文章编号:1671-4598(2022)06-0266-06 DOI:10.16526/j.cnki.11-4762/tp.2022.06.041 中图分类号:TP311 文献标识码:A

一种多尺度协同变异的萤火虫粒子群混合算法及在车间调度中的应用

周艳平，刘永娟
（青岛科技大学 信息科学技术学院，山东 青岛 266061）

摘要：车间调度对于制造企业提高生产效率、降低生产成本具有重要的作用，针对单一优化算法在解决调度优化问题时存在的不足，探索求解速度和求解质量的均衡，提出了一种多尺度协同变异的萤火虫粒子群混合算法；引入动态自适应策略把种群分为两组，对两组族群平行进化，在保持种群多样性的同时提高求解速度；引入多尺度协同变异算子，利用不同大小方差的自适应高斯变异机制使种群以尽量分散的变异尺度来搜索解空间，通过混沌初始化种群进一步提高算法的局部检索能力；将提出的算法应用于函数优化和流水车间调度问题求解，实验结果显示，算法在求解效率、精度方面优于对比算法，具有较好的性能和应用价值。

关键词：多尺度协同变异；动态自适应策略；平行进化；混沌初始化；流水车间调度

A Hybrid Firefly and Particle Swarm Optimization Algorithm with Multi-scale Cooperative Variation and Its Application in Shop Scheduling

ZHOU Yanping, LIU Yongjuan
(College of Information Science and Technology, Qingdao University of Science & Technology, Qingdao 266061, China)

Abstract: Job shop scheduling plays an important role in improving production efficiency and reducing production cost for manufacturing enterprises, aiming at the shortcoming of single optimization algorithm in solving the scheduling optimization problem, the balance between solution speed and solution quality is researched in this paper, and a hybrid firefly and particle swarm optimization algorithm with multi-scale cooperative variation is proposed. The dynamic adaptive strategy is introduced to divide the population into two groups, and the parallel evolution of the two groups can improve the solution speed while maintaining the diversity of the population; The multi-scale cooperative mutation operator is introduced, and the adaptive Gaussian mutation mechanism with different variances is used to make the population search the solution space with the most dispersed mutation scale, and the local retrieval ability of the algorithm is further improved by initializing the population through the chaos; The proposed algorithm is applied to the function optimization and flow shop scheduling, the experimental results show that the algorithm is superior to the comparison algorithm in solving efficiency and accuracy, which has good performance and application value.

Keywords: multi-scale cooperative variation; dynamic adaptive strategy; parallel evolution; chaos initialization; flow shop scheduling

0 引言

作为有效的资源配置与优化手段，生产调度在整个制造体系中扮演关键的角色，应用范围非常广泛，包括工业、商业等各方面。作为整个生产调度的关键部分，车间调度同时也是制造类企业的核心问题，是一个典型的 NP-hard 问题[1]，一直以来也是组合优化领域内的重点内容，对整个生产流程能否顺利完成有很大的影响。车间调度问题的本质是在符合约束条件的基础上，根据生产指标，给出每个工件的加工顺序、机器、操作方式等的组合问题。流水车间调度作为其中的一类，是流水线生产调度的简化模型，对流水车间调度模型进行研究，有非常重要的意义。

在研究的初期，学者们通过使用精确算法来求解流水车间调度问题，但存在难度大、耗时长等问题，求解效果并不理想。后来研究者们采用近似算法去寻找问题的最优解，近似算法又叫做启发式算法，不断被研究改进的启发式算法成为了调度问题的主要求解办法。

粒子群算法（PSO，particle swarm optimization）是美

收稿日期:2022-03-30； **修回日期**:2022-04-01。
基金项目:国家自然科学基金资助项目(61104004)。
作者简介:周艳平(1976-),男,山东临沂人,博士,副教授,主要从事智能优化算法、生产计划与生产调度、计算机软硬件开发方向的研究。
通讯作者:刘永娟(1997-),女,山东临沂人,硕士研究生,主要从事智能优化算法、生产计划与生产调度方向的的研究。
引用格式:周艳平,刘永娟.一种多尺度协同变异的萤火虫粒子群混合算法及在车间调度中的应用[J].计算机测量与控制,2022,30(6):266-271,278.

