基于遗传杂交和 PSO 的置换流水车间调度算法

王圣富

- 北京理工大学计算机学院 2016 级
- 学号：1120161848
- 班号：07111605

摘　要

本文分析了粒子群优化算法 PSO 的进化式，针对其容易发生早熟、收敛速度慢、后期搜索性能和个体寻优能力降低等缺点，结合遗传算法的思想，提出一种混合 PSO 算法。该算法是在 PSO 算法的更新过程中，对粒子速度引入遗传算法的变异操作，对粒子位置引入遗传算法交叉操作。对速度的变异降低了算法后期因种群过于密集而陷入局部最优的可能，对位置的交叉使得父代中优良个体的基因能够更好地遗传给下一代，从而得到更优、更多样化的后代，加快进化过程，提高了收敛速度和群体搜索性能。最后比较了标准 PSO 算法和本文提出的混合 PSO 算法，证明了混合 PSO 有优于标准 PSO 的时间复杂度和稳定性。

关键词

流水车间；
粒子群优化；
遗传杂交；
甘特图

§1 引言

粒子群优化算法是一种基于种群搜索的自适应进化计算技术。PSO 算法实现容易、参数较少，能有效解决复杂优化任务，在过去几年中得到了飞速发展，并在图像处理、模式识别、优化等领域得到了广泛应用。

PSO 算法作为一种崭新的随机搜索算法存在着易陷入局部极值点、进化后期收敛速度慢、

精度较差等不足。针对这些问题，本文对 PSO 算法进行了改进，文献[1]给出了一种自适应逃逸粒子群算法，算法中的逃逸行为是一种简化的速度变异操作，通过速度的自适应变化改变微粒在搜索空间内的飞行速度，使得 PSO 算法收敛速度得到提高。但是该变异操作仅仅增加了个体微粒的搜索性能，在算法迭代后期没有充分利用种群间的相互信息，在一定条件下仍然存在收敛速度较慢、不能快速逃出局部极小点的缺点。

本文结合遗传算法的思想和文献[1]中的逃逸思想，提出混合 PSO 算法——遗传 PSO。该算法是在速度变异的同时，对粒子位置执行一种交叉操作，通过交叉将上一代种群优良个体的基因遗传给下一代，使得一次进化得到的新种群趋于更优、更多样化状态。新算法充分显示了新算法的快速收敛性和全局收敛性。

§2 模型假设

2.1 背景

1. 所有工件均在 0 时刻释放且在各机器上的加工顺序相同，每个工件在每台机器上只加工一次。
2. 每个机器某一个时刻最多只能加工一个工件，而且执行过程是非抢占的。

2.2 符号

$v_{id}{}^{k}$：　名字为id的个体在第k次迭代中的速度；

c：　加速常数；

r：　随机变量；

$pbest$：　个体最优；

$gbest$：　全局最优；

w_i：　惯性系数；

ΔE：　评价值增量（温度差）；

$P(e)$：　接受函数；

$T(t)$：　冷却进度；

§3 算法设计

3.1 基本 PSO

PSO 初始化为一群随机粒子，然后通过迭代找到最优解。在每一次迭代中。粒子通过跟踪两个极值来更新自己：第一个就是粒子本身所找到的最优解，这个解叫做个体极值$pbest$；另一个极值是整个种群目前找到的最优解，这个极值是全局极值$gbest$。在找到上述两个最优值时，粒子根据如下公式来更新自己的速度和新的位置：

$$v_{id}{}^{k+1} = v_{id}{}^{k} + c_1 r_1 \left(pbest_{id}{}^{k} - x_{id}{}^{k}\right) + c_2 r_2 \left(gbest_{id}{}^{k} - x_{id}{}^{k}\right)$$

$$x_{id}{}^{k+1} = x_{id}{}^{k} + v_{id}{}^{k+1}$$

其中，c_1和c_2为加速常数，通常在 0~4 之间，一般取$c_1 = c_2 = 2$。r_1和r_2为两个在[0,1]内服从均匀分布的随机变量。每个粒子的位置和速度都以随机方式进行初始化，而后粒子的速度就朝着全局最优和个体最优的方向靠近。

3.2 标准 PSO（SPSO）

为了使粒子保持运动惯性，使其有扩展搜索空间的趋势，有能力探索新的区域。Shi 等人在文献[4]中提出了对基本粒子群算法的改进，即对速度更新方程加惯性权重w_i：

$$v_{id}{}^{k+1} = w_i v_{id}{}^{k} + c_1 r_1 \left(pbest_{id}{}^{k} - x_{id}{}^{k}\right) + c_2 r_2 \left(gbest_{id}{}^{k} - x_{id}{}^{k}\right)$$

粒子群到达局部最优附近时，粒子速度的更新主要由第一项来决定。由于固定参数的 PSO 算法的惯性权重加通常小于 1，粒子的速度将会越来越小，甚至停止运动，发生早熟收敛。对此，很多学者研究了自适应改变惯性权重，如线性递减惯性权重。

3.3 流水车间的数学模型

Flow-Shop 调度研究 m 台机器上 n 个工件的流水加工过程，每个零件在各机器上加工顺序相同，同时约定每个工件在每台机器上只加工一次，每台机器一次在某一时刻只能加工一个工件，各工件在各机器上所需的加工时间和准备时间已知，要求得到某调度方案时的某项指标最优。进一步，若约定每台机器加工的各工件顺序相同，则称其为置换 Flow-Shop 问题。

