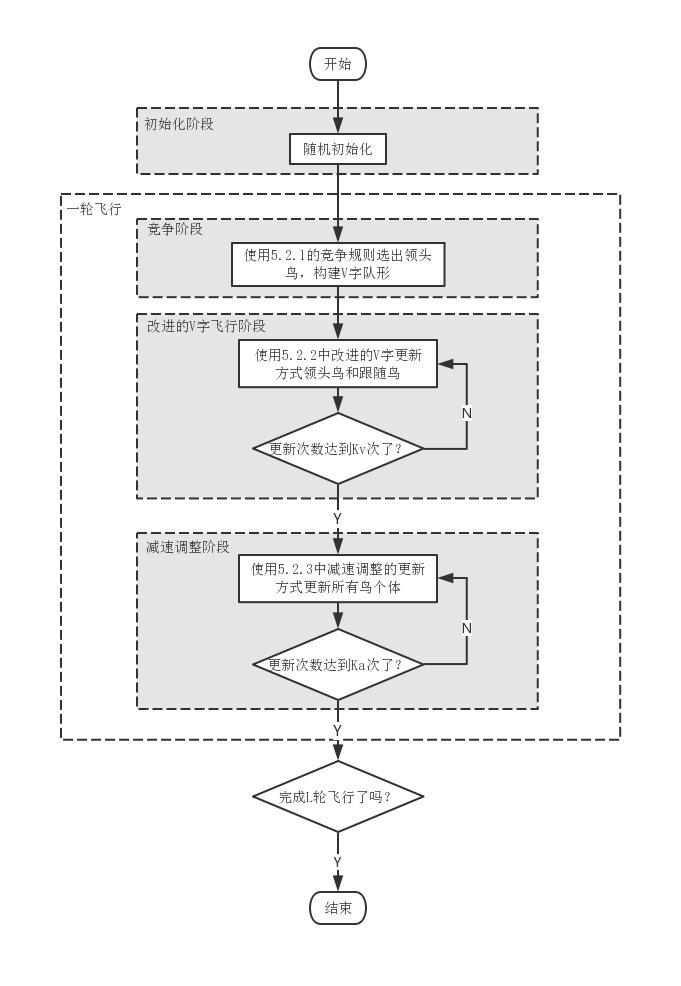
（1）初始化阶段。为了体现算法的鲁棒性，使用随机初始化，不使用加入启发式的初始化。

（2）竞争阶段。每只鸟使用5.2.3的方法选出领头鸟，并构成V字队形，准备下一轮飞行。

（3）改进的V字飞行阶段。保持V字队形不变，所有鸟使用5.2.1的更新方法更新代。

（4）减速调整阶段。打破原有的V字队形，所有鸟使用5.2.2的更新方法在散乱的鸟群中更新代。

算法流程图如下：



初始化阶段结束之后，种群进入一轮飞行。首先进入竞争阶段，按照5.2.1的竞争的规则选出领头鸟，确定V字队形的排列顺序，然后进入V字飞行阶段，使用5.2.2的更新方法对领头鸟和跟随鸟更新代，最后进入减速调整阶段，使用5.2.3的更新方法对每只鸟更新代。竞争阶段只是改变队形，并没有对个体进行更新，因此在一轮飞行中，整个种群一共进行次的更新。完成一轮飞行之后，如果还未达到搜索结束条件，就使用新的V字队形进行下一轮的飞行。

令鸟群大小为，染色体维度为，在最坏的情况下，CCMBO的时间复杂度为：

（1）初始化阶段，对个个体做随机初始化，时间复杂度是

（2）竞争阶段，个个体进行排序，选出领头鸟，构建新的V字型队伍，本阶段时间复杂度是；

（3）改进的V字飞行阶段，个个体通过交叉或者邻域搜索的方法各自生成个候选解，如此更新代，本阶段的时间复杂度是；

（4）减速调整阶段，个个体通过邻域搜索各自生成个候选解，如此更新代，本阶段的时间复杂度是。

综合以上的分析，和代表着优化问题的规模，、和是与和无关的常量，因此对任何规模的优化问题来说，CCMBO算法的时间复杂度为：



# 5.3 实例仿真以及性能评价

为了验证CCMBO算法的效果，本节首先把CCMBO应用在标准的FJSP分批调度问题上，对CCMBO各个改进的部分进行实验，与基本MBO进行对比以验证改进的效果，并且与使用该标准问题的其他算法进行对比。然后把CCMBO算法应用到免疫分析检测设备分批调度问题上，验证分批调度的效果。

