ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.21, No.12, December 2010, pp.3082−3093
doi: 10.3724/SP.J.1001.2010.03687

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 基于记忆库拉马克进化算法的作业车间调度*

夏柱昌[1,2], 刘 芳[1,2+], 公茂果[2,3], 戚玉涛[1,2]

[1](西安电子科技大学 计算机学院,陕西 西安 710071)

[2](西安电子科技大学 智能感知与图像理解教育部重点实验室,陕西 西安 710071)

[3](西安电子科技大学 智能信息处理研究所,陕西 西安 710071)

## Memory Based Lamarckian Evolutionary Algorithm for Job Shop Scheduling Problem

XIA Zhu-Chang[1,2], LIU Fang[1,2+], GONG Mao-Guo[2,3], QI Yu-Tao[1,2]

[1](School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

[2](Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, Xidian University, Xi'an 710071, China)

[3](Institute of Intelligent Information Processing, Xidian University, Xi'an 710071, China)

+ Corresponding author: E-mail: f63liu@163.com

**Xia ZC, Liu F, Gong MG, Qi YT. Memory based lamarckian evolutionary algorithm for job shop scheduling problem. *Journal of Software*, 2010,21(12):3082−3093.** http://www.jos.org.cn/1000-9825/3687.htm

**Abstract**: Compared with the Genetic Algorithm, a multi-population genetic algorithm has an enhancement in performance, but for a job shop scheduling problem, which has a lot of local optima, it also has the shortcomings of an easy-to-fall into local optima and a poor ability of local search. Therefore, an effective algorithm is proposed to solve job shop scheduling problem. The proposed algorithm, based on multi-population genetic algorithm, involves the strategy of memory-base and a mechanism of the Lamarckian evolution. Not only does the memory-base make individuals between sub-populations exchange information, but it can maintain the diversity of the population. The local search operator, based on Lamarckian evolution, is adupted to enhance the individual's ability of local search. The simulated annealing algorithm that has a stronger ability to jump out local optima than the genetic algorithm is used, thus, alleviated the problem and enhances the performance of the algorithm for job shop scheduling. The experimental results on the well-known benchmark instances show the proposed algorithm is very effective in solving job shop scheduling problems.

* Supported by the National Natural Science Foundation of China under Grant Nos.60703107, 60703108, 60803098, 60803706, 60872135 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.20060101Z1119, 2009AA12Z210 (国家高技术研究发展计划(863)); the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant Nos.20060701007, 20070701022 (国家教育部博士点基金); the Key Scientific and Technological Innovation Special Projects of Shaanxi "13115" of China under Grant No.2008ZDKG-37 (陕西省"13115"科技创新工程重大科技专项项目); the Program for Cheung Kong Scholars and Innovative Research Team in University of China under Grant No.IRT0645 (国家教育部长江学者和创新团队支持计划); the China Postdoctoral Science Foundation Funded Project under Grant Nos.200801426, 20080431228 (中国博士后科学基金资助项目)

Received 2008-06-23; Revised 2008-10-09; Accepted 2009-07-06

**Key words**: job shop scheduling; multi-population genetic algorithm; memory-base; Lamarckian evolution; local search; simulated annealing

**摘 要**: 多种群遗传算法相比遗传算法在性能上能够有所提高,但对具有较多局部最优解的作业车间调度问题,多种群遗传算法仍然难以改善易陷入局部最优解和局部搜索能力差的缺点.因此,提出了一种求解作业车间调度问题的新算法 MGA-MBL(multi-population genetic algorithm based on memory-base and Lamarckian evolution for job shop scheduling problem).MGA-MBL 在多种群遗传算法的基础上通过引入记忆库策略,不但使子种群间的个体可以进行信息交换,而且有利于保持整个种群的多样性;通过构造基于拉马克进化机制的局部搜索算子来提高多种群遗传算法中子种群进化的局部搜索能力.由于 MGA-MBL 采用了全局寻优能力较强的模拟退火算法对记忆库中的个体进行优化,从而缓解了多种群遗传算法易陷入局部最优解的问题,并提高了算法求解作业车间调度问题的性能.对著名的 benchmark 数据进行测试,实验结果证实了 MGA-MBL 在求解作业车间调度问题上的有效性.

**关键词**: 作业车间调度;多种群遗传算法;记忆库;拉马克进化;局部搜索;模拟退火

**中图法分类号**: TP301 **文献标识码**: A

作业车间调度问题(job shop scheduling problem,简称JSSP)[1]是一个典型的NP-难问题,其研究的方法有两大类:精确算法和近似算法.精确算法主要采用基于析取图模型的枚举方法,或基于活动调度生成、混合整数规则模型等的方法.近似算法有优先权规则调度算法、启发式算法和随机搜索的改进算法,尤其是遗传算法[2]、模拟退火算法(simulated annealing,简称SA)[3]和禁忌搜索算法[4]倍受关注[5].

遗传算法[6]善于探索问题的整个空间,却不擅长对局部区域的精细调整,而且容易陷于局部最优.因此,标准遗传算法的一系列改进算法被提出来.多种群遗传算法[7]就是标准遗传算法的改进算法,每个子种群搜索解空间中的一部分,并且子种群之间可以进行信息交换,因此在一定程度上可以避免陷入局部最优问题.一般的多种群遗传算法采用普通的通信机制,即相邻子种群中最优个体替换当前子种群中适应度较低的个体,信息交换以染色体为单位.我们提出的方法MGA-MBL也是多种群遗传算法,与一般的多种群遗传算法不同的是,MGA-MBL通过引入了记忆库策略改进了子种群间的普通通信机制.MGA-MBL利用记忆库记录各子种群中的优秀个体,每个子种群中个体以一定概率同记忆库中个体进行交叉操作,即子种群之间的信息交换是以基因为单位,从而增加了整个种群的多样性,有利于全局最优解的搜索.

然而对于局部最优解比较多的作业车间调度问题而言,常常会使种群收敛到局部最优.因此,如何跳出局部最优,成为一个迫切需要解决的问题.模拟退火算法[8]采用Metropolis准则,以一定概率接受差的解[9],所以SA也能以一定概率跳出当前的局部最优.在跳出局部最优能力这点上,SA算法要比标准遗传算法强,但SA算法的性能与初始解选取的好坏有很大关系.另外,其收敛速度慢.在方法MGA-MBL中,记忆库实际上是一个特殊的子种群,它除了承担子种群间信息交换的功能外,还记录着各个子种群中的优秀个体(这点正好满足SA算法对初始解的要求).因此,在MGA-MBL中,我们对记忆库采用SA算法进行优化,而其他子种群采用遗传算法进行优化,缓解了多种群遗传算法易陷入局部最优解的问题.

在多种群遗传算法中,子种群的进化主要采用标准遗传算法.因此,标准遗传算法存在的局部搜索能力差的缺点,多种群遗传算法同样存在.针对作业车间调度问题,如何充分利用个体的有效信息,提高遗传算法的局部搜索能力也是本文的工作之一.拉马克进化理论为我们提供了有力的工具.Lamarck认为,有机体后天学习获得的性状能直接反馈在基因型上,通过基因遗传给后代.局部搜索[10,11]可以看作是一个有机个体在他生命周期中的后天学习过程,搜索到好的基因片段直接反馈到染色体上,并且修改个体的适应度,通过遗传作用,将好的基因片段遗传给下一代.在MGA-MBL中,受拉马克进化学说思想的启发,我们构造了适合作业车间调度问题的局部搜索算子.