置换 Flow-Shop 调度问题的数学描述如下：n 个工件、m 台机器的流水车间调度问题的完工时间可表示为$c(j_1,1) = t_{j_1 1}$，。；令$c(j_i,k)$表示工件j_i在机器k上的加工完成时间，$\{j_1, j_2, j_3, \ldots, j_n\}$表示工件的调度，那么，

$$c(j_1,k)=c(j_1,k-1)+t_{j_1k},k=2,\dots,m$$

$$c(j_i,1)=c(j_{i-1},1)+t_{j_i1},i=2,\dots,m$$

$$c(j_i,k)=\max\{c(j_{i-1},k),c(j_i,k-1)\}+t_{j_ik},k=2,\dots,m,i=2,\dots,m$$

那么最大流程时间为$c_{max}=c(j_n,m)$。调度目标就是确定$\{j_1,j_2,j_3,\dots,j_n\}$，使得$c_{max}$最小。

3.4 流水车间的 PSO 算法

为了将微粒群算法运用到 Flow-Shop 调度问题等组合优化问题上，需要对微粒的位置和速度做出重新定义。设一共有 n 个工件，不妨设某个位置$X=(x_1,x_2,x_3,\dots,x_n)$，该位置的速度为$V=(v_1,v_2,v_3,\dots,v_n)$。通俗地说，每一个微粒的位置就是该批工件的加工顺序，而速度就是对加工顺序的修正数。$P=(p_1,p_2,p_3,\dots,p_n)$，$P_i$为微粒 i 所经历的最好位置，也就是微粒 i 所经历的具有最好适应值的位置，成为个体最好位置。P_g是指微粒群中所有微粒所经历过的最好位置，成为全局最好位置。

值得说明的是，$c_i=(c_1,c_2,c_3,\dots,c_n)$为常量，$r_i=(r_1,r_2,r_3,\dots,r_n)$为由随机函数产生的，定义$c_ir_i=(c_1r_1,c_2r_2,c_3r_3,\dots,c_nr_n)$

3.5 混合 PSO 算法

文献[1]结合生物界中物种发现生存密度过大时会自动分家迁移的习性，给出了一种自适应逃逸微粒群算法。本文在文献[1]算法思想的基础上，结合生物进化特征，给出一种不仅可以提高收敛速度，而且可增加种群多样性，提高种群进化质量的混合 PSO 算法。

该算法中速度变异操作进行改进的同时引入对位置的一种交叉操作。该交叉操作可使 PSO 算法更好地摆脱局部极值点，提高算法的收敛速度和全局收敛性。该交叉操作的思想如下：

对一次更新后的种群，从中随机挑出 m 个粒子，将这 m 个粒子的当前位置X_i与其当前个体极值p_i及按适应度值排序后的个体极值中的前 m 个极值(sp_i)进行交叉，得到 m 个新的粒子位置X_i'。如果新位置的适应值$f(X_i')$优于排序后对应个体极值的历史最优适应值$f(sp_i)$，则用$f(X_i')$取代$f(X_i)$，同时用X_i'取代X_i。显然，这样的交叉使得粒子在一次进化中不但利用了自己的历史经验信息，而且利用了种群中优良个体的经验信息，增加了粒子的多样性，更增加了种群的进化质量，使粒子找到全局最优的可能性增大。

迭代后期，当某些粒子的位置x_i及其$pbest$接近群体的$gbest$时，其速度更新由wv决定。由于$w < 1$，粒子的运行速度将迅速趋于 0，粒子运行出现惰性。随着迭代的进行，其他粒子将很快聚集到这些惰性粒子的周围，使粒子过早地收敛于$gbest$而停止运动(粒子速度变为 0)，而$gbest$只是种群目前找到的最好点，并不能保证一定是优化问题的全局最优解。

从以上分析可见，要让算法收敛到全局最优，就要使惰性粒子逃离局部最优点，而粒子接近$gbest$的程度与其速度大小有关，即可通过干扰惰性粒子速度使其跳出局部最优。要使优化算法摆脱局部最优，本文对惰性粒子，即速度小于某个阈值的粒子速度进行变异，降低算法陷入局部最优的可能性。

3.6 设计

首先我们初始化种群，经过测验，我们选择 2500 作为进化迭代次数，150 作为种群个体数目。

```
numOfIterations=2500;%进化次数
numOfIndivisual=150;%个体数目
individual=zeros(numOfIndivisual,numOfJobs);%种群信息

for i=1:numOfIndivisual
    individual(i,:)=randperm(numOfJobs);%使用随机置换函数生成随机序列
end
```

接着计算种群适应度，这是更新操作的主要部分。

```
fitOfIndivisual=fitness(individual,TimeOfJobs);%计算种群适应度
[value,index]=min(fitOfIndivisual);%记录最优下标
flowPbest=individual;%当前个体最优
flowGbest=individual(index,:);%当前全局最优
recordPbest=inf*ones(1,numOfIndivisual);%个体最优记录
recordGbest=fitOfIndivisual(index);%群体最优记录
newIndivisual=individual;%创建种群信息的备份，以便更新
```

利用交叉操作，删除与交叉区域相同元素。同时，进行与个体最优进行交叉和与全体最优进行交叉操作。

```
c1=round(rand*(numOfJobs-2))+1;  %产生交叉位
c2=round(rand*(numOfJobs-2))+1;  %产生交叉位
while c1==c2
    c1=round(rand*(numOfJobs-2))+1;
    c2=round(rand*(numOfJobs-2))+1;
end
chb1=min(c1,c2);
chb2=max(c1,c2);
cros=flowGbest/ flowPbest(chb1:chb2);
ncros=size(cros,2);
```

利用变异操作，对惰性粒子进行变异处理。同时，进行更新操作。

```
c1=round(rand*(numOfJobs-1))+1;   %产生变异位
c2=round(rand*(numOfJobs-1))+1;   %产生变异位
while c1==c2
    c1=round(rand*(numOfJobs-2))+1;
    c2=round(rand*(numOfJobs-2))+1;
end
swap(newIndivisual(i,c1),newIndivisual)
```

绘制甘特图也是本文程序重要的一部分。我们绘画的基本元素为矩形，为了方便区分，需要给不同任务的矩形做不同的颜色处理，并加上任务序号标记。在用于计算个体适应度值的 fitness 函数中，就需要利用流水车间的任务完成时间的递推关系得出当前的适应值。

§4 编程实验

4.1 前期设置

混合 PSO 算法虽然能达到很好的实验效果，但需要大量矩阵运算和更新操作，C++作为面向对象操作语言不能很好地实现该算法，所以本文选择 Matlab R2018a 作为编程环境。