本节使用Python3.6编写算法，在3.20GHz，16.0GB的计算机上进行实验。

## 5.3.1 测试例子

5.3.1.1 FJSP分批调度标准测试用例

此处使用【赵】 中的测试用例，一共四个测试用例，类型都属于混排，带有准备操作，准备操作可分离，以完工时间最小化作为目标。使用来描述测试用例的规模，其中代表批的数量，代表机器数，代表每个批的工件总数。一下是四个测试用例的规模：

P1：规模是，即4个批，6台机器，每一批都包含8个工件，这属于中小规模的分批调度问题。每类工件的每个工序在不同机器上的加工时间以及准备操作时间如表所示；

P2：规模是，批数与机器数跟P1相同，每一批都包含20个工件，这属于中小规模的分批调度问题。加工时间与准备操作时间与P1也相同，如表所示；

P3：规模是，即6个批，6台机器，每一批都包含10个工件，这属于中小规模的分批调度问题。每类工件的每个工序在不同机器上的加工时间以及准备操作时间如表所示；

P2：规模是，批数与机器数跟P3相同，每一批都包含20个工件，这属于大规模的分批调度问题。加工时间与准备操作时间与P3也相同，如表所示；

（不知道要不要把具体数据列在正文）

5.3.1.2 免疫分析检测设备分批调度问题

## 5.3.2 实验结果与分析

六、基于多领头鸟分化协同候鸟迁移算法的优化调度

# 6.1 引言

经过第五章的改进和实验，CCMBO算法的寻优性能已经比基本MBO的性能好了，但是寻优精度还有待提升，寻优的稳定性和鲁棒性也也稍显不足。

Nathan和Barbosa指出，在候鸟群的长途迁徙过程中，鸟群的队形有时也会形成除了V字型以外的编队，有时还会形成多个组群[28]。对于基于群体的进化算法来说，多种群协同进化是一种有效提升算法寻优能力的手段[29]。Defersha指出，多种群协同进化并不是算法的简单并行版本，而是一种与传统方式不同的方式来搜索解空间[9]。多种群协同允许不同特征的子种群形成，可以进化出不同特征的个体，有更好的多样性，同时子种群内部也能形成很好的局部搜索环境。

本章在CCMBO的基础上，对算法进行多种群协同的改进，提出多领头鸟分化协同候鸟迁移算法（Multi-Leader Competitive Cooperative Migrating Bird Optimization，ML-CCMBO）。首先设计了多领头鸟的分化协同机制，不同的领头鸟带领不同的子鸟群，不同的子鸟群分化出不同的进化方向，通过基于交换的子种群迁移来进行协同进化。其次设计了阶段性邻域搜索，对不同阶段的个体执行不同的邻域搜索策略，在停滞阶段，执行不同程度的退化，以激发个体的搜索能力。

