DOI:10.3969/j.issn.1673-4785.201205012
网络出版地址:http://www.cnki.net/kcms/detail/23.1538.TP.20130125.1427.001.html

改进的萤火虫算法求解阻塞流水线调度问题

郭丽萍[1],李向涛[1],谷文祥[1,2],殷明浩[1]
(1.东北师范大学 计算机科学与信息技术学院,吉林 长春 130117;2.长春建筑学院 基础教学部,吉林 长春 130607)

摘 要:为了提高阻塞流水线调度问题的求解性能,提出了一种改进的萤火虫算法来求解阻塞流水线调度问题.首先,提出一种离散机制把个体的实数编码形式转换成离散的作业序列,从而使算法能够应用于离散问题求解;其次,设计一种双重初始化方法,并将NEH启发式方法应用到初始化中来,使算法有一个较优的初始化环境,提高初始种群的解的质量;此外,重新设计了算法中个体的移动方式来增大搜索域;最后,以一定概率对种群中的个体进行局部搜索,加强算法的局部搜索性能.通过对Taillard数据集中部分实例进行求解,实验结果验证了新算法的有效性.

关键词:阻塞流水线调度问题;萤火虫算法;离散机制;NEH启发式;局部搜索
中图分类号:TP301.6 **文献标志码**:A **文章编号**:1673-4785(2013)01-0033-06

An improved firefly algorithm for the blocking flow shop scheduling problem

GUO Liping[1], LI Xiangtao[1], GU Wenxiang[1,2], YIN Minghao[1]
(1. Department of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China; 2. Department of Basic Subjects Teaching, Changchun Architecture & Civil Engineering College, Changchun 130607, China)

Abstract: In order to improve the solving performance of the blocking flow shop scheduling problem, the researcher proposes to examine an improved firefly algorithm. Firstly, a discrete mechanism was proposed to convert the real coded form of individuals to discrete job sequences, so that the algorithm could be used to deal with discrete problems. Secondly, a double-initialization method was designed and the NEH (Nawaz-Enscore-Ham) heuristic method was applied to the initialization, which provided a superior initial environment and improved the quality of the solution of the initial population. In addition, the research paper focused on redesigning the movement pattern of individuals to enhance the search space. Finally, a local search algorithm was applied to individuals with a certain probability, for the purpose of enhancing the ability to search locally. The improved firefly algorithm was then tested utilizing the Taillard datasets, and the results verified the effectiveness of the new algorithm.

Keywords: blocking flow shop scheduling problem; firefly algorithm; discrete mechanism; NEH heuristic method; local search

流水线调度问题是一类重要的组合优化问题,阻塞流水线调度问题作为其中的一个重要分支,由于具有很强的工程背景和实际意义,而得到了越来越多学者的关注,目前人们已经证明包含2台机器以上的阻塞流水线调度问题的计算复杂性为NP-hard[1].该问题的解决方法主要分为构造性启发式方法[2-3]和元启发式方法[4-7].萤火虫算法[8]是由剑桥大学的Yang在2008年提出来的一个模拟自然界中萤火虫发光行为的仿生算法,目前该算法在函数优化方面表现出较优的性能[9].为了使萤火虫算法适用于求解阻塞流水线调度问题,本文对萤火虫算法进行了一些改进,加强了算法的搜索性能,且避免了萤火虫算法早熟的问题.

收稿日期:2012-05-07. 网络出版日期:2013-01-25.
基金项目:国家自然科学基金资助项目(60803102, 61070084).
通信作者:郭丽萍. E-mail: guolp281@nenu.edu.cn.