本文第 1 节阐述多种群遗传算法在作业车间调度问题上的应用.第 2 节描述记忆库及其优化、基于拉马克进化机制的局部搜索算子和 MGA-MBL 算法.第 3 节是 MGA-MBL 在 benchmark 标准测试问题上的测试结果

及其分析.第 4 节是结论.

## 1 多种群遗传算法在作业车间调度问题上的应用

### 1.1 Job-Shop调度问题描述

JSSP问题的描述为:一个加工系统有$M$台机器,要求加工$N$个作业,其中,作业$i$包含工序数为$L_i$.令$L=\sum_{i=1}^{N}L_i$,则 $L$ 为任务集的总工序数.其中,各工序的加工时间已确定,并且每个作业必须按照工序的先后顺序加工.调度的任务是安排所有作业的加工调度排序,约束条件被满足的同时,使性能指标得到优化.

JSSP 需要考虑如下约束:

I. 每道工序在指定的机器上加工,且必须在其前一道工序加工完成后才能开始加工;

II. 某一时刻 1 台机器只能加工 1 个作业;

III. 每个作业只能在 1 台机器上加工 1 次;

IV. 各作业的工序顺序和加工时间已知,不随加工排序的改变而改变.

例如,我们以优化目标为最大完工时间的最小化为例:令$(i,j)$表示作业$i$的第$j$个工序.$S_{ij}$和$T_{ij}$分别表示$(i,j)$的加工起始时刻和加工时间.$Z_{ijk}$表示$(i,j)$是否在第$k$台机器上加工:如果$(i,j)$在第$k$台机器上加工,$Z_{ijk}$=1;否则,$Z_{ijk}$=0.$C_k$为第$k$台机器的完工时间,则

$$F=\min(\max(C_k)),k=1,2,\ldots,M \tag{1}$$

$$\text{s.t. } S_{ij}-S_{i(j+1)}+T_{ij}\leqslant 0,i=1,2,\ldots,N;j=1,2,\ldots,L_i-1 \tag{2}$$

$$S_{i1}\geqslant 0,i=1,2,\ldots,N \tag{3}$$

$$Z_{ijk}=Z_{pqk}=1 \text{ 且} S_{ij}-S_{pq}+T_{ij}\leqslant 0 \text{ 或} S_{pq}-S_{ij}+T_{pq}\leqslant 0,i\neq p \tag{4}$$

公式(1)为目标函数,使最迟完工的机器尽早完成,即加工时间最短;公式(2)表示 1 个作业只能在加工完成前一道工序后才可以加工后一道工序;公式(3)表示 1 个作业的第 1 道工序的起始加工时刻大于或等于 0;公式(4)表示在 1 台机床上不会同时加工 1 个以上的作业.

### 1.2 编码设计

本文采用基于工序的编码方式,染色体由所有工序的排序构成,每个基因代表一个工序,同一作业的所有工序采用同一作业序号表示,根据它们在染色体排序中的顺序决定它们在不同机器上的加工顺序.一个 $N\times M$ job-shop 问题,其染色体有 $N\times M$ 个基因组成,每个作业序号在染色体中出现 $M$ 次,对于某一作业序号,它的第 $i$ 次出现表示该作业的第 $i$ 道工序($i$=1,2,…,$M$).设 3×3 job shop 问题的一个染色体为:112132233,其中,1 表示作业 1,2 和 3 意义相同.染色体中的 3 个 1 表示作业 1 的 3 个工序,第 1 个 1 对应于作业 1 的第 1 道加工工序,以此类推.

### 1.3 解码设计

本文采用前插式解码[12],例如染色体 112132233,根据表 1 所给的数据,对每个当前工序,从其所在机器的前面开始遍历,如果有时间间隔和约束条件可以满足当前工序,则将其插入该位置,其Gantt图表示如图 1 所示.不难看出,此解码方法可以解码为活动调度,相比顺序插入的半活动调度方法可以节省大量的时间.

**Table 1** Data for a jssp instance of 3 jobs and 3 machines

**表 1** 一个 3 作业 3 机器的 jssp 加工数据

| Job | Machine (processing time) | | |
|---|---|---|---|
| 1 | 1(4) | 2(3) | 3(3) |
| 2 | 2(3) | 1(2) | 3(2) |
| 3 | 2(4) | 1(3) | 3(1) |

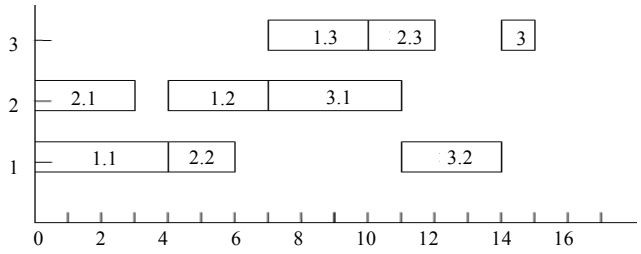


Fig.1 Gantt chart for chromosome [12132233]

图 1 染色体[112132233]的 Gantt 图

### 1.4 适应度函数及交叉算子设计

#### 1.4.1 适应度函数

根据第 1.3 节所给出的解码方法,每个染色体经过解码都有一个对应的调度时间(完成所有作业加工的时间).设$E_i$为作业$i$的完工时间,则一个染色体对应的调度时间为$Time=\max(E_i),i=1,2,\ldots,N$.我们的目标是要寻找调度时间短的合理调度顺序,因此我们选择适应度函数为$f=1/Time$.

#### 1.4.2 交叉算子

交叉操作的目的是利用个体组合出后代新个体,在尽量降低有效模式破坏概率的基础上对解空间进行高效搜索.交叉操作是主要的遗传操作,GA 的性能在很大的程度上依赖所使用的交叉算子.鉴于 JSSP 的特殊性,GA 必须结合所采用的编码技术设计相应的交叉算子.

本文应用双点交叉算子,其原理如下:随机选取染色体中的 2 个交叉点,设为$P_1,P_2$,且 $1\leqslant P_1<P_2\leqslant N\times M$,父代个体$A$和$B$分别为

$$A:a_1a_2\ldots a_{P_1}\ldots a_{P_2}\ldots a_{N\times M};\ B:b_1b_2\ldots b_{P_1}\ldots b_{P_2}\ldots b_{N\times M},$$

其中,$1\leqslant a_i\leqslant N,1\leqslant b_i\leqslant N,1\leqslant i\leqslant N\times M$.对于父代个体$A$的子代个体$A_s$,$P_1\sim P_2$基因位之间的基因与父代个体$A$的$P_1\sim P_2$基因位之间的基因相同,其余 $1\sim P_1-1$ 基因位和$P_2+1\sim N\times M$基因位的基因由以下操作得到的基因片段填充:首先从左到右遍历父代个体$B$,找到第 1 个出现与 $a_{P_1}$ 相等的基因值$b_j(1\leqslant j\leqslant N\times M)$,从父代染色体$B$中删除第 1 个出现的基因值为$b_j$的基因,得到基因片段$B_1:b_1b_2\ldots b_{j-1}b_{j+1}\ldots b_{N\times M-1}$;然后从左到右遍历$B_1$,找到第 1 个出现与 $a_{P_1+1}$ 相等的基因值$b_j(1\leqslant j\leqslant N\times M-1)$,从染色体$B_1$上删除第 1 个出现的基因值为$b_j$的基因,得到基因片段$B_2:b_1b_2\ldots b_{j-1}b_{j+1}\ldots b_{N\times M-2}$;重复上述删除操作,直到从染色体 $B_{P_2-P_1}$ 上删除第 1 个出现与 $a_{P_2}$ 相等的基因值为$b_j$的基因$(1\leqslant j\leqslant N\times M-(P_2-P_1))$,得到基因片段 $B_{P_2-P_1+1}:b_1b_2\ldots b_{j-1}b_{j+1}\ldots b_{N\times M+P_1-P_2-1}$;将基因片段 $B_{P_2-P_1+1}$ 中基因按照顺序分别放到$A_s$的 $1\sim P_1-1$ 基因位和$P_2+1\sim N\times M$基因位上,即得到子代个体$A_s$.父代个体$B$的子代个体$B_s$产生过程同上.