国的社会心理学家 Eberhart 和电气工程师 Kennedy[2] 受达尔文“物竞天择、适者生存”的启发，模拟了鸟群群聚、觅食的群体性行为，于 1995 年提出的，算法步骤简单、参数少并且易于实现；英国剑桥大学工程系的 X. S. Yang[3] 在对萤火虫种群的发光行为进行建模的基础上提出了萤火虫算法（FA，firefly algorithm），算法的提出时间不长，对算法的改进和优化仍然处于初级阶段。但不断被研究改进的萤火虫、粒子群算法已被应用于车间调度、资源调度、组合优化等领域。

J. I. FISTER 等人[4]提出了一种模因萤火虫算法（MFFA），将萤火虫算法与局部搜索启发式算法进行结合，并应用于组合优化问题中，具有一定的实用性。郑捷等[5]采用机器选择策略进行种群生成，并对适应度较优的个体之间进行局部搜索，从而产生了一种改进的离散萤火虫算法，并成功地应用在柔性作业车间调度模型中。刘长平等[6]提出了一种 DFA 算法，用 NEH 构造初始解，并根据问题的特性，重新定义了距离公式，通过交叉、变异操作实现位置的更新，并对处在最佳位置的个体进行局部搜索，成功地应用在零空闲置换流水车间调度中。张其亮等[7]结合了粒子群、贪婪算法，通过变异操作来改变粒子停滞或群体陷入局部最优状态，并依据模拟退火接收变异个体，在置换流水车间调度问题的求解质量上有一定的提高。吴琼等[8]结合了粒子群、引力搜索算法，当粒子群进化停滞时，引入引力搜索算法，并采用协同进化的思想，使得两种算法并行优化，并且子种群之间交流，以获得最优粒子，并应用到对作业车间调度问题的求解中。从实验数据来看，以上算法在不同程度上都提高了算法的性能，但在求解精度上还可以继续改进，而且随着车间的日益复杂化，求解算法也应该不断的被改进完善。

单一算法在适应性上是有限的，萤火虫和粒子群算法有一定的相似性，在全局搜索能力方面也有一定的互补性，为更好地发挥两种算法的潜能，取长补短获得更优的解，本文在萤火虫粒子群混合算法的基础上，做出了进一步的改进，提出了一种多尺度协同变异的萤火虫粒子群混合算法（HFPMCV，a hybrid firefly and particle swarm optimization algorithm with multi-scale cooperative variation），引入多尺度协同变异算子，提高求解精度，混沌初始化种群以提高初始解质量，根据高斯拟合把种群分为两组；平行进化，提高求解速度，函数优化实验验证了改进算法的有效性，流水车间调度问题实验，表明了改进算法的优越性。

1　多尺度协同变异的萤火虫粒子群混合算法

1.1　萤火虫算法

萤火虫算法的提出是建立在萤火虫种群中个体的种种社会行为的基础之上的，在实际的应用中，只考虑发光特性，通过发光特性搜索目标附近特定区域，向一定范围内定位较优的萤火虫移动[9]，萤火虫进化和目标函数寻优的最主要的两个关键因素就是荧光亮度和吸引度。亮度表明萤火虫所处方位的好坏，决定了飞行方向，吸引度影响萤火虫的移动距离，利用吸引度和亮度更新、迭代来寻求目标函数最优解。数学描述如下：

在 D 维搜索空间内萤火虫 i 的位置向量 $\boldsymbol{X}_i=(x_{i1},x_{i2},x_{i3},\cdots x_{iD})$，萤火虫荧光亮度公式：

$$I=I_0e^{-\gamma r_{ij}} \tag{1}$$

式中，I_0 为自身亮度，γ 为光吸收系数，r 为距离。

萤火虫之间的相对吸引度正比于亮度：

$$\beta=\beta_0e^{-\gamma r_{ij}^2} \tag{2}$$

式中，β_0 为其初始吸引度。

萤火虫移动距离：

$$X'_i=X_i+\beta_0e^{-\gamma r_{ij}^2}(X_j-X_i)+\alpha rand() \tag{3}$$

α 是扰动的步长因子，$rand()$ 是一个随机扰动，取值为 $[-0.5,0.5]$ 范围内的均匀分布，α 取值为 $[0,1]$ 范围。

最亮的萤火虫位置移动：

$$X'_i=X_i+\alpha randGuass() \tag{4}$$

中心思想就是，所有个体都飞向比自己更亮的个体四周，飞行的最后结果即所有的个体都围在了亮度最强的个体周围，从而达到算法求解的目的。

萤火虫算法 FA：

1)初始化参数，萤火虫数目 n，吸引度 β_0，步长因子 α，迭代次数 Generation，搜索精度 ε。

2)初始化个体位置，最大荧光亮度 I_0 用萤火虫的目标函数值表示。

3)由 1)、2)计算群体中个体的 I、β，飞行方向由相对亮度决定。

4)由 3)更新个体位置，由 4)移动最佳个体。

5)计算个体更新位置后的亮度。

6)满足终止条件则输出最优解，否则转 3)步。

1.2　粒子群算法

粒子群算法的核心思想在于利用粒子与种群之间的信息改变粒子的速率和定位，在飞行范围内，粒子随机飞行，飞行过程中通过学习、模仿周围粒子的信息来获取个体、全体最佳位置，综合分析经验后，调整自身的飞行经验以实现更好的飞行，完成种群间个体协作探索，利用速率和定位的不断更新迭代来寻求目标函数的最佳解[10]。数学描述如下：