# 6.2 多领头鸟分化协同候鸟迁移算法

## 6.2.1 多领头鸟分化协同机制

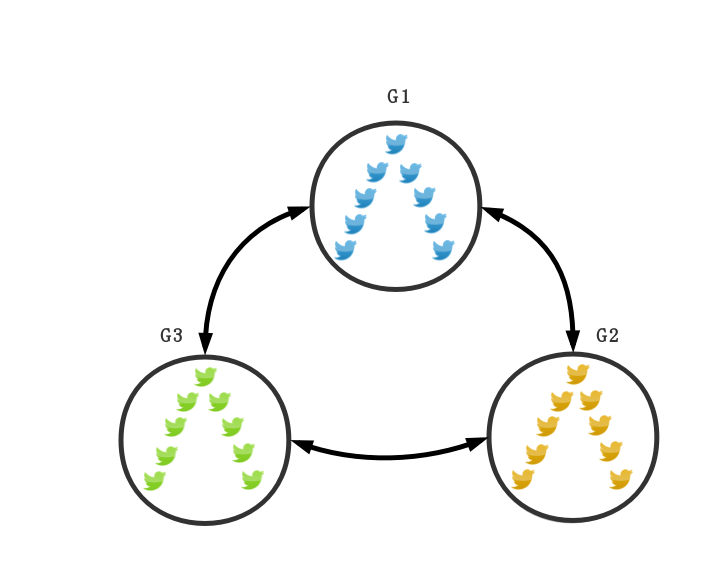
多种群协同不是简单地把一个大种群分为多个种群，而是通过子种群独立进化来维护多样性，通过子种群间的信息交流来提高效率，通过各种种群迁移拓扑来实现高效的进化模式。

Kurdi提出，可以对不同子种群使用不同的算子，以产生不同的子种群环境[30]。不同子种群中的个体根据所处环境进化，可以产生不同特征的个体。

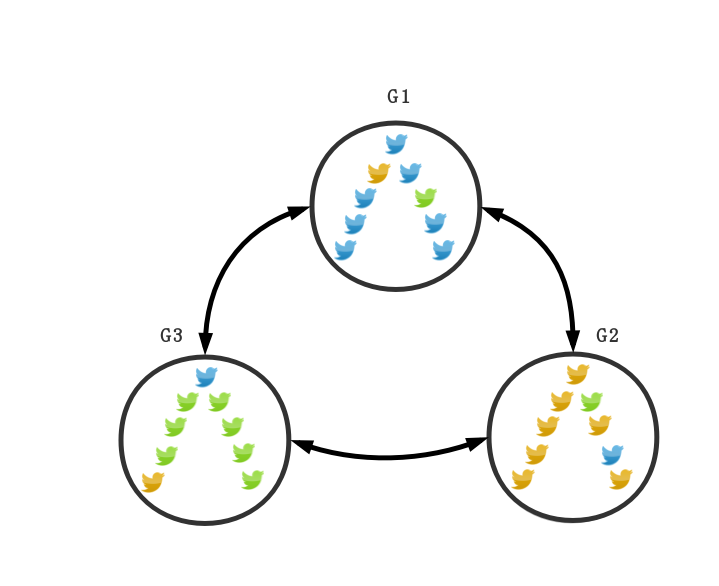
由于分批调度问题可以被分解为两个子问题，即分批子问题和调度子问题，因此本章打算让第一个子鸟群（G1）侧重解决分批子问题，朝着寻找最优分批方案的方向去进化，同时让第二个子鸟群（G2）侧重解决调度子问题，朝着寻找最优调度方案的方向去进化。具体来说，就是让G1使用针对分批矩阵（S1）的邻域搜索算子，让个体在分批搜索域上得到充分的搜索，然后让G2使用针对调度矩阵（S2）的邻域搜索算子，让个体在调度顺序搜索域上得到充分的搜索。

两个子问题并不是完全独立的，它们之间有时存在复杂耦合。例如，针对原来的分批方案有一个最优的调度方案，当分批方案发生变化，原来最优的调度方案对于新的分批方案来说可能已经不是最优的了。因此还需要把这两个子问题当成一个整体来优化。本章打算让第三个子鸟群（G3）同时解决两个子问题，同时朝着最优分批方案和最优调度方案去进化。具体来说，就是让G3使用同时针对S1和S2的邻域搜索算子，使每次邻域搜索都能在分批搜索域和调度搜索域上同时搜索。

三个子鸟群具有不同的功能，分化出不同的搜索环境和特点，适用于不同类型的个体，如下图所示。有些个体只需要改变分批方案，就能得到进化，有些个体只需要改变调度顺序，就能得到进化，而有些个体需要同时改变分批方案和调度顺序才能得到进化。特别是在尝试跳出局部最优的阶段，这三种不同的功能可以给不同的个体提供多样的尝试，帮助跳出局部最优。