例如,父代染色体为$A$:231123321,$B$:223112313,$P_1,P_2$分别为 3 和 6.

父代个体$A$的子代个体$A_s$的产生过程如下:

Step 1. 子代个体$A_s$的 3~6 基因位之间的基因与父代个体$A$的 3~6 基因位之间的基因相同,即 _ _ 1 1 2 3 _ _;

Step 2. 从左到右遍历$B$,首先删除第 1 个出现与染色体$A$中第 3 位上基因值相同的一个基因(223+12313),得到基因片段$B_1$:22312313;然后从左到右遍历$B_1$,删除第 1 个出现与染色体$A$中第 4 位上基因值相同的一个基因(223+2313),得到基因片段$B_2$:2232313;从左到右遍历$B_2$,删除第 1 个出现与染色体$A$中第 5 位上基因值相同的一个基因(2232313),得到基因片段$B_3$:232313;最后从左到右遍历$B_3$,删除第 1 个出现与染色体$A$中第 6 位上基因值相同的一个基因(23+2313),得到基因片段$B_4$:22313;

Step 3. 子代个体$A_s$中的 1~2 基因位和 7~9 基因位按照得到的基因片段$B_4$按顺序补全,得到子代个体$A_s$:221123313.

同样操作,可以得到父代个体 $B$ 的子代个体为 233112321.

这种交叉操作简单,保持了父代个体中的一些基因段不变,保证了优良的交叉性能.

## 2 基于记忆库策略和拉马克进化机制的作业车间调度

多种群遗传算法相比遗传算法在求解较为简单的问题时表现出优越的性能,但在求解比较复杂的问题时,仍然表现出遗传算法本质上的易陷入局部最优和局部搜索能力差的缺点.MGA-MBL 对多种群遗传算法进行了改进,通过引入记忆库策略和基于拉马克进化机制的局部搜索算子来弥补多种群遗传算法表现出来的不足.

### 2.1 记忆库及其优化

多种群遗传算法中的每个子种群独立进化,独立搜索解空间中的一部分.我们提取出每个子种群的优秀个体组成记忆库,因此记忆库记录了解的各个子空间的优秀个体.我们用 SA 算法对这些优秀个体进行优化,子种群通过优化后的记忆库进行通信.因此,记忆库可以有效指导个体朝向解空间中的全局最优移动.

模拟退火算法由 Metropolis 等人于 1953 年提出,其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性.该算法是在某一初温下,伴随温度的不断下降,结合概率突跳特性在解空间中随机寻找目标函数的全局最优解.

状态产生函数:设计状态产生函数的出发点是尽可能保证产生的候选解遍布全部解空间.车间调度问题的常用状态产生函数有基因互换、逆转和插入.本文借鉴遗传算法中的交叉算子,状态产生函数采用单性交叉的方法.单性交叉过程是:随机产生 4 个随机数$P_1$,$P_2$,$P_3$,$P_4$,使他们满足 $1\leqslant P_1<P_2<P_3<P_4\leqslant N\times M$,然后交换$P_1$~$P_2$位和$P_3$~$P_4$位之间的基因,即可得到子代个体.例如,父代染色体为 2111223333,产生的随机数为 2,4,6,8,然后使 2~4 位与 6~8 位之间的基因交换,交换后的子代个体为 2233211133.

状态接受函数(metropolis准则):状态接受函数是SA产生概率突跳能力的关键.本文采用的状态接受函数为$P(S_i\rightarrow S_j)=\min\{1,\exp(-\Delta/t)\}$.其中,$\Delta$为新旧个体的适应度之差.这里的能量函数不一定非要选取适应度,当然也可以利用调度时间.

停止准则:本文采用了简单的最低温度要求策略,即当前温度小于最低温度则停止迭代.

**SA 算法**. 优化记忆库.

Step 1. 对于记忆库中的当前个体,使用本文的状态产生函数产生新个体;

Step 2. 计算新个体的适应度;

Step 3. 如果新个体适应度高,则替换旧个体;否则,计算新旧个体适应度之差,使用状态接受函数判断是否接受新个体;

Step 4. 退温,$t=\lambda t(0<\lambda<1)$;

Step 5. 判断停止准则是否满足:如果满足则结束,否则转 Step 1.

由于 MGA-MBL 采用了较为简单的最低温度停止准则,为了防止迭代过程中遗失最优个体,在迭代过程中保留了搜索到的最优个体.

### 2.2 基于拉马克进化机制的局部搜索算子

在 Darwin 提出基于自然选择的进化论之前,Lamarck 提出了生物进化的多面理论.拉马克进化理论认为,生物体在生存期内存在自身学习的过程,并能将学习到的知识传递给后代.然而,拉马克进化论认为生物体后天学习获得的性状是通过基因遗传给后代的,这已被证明在生物进化中是错误的.但是,Dawkins 密母学理论重新让拉马克进化理论焕发了生机.虽然拉马克进化理论被证明在生物进化中是错误的,但是在文化进化(cultural evolution)中却是正确的.近几年,拉马克进化理论被抽象为一种局部搜索机制引入进化计算中,逐步成为进化计算的研究热点之一.

局部搜索是父代个体在其邻域上的学习过程,通过学习,可以更加适应环境,即适应度得到提高.表现型空间表现为更加适应环境,基因型空间为局部基因排列的改变.根据拉马克进化理论,得到改进的父代个体可以将学习到的好的基因遗传给子代,得到更加适应环境的子代个体.因此,局部搜索在局部空间上引导个体向更好的

解移动.

局部搜索是以牺牲计算时间为代价的.因此,拉马克进化的贡献必须与所研究问题的代价相权衡.作业车间调度问题的基因型空间比较庞大,对于局部搜索而言,如何解决搜索的局部充分性和计算时间之间的矛盾是我们需要考虑的一个首要问题.