D 维搜索区域内，任意一个粒子都在以一定的初速度随机飞行，粒子 i 的当前位置向量 $\boldsymbol{X}_i=(x_{i1},x_{i2},x_{i3},\cdots x_{iD})$，第 i 个粒子当前速率：

$$V_i=(v_{i1},v_{i2},\cdots v_{iD}),\ i=1,2,\cdots N \tag{5}$$

$P_{\text{best}}=(p_{i1},p_{i2},\cdots p_{iD})i=1,2,\cdots,N$ 为个体极值，全局极值为 $g_{\text{best}}=(p_{g1},p_{g2},\cdots p_{gD})$。

粒子进行速度更新：

$$v_{id}=w*v_{id}+c_1r_1(p_{id}-x_{id})+c_2r_2(p_{gd}-x_{id}) \tag{6}$$

ω 是惯性权重，平衡全局、局部搜索能力，c_1、c_2 是学习因子，r_1、r_2 是 $[0,1]$ 里的随机数。速度公式包含三部分，$w*v_{id}$ 表示个体之前的速度对现在速度的影响，$c_1r_1(p_{id}-x_{id})$ 是个体历史最好位置对现在速度的影响，$c_2r_2(p_{gd}-x_{id})$ 是个体之间的信息传递对现在速度的影响。

粒子进行位置更新：

$$x_{id} = x_{id} + v_{id} \tag{7}$$

c_1、c_2 是学习因子，r_1、r_2 是 [0,1] 区间的随机数。

步骤如下：

粒子群算法 PSO：

1)设置参数，包括种群规模、个体的速度和位置，并设置其它参数。

2)初始化种群。

3)计算适应度值，选出个体、全局极值。

4)粒子飞行，据式(6)、(7)更新个体的速度、位置。

5)符合终止条件就输出最优解，反之返回 3)。

1.3 多尺度协同变异的萤火虫粒子群混合算法

考虑到萤火虫算法和粒子群算法的优缺点，杨小东等提出了萤火虫粒子群混合算法 FAPSOMA[11]，本文在此基础上引入多尺度协同变异算子，提出了 HFPMCV 算法，初期，通过多尺度变异算子快速找到全局最优解空间，不断迭代，靠近最优解领域时，通过小尺度变异算子快速收敛到最优解，以有效的进行变异，并且对种群进行混沌初始化，同时动态自适应策略把种群分成两组，两组种群平行进化以寻找优化问题的精确解。

1.3.1 改进策略

1) 混沌初始化：随机初始化会造成个体位置分布不均，个体间差异较大，阻碍全局寻优，影响求解效果[12]，为了获得质量较好的初始解，对种群进行混沌初始化，将变量映射到混沌变量的取值区域，通过线性变换把混沌序列转化到搜索空间[13]。

由 Logistic 映射：

$$Z_{n+1} = \mu Z_n(1 - Z_n),\ n = 0,1,2,\cdots \tag{8}$$

得混沌序列，μ 为控制参量，$\mu \in [0,4]$，当 $\mu = 4$ 时为完全混沌状态。

将混沌序列映射为求解空间范围的混沌变量：

$$p_{n,i} = a_i + (b_i - a_i)Z_{n,i}, i = 1,2,\cdots,I \tag{9}$$

$Z_{n,i}$ 为个体在第 i 维迭代 n 次得到的混沌序列值，I 为搜索空间维度，$p_{n,i}$ 为一个混沌变量，$[p_{n,1}, p_{n,2},\cdots p_{n,I}]$ 代表一个可行解。