PSO 作为经典的元启发算法，对于不同的种群个体数和迭代次数有着不同的复杂度。种群个体数越高，则搜索到全局最优解的可能性也越大，但计算复杂度也显著增大。在本次实验中，我们利用 Matlab 高效的矩阵运算和内存处理使得时间控制在 20 秒以内。

硬件方面，本人使用的电脑型号为 Lenovo Yoga 710-14，Windows10 2018 春季创意者版，超极本类别，6 代中央处理器。

4.2 结果

首先我们运行编写好的程序，得到如下结果。其中，中文数字表示工件排在调度序列的第几位。

表 1：混合 PSO 算法的结果

样例	01	02	03	04	05
解	7038	7166	7312	8003	7720
一	8	7	11	4	5
二	1	3	12	12	2
三	5	4	6	13	4
四	3	11	5	14	1
五	4	8	10	10	3
六	11	13	9	9	8
七	9	2	3	7	6
八	7	12	2	6	10
九	2	5	4	11	9
十	10	1	7	8	7
十一	6	9	8	1	
十二		6	1	2	
十三		10		3	
十四				5	

根据已经得到的针对五个样例的流水车间调度，我们可以制作如下的甘特图。其中，横坐标代表工件的加工时间，纵坐标从上到下依次是 M1,M2,M3,M4,…,Mm 的机器序列，图中应为反向。