针对作业车间调度问题,我们利用一种局部循环左移的思想构造了基于拉马克进化机制的局部搜索算子,其操作是:随机产生基因位$P_1$,$P_2$,且满足 $1\leqslant P_1<P_2\leqslant N\times M$.对于当前染色体,我们保持 1~$P_1$−1 基因位和$P_2$+1 至$N\times M$基因位上的基因不变,对$P_1$~$P_2$基因位之间的基因,首先从第$P_1$+1 位开始将$P_1$+1~$P_2$基因位的基因左移 1 位,原$P_1$位上的基因放到$P_2$位上,得到一个新个体;然后从原染色体第$P_1$+2 位开始,将$P_1$+2~$P_2$位的基因左移 2 位,原$P_1$~$P_1$+1 位上的基因放到$P_2$−1~$P_2$位上,得到另一个新个体;重复操作,直到从原染色体第$P_2$位开始,将$P_2$位上的基因左移$P_2$−$P_1$位,原$P_1$~$P_2$−1 位的基因放到$P_1$+1~$P_2$位上,得到一个新个体.

例如,待学习的染色体为 211122333,产生的随机数为$P_1$=4,$P_2$=7,那么我们保持 1~3 基因位和 8~9 基因位的基因不变,首先从第 5 位基因开始,把 5~7 位的基因依次左移到 4~6 位上,原第 4 位上的基因放到第 7 位上,这样我们得到的新染色体为 211223133.然后从原染色体的第 6 位基因开始,把 6~7 位的基因依次左移到 4~5 位上,原 4~5 位上的基因放到 6~7 位,操作后得到的新染色体为 211231233.重复上面的操作,直到完成对第 7 位基因的操作.此例中,我们得到的局部搜索空间为 211223133,211231233,211312233.

本文局部搜索算子应用在 JSSP 上能保证搜索空间比较小(最大为 $N\times M$−1),而且在一定程度上能维持染色体上基因的相对顺序保持不变.在搜索过程中可能产生相同的子代个体,我们没有去除,原因为:1) 本文局部搜索算子应用在作业车间调度上,产生相同子代的可能性很小.一般来说,只有在局部基因相同或者局部基因有着重复的序列时(例如局部基因为 123123)才有可能产生相同子代个体;2) 如果要去除相同的个体,那么花费在个体比较上的时间会较大.考虑到以上两点,我们不对相同的子代个体进行处理.

### 2.3 基于记忆库策略和拉马克进化机制的作业车间调度算法

MGA-MBL 算法流程如下:

Step 1. 设定参数并初始化各个子种群:种群规模$P$,子种群个数$Size$,交叉概率$P_c$,变异概率$P_m$,迁移概率$P_t$,初始温度$T_0$,退火系数$r_sa$,停止温度$T$_MIN,按照设定的参数,随机初始化各个子种群,$i$=0;

Step 2. 计算各子种群中个体适应度,提取每个子种群中最优个体组成初始的记忆库,执行 SA 算法优化记忆库;

Step 3. 判断是否达到算法终止条件,若是则输出最优解;否则转 Step 4;

Step 4. 对子种群 $i$ 执行如下的步骤:

1) 个体按交叉概率$P_c$执行第 1.4.2 节中的交叉算子,生成子种群$i$的子代个体,子代个体以概率$P(S_s\rightarrow S_f)=\min\{1,\exp(-\Delta/t)\}$替换父代个体.其中,$\Delta$为子代个体$S_s$和父代个体$S_f$的调度时间之差;

2) 个体按变异概率$P_m$执行第 2.2 节的拉马克局部搜索算子,局部空间中最好个体作为子代个体,子代个体以概率$P(S_{sbest}\rightarrow S_f)=\min\{1,\exp(-\Delta/t)\}$替换父代个体.其中,$\Delta$为局部空间中最好个体$S_{sbest}$和父代个体$S_f$的调度时间之差;

3) $i$=$i$+1;判断 $i$=$Size$?若是则 $i$=0,转 Step 5;否则,转 Step 4;

Step 5. 对子种群 $i$ 执行如下的步骤:

1) 个体以概率$P_t$同记忆库中的个体执行第 1.4.2 节中的交叉操作(记忆库中的个体采用轮盘赌的方式产生),产生的子代个体以概率$P(S_s\rightarrow S_f)=\min\{1,\exp(-\Delta/t)\}$替换子种群$i$中父代个体.其中,$\Delta$为子代个体和子种群$i$中父代个体的调度时间之差;

2) $i$=$i$+1;判断 $i$=$size$?若是转 Step 6,否则转 Step 5;

Step 6. 更新记忆库:每个子种群中的所有个体同记忆库中的个体比较,如果子种群中个体好于记忆库中某个个体,则替换,执行 SA 算法,$i$=0,转 Step 3.

## 3 仿真实验及结果分析

两类数据用来检测MGA-MBL的性能,分别是*FT*和*LA*[13],它们都是广泛应用于调度类问题的标准测试数据.为了使仿真结果具有可比性,我们在实验中对每个方法涉及的参数做了如下选择.多种群遗传算法(MGA):种群规模为 100,子种群个数*size*为 5,交叉概率*pc*为 0.8,变异概率*pm*为 0.1.带有记忆库的多种群遗传算法(MGA+MB):种群规模为 100,子种群个数*size*为 20,交叉概率*pc*为 0.8,变异概率*pm*为 0.1,迁移概率*pt*为 0.8.带有记忆库和模拟退火的多种群遗传算法(MGA+MB+SA):种群规模为 100,子种群个数*size*为 20,交叉概率*pc*为 0.8,变异概率*pm*为 0.1,迁移概率*pt*为 0.8,初温$T_0$为 100,退火系数*r_sa*为 0.9,最低温度*T_MIN*为 0.1.本文算法MGA-MBL:种群规模为 100,子种群个数*size*为 20,交叉概率*pc*为 0.8,变异概率*pm*为 0.1,迁移概率*pt*为 0.8,初温$T_0$为 100,退火系数*r_sa*为 0.9,最低温度*T_MIN*为 0.1.多种群遗传算法采用的是普通通信机制,子种群个数*size*为 20 求得的解远不如子种群个数*size*为 5 时的好.因此,对多种群遗传算法,我们选择子种群个数*size*为 5.结束准则设为适应度评估次数达到 800 000.算法采用C++编写,运行在 3.0 GHz CPU,1G内存的个人电脑上,操作系统为Windows XP Professional.实验独立运行 20 次.

图 2~图 4 分别是对文中 4 种算法所求出的最好解、解平均值和最差解的比较.算法 MGA-MBL 为本文算法,在多种群遗传算法的基础上添加了记忆库、模拟退火和拉马克局部搜索算子.