因为 PSO 算法主要更新速度、位置、目标函数值，而 FA 算法主要更新位置和目标函数值，本文用 FA 算法中的个体位置和目标值来替代 PSO 算法中的位置和目标值，并在 FA 算法中创建速度信息，所以在本改进算法中，创建 FA 算法的速度序列：

$$z_i(n) = \mu z_i(n-1)(1 - z_i(n-1)) \tag{10}$$

$z_i(n)$ 是 [0, 1] 中符合正态分布的随机数，n 是迭代数，μ 值为 4。

2) 动态自适应控制策略：在传统算法中，种群分组采用随机分组、平均分组的方法，没有充分考虑种群中个体的运动轨迹。

HFPMCV 算法采用高斯拟合函数[14]进行种群分组，在混沌初始化种群后，计算并按照种群中各个体的适应度值进行高斯拟合，因适应度值是由目标函数得来，因此拟合曲线与目标函数曲线有较强关联度[15]，求解拟合曲线峰值的加权平均数[16]，并以此为根据把整个种群划分为两组族群。

3) 多尺度协同变异算子：为了提高求解精度，避免陷入局部最优，引入多尺度协同变异算子[17]，多尺度协同变异利用方差不同的高斯变异机制探索解空间，初期，通过大尺度变异算子快速找到全局最优解空间，不断迭代，靠近最优解领域时，通过小尺度变异算子快速收敛到最优解，以有效的进行变异[18]，设尺度个数为 M，首先初始化方差：

$$\sigma^{(0)} = (\sigma_1^{(0)}, \sigma_2^{(0)}, \cdots \sigma_M^{(0)}) \tag{11}$$

对方差、优化变量设置一样的取值范围，随着迭代次数的增加，方差会一直变化，所有个体生成 M 个子群，个体个数为 $P = N/M$，迭代次数为 K，子种群适应度：

$$FitX_m^{(K)} = \sum_{i-1}^{p} f(x_i^m) * P^{-1}, m = 1,2,\cdots M \tag{12}$$

f 是适应度函数，第 m 个变异算子的标准差：

$$\sigma_m^{(K)} = \sigma_m^{(K-1)} \exp\left(\frac{M * FitX_m^{(K)} - \sum_{m=1}^{M} FitX_m^{(K)}}{FitX_{\max} - FitX_{\min}}\right) \tag{13}$$

$$FitX_{\max} = max(FitX_1^{(K)}, FitX_2^{(K)}, \cdots FitX_M^{(K)}) \tag{14}$$

$$FitX_{\min} = min(FitX_1^{(K)}, FitX_2^{(K)}, \cdots FitX_M^{(K)}) \tag{15}$$

变异算子的进化过程为一个递归的过程，HFPMCV 算法要求的是随着迭代次数的增加变异算子逐渐减小，因此需要规范变异算子的标准差：

$$\begin{aligned} &\text{if} \quad \sigma_i^{(k)} > W/4 \\ &\sigma_i^{(k)} = |\ W/4 - \sigma_i^{(k)}\ | \end{aligned} \tag{16}$$

W 为待优化变量空间的宽度。

位置更新公式如下：

$$\begin{aligned} &\text{If}(v_{id} < T_d)\text{then} \\ &f(x_i + randn * \sigma_i^{(K)}) = \min_{0 \leqslant j \leqslant M} f(x_i + randn * \sigma_j^{(K)}) \\ &If f(x_i + randn * \sigma_i^{(K)}) < f(x_i + rand * V_{\max}) \\ &v_{id} = randn * \sigma_i^{(K)} \\ &Else v_{id} = rand * V_{\max} \end{aligned} \tag{17}$$

其中：T_d 是设置好的阈值，阈值 T_d 过大会降低算法的深度局部搜索能力，阈值过小会降低算法的搜索速度，因此，自适应设定阈值，不同维的速度设置不同的阈值，当达到该值时，阈值会自动下降，当速度小于阈值时，个体进行逃逸：

$$G_d(t) = G_d(t-1) + \sum_{i=1}^{Size} c_{id}(t)$$

其中：

$$c_{id}(t) = \begin{cases} 0 & v_{id}(t) > T_d \\ 1 & v_{id}(t) < T_d \end{cases} \tag{18}$$

$$\begin{aligned} &\text{if} \quad G_d(t) > k_1 \text{ then} \\ &G_d(t) = 0; T_d = T_d/k_2 \end{aligned} \tag{19}$$

k_1、k_2 是常数，$Size$ 为种群大小，$G_d(t)$ 是种群在第 d 维

速度的逃逸次数。

4）平行进化：结合 FA 和 PSO 的优点，使 FA 和 PSO 平行的单独进化[19]，每一代进化结束后，各自保存最优个体，选出较优个体，并与父代最优个体进行比较选出更优个体作为本代最优个体，全部迭代结束后，输出整体最优值，大大提升了求解速度[20]。