三个功能分化的子鸟群不是孤立种群，它们通过一定的拓扑结构和种群迁移策略组织成一个有机的整体，从而实现子种群之间的协同。有了合适的拓扑结构和种群迁移，三个种群的分化功能才能得到最大的发挥。拓扑结构就是子种群连接的方式，ML-CCMBO使用的是环状的拓扑，三个子鸟群分布在一个环上，互相连接，每一个子鸟群都与另外两个子鸟群有连接。种群迁移策略是各个子种群进行信息交流的方式。当某个子种群陷入局部最优了，适当的种群迁移可以引入局部最优以外的个体，刺激子种群跳出局部最优。当某个个体在它的子种群内得不到进化，迁移到另外一个子种群可以给它一个新的进化环境，促进它继续进化。经典的多种群协同进化算法会在一个子种群中选择某些个体去替换另一个子种群的部分个体。如下图所示，种群迁移之后，子鸟群引入了不同特性的个体，提高了子种群内的多样性，促进子种群的进化，而迁移到其他子种群的个体则在新的环境里尝试不同方向的进化。考虑到直接替换部分个体会导致部分个体的消失和部分个体的重复出现，会加剧多样性的损失，因此ML-CCMBO不使用替换的方法，而是使用交换的方法。



具体来说，ML-CCMBO的子鸟群会选出最优的个个体，与另一个子鸟群的任意个个体交换。这样能保证每个个体都有迁移到其他子鸟群的机会，同时也能让最优个体停滞不前的时候有更多机会在不同环境的刺激下继续进化。ML-CCMBO使用同步的多种群进化模式，即每个子鸟群独立进化代之后就进行种群迁移。ML-CCMBO子鸟群迁移的步骤如下：

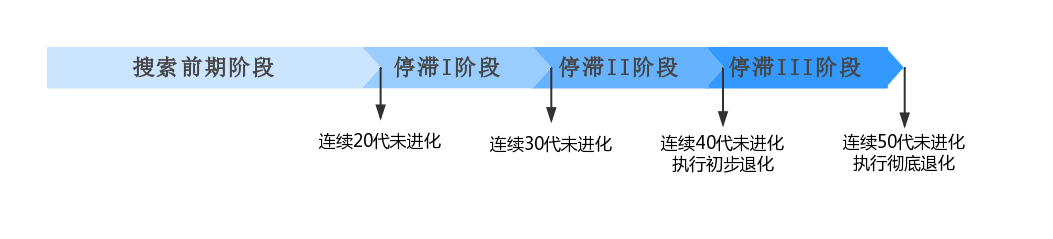
step1：分别选出三个子种群G1，G2，G3最优的个个体，分别是G1b，G2b，G3b；

step2：分别从三个子种群G1，G2，G3中未被选择的个体中随机选择个个体，分别是G1r，G2r，G3r；

step3：令G1b与G2r交换，令G2b与G3r交换，令G3b与G1r交换。

## 6.2.2 阶段性邻域搜索策略

CCMBO算法虽然使用了不同的邻域算子，但是只使用了一种邻域搜索策略。为了适应不同阶段个体的特点，ML-CCMBO使用了4.3.3.4中提到的其他搜索策略，提出了阶段性邻域搜索，可以提高算法在复杂搜索域里面的搜索能力。在个体进行邻域搜索之前，要判断个体处于什么阶段，针对不同阶段的个体使用不同强度的邻域搜索策略。在停滞阶段后期，执行不同程度的退化，帮助个体跳出局部最优。各个阶段的划分如下图所示。



当个体20代以内无进化时，认为这是搜索前期阶段，搜索策略与CCMBO相同，都使用1greedy-1step的简单策略，有活力的个体一般在20代以内就能得到再一次的进化，因此处于这个阶段的不需要特殊处理。

如果个体连续20代以上没有进化，则进入“停滞I”阶段，被视为有停滞的倾向，需要加强邻域搜索的强度。此时使用4.3.3.4中的2greedy-1step邻域搜索策略，即每次做两次邻域搜索，按照贪心的法则选取较优者作为邻域搜索的结果。