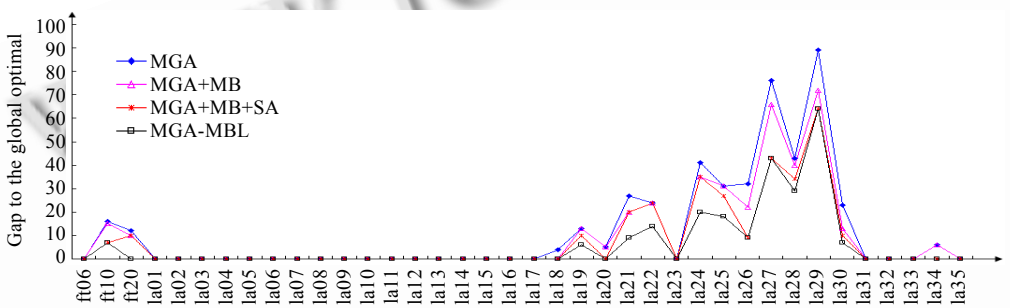


Fig.2 Comparison of the best solutions of algorithms

图 2 算法求出的最好解的比较

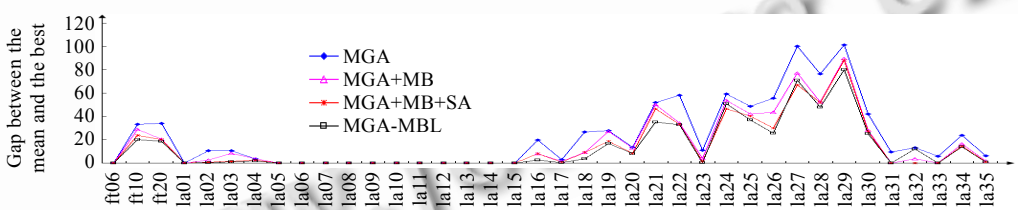


Fig.3 Comparison of mean values of solutions of algorithms

图 3 算法求出的解平均值的比较

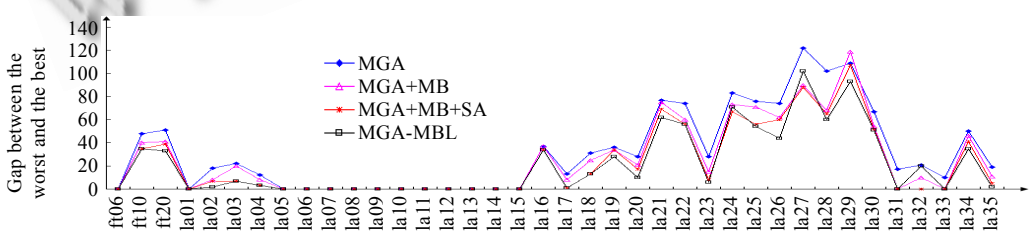


Fig.4 Comparison of the worst solutions of algorithms

图 4 算法求出的最差解的比较

如图 2 所示,对于问题 ft06,la01~la17,la23,la31~la33 和 la35,4 种算法曲线上的点都为 0,说明 4 种算法都可以求出最优解.对于其他问题,本文算法 MGA-MBL 所求出的最好解在其他算法的下方,说明 MGA-MBL 求出的最好解要好于其他算法.

图 3 是 4 种算法求出的解平均值的比较.可以看出,对于比较容易求解的问题 ft06,la01,la05~la15,4 个算法在曲线上的点都为 0,说明这 4 种算法每次运行都可以找到最优解.问题 la24,la27 和 la32,MGA-MBL 求出解的平均值稍差.其余问题,MGA-MBL 求出解的平均值都要好于其他算法.

图 4 是 4 种算法求出的最差解的比较.可以看出,容易求解的问题 ft06,la01,la05~la15,4 种算法求出的最差解与最优解的差值为 0,说明 4 种算法每次运行都可以求出最优解.问题 la24,la27 和 la32,MGA-MBL 稍差,说明有个别最差解与最优解的偏差比较大.其余问题,本文算法 MGA-MBL 求出的最差解都小于其他算法,显示出 MGA-MBL 的稳定性.

表 2 是对记忆库性能的测试.

**Table 2** Performance comparisons of multi-population genetic algorithms with the memory base

**表 2** 具有记忆库的多种群遗传算法的性能比较

| Inst. | Scale | Optimal | Best solutions | | Average | | Mean relative deviation (%) | | Worst solutions | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | MGA | MGA+MB | MGA | MGA+MB | MGA | MGA+MB | | MGA |
| ft06 | 6×6 | 55 | 55 | **55** | 55 | 55 | 0.00 | 0.00 | 55 | 55 |
| ft10 | 10×10 | 930 | 946 | **945** | 963.4 | 959 | 3.59 | 3.12 | 978 | 970 |
| ft20 | 20×5 | 1 165 | 1 177 | **1 175** | 1199.2 | 1185.6 | 2.94 | 1.77 | 1 216 | 1 206 |
| la01 | 10×5 | 666 | 666 | **666** | 666 | 666 | 0.00 | 0.00 | 666 | 666 |
| la02 | 10×5 | 655 | 655 | **655** | 665.6 | 657.4 | 1.62 | 0.37 | 673 | 663 |
| la03 | 10×5 | 597 | 597 | **597** | 607.6 | 605.4 | 1.78 | 1.41 | 619 | 617 |
| la04 | 10×5 | 590 | 590 | **590** | 593.6 | 593.7 | 0.61 | 0.63 | 602 | 598 |
| la05 | 10×5 | 593 | 593 | **593** | 593 | 593 | 0.00 | 0.00 | 593 | 593 |
| la06 | 15×5 | 926 | 926 | **926** | 926 | 926 | 0.00 | 0.00 | 926 | 926 |
| La07 | 15×5 | 890 | 890 | **890** | 890 | 890 | 0.00 | 0.00 | 890 | 890 |
| La08 | 15×5 | 863 | 863 | **863** | 863 | 863 | 0.00 | 0.00 | 863 | 863 |
| La09 | 15×5 | 951 | 951 | **951** | 951 | 951 | 0.00 | 0.00 | 951 | 951 |
| La10 | 15×5 | 958 | 958 | **958** | 958 | 958 | 0.00 | 0.00 | 958 | 958 |
| La11 | 20×5 | 1 222 | 1 222 | **1 222** | 1 222 | 1 222 | 0.00 | 0.00 | 1 222 | 1 222 |
| La12 | 20×5 | 1 039 | 1 039 | **1 039** | 1 039 | 1 039 | 0.00 | 0.00 | 1 039 | 1 039 |
| La13 | 20×5 | 1 150 | 1 150 | **1 150** | 1 150 | 1 150 | 0.00 | 0.00 | 1 150 | 1 150 |
| La14 | 20×5 | 1 292 | 1 292 | **1 292** | 1 292 | 1 292 | 0.00 | 0.00 | 1 292 | 1 292 |
| La15 | 20×5 | 1 207 | 1 207 | **1 207** | 1 207 | 1 207 | 0.00 | 0.00 | 1 207 | 1 207 |
| la16 | 10×10 | 945 | 945 | **945** | 964.8 | 953.2 | 2.10 | 0.87 | 982 | 982 |
| la17 | 10×10 | 784 | 784 | **784** | 787 | 785.3 | 0.38 | 0.17 | 797 | 792 |
| la18 | 10×10 | 848 | 852 | **848** | 874.6 | 857.3 | 3.14 | 1.10 | 879 | 873 |
| la19 | 10×10 | 842 | 855 | **855** | 869.8 | 869.2 | 3.30 | 3.23 | 878 | 876 |
| la20 | 10×10 | 902 | 907 | **907** | 915.4 | 914.8 | 1.49 | 1.42 | 930 | 923 |
| la21 | 15×10 | 1 046 | 1 073 | **1 066** | 1097.8 | 1096.1 | 4.95 | 4.79 | 1 123 | 1 121 |
| la22 | 15×10 | 927 | 951 | **951** | 985.2 | 961.5 | 6.28 | 3.72 | 1 001 | 987 |
| la23 | 15×10 | 1 032 | 1 032 | **1 032** | 1042.8 | 1036.1 | 1.05 | 0.40 | 1 060 | 1 047 |
| la24 | 15×10 | 935 | 976 | **970** | 994.2 | 988.9 | 6.33 | 5.76 | 1 018 | 1 008 |
| la25 | 15×10 | 977 | 1 008 | **1 008** | 1025.8 | 1 019 | 4.99 | 4.30 | 1 053 | 1 048 |
| la26 | 20×10 | 1 218 | 1 250 | **1 240** | 1273.6 | 1261.6 | 4.56 | 3.58 | 1 292 | 1 280 |
| la27 | 20×10 | 1 235 | 1 311 | **1 301** | 1335.4 | 1312.1 | 8.13 | 6.24 | 1 357 | 1 325 |
| la28 | 20×10 | 1 216 | 1 259 | **1 256** | 1292.4 | 1268.3 | 6.28 | 4.30 | 1 318 | 1 284 |
| la29 | 20×10 | 1 152 | 1 241 | **1 224** | 1253.2 | 1241.6 | 8.78 | 7.78 | 1 261 | 1 271 |
| la30 | 20×10 | 1 355 | 1 378 | **1 368** | 1 397 | 1383.3 | 3.10 | 2.09 | 1 422 | 1 409 |
| la31 | 30×10 | 1 784 | 1 784 | **1 784** | 1793.6 | 1 784 | 0.54 | 0.00 | 1 801 | 1 784 |
| la32 | 30×10 | 1 850 | 1 850 | **1 850** | 1863.2 | 1853.5 | 0.71 | 0.19 | 1 871 | 1 860 |
| la33 | 30×10 | 1 719 | 1 719 | **1 719** | 1724.8 | 1 719 | 0.34 | 0.00 | 1 729 | 1 719 |
| la34 | 30×10 | 1 721 | 1 727 | **1 727** | 1744.8 | 1737.3 | 1.38 | 0.95 | 1 771 | 1 767 |
| la35 | 30×10 | 1 888 | 1 888 | **1 888** | 1894.2 | 1 890 | 0.33 | 0.11 | 1 907 | 1 899 |