1.3.2　算法流程

HFPMCV 算法流程如图 1 所示。

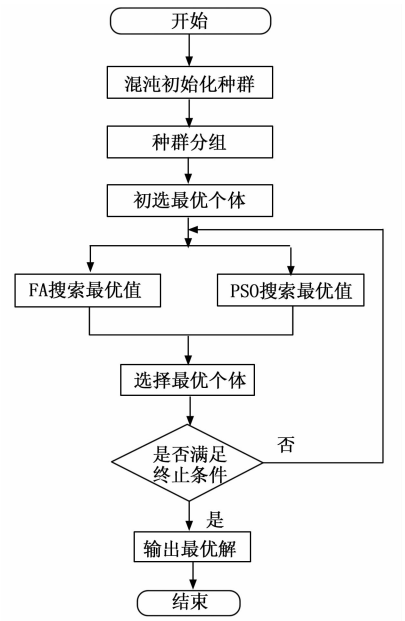

图 1　HFPMCV 算法流程图

1.3.3　函数优化测试

为了测试和解释 HFPMCV 算法的性能和优越性，将 HFPMCV 算法与 FA 算法、FAPSOMA 算法进行了对比，分别采用 HFPMCV 算法以及 FA 算法、FAPSOMA 算法对以下函数[21]进行寻优：

$$y = \sum_{i=1}^{N}(x_i^2 - 10\cos(2\pi x_i) + 10)$$

$$x_i \in [-5.12, 5.12], i = 1,2,\cdots,30 \qquad (20)$$

是一个多峰函数，在 30 维可行域中有 $2^{30}-1$ 个局部求解，最优值是 0，对 3 个算法中的各参数设定一样的值以提高计算有效性和结果精确度，迭代 500 次，种群规模是 100，维度是 30 等。

3 种算法的性能对比如图 2 所示，图中可以看出，迭代次数相同时，HFPMCV 算法收敛更快，精度更高，求解结果相同时，FA、FAPSOMA 需要迭代更多次才能得到最优解，通过函数测试验证了 HFPMCV 算法的优良性能。

2　HFPMCV 算法求解流水车间调度问题

2.1　流水车间调度问题模型

FSP 问题[22]：n 个工件在 m 台机器上加工，i 工件工序数为 L_i，总工序数 L，工序加工时间固定，并按工序加工。在约束条件下安排工件加工顺序以优化指标。

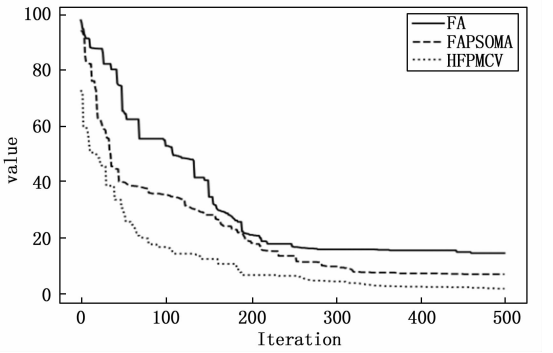

图 2　3 种算法性能比较图

具体表述如下：

$$\begin{cases} T_{1,1} = t_{1,1} \\ T_{i,1} = T_{i-1,1} + t_{i,1}, i = 2,\cdots,n \\ T_{1,j} = T_{1,j-1} + t_{1,j}, j = 2,\cdots,m \\ T_{i,j} = max\{T_{i-1,j}, T_{i,j-1}\} + t_{i,j}, i = 2,\cdots,n; j = 2,\cdots,m \end{cases} \qquad (21)$$

$t_{i,j}$ 是工件 i 在机器 j 上进行加工操作所耗费的时间；$\theta_{i,j}^{k}$ 是在机器 k 上，i 工件加工完成，j 工件的预备时间，本实验中设为 0；$T_{i,j}$ 是工件 i 在机器 j 上的完工时间。

凡事都有一个标准，评价调度方案好坏的标准就是是否达到指标，如客户满意度指标、完工时间指标、性能指标等。本章选取的指标为最大完工时间，以最大完工时间中的最小值作为优化目标[23]，目标函数如下：