1 阻塞流水线调度问题

阻塞流水线调度问题可以描述为:n 个作业 J_1, $J_2,\cdots,J_n$ 要在 m 台机器 $M_1,M_2,\cdots,M_m$ 上运行,每个作业包含 m 道工序,每个作业的工序必须按照统一顺序在 m 台机器上执行,每个作业在每台机器上的运行时间是固定的,要求协调不同作业的执行顺序,使得某种生产性能指标达到最优. 阻塞流水线调度问题的约束条件包括如下:

1)每个作业在每台机器上只能加工 1 次;

2)每台机器每次最多只能加工 1 道工序;

3)每台机器上的作业加工顺序相同;

4)作业的某道工序在某台机器上的加工一旦开始,便不能终止;

5)一个作业完成某道工序后,该作业将在当前机器上滞留直到下游机器变为可用为止.

在该问题中,用 $\pi=\{\pi_1,\pi_2,\cdots,\pi_n\}$ 表示一个调度序列,$o_{j,k}(j=1,2,\cdots,n;\ k=1,2,\cdots,m)$ 表示作业 j 在机器 k 上执行一个无间断的工序,其所需时间为 $p_{j,k}(j=1,2,\cdots,n;\ k=1,2,\cdots,m)$,$e_{j,k}(j=1,2,\cdots,n;\ k=0,1,\cdots,m)$ 表示第 j 个作业在第 k 台机器上的离开时间. 则有:

$$e_{1,0}=0, \tag{1}$$

$$e_{1,k}=e_{1,k-1}+p_{1,k},\ k=1,2,\cdots,m-1, \tag{2}$$

$$e_{j,0}=e_{j-1,1},\ j=2,3,\cdots,n, \tag{3}$$

$$e_{j,k}=\max\{e_{j,k-1}+p_{j,k},e_{j-1,k+1}\},\ j=2,3,\cdots,n,\ k=1,2,\cdots,m-1, \tag{4}$$

$$e_{j,m}=e_{j,m-1}+p_{j,m},\ j=1,2,\cdots,n. \tag{5}$$

本文选取最大完工时间最小为优化指标,那么最大完工时间 $C_{\max}(\pi)=e_{n,m}$,且它的计算复杂性为 $O(mn)$.

以 3 个作业 J_1、J_2、J_3 为例,下面举例说明如何求解阻塞流水线调度问题,3 个作业在 3 台机器上运行所需时间如表 1 所示. 第 1 个作业在第 1 台机器上运行所需要的时间为 3,第 1 个作业在第 2 台机器上运行所需要的时间为 4,依此类推.

表 1 时间矩阵

Table 1 Time matrix of an example

作 业	机 器		
	M_1	M_2	M_3
J_1	3	4	3
J_2	6	5	2
J_3	2	7	4

由式(1)~(5)可以计算出该问题的最大完工时间的最小值 $e_{3,3}$ 为 21,对应的作业执行顺序为 $\{J_3,J_2,J_1\}$,如图 1 所示.

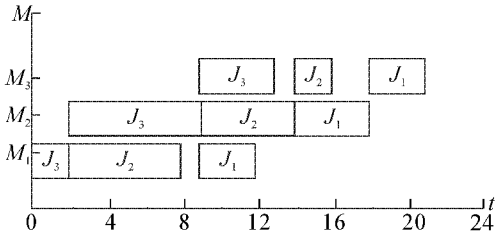


图 1 问题的甘特图表示

Fig. 1 The Gantt chart of the example

2 萤火虫算法

萤火虫算法是一种通过模拟自然界中萤火虫发光行为的仿生算法. 目前,人们对自然界中萤火虫的发光行为的意义还存在很多争议,不过有 2 点是确定的:吸引配偶和吸引潜在的猎物.

一般情况下,可见光的强度随着当前位置到光源距离的增大而减小,另外,空气还要吸收一部分光源. 因此,萤火虫发出光只在一定的距离范围内可以被其他萤火虫看见. 萤火虫算法约定[9]:1)萤火虫之间不存在性别之分,每只萤火虫都可以吸引其他的萤火虫,也可以被其他的萤火虫吸引;2)萤火虫的吸引力和它们发出的光的亮度成正比,因此,对于任意 2 只萤火虫来说,亮度较弱的萤火虫会朝着亮度较强的萤火虫飞行;如果当前可见光距离范围内,没有比自己更亮的萤火虫时,则该萤火虫实行随机飞行策略;3)光的亮度一般由求解问题的目标函数决定.