由表 2 中的数据可以看出,对所有问题,采用了记忆库的多种群遗传算法 MGA+MB 求出的最好解都要好于和等于普通的多种群遗传算法 MGA 求出的解,这一点也可从图 2 中得到证实;而且在多数问题上,MGA+MB 求

出的最好解要比 MGA 的解好很多,例如 la26 和 la29 等.在算法的其他性能上,平均解、最差解和平均相对偏差(平均相对偏差=(平均解问题的最优解)/问题的最优解)上,除了问题 la04 以外,对其他问题 MGA+MB 都要好于 MGA.因此,记忆库对算法的性能起到重要的作用.

表 3 是对模拟退火算法性能的测试.由表中数据可以看出,对所有问题,MGA+MB+SA 求出的最好解都要好于和等于 MGA+MB 求出的解,在问题 la27 上,MGA+MB+SA 的最好解要比 MGA+MB 好很多,这一点也可从图 2 中得到证实.MGA+MB+SA 在解的平均值、最差解和平均相对偏差上也都要好于和等于 MGA+MB,这一点也可从图 3 和图 4 中得到证实.由此可以看出,模拟退火算法是一种解决算法陷入局部最优问题的有效方法,对算法的整体性能起到重要的作用.

**Table 3** Performance comparisons of our algorithm with the improved memory base by SA

**表 3** 用模拟退火方法优化记忆库的性能比较

| Instance | Scale | Optimal | Best solutions | | Average | | Mean relative deviation (%) | | Worst solutions | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | MGA+MB | MGA+MB+SA | MGA+MB | MGA+MB+SA | MGA+MB | MGA+MB+SA | MGA+MB | MGA+MB+SA |
| ft06 | 6×6 | 55 | 55 | **55** | 55 | 55 | 0.00 | 0.00 | 55 | 55 |
| ft10 | 10×10 | 930 | 945 | **937** | 959 | 953.6 | 3.12 | 2.54 | 970 | 965 |
| ft20 | 20×5 | 1 165 | 1 175 | **1 175** | 1185.6 | 1 185 | 1.77 | 1.72 | 1 206 | 1 204 |
| la01 | 10×5 | 666 | 666 | **666** | 666 | 666 | 0.00 | 0.00 | 666 | 666 |
| la02 | 10×5 | 655 | 655 | **655** | 657.4 | 655.7 | 0.37 | 0.11 | 663 | 662 |
| la03 | 10×5 | 597 | 597 | **597** | 605.4 | 598.3 | 1.41 | 0.22 | 617 | 604 |
| la04 | 10×5 | 590 | 590 | **590** | 593.7 | 592.6 | 0.63 | 0.44 | 598 | 593 |
| la05 | 10×5 | 593 | 593 | **593** | 593 | 593 | 0.00 | 0.00 | 593 | 593 |
| la06 | 15×5 | 926 | 926 | **926** | 926 | 926 | 0.00 | 0.00 | 926 | 926 |
| la07 | 15×5 | 890 | 890 | **890** | 890 | 890 | 0.00 | 0.00 | 890 | 890 |
| la08 | 15×5 | 863 | 863 | **863** | 863 | 863 | 0.00 | 0.00 | 863 | 863 |
| la09 | 15×5 | 951 | 951 | **951** | 951 | 951 | 0.00 | 0.00 | 951 | 951 |
| la10 | 15×5 | 958 | 958 | **958** | 958 | 958 | 0.00 | 0.00 | 958 | 958 |
| la11 | 20×5 | 1 222 | 1 222 | **1 222** | 1 222 | 1 222 | 0.00 | 0.00 | 1 222 | 1 222 |
| la12 | 20×5 | 1 039 | 1 039 | **1 039** | 1 039 | 1 039 | 0.00 | 0.00 | 1 039 | 1 039 |
| la13 | 20×5 | 1 150 | 1 150 | **1 150** | 1 150 | 1 150 | 0.00 | 0.00 | 1 150 | 1 150 |
| la14 | 20×5 | 1 292 | 1 292 | **1 292** | 1 292 | 1 292 | 0.00 | 0.00 | 1 292 | 1 292 |
| la15 | 20×5 | 1 207 | 1 207 | **1 207** | 1 207 | 1 207 | 0.00 | 0.00 | 1 207 | 1 207 |
| la16 | 10×10 | 945 | 945 | **945** | 953.2 | 953 | 0.87 | 0.85 | 982 | 979 |
| la17 | 10×10 | 784 | 784 | **784** | 785.3 | 784.8 | 0.17 | 0.10 | 792 | 785 |
| la18 | 10×10 | 848 | 848 | **848** | 857.3 | 857 | 1.10 | 1.06 | 873 | 861 |
| la19 | 10×10 | 842 | 855 | **852** | 869.2 | 860.6 | 3.23 | 2.21 | 876 | 876 |
| la20 | 10×10 | 902 | 907 | **902** | 914.8 | 910.3 | 1.42 | 0.92 | 923 | 919 |
| la21 | 15×10 | 1 046 | 1 066 | **1 066** | 1096.1 | 1092.6 | 4.79 | 4.46 | 1 121 | 1 115 |
| la22 | 15×10 | 927 | 951 | **951** | 961.5 | 960.2 | 3.72 | 3.58 | 987 | 983 |
| la23 | 15×10 | 1 032 | 1 032 | **1 032** | 1036.1 | 1033.5 | 0.40 | 0.15 | 1 047 | 1 040 |
| la24 | 15×10 | 935 | 970 | **970** | 988.9 | 981.6 | 5.76 | 4.98 | 1 008 | 1 002 |
| la25 | 15×10 | 977 | 1 008 | **1 004** | 1 019 | 1017.1 | 4.30 | 4.10 | 1 048 | 1 033 |
| la26 | 20×10 | 1 218 | 1 240 | **1 227** | 1261.6 | 1247.9 | 3.58 | 2.45 | 1 280 | 1 278 |
| la27 | 20×10 | 1 235 | 1 301 | **1 278** | 1312.1 | 1302.1 | 6.24 | 5.43 | 1 325 | 1 323 |
| la28 | 20×10 | 1 216 | 1 256 | **1 250** | 1268.3 | 1268.1 | 4.30 | 4.28 | 1 284 | 1 281 |
| la29 | 20×10 | 1 152 | 1 224 | **1 216** | 1241.6 | 1 240 | 7.78 | 7.64 | 1 271 | 1 259 |
| la30 | 20×10 | 1 355 | 1 368 | **1 365** | 1383.3 | 1381.9 | 2.09 | 1.99 | 1 409 | 1 407 |
| la31 | 30×10 | 1 784 | 1 784 | **1 784** | 1 784 | 1 784 | 0.00 | 0.00 | 1 784 | 1 784 |
| la32 | 30×10 | 1 850 | 1 850 | **1 850** | 1853.5 | 1 850 | 0.19 | 0.00 | 1 860 | 1 850 |
| la33 | 30×10 | 1 719 | 1 719 | **1 719** | 1 719 | 1 719 | 0.00 | 0.00 | 1 719 | 1 719 |
| la34 | 30×10 | 1 721 | 1 727 | **1 721** | 1737.3 | 1735.5 | 0.95 | 0.84 | 1 767 | 1 762 |
| la35 | 30×10 | 1 888 | 1 888 | **1 888** | 1 890 | 1 889 | 0.11 | 0.05 | 1 899 | 1 893 |