$$min\{maxT_{i,j}\}, i = 1,\cdots,n; j = 1,\cdots,m \qquad (22)$$

$maxT_{i,j}$ 为所有工件中最后完成加工的结束时间；$min\{maxT_{i,j}\}$ 为最大完工时间中的最小值。

以流水车间调度问题为模型，公式（21）中的约束如下：

1）一个机器不可同时加工多个工件。

2）每个待加工工件的加工工序是固定的，只有完成了前面工序的加工工作，才可以进行之后工序的加工工作。

3）工件零等待，完成当前工序加工的工件必须要立刻被运输到下台机器，准备加工。

4）当前机器没有需要加工的工件时可以进行等待，直到工件运输到该机器。

5）工件、机器已知，明确机器上加工工件的时间。

2.2　HFPMCV 算法求解流水车间调度问题步骤

HFPMCV 算法求解流水车间调度问题的求解步骤如下：

输入：种群规模 popsize、迭代次数 iteration、搜索范围 bound、维度 vardim

输出：工件加工最优次序、最小完工时间

步骤 1：根据流水车间调度问题，采用基于工件的编码方式，对于整个种群，将位置向量 $x_i[D]$，由公式（8）生成向量 $x_{i+1}[D], x_{i+2}[D], \cdots, x_N[D]$，$N$ 为种群数，通过公式（9）将 $x_i[D], x_{i+1}[D], x_{i+2}[D], \cdots, x_N[D]$ 映射到位置变量

的取值空间，获取最初位置 $X_1[D],X_2[D],\cdots,X_N[D]$，生成 n 个类似 $[p_{n,1},p_{n,2},\cdots p_{n,I}]$ 的可行解，即采用混沌映射的方式生成一个可以进化的初始种群。

步骤 2：根据动态自适应控制策略，计算并依据初始适应度值进行高斯拟合，把整个种群划分为两组族群。

步骤 3：根据初始适应度值，选出两族群各自的最优个体，通过比较，选出较优个体作为本代最优个体，即第一代最优加工次序。

步骤 4：开始平行进化，FA 和 PSO 分别搜索最优值：PSO 加入多尺度协同变异算子，据式（17）更新速度、位置，重新计算适应度值，FA 加入多尺度协同变异算子，据式（1）、（2）更新亮度、吸引度，据式（17）更新位置、速度，重新计算适应度值，FA 和 PSO 各自保存本代最优个体。根据式（13）～（15）分别更新两族群中多尺度协同变异算子的方差，根据变异次数更新阈值。

步骤 5：判断两者保存的最优值哪一个更优，比较更优个体与上一代最优个体，选出较优者作为本代最优个体，即最优方案。

步骤 6：达到最大迭代次数则输出最优加工次序和相对应的最优加工时间，否则转步骤 4，开始下一次并行迭代。

2.3 HFPMCV 算法求解流水车间调度问题复杂度分析

种群规模为 M，个体维度 D，迭代次数 G，初值的混沌序列 m，从算法求解步骤中可以看出，步骤 1 对种群进行混沌初始化，此时的复杂度为 $O(m*D)$，步骤 2 计算初始适应度值的复杂度为 $O(M*D)$，高斯拟合进行种群划分的时间复杂度为 $O(M*D)$，步骤 3 保留最优个体的时间复杂度为 $O(M)$，步骤 4 是通过加入多尺度协同变异算子的萤火虫种群和粒子种群平行进化，因而每一代的时间复杂度为两种群的复杂度之和，对于粒子种群来说，位置、速度更新与自身状态、个体极值、全局极值有关，复杂度是 $3*O(\frac{1}{2}M*D)$，计算适应度值的复杂度是 $O\left(\frac{1}{2}M*D\right)$，对于萤火虫种群来说，因为创建了速度序列，所以与粒子种群类似，种群更新的复杂度是 $3*O(\frac{1}{2}M*D)$，计算适应度值的复杂度是 $O\left(\frac{1}{2}M*D\right)$，一次自动调节变异算子的复杂度是 $O(M*D)$，步骤 5 保存最优个体的复杂度是 $O(M)$，故迭代 G 次，总的时间复杂度为：

$$G*O[O(m*D)+2*O(M*D)+O(M)+O(M)$$
$$2*3*O(\frac{1}{2}M*D)+2*O\left(\frac{1}{2}M*D\right)+O(M*D)]$$

整理后为：

$O(G*m*D)+7*O(G*M*D)+2*O(G*M)$ 可近似看作 $O(G*M*D)$，即复杂度与基本算法仍处于一个数量级。

2.4 仿真实验

2.4.1 实验准备

在 Windows10 操作系统下的 PyCharm 软件上运行程序，采用 python 语言编程。种群规模为 50，迭代 800 次，保证几个算法参数一致，基于工件编码，选择 TA 类车间调度数据集[24]，用 20 * 5、50 * 5 及 100 * 5 这 3 个数据集为基础，分别选择每一个数据集下面的一个典型问题，来进行流水车间调度问题的实验以验证算法。