当个体连续30代以上没有进化，则进入“停滞II”阶段，此时处于更高的停滞水平，此阶段的个体更难得到进化。除了继续保持搜索的强度以外，还需提高邻域搜索的深度。此时使用4.3.3.4中的2greedy-2step邻域搜索策略，尝试使用更大的步幅做更深入的搜索，促进停滞的个体继续进化。

当个体连续40代以上没有进化，则进入“停滞III”阶段。到达此阶段的个体在经历了“停滞I”和“停滞II”之后还未能进化，已经到达很严重的停滞了，个体似乎遇到了进化的天花板。因此可以在第40代无进化的时候，尝试让个体退几步，进行初步的退化。即退化到某个比当前差的状态，以尝试跳出局部最优。在本阶段同样使用2greedy-1step邻域搜索策略，保持搜索的强度。大多数个体在本阶段都能得到进化了。

当个体在第50代都无进化时，可以认为该个体彻底停滞，只有少数个体能达到这样的状态。这样的个体并不具备进化的能力，因此可以进行彻底的退化。即重新初始化该个体，彻底逃离局部最优的位置，重新开始搜索，回到搜索前期阶段。

## 6.2.3 算法流程及复杂度分析

ML-CCMBO算法的每个子鸟群会并行运行具有不同邻域针对性的CCMBO算法，同时对每个个体都执行阶段性的邻域搜索策略，三个子鸟群朝着不同的方向进化，经过竞争阶段、改进的V字飞行阶段、减速调整阶段，完成一轮飞行之后，就进行一次种群迁移，迁移结束后进行下一轮的飞行。算法的流程图如下所示：



下面分析ML-CCMBO算法的时间复杂度。令字鸟群大小为，染色体维度为，在最坏的情况下，ML-CCMBO的单个子鸟群寻优的时间复杂度为：

（1）初始化阶段，对个个体做随机初始化，时间复杂度是

（2）竞争阶段，个个体进行排序，选出领头鸟，构建新的V字型队伍，本阶段时间复杂度是；

（3）使用阶段性邻域搜索策略后，改进的V字飞行阶段，在最坏情况下，使用最复杂的邻域策略，个个体通过交叉或者邻域搜索的方法各自生成个候选解，如此更新代，本阶段的时间复杂度是；

（4）使用阶段性邻域搜索策略后，减速调整阶段，在最坏情况下，使用最复杂的邻域策略，个个体通过邻域搜索各自生成个候选解，如此更新代，本阶段的时间复杂度是。

（5）种群迁移阶段，需要选出个个体中最优的个体去迁移，排序所需要的时间复杂度是。

综合以上的分析，和代表着优化问题的规模，、和是与和无关的常量，因此对任何规模的优化问题来说，ML-CCMBO的三个子鸟群寻优的时间复杂度为：



# 6.3 实例仿真以及性能评价

七、基于多微鸟群协同候鸟迁移算法的多目标分批优化调度

参考文献

[1] REITER S. A system for managing job-shop production[J]. The Journal of Business, 1966, 39(3): 371–393.

[2] CHEN J, STEINER G. On discrete lot streaming in no-wait flow shops[J]. IIE Transactions, 2003, 35(2): 91–101.

[3] KALIR A A, SARIN S C. Optimal Solutions for the Single Batch, Flow Shop, Lot-streaming Problem with Equal Sublots[J]. Decision Sciences, 2007, 32(2): 387–398.

[4] LIU J. Single-job lot streaming in m-1 two-stage hybrid flowshops[J]. European Journal of Operational Research, 2008, 187(3): 1171–1183.

[5] KIM Y D, SHIM S O, KIM S B等. Parallel machine scheduling considering a job-splitting property[J]. International Journal of Production Research, 2004, 42(21): 4531–4546.

[6] NAIT TAHAR D, YALAOUI F, CHU C等. A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times[J]. International Journal of Production Economics, 2006, 99(1–2): 63–73.

[7] CHENG M, MUKHERJEE N J, SARIN S C. A review of lot streaming[J]. International Journal of Production Research, Taylor & Francis, 2013, 51(23–24): 7023–7046.

[8] LEI D, GUO X. Scheduling job shop with lot streaming and transportation through a modified artificial bee colony[J]. International Journal of Production Research, 2013, 51(16): 4930–4941.