表 4 是对拉马克局部搜索算子的验证.由表中的数据可以看出,采用拉马克局部搜索算子的 MGA-MBL 在所有问题上求出的最好解都要好于和等于采用传统变异算子的 MGA+MB+SA,并且也要好于和等于其他的算法即 MGA 和 MGA+MB,这一点可从图 2 中得到证实.特别值得一提的是,对于问题 ft20,只有本文的算法 MGA-MBL 可以求出最优解,而前面几种算法均不能求出.这些实验结果是与我们使用拉马克局部搜索算子分不开的,该算子能够在一代中对同一个体沿着多个方向进行搜索,因此能够在一代中快速搜索到当前解的局部最优

或者是向个体的更好方向搜索.相对于传统的变异算子来说(MGA,MGA+MB 和 MGA+MB+SA 算法都采用传统的变异算子),有效地减小了搜索的盲目性.对于算法的其他指标,即解的平均值、最差解和平均相对偏差,只有问题 la24,la27 和 la32 本文算法 MGA-MBL 不如 MGA+MB+SA,但差距不大;其余问题,本文算法仍有着较好的性能.由此可以看出,采用拉马克局部搜索算子可使算法的性能有较大的改进,特别是在找到最好解方面.

**Table 4** Comparisons of Lamarckian local search operator and conventional mutation operator

**表 4** 拉马克局部搜索算子与传统变异算子的性能比较

| Instance | Scale | Optimal | Best solutions | | Average | | Mean relative deviation (%) | | Worst solutions | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | MGA+MB+SA | MGA-MBL | MGA+MB+SA | MGA-MBL | MGA+MB+SA | MGA-MBL | MGA+MB+SA | MGA-MBL |
| ft06 | 6×6 | 55 | 55 | **55** | 55 | 55 | 0.00 | 0.00 | 55 | 55 |
| ft10 | 10×10 | 930 | 937 | **937** | 953.6 | 950.2 | 2.54 | 2.17 | 965 | 965 |
| ft20 | 20×5 | 1 165 | 1 175 | **1 165** | 1 185 | 1183.6 | 1.72 | 1.60 | 1 204 | 1 198 |
| la01 | 10×5 | 666 | 666 | **666** | 666 | 666 | 0.00 | 0.00 | 666 | 666 |
| la02 | 10×5 | 655 | 655 | **655** | 655.7 | 655.3 | 0.11 | 0.05 | 662 | 657 |
| la03 | 10×5 | 597 | 597 | **597** | 598.3 | 598.2 | 0.22 | 0.20 | 604 | 604 |
| la04 | 10×5 | 590 | 590 | **590** | 592.6 | 591.8 | 0.44 | 0.31 | 593 | 593 |
| la05 | 10×5 | 593 | 593 | **593** | 593 | 593 | 0.00 | 0.00 | 593 | 593 |
| la06 | 15×5 | 926 | 926 | **926** | 926 | 926 | 0.00 | 0.00 | 926 | 926 |
| la07 | 15×5 | 890 | 890 | **890** | 890 | 890 | 0.00 | 0.00 | 890 | 890 |
| la08 | 15×5 | 863 | 863 | **863** | 863 | 863 | 0.00 | 0.00 | 863 | 863 |
| la09 | 15×5 | 951 | 951 | **951** | 951 | 951 | 0.00 | 0.00 | 951 | 951 |
| la10 | 15×5 | 958 | 958 | **958** | 958 | 958 | 0.00 | 0.00 | 958 | 958 |
| la11 | 20×5 | 1 222 | 1 222 | **1 222** | 1 222 | 1 222 | 0.00 | 0.00 | 1 222 | 1 222 |
| la12 | 20×5 | 1 039 | 1 039 | **1 039** | 1 039 | 1 039 | 0.00 | 0.00 | 1 039 | 1 039 |
| la13 | 20×5 | 1 150 | 1 150 | **1 150** | 1 150 | 1 150 | 0.00 | 0.00 | 1 150 | 1 150 |
| la14 | 20×5 | 1 292 | 1 292 | **1 292** | 1 292 | 1 292 | 0.00 | 0.00 | 1 292 | 1 292 |
| la15 | 20×5 | 1 207 | 1 207 | **1 207** | 1 207 | 1 207 | 0.00 | 0.00 | 1 207 | 1 207 |
| la16 | 10×10 | 945 | 945 | **945** | 953 | 947.7 | 0.85 | 0.29 | 979 | 979 |
| la17 | 10×10 | 784 | 784 | **784** | 784.8 | 784.4 | 0.10 | 0.05 | 785 | 785 |
| la18 | 10×10 | 848 | 848 | **848** | 857 | 851.7 | 1.06 | 0.44 | 861 | 861 |
| la19 | 10×10 | 842 | 852 | **848** | 860.6 | 858.8 | 2.21 | 2.00 | 876 | 870 |
| la20 | 10×10 | 902 | 902 | **902** | 910.3 | 909.9 | 0.92 | 0.88 | 919 | 912 |
| la21 | 15×10 | 1 046 | 1 066 | **1 055** | 1092.6 | 1 081 | 4.46 | 3.35 | 1 115 | 1 108 |
| la22 | 15×10 | 927 | 951 | **941** | 960.2 | 960.1 | 3.58 | 3.57 | 983 | 983 |
| la23 | 15×10 | 1 032 | 1 032 | **1 032** | 1033.5 | 1032.4 | 0.15 | 0.04 | 1 040 | 1 038 |
| la24 | 15×10 | 935 | 970 | **955** | 981.6 | 985.7 | 4.98 | 5.42 | 1 002 | 1 006 |
| la25 | 15×10 | 977 | 1 004 | **995** | 1017.1 | 1 014 | 4.10 | 3.79 | 1 033 | 1 031 |
| la26 | 20×10 | 1 218 | 1 227 | **1 227** | 1247.9 | 1243.5 | 2.45 | 2.09 | 1 278 | 1 262 |
| la27 | 20×10 | 1 235 | 1 278 | **1 278** | 1302.1 | 1305.8 | 5.43 | 5.73 | 1 323 | 1 337 |
| la28 | 20×10 | 1 216 | 1 250 | **1 245** | 1268.1 | 1263.8 | 4.28 | 3.93 | 1 281 | 1 276 |
| la29 | 20×10 | 1 152 | 1 216 | **1 216** | 1 240 | 1 232 | 7.64 | 6.94 | 1 259 | 1 245 |
| la30 | 20×10 | 1 355 | 1 365 | **1 362** | 1381.9 | 1380.3 | 1.99 | 1.87 | 1 407 | 1 406 |
| la31 | 30×10 | 1 784 | 1 784 | **1 784** | 1 784 | 1 784 | 0.00 | 0.00 | 1 784 | 1 784 |
| la32 | 30×10 | 1 850 | 1 850 | **1 850** | 1 850 | 1 862 | 0.00 | 0.65 | 1 850 | 1 870 |
| la33 | 30×10 | 1 719 | 1 719 | **1 719** | 1 719 | 1 719 | 0.00 | 0.00 | 1 719 | 1 719 |
| la34 | 30×10 | 1 721 | 1 721 | **1 721** | 1735.5 | 1734.8 | 0.84 | 0.80 | 1 762 | 1 756 |
| la35 | 30×10 | 1 888 | 1 888 | **1 888** | 1 889 | 1888.8 | 0.05 | 0.04 | 1 893 | 1 890 |