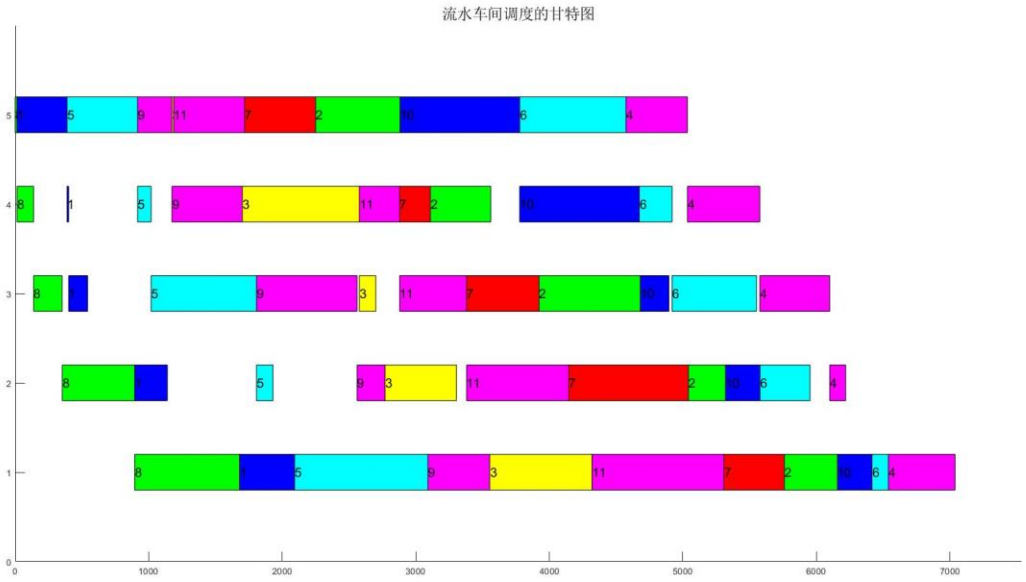

图 1：用混合 PSO 算法解样例一

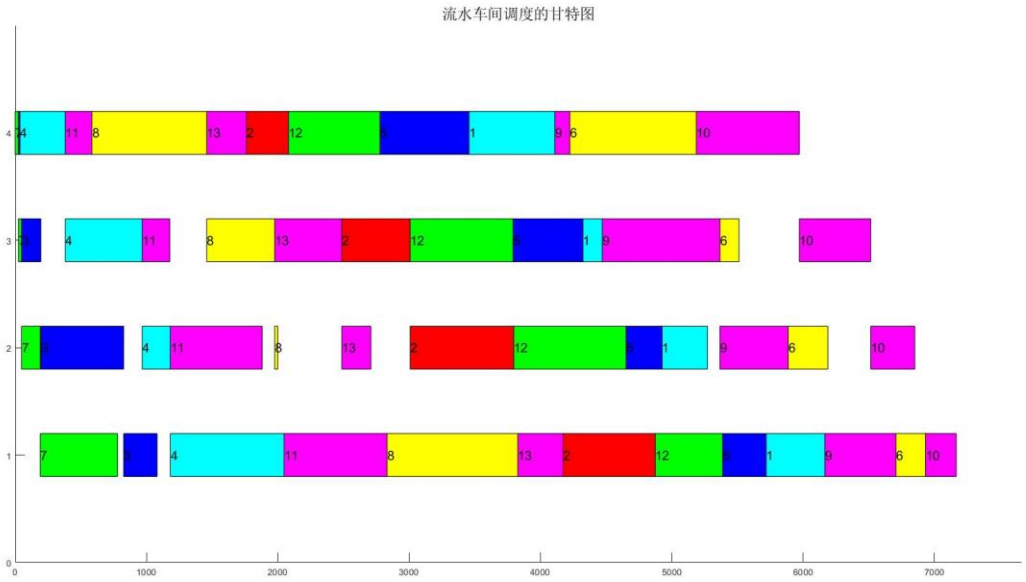

图 2：用混合 PSO 算法解样例二

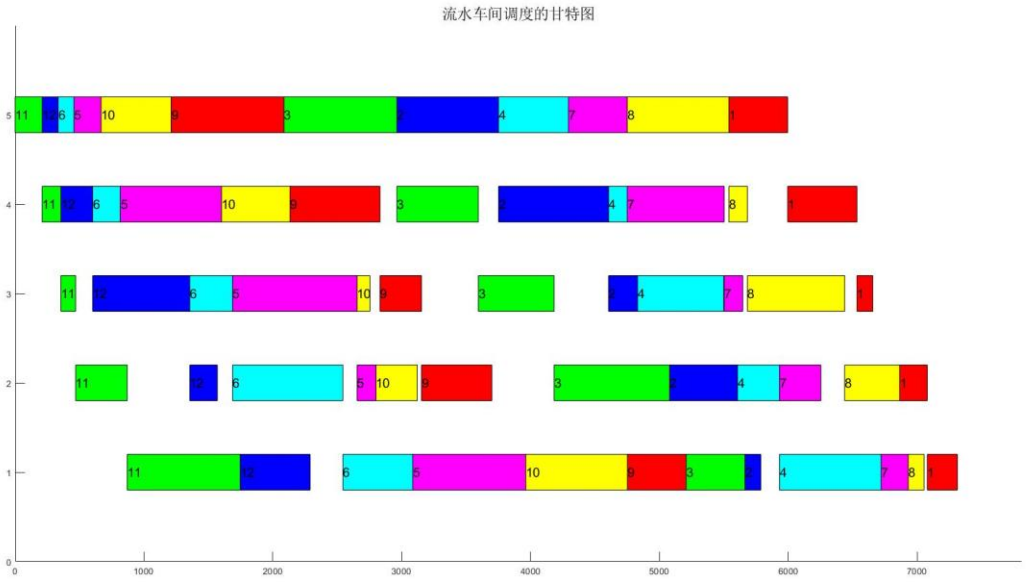

图 3：用混合 PSO 算法解样例三

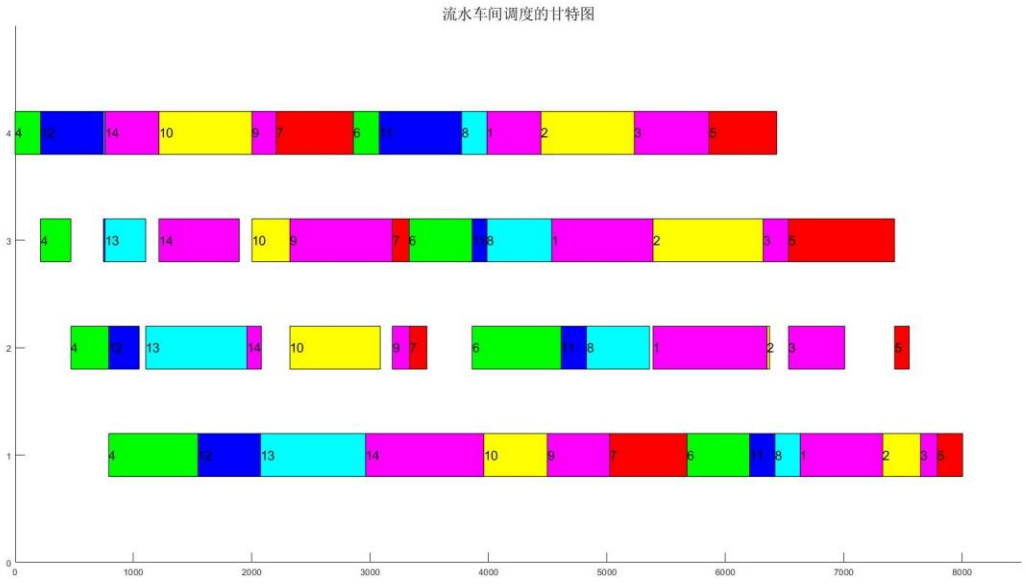

图 4：用混合 PSO 算法解样例四

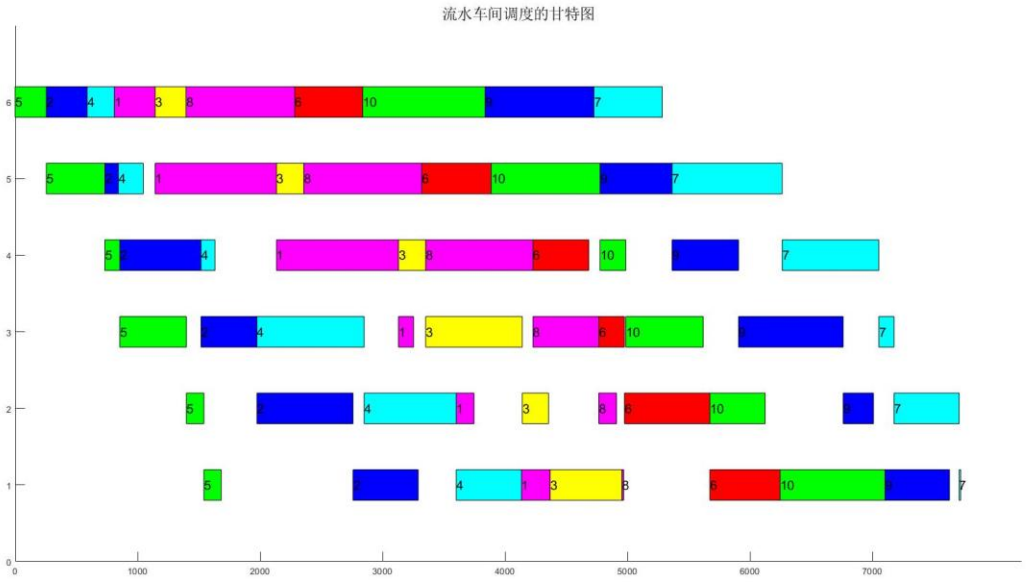