2.4.2 实验结果与分析

为了测试和解释 HFPMCV 算法的性能和优越性，分别使用 FA、FAPSOMA、HFPMCV 对 3 种不同规模的调度集进行对比实验，并以最小化最大完工时间为指标，为避免实验的偶然性，3 种算法在对 3 个不同规模的数据集进行求解的时候，分别运行 10 次，把 10 次结果的平均值和最优值作为结果参考指标。实验结果如表 1～3 所示。

表 1 3 种算法在规模为 20 * 5 下的输出结果

算法名称	输出结果	最优值	平均值
FA	1 363/1 368/1 339/1 358/1 360/ 1 339/1 349/1 362/1 353/1 360	1 339	1 355.1
FAPSOMA	1 329/1 332/1 322/1 338/1 321/ 1 320/1 324/1 349/1 347/1 321	1 320	1 330.3
HFPMCV	1 297/1 297/1 305/1 310/1 291/ 1 294/1 295/1 302/1 297/1 291	1 291	1 297.9

表 2 3 种算法在规模为 50 * 5 下的输出结果

算法名称	输出结果	最优值	平均值
FA	2 936/2 910/2 936/2 886/2 895/ 2 902/2 891/2 895/2 904/2 924	2 886	2 907.9
FAPSOMA	2 874/2 867/2 853/2 837/2 874/ 2 867/2 875/2 879/2 867/2 879	2 837	2 867.2
HFPMCV	2 777/2 739/2 768/2 768/2 759/ 2 787/2 787/2 752/2 757/2 769	2 739	2 766.3

表 3 3 种算法在规模为 100 * 5 下的输出结果

算法名称	输出结果	最优值	平均值
FA	5627/5616/5628/5694/5655/ 5591/5638/5588/5590/5657	5588	5628.4
FAPSOMA	5582/5564/5575/5569/5544/ 5556/5569/5569/5588/5543	5543	5565.9
HFPMCV	5508/5507/5505/5494/5498/ 5505/5510/5512/5515/5516	5494	5507

仿真结果如表 3 所示，对于同一个规模的数据集，单从最优值来看，例如 50 * 5 规模，FA 的最优值为 2 886，FAPSOMA 的最优值为 2 837，HFPMCV 算法最优值为 2 739，即 HFPMCV 算法可以得到比 FA、FAPSOMA 算法更佳的完工时间，FAPSOMA 算法相比 FA 算法所得到的完工时间更佳。对于同一规模，单从十次结果的平均值来看，例如 100 * 5 规模，HFPMCV 算法的平均值为5 507，