在萤火虫算法中,每只萤火虫称为一个“个体”,每个个体包含亮度、吸引力、位置等属性. 在该算法中,个体在某个固定位置的亮度 I 是固定的,该变量的设定取决于具体问题的目标函数. 其他个体看到该个体的亮度则随着它们之间距离的增大而减小,此外,传播介质也会吸收一部分光线,因此,个体在距离当前位置 r 处的亮度还和介质的吸收率有关[9]. 一般把一个个体在距离该个体 r 处的亮度表示为

$$I(r)=I_0\mathrm{e}^{-\gamma r^2}. \tag{6}$$

式中:I_0 为个体在当前所处位置的亮度,γ 为介质的吸收率. 有时,为了减小个体亮度变弱的速率,式(6)也可用如下公式代替:

$$I(r)=\frac{I_0}{1+\gamma r^2}.$$

每个个体对其他个体的吸引力 β 也是相对的,它随着 2 个个体之间距离的增大而减小;此外,吸引

力还和介质的吸收率有关[9]. 因此把个体在距离该个体为 r 的位置对其他个体的吸引力定义为

$$\beta(r) = \beta_0 e^{-\gamma r^2}.$$

式中：β_0 表示 2 个个体距离为 0 时，当前个体对另外 1 个个体的吸引力，γ 为介质的吸收率.

在该算法中，2 个个体之间的距离 r 采用欧式距离. 个体 i 向着比它更亮的个体 j 进行更新的公式定义为

$$x_i' = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(R - \frac{1}{2}). \quad (7)$$

式中：x_i 和 x_j 分别表示个体 i 和个体 j 在整个种群更新之前的位置，x_i' 表示个体 i 朝着比自己更亮的个体 j 更新之后的位置，r_{ij} 代表个体 i 和个体 j 之间的距离，$\beta_0 e^{-\gamma r_{ij}^2}$ 表示个体 j 对个体 i 的吸引力，α 是一个[0, 1]内的随机数，R 为一个随机数生成函数，生成[0, 1]内的随机数.

3　改进的萤火虫算法求解阻塞流水线调度问题

虽然萤火虫算法在函数优化方面表现出了较好的计算性能，但是，它在求解阻塞流水线调度问题中仍然存在早熟现象，为了提高算法的寻优性能，本文对萤火虫算法进行了一些改进.

萤火虫算法应用于求解阻塞流水线调度问题时，首先遇到的问题就是如何将实数编码离散化. 本文提出一种最小排序方法，这种方法可以将个体的实数编码形式转化成离散的调度序列. 该方法对初始化后的实数序列从小到大进行排序，它们的排序序号构成一个离散序列，将这个序列作为该实数序列对应的离散调度序列. 如表 2 所示，0.43 是实数编码序列中最小的实数，从小到大排序后，它的序号应该为 1；同理，0.91 是实数序列中最大的，因此其序号为 5. 新生产的序列{4, 2, 5, 1, 3}作为实数序列{0.85, 0.56, 0.91, 0.43, 0.76}对应的离散调度序列.

表 2　个体表示形式

Table 2　The presentation of an object

维　数	X^i	π^i
1	0.85	4
2	0.56	2
3	0.91	5
4	0.43	1
5	0.76	3

其次，在种群初始化时，本文设计了一种双重初始化方法：分别生成 2 个种群，计算 2 个种群中每个个体的最小完工时间. 让 2 个种群中的个体按序号两两进行比较：种群 1 中的第 1 个个体和种群 2 中的第 1 个个体进行比较，选择其中最小完工时间较小的个体，作为新种群的第 1 个个体；种群 1 中的第 2 个个体和种群 2 中的第 2 个个体进行比较，选择其中最小完工时间较小的一个，作为新种群的第 2 个个体；依此类推，新种群作为初始化种群. 此外，由于 NEH(Nawaz-Enscore-Ham)[10]启发式算法是目前求解流水线调度最有效的启发式方法之一，因此，在初始化种群时还引入了 NEH 的思想. 种群的第 1 个个体由 NEH 启发式方法生成，其他个体由双重初始化方式生成，从而使得算法有一个较好的初始化环境.