图 5：用混合 PSO 算法解样例五

为了充分了解模拟退火算法在迭代退火的训练过程当中，目标函数总延迟的变化情况，我们需要单机调度的训练过程图。每一个样例的模拟退火训练过程如下：

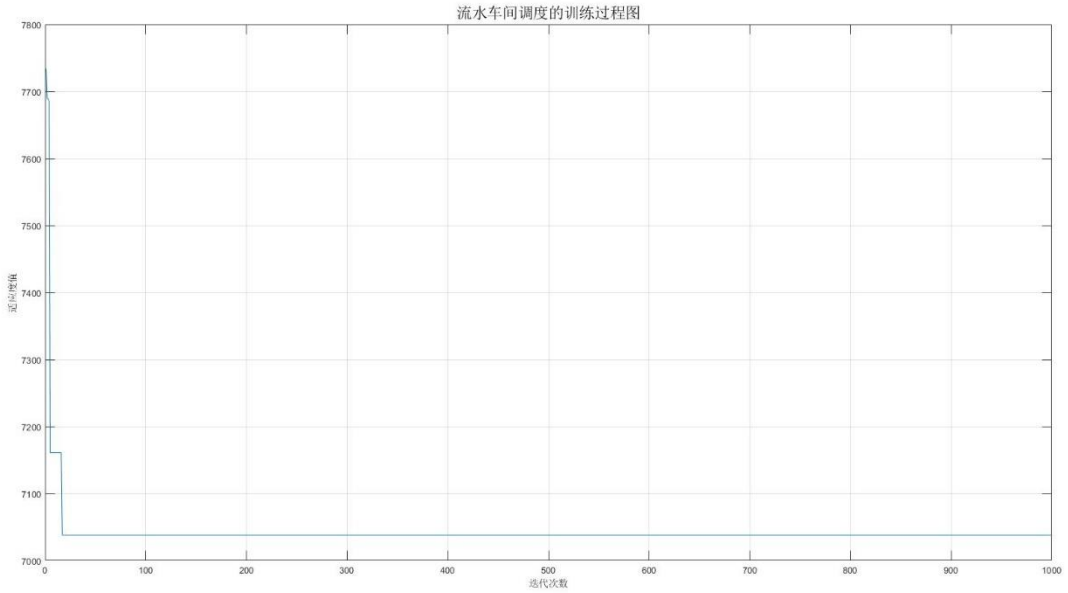

图 6：混合 PSO 算法下样例一的适应度变化

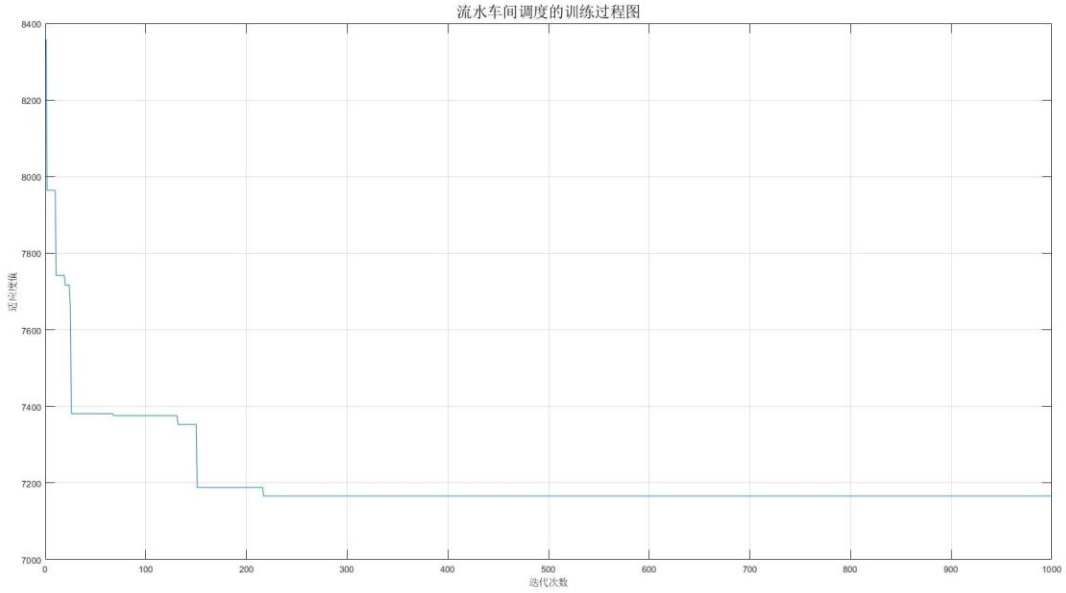

图 7：混合 PSO 算法下样例二的适应度变化

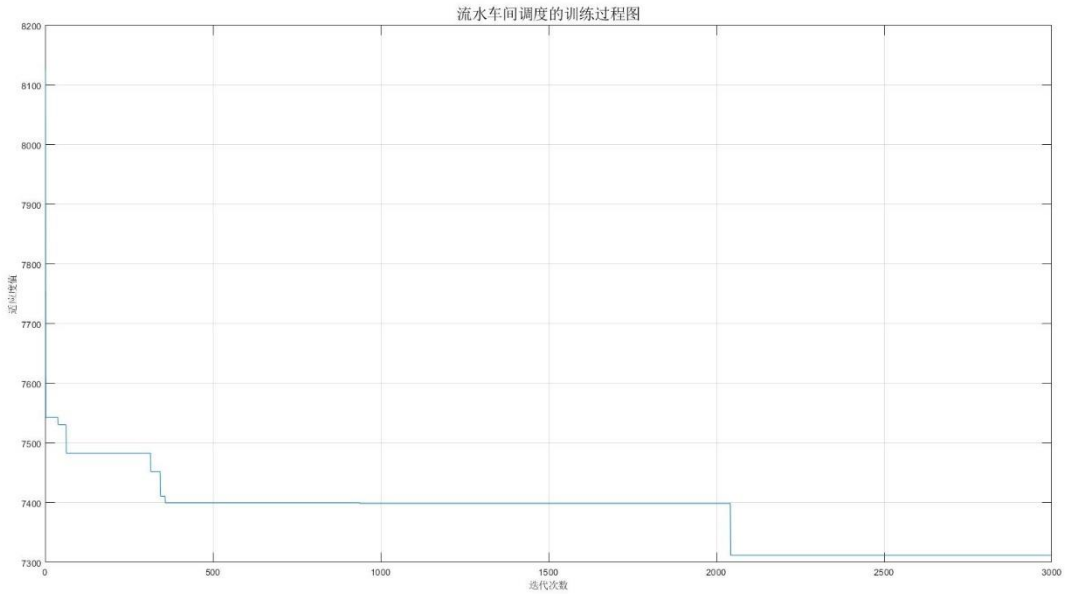

图 8：混合 PSO 算法下样例三的适应度变化