[9] DEFERSHA F M, CHEN M. A coarse-grain parallel genetic algorithm for flexible job-shop scheduling with lot streaming[J]. Proceedings - 12th IEEE International Conference on Computational Science and Engineering, CSE 2009, 2009, 1: 201–208.

[10] DEFERSHA F M, BAYAT MOVAHED S. Linear programming assisted (not embedded) genetic algorithm for flexible jobshop scheduling with lot streaming[J]. Computers and Industrial Engineering, Elsevier, 2018, 117: 319–335.

[11] HUANG R H, YU T H. An effective ant colony optimization algorithm for multi-objective job-shop scheduling with equal-size lot-splitting[J]. Applied Soft Computing, Elsevier B.V., 2017, 57: 642–656.

[12] LOW C, HSU C M, HUANG K I. Benefits of lot splitting in job-shop scheduling[J]. International Journal of Advanced Manufacturing Technology, 2004, 24(9–10): 773–780.

[13] 白俊杰，龚毅光，王宁生，唐敦兵. 多目标柔性作业车间分批优化调度[J]. 计算机集成制造系统, 2010, 16(2): 0–7.

[14] DEMIR Y, IŞLEYEN S K. An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations[J]. International Journal of Production Research, 2014, 52(13): 3905–3921.

[15] ZHIJUN S, JIN A. 作业车间多工艺路线批量作业计划优化[J]. 中国机械工程, 2006, 19(2): 183–187.

[16] HAN Y, LI J, SANG H等. An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem[J]. Swarm and Evolutionary Computation, Elsevier B.V., 2018, 38: 64–78.

[17] 王万良，范丽霞，徐新丽，赵燕伟，张静. 多目标差分进化算法求解柔性作业车间批量调度问题[J]. 2013, 19(10): 2481–2492.

[18] YANG X, FAN L, LI L等. Hybrid Discrete Differential Evolution Algorithm for Lot Splitting with Capacity Constraints in Flexible Job Scheduling[J]. Mathematical Problems in Engineering, 2013: 1–10.

[19] 王云，冯毅雄，谭建荣，高一聪. 柔性作业车间分批调度多目标优化方法[J]. 浙江大学学报（工学版）, 2011, 45(4): 719–764.

[20] DEFERSHA F M, CHEN M. Jobshop lot streaming with routing flexibility, sequence-dependent setups, machine release dates and lag time[J]. International Journal of Production Research, 2012, 50(8): 2331–2352.

[21] WONG T C, NGAN S C. A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop[J]. Applied Soft Computing Journal, Elsevier B.V., 2013, 13(3): 1391–1399.

[22] DAVIS L. Job Shop Scheduling with Genetic Algorithms[C]//international conference on genetic algorithms. 1985: 136–140.

[23] PONNAMBALAM S G, ARAVINDAN P, SREENIVASA RAO P. Comparative evaluation of genetic algorithms for job-shop scheduling[J]. Production Planning and Control, 2001, 12(6): 560–574.

[24] ROCE F, TADEI R C, VOLTA G. A genetic algorithm for the job shop problem[J]. Computers & Operations Research, 1995, 22(1): 15–24.

[25] SHA D Y, HSU C-Y. A hybrid particle swarm optimization for job shop scheduling problem[J]. Computers & Industrial Engineering, 2006, 51(4): 791–808.

[26] B GIFFLER G L T. Algorithms for solving production-scheduling problems[J]. Operations Research, 1959, 8(4): 487–503.

[27] LEI D. A Pareto archive particle swarm optimization for multi-objective job shop scheduling[J]. Computers and Industrial Engineering, 2008, 54(4): 960–971.

[28] NATHAN A, BARBOSA V C. V-like formations in flocks of artificial birds[J]. Artificial Life, 2008, 14(2): 179–188.

[29] ASADZADEH L. A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy[J]. Computers and Industrial Engineering, Elsevier Ltd, 2016, 102: 359–367.

[30] KURDI M. An effective new island model genetic algorithm for job shop scheduling problem[J]. Computers & Operations Research, Elsevier, 2016, 67: 132–142.