目前，人们已经发现很多昆虫存在莱维飞行(Lévy flights)[11-12]，且莱维飞行已经应用于优化领域，并取得了预期的良好效果. 为了增强算法的全局搜索性能，避免种群陷入局部最优，在萤火虫算法的搜索过程中，如果当前个体找不到比自己更优的个体时，选择莱维飞行代替原算法中的随机飞行. 此外，对于那些种群中非最优的个体，对它们的飞行公式进行改进：当它们发现比自己更亮的个体时，首先由系统产生一个随机数 q，如果 q 小于 0.5，则按照式(8)进行更新；否则，仍采用式(7)对个体的位置进行更新.

$$x_i'' = x_i' + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i') + \alpha(R - \frac{1}{2}). \quad (8)$$

式中：x_j 仍然表示个体 j 在整个种群更新之前的位置，x_i' 表示的是第 i 个个体朝着前 $j-1$ 个体中比自己亮的个体更新之后的新位置，x_i'' 表示 x_i' 朝着比自己亮的个体 j 更新之后的位置，r_{ij} 代表个体 i 和个体 j 之间的距离，$\beta_0 e^{-\gamma r_{ij}^2}$ 表示个体 j 对个体 i 的吸引力. 可以看出，式(8)是实时更新的，式(7)仅仅依赖于整个种群移动之前的个体位置. 这样的飞行更新方式能够增加飞行的随机性，有利于保持种群的多样性，增大种群搜索空间.

为了加速种群的收敛，本文还提出了一种 α 的更新方式，使得 α 随着迭代次数逐渐减小，更新公式为

$$\alpha = \alpha_0 - e^{0.0015 \times t}.$$

式中：α_0 选用 0.9，t 为迭代次数.

对于一个调度序列而言，目前主要有 3 种产生新序列的方法：插入、交换和逆序[6]. 其中，插入方法被证明为最适合产生一个新序列的方法. 为了加强萤火虫算法的局部搜索能力，本文引入了一个局部搜索算法. 局部搜索算法可以描述如下[7]：

1)选定调度序列 $\pi^r=\{\pi_1^r,\pi_2^r,\cdots,\pi_n^r\}$,$i=0$,$j=1$.

2)$i=i+1$,如果 $i>n$,则 $i=i-n$.

3)把 π^r 赋给中间序列 U,把序列 U 中的第 i 个作业 π_i^r 从 U 中去除,将 π_i^r 插入 U 中所有可以插入的位置,选取其中最好的调度序列 π^{best}.

4)分别求解序列 π^{best} 和序列 U 的最小完工时间 $f(\pi^{best})$ 和 $f(U)$,如果 $f(\pi^{best})<f(U)$,则 $U=\pi^{best}$,$j=0$;否则 $j=j+1$.

5)如果 $j\leqslant n$,转步骤2);否则,算法结束,$\pi^r=U$.

在这个算法中,i 和 j 用来控制循环次数,没有实际意义,n 是作业的数目,也是解的维数.

这种局部搜索算法的好处在于:首先,局部搜索策略并没有直接对所选序列进行修改,而是应用于一个中间序列 U,避免算法陷入局部最优;其次,在每次搜索过程中,只有找到更好的解时,才对中间序列进行更新,有利于算法逐步向更优的解靠近. 因此,局部搜索算法在一定程度上加强了萤火虫算法的局部搜索能力.

改进后的萤火虫算法和原算法相比,初始化部分使得算法有一个较好的搜索域,局部搜索算法又进一步提高了算法的搜索性能. 改进后的萤火虫算法求解阻塞流水线调度问题的步骤如下:

1)初始化种群,包括迭代次数 $t=0$、最大迭代次数 t_{max}、随机参数 α_0、个体吸引力 β_0、介质吸收率 γ、局部搜索概率 p.