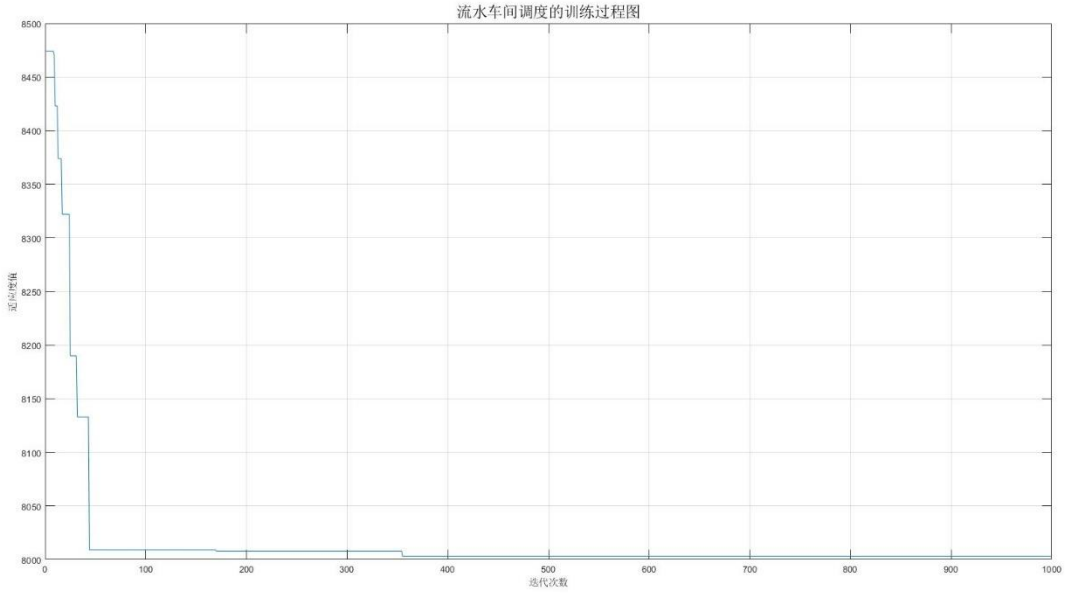

图 9：混合 PSO 算法下样例四的适应度变化

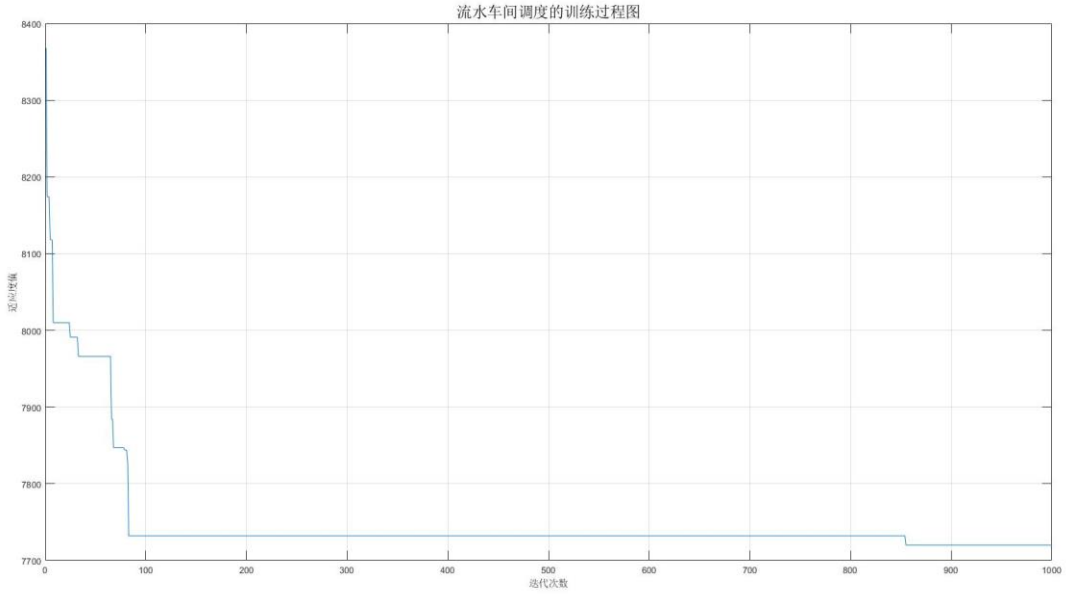

图 10：混合 PSO 算法下样例五的适应度变化

从图中可以看出，混合 PSO 算法会随着迭代次数的增加，适应度变化趋势是越来越小，并逐渐稳定在某一阈值内。当增大种群规模时，搜索的速度将更快。因此,混合 PSO 算法在 Flow-Shop 调度问题中的运用取得了较好的效果，表明了混合 PSO 算法应用于流水车间调度中的可行性。