表 5 是本文算法 MGA-MBL 同文献[13]中算法的比较.在实验中,我们采用的结束准则与文献[13]一致,即对不同问题采用与文献[13]相同的适应度评估次数作为结束条件,见表 5.可以看出,对所有问题,MGA-MBL 求出的最好解都要好于和等于文献[13]中的方法求出的解.对于问题 ft20,文献[13]的方法与前面提到的算法即 MGA,MGA+MB 和 MGA+MB+SA 一样都未能求出最优解;而本文方法 MGA-MBL 在结束准则与文献[13]相同的条件下却能求出最优解.对比两个算法求出的最好解与最优解的相对偏差(相对偏差=(求出的最好解−问题的最优解)/问题的最优解),对所有问题,本文算法 MGA-MBL 要明显小于和等于文献[13]中的相对偏差.对于有些问题,它们之间的差别还是较明显的.例如问题 la25 和 la28,文献[13]的偏差为 5.2%和 6.3%;而本文算法仅为 1.8%和 1.5%.从这些实验结果的比较中,可以看出,表明了本文的方法 MGA-MBL 在求解作业车间调度问题上更有效.

**Table 5** Performance comparison between our approach and that in Ref.[13]

**表 5** 本文算法与文献[13]的比较

| Instance | Scale | Optimal | Fitness evaluation times ($\times 10^6$) | Best solutions | | Relative deviation (%) | |
|---|---|---|---|---|---|---|---|
| | | | | Ref.[13] | MGA-MBL | Ref.[13] | MGA-MBL |
| ft06 | 6×6 | 55 | 0.1 | 55 | **55** | 0.0 | 0.0 |
| ft10 | 10×10 | 930 | 90.1 | 938 | **937** | 0.9 | 0.8 |
| ft20 | 20×5 | 1 165 | 90.1 | 1 169 | **1 165** | 0.3 | 0.0 |
| la01 | 10×5 | 666 | 0.1 | 666 | **666** | 0.0 | 0.0 |
| la02 | 10×5 | 655 | 0.1 | 655 | **655** | 0.0 | 0.0 |
| la03 | 10×5 | 597 | 50.1 | 604 | **597** | 1.2 | 0.0 |
| la04 | 10×5 | 590 | 0.1 | 590 | **590** | 0.0 | 0.0 |
| la05 | 10×5 | 593 | 0.1 | 593 | **593** | 0.0 | 0.0 |
| la06 | 15×5 | 926 | 0.1 | 926 | **926** | 0.0 | 0.0 |
| la07 | 15×5 | 890 | 0.1 | 890 | **890** | 0.0 | 0.0 |
| la08 | 15×5 | 863 | 0.1 | 863 | **863** | 0.0 | 0.0 |
| la09 | 15×5 | 951 | 0.1 | 951 | **951** | 0.0 | 0.0 |
| la10 | 15×5 | 958 | 0.1 | 958 | **958** | 0.0 | 0.0 |
| la11 | 20×5 | 1 222 | 0.1 | 1 222 | **1 222** | 0.0 | 0.0 |
| la12 | 20×5 | 1 039 | 0.1 | 1 039 | **1 039** | 0.0 | 0.0 |
| la13 | 20×5 | 1 150 | 0.1 | 1 150 | **1 150** | 0.0 | 0.0 |
| la14 | 20×5 | 1 292 | 0.1 | 1 292 | **1 292** | 0.0 | 0.0 |
| la15 | 20×5 | 1 207 | 0.1 | 1 207 | **1 207** | 0.0 | 0.0 |
| la16 | 10×10 | 945 | 50.1 | 946 | **945** | 0.1 | 0.0 |
| la17 | 10×10 | 784 | 20.1 | 784 | **784** | 0.0 | 0.0 |
| la18 | 10×10 | 848 | 20.1 | 848 | **848** | 0.0 | 0.0 |
| la19 | 10×10 | 842 | 10.1 | 842 | **842** | 0.0 | 0.0 |
| la20 | 10×10 | 902 | 50.1 | 907 | **907** | 0.6 | 0.6 |
| la21 | 15×10 | 1 046 | 50.1 | 1 091 | **1 063** | 3.6 | 1.6 |
| la22 | 15×10 | 927 | 50.1 | 960 | **939** | 3.6 | 1.3 |
| la23 | 15×10 | 1 032 | 10.1 | 1 032 | **1 032** | 0.0 | 0.0 |
| la24 | 15×10 | 935 | 10.1 | 978 | **959** | 4.6 | 2.6 |
| la25 | 15×10 | 977 | 10.1 | 1 028 | **995** | 5.2 | 1.8 |
| la26 | 20×10 | 1 218 | 10.1 | 1 271 | **1 218** | 4.4 | 0.0 |
| la27 | 20×10 | 1 235 | 10.1 | 1 320 | **1 272** | 4.0 | 3.0 |
| la28 | 20×10 | 1 216 | 10.1 | 1 293 | **1 234** | 6.3 | 1.5 |
| la29 | 20×10 | 1 152 | 10.1 | 1 293 | **1 204** | 8.2 | 4.5 |
| la30 | 20×10 | 1 355 | 10.1 | 1 368 | **1 355** | 1.0 | 0.0 |
| la31 | 30×10 | 1 784 | 10.1 | 1 784 | **1 784** | 0.0 | 0.0 |
| la32 | 30×10 | 1 850 | 10.1 | 1 850