2)按照式(1)~(5)计算每个个体的最小完工时间.

3)萤火虫算法.

①更新 α;

②对当前个体,如果有比自己更亮的个体,则按照改进后的更新方式,利用式(7)或(8)对当前个体的位置进行更新;否则,采用莱维飞行对个体位置进行更新.

4)按照式(1)~(5)计算每个个体的最小完工时间.

5)系统产生一个随机数,如果这个随机数小于局部搜索概率 p,则对种群个体进行局部搜索,并更新种群.

6)$t=t+1$,如果 $t\geqslant t_{max}$ 或算法达到其他标准,则算法终止;否则,转步骤3).

4 仿真实验

实验所用 PC 机的处理器为 AMD Athlon64 X2 3600 + 1.91 GHz,内存为 960 MB,编程工具采用 Matlab 7.0.

4.1 数据集

本文采用著名的 Taillard 数据集作为测试实例,该数据集包含 12 个子集,每个子集包含 10 个测试实例. 为了检验算法的有效性,从这些子集中选取了部分具有代表性的实例作为测试数据,所选实例涵盖了从 20 个作业 5 台机器到 200 个作业 10 台机器不同难度的测试实例,具有很好的代表性. 所选实例的规模如表 3 所示. 例如在 Ta04 实例中,20×5 表示 20 个作业在 5 台机器上运行.

表 3 实例规模

Table 3 The size of instances

实 例	规 模	实 例	规 模
Ta04	20×5	Ta54	50×20
Ta14	20×10	Ta64	100×5
Ta24	20×20	Ta74	100×10
Ta34	50×5	Ta84	100×20
Ta44	50×10	Ta94	200×10

4.2 实验结果及分析

实验中,种群大小为 10,最大迭代次数 t_{max} 为 100,α_0、β_0、γ 的设置参考了萤火虫算法相关文献[9,13-14]中的参数设置,α_0 设置为 0.9,β_0 设置为 1.0,γ 设置为 0.9,局部搜索概率 p 设置为 0.2. 对于前 6 个实例,每个实例进行 10 次独立实验;对于后 4 个实例,由于数据量较大,每个个体进行 5 次独立实验. 为了测试改进后的萤火虫算法在阻塞流水线调度问题中的性能,还对同等条件下的未改进的萤火虫算法和标准的粒子群算法进行了测试. 表 4 为未改进的萤火虫算法的测试结果,表 5 为标准粒子群算法的测试结果,改进的萤火虫算法的测试结果列于表 6 中. 其中,在标准的粒子群算法中,2 个学习因子 c_1 和 c_2 取值为 2,惯性权重 ω 设置为 0.7.

由表 4、5 可以看出,无论是最小值、最大值,还是平均值,萤火虫算法的求解结果普遍优于标准粒子群算法. 对比表 4 和表 6,可以发现,改进的萤火虫算法在求解效果上要远远优于基本的萤火虫算法,而且求解结果更加稳定. 综合表 4~6,改进后的萤火虫算法的求解效果明显优于基本的萤火虫算法和标准的粒子群算法. 并且随着种群的增大,这种差距更加明显,充分体现了算法的有效性,而且新算法的求解效果更加稳定.