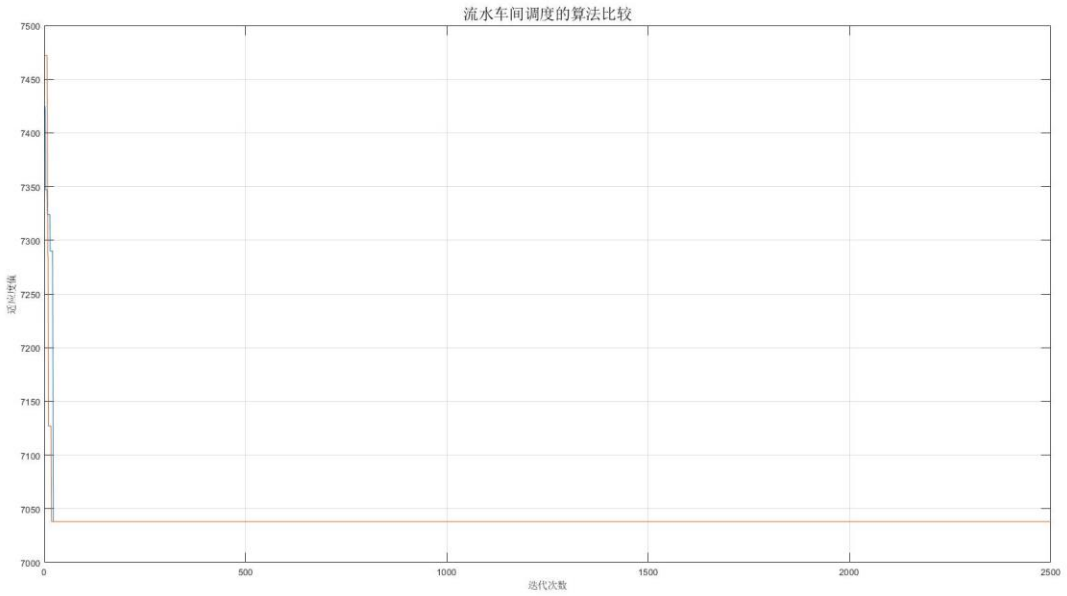

图 11：样例一中标准 PSO 算法（蓝色）和混合 PSO 算法（橙色）的适应度趋势比较

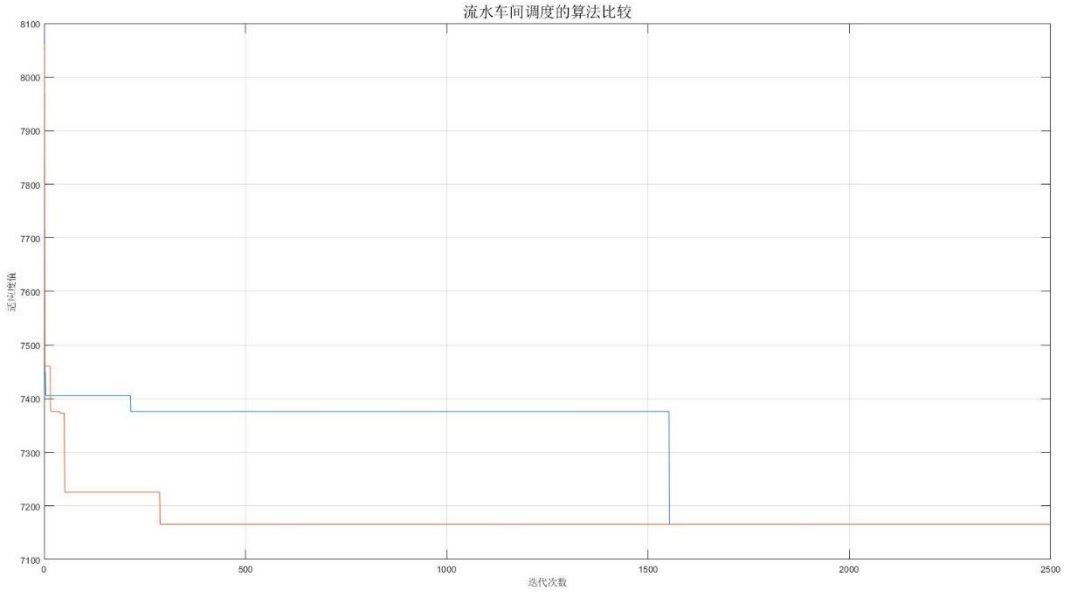

图 12：样例二中标准 PSO 算法（蓝色）和混合 PSO 算法（橙色）的适应度趋势比较

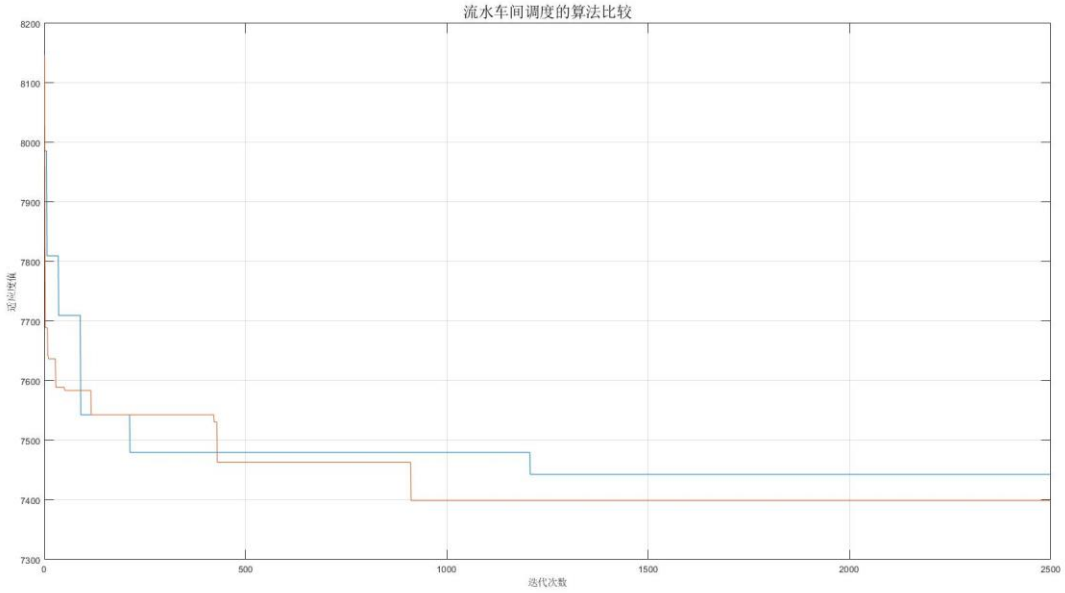

图 13：样例三中标准 PSO 算法（蓝色）和混合 PSO 算法（橙色）的适应度趋势比较

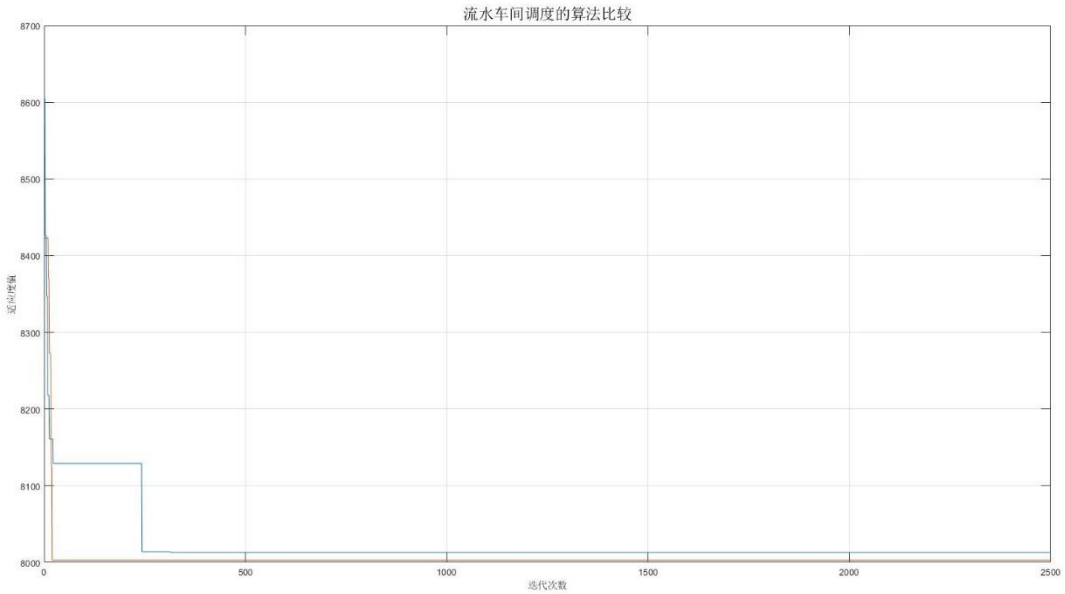

图 14：样例四中标准 PSO 算法（蓝色）和混合 PSO 算法（橙色）的适应度趋势比较

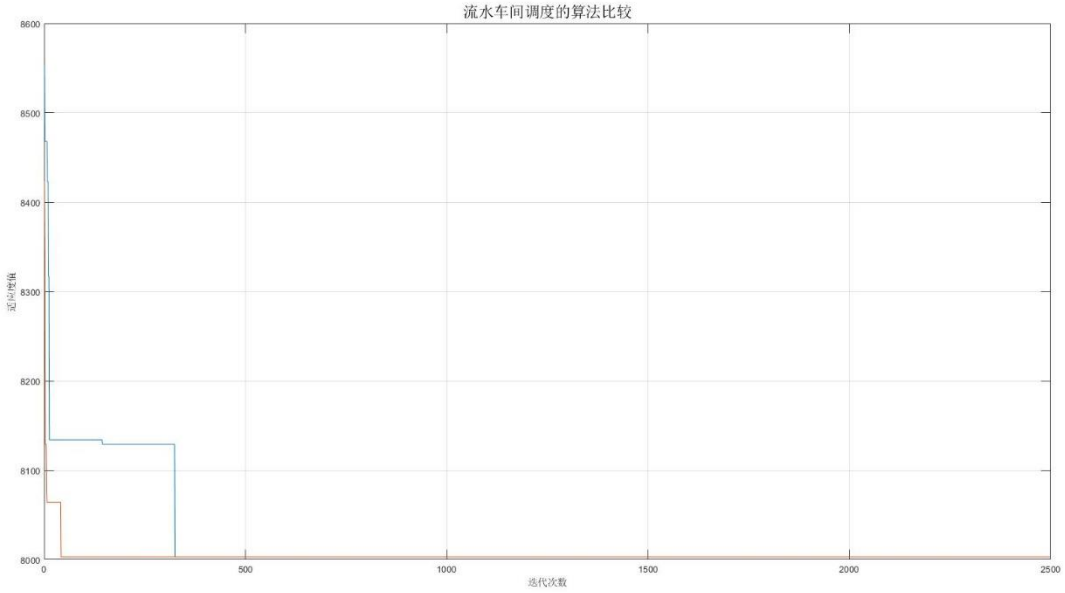

图 15：样例五中标准 PSO 算法（蓝色）和混合 PSO 算法（橙色）的适应度趋势比较

我们可以清楚地看到，标准 PSO 算法与混合 PSO 算法相比逊色了不少，不光适应度下降的趋势较混合 PSO 更慢，而且有些情况下标准 PSO 算法得出的解明显比混合 PSO 算法得出的解差。对于流水车间问题，由于很难辨别搜索方向，标准 PSO 算法单次实验结果存在较大差异。而混合 PSO 算法次实验结果比较稳定，数据差异相对较小，在流水车间问题上具有更好的稳定性和健壮性。

§5 总结评价

本文分析了微粒群算法速度、种群最优值以及全局最优解之间的关系，将遗传算法的交叉和变异思想引入到粒子的位置和速度更新过程中，给出了一种解决数值优化问题的新的混合 PSO 算法。实验结果显示：