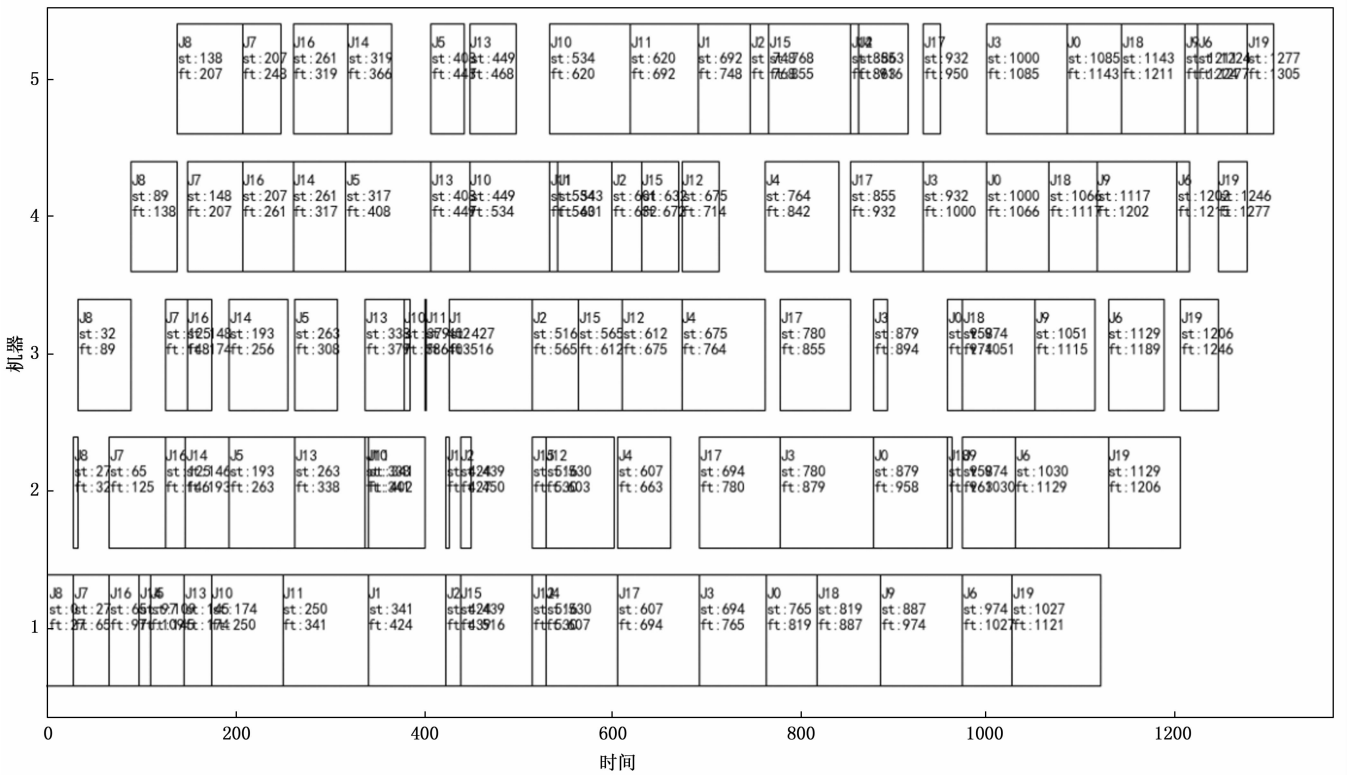

图 3　HFPMCV 算法求解车间调度问题的甘特图

FAPSOMA 的平均值为 5 565.9，FA 的平均值为 5 628.4，HFPMCV 算法的平均值小于 FA、FAPSOMA 算法，FAPSOMA 算法相比 FA 算法得到的平均值更小。综合两个指标来看，HFPMCV 算法在求解流水车间调度问题方面与 FA、FAPSOMA 算法相比，性能更优、强度更高、求解结果更加精准。

甘特图，别名条状图，较为简单、直观，以图示的方式来向人们展示工程进度安排，甘特图上面标识着工件加工顺序，每一个工件在每一台机器上的开始加工时间、结束加工时间，整个优化方案的用时等。通过生成的甘特图可以直观的看到算法求解流水车间调度的结果，即生成的最优加工顺序以及最小完工时间。

图 3 展示了 HFPMCV 算法求解一次 20 * 5 的车间调度问题时所得的甘特图，X 轴表示加工时间，Y 轴表示机器号，J 代表工件号，st 代表工件开始加工时间，ft 代表工件结束加工时间，如工件 8 在第 5 台机器上面的开始加工时间为 138，结束加工时间为 207，加工用时 69，所有的工件在第一台机器上全部加工完成的时间为 1 121，所有的工件在第五台机器上全部加工完成的时间为 1 305，从图中可以看出工件的加工顺序为 [8，7，16，14，5，13，10，11，1，2，15，12，4，17，3，0，18，9，6，19]，最小完工时间为 1 305。

3　结束语

本文针对流水车间调度模型求解问题，并考虑到单种算法对全局最优解的收敛局限性，并且结合前人提出的萤火虫粒子群混合算法，在萤火虫粒子群混合算法的基础上，提出了 HFPMCV 算法，通过引入多尺度协同变异算子、混沌初始化、动态自适应策略以及平行进化模式，提高了算法的求解速率和精度，函数优化实验验证了 HFPMCV 算法的可行、有效，用 HFPMCV、FA、FAPSOMA 算法解决以最小化最大完工时限为指标的流水车间调度问题，对比实验结果显示 HFPMCV 算法得到的完工时间均小于 FA、FAPSOMA 算法，说明了 HFPMCV 算法速率更快、精度更高，与基本算法相比较，在求解流水车间调度模型时，更具有一定的实用价值。但萤火虫位置更新公式仍需要优化，以改善算法时间复杂度，获得更优的求解结果。接下来将着重于研究 HFPMCV 算法在其他车间调度问题中的应用。

参考文献：

[1] GAREY M R，JOHNSON D S，SETHI R. TheComplexity of Flow-shop and Job-shop Scheduling [J]. Mathematics of Operations Research，1976，1 (2)：117－129.

[2] KENNEDY J，EBERHART R. Particle swarm optimization [C] //Neural Networks. Piscataway：IEEE Service Center，1995：1942－1948.

[3] YANG X S. Firefly algorithm Levy flight and global optimization，in：research and development in intelligent system [M]. Springer London，2010：209－218.

[4] FISTER JR I，YANG X S，FISTER I，et al. Memetic firefly algorithm for combinatorial optimization [J]. Arxiv Preprint ArXiv：1204.5165，2012.

（下转第 278 页）