表 4　萤火虫算法测试结果

Table 4　The experiment results of firefly algorithm

次　数	Ta04	Ta14	Ta24	Ta34	Ta44	Ta54	Ta64	Ta74	Ta84	Ta94
1	1 655	1 730	2 598	3 799	4 404	5 065	7 334	8 599	9 174	16 488
2	1 704	1 768	2 591	3 872	4 404	5 042	7 392	8 551	9 115	16 507
3	1 671	1 786	2 632	3 861	4 412	4 968	7 318	8 581	9 147	16 471
4	1 691	1 799	2 635	3 846	4 345	5 010	7 355	8 607	9 174	16 477
5	1 700	1 759	2 636	3 848	4 365	5 048	7 361	8 627	9 197	16 472
6	1 672	1 742	2 638	3 781	4 394	5 031	—	—	—	—
7	1 694	1 739	2 600	3 825	4 350	5 032	—	—	—	—
8	1 687	1 745	2 645	3 850	4 368	4 994	—	—	—	—
9	1 696	1 796	2 629	3 794	4 361	5 034	—	—	—	—
10	1 663	1 764	2 582	3 801	4 411	5 010	—	—	—	—
最小值	1 655	1 730	2 582	3 781	4 345	4 968	7 318	8 551	9 115	16 471
最大值	1 704	1 799	2 645	3 872	4 412	5 065	7 392	8 627	9 197	16 507
平均值	1 683.3	1 762.8	2 618.6	3 827.7	4 381.4	5 023.4	7 352.0	8 593.0	9 161.4	16 483.0

表 5　粒子群算法求解结果

Table 5　The experiment results of particle swarm algorithm

次　数	Ta04	Ta14	Ta24	Ta34	Ta44	Ta54	Ta64	Ta74	Ta84	Ta94
1	1 803	1 822	2 800	3 994	4 427	5 194	7 432	8 761	9 261	16 754
2	1 815	1 892	2 750	4 029	4 575	5 199	7 583	8 824	9 361	16 899
3	1 857	1 772	2 758	4 030	4 480	5 195	7 633	8 828	9 458	16 705
4	1 789	1 911	2 810	4 006	4 538	5 277	7 614	8 801	9 373	16 700
5	1 775	1 774	2 780	4 001	4 497	5 118	7 603	8 773	9 456	16 832
6	1 900	1 910	2 797	4 007	4 486	5 224	—	—	—	—
7	1 833	1 951	2 796	3 970	4 499	5 253	—	—	—	—
8	1 778	1 926	2 722	3 964	4 593	5 218	—	—	—	—
9	1 879	1 901	2 753	4 005	4 496	5 225	—	—	—	—
10	1 835	1 893	2 835	4 007	4 574	5 166	—	—	—	—
最小值	1 775	1 772	2 722	3 964	4 427	5 118	7 432	8 761	9 261	16 700
最大值	1 900	1 951	2 835	4 030	4 593	5 277	7 633	8 828	9 458	16 899
平均值	1 826.4	1 875.2	2 780.1	4 001.3	4 516.5	5 206.9	7 573.0	8 797.4	9 381.8	16 778.0

表 6　改进的萤火虫算法求解结果

Table 6　The experiment results of improved firefly algorithm

次　数	Ta04	Ta14	Ta24	Ta34	Ta44	Ta54	Ta64	Ta74	Ta84	Ta94
1	1 448	1 535	2 348	3 217	3 763	4 449	5 943	7 357	8 104	13 885
2	1 448	1 535	2 348	3 209	3 768	4 452	5 914	7 319	8 051	13 875
3	1 448	1 535	2 348	3 216	3 760	4 465	5 976	7 361	8 050	13 864
4	1 449	1 535	2 348	3 199	3 723	4 434	5 958	7 320	8 072	13 802
5	1 448	1 535	2 353	3 213	3 762	4 456	5 962	7 381	8 021	13 889
6	1 449	1 535	2 348	3 194	3 767	4 440	—	—	—	—
7	1 448	1 535	2 348	3 205	3 747	4 438	—	—	—	—
8	1 448	1 538	2 348	3 214	3 755	4 454	—	—	—	—
9	1 448	1 535	2 348	3 200	3 748	4 448	—	—	—	—
10	1 448	1 535	2 348	3 216	3 747	4 411	—	—	—	—
最小值	1 448	1 535	2 348	3 194	3 723	4 411	5 914	7 319	8 021	13 802
最大值	1 449	1 538	2 353	3 217	3 768	4 465	5 976	7 381	8 104	13 889
平均值	1 448.2	1 535.3	2 348.5	3 208.3	3 754.0	4 444.7	5 950.6	7 347.6	8 059.6	13 863.0