1. 混合 PSO 算法不论是在解的稳定性或是求解的速度上均好于标准 PSO 算法，对位置更新引入的交叉思想增加了粒子的多样性和种群的进化质量，使粒子更容易找到正确的搜索方向，从而易越过局部极值而向全局极值收敛；对速度变异加上判定条件，保证了变异操作不会使粒子远离全局极值，从而能快速地向全局极值收敛。
2. 交叉和变异操作的引入所增加的时间消耗在可接受的范围之内。

本文提出的混合 PSO 算法，其中对关键参数作了深入分析，给出了 PSO 算法优化的动态过

程，并对目前的标准 PSO 方法的仿真结果作了综合对比，是优化问题的典型代表，以期对 PSO 的深入研究作好基础的分析，并对以后研究 PSO 算法在各种实际问题中的应用有所帮助。

§6 参考文献

[1] 赫然;王永吉;王青 一种改进的自适应逃逸微粒群算法及实验分析[期刊论文]-软件学报 2005(12)

[2] 宋洁;董永峰;侯向丹 改进的粒子群优化算法[期刊论文]-河北工业大学学报 2008(04)

[3] SUN Jun;FENG Bin;XU Wen-bo Particle swarm optimization with particles having quantum behavior 2004

[4] HI Y，EBERHAART R C. A modified particle Swarm optimizer[C]／／Proc of Congress on Evolutionary Computation. Piseataway：IEEE Press，1998：69·73.