5 结束语

本文提出了一种改进的萤火虫算法,设计了双重初始化方法,引入了 NEH 启发式方法,重新设计了算法中个体位置的更新方式,并引入了莱维飞行来增大种群的搜索域. 在求解阻塞流水线调度问题时,设计了一种将实数编码离散化的机制,实现了实数编码算法对离散化问题的求解. 此外,本文还引入了一种局部搜索算法来加强算法的局部搜索能力. 实验结果表明,改进后的萤火虫算法在求解阻塞流水线调度问题时,性能有了明显提高,而且随着问题规模的增大,这种提高更加明显,体现了算法的有效性和鲁棒性.

由于阻塞流水线调度问题具有很重要的实际应用价值,因此将来的工作会考虑把其他优秀的算法引入到这个问题中进行求解. 另外,由于元启发式算法普遍存在计算量大的问题,将来的工作还应该考虑如何提高求解效率.

参考文献:

[1] ALLAHYERDI A, NG C T, CHENG T C E, et al. A survey of scheduling problems with setup times or costs[J]. European Journal of Operational Research, 2008, 187(3): 985-1032.

[2] MCCORMICK S T, PINEDO M L, SHENKER S, et al. Sequencing in an assembly line with blocking to minimize cycle time[J]. Operations Research, 1989, 37(6): 925-935.

[3] RONCONI D P. A note on constructive heuristics for the flowshop problem with blocking[J]. International Journal of Production Economics, 2004, 87(1): 39-48.

[4] CARAFFA V, IANES S, BAGCHI T P, et al. Minimizing makespan in a blocking flowshop using genetic algorithms [J]. International Journal of Production Economics, 2001, 70(2): 101-115.

[5] RONCONI D P. A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking[J]. Annals of Operations Research, 2005, 138(1): 53-65.

[6] GRABOWSKI J, PEMPERA J. The permutation flow shop problem with blocking. A tabu search approach[J]. Omega, 2007, 35(3): 302-311.

[7] WANG Ling, PAN Quanke, SUGANTHAN P N, et al. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems[J]. Computers & Operations Research, 2010, 37(3): 509-520.

[8] YANG Xinshe. Nature-inspired metaheuristic algorithms [M]. Frome, UK: Luniver Press, 2008: 81-96.

[9] YANG Xinshe. Firefly algorithms for multimodal optimization[C]//Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications. Berlin/Heidelberg, Germany: Springer-Verlag, 2009: 169-178.

[10] NAWAZ M, ENSCORE E E J, HAM I. A heuristic algorithm for the m-machine, n-job flow shop sequencing problem[J]. Omega, 1983, 11(1): 91-95.

[11] BROWN C T, LIEBOVITCH L S, GLENDON R. Lévy flights in Dobe Ju/'hoansi foraging patterns[J]. Human Ecology, 2007, 35(1): 129-138.

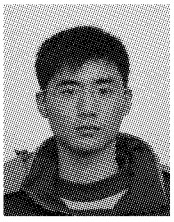
[12] PAVLYUKEVICH I. Lévy flights, non-local search and simulated annealing[J]. Journal of Computational Physics, 2007, 226(2): 1830-1844.

[13] YANG Xinshe. Firefly algorithm, Lévy flights and global optimization[C]//Research and Development in Intelligent Systems XXVI. London, UK: Springer, 2010: 209-218.

[14] YANG Xinshe. Firefly algorithm, stochastic test functions and design optimization[J]. International Journal of Bio-Inspired Computation, 2010, 2(2): 78-84.

作者简介:

郭丽萍,女,1989 年生,硕士研究生,主要研究方向为智能规划、智能信息处理.



李向涛,男,1987 年生,博士研究生,主要研究方向为智能规划、智能信息处理,发表学术论文多篇.

谷文祥,男,1947 年生,教授,博士生导师,主要研究方向为智能规划与规划识别、形式语言与自动机理论、模糊数学及其应用. 主持国家自然科学基金项目 3 项,发表学术论文 100 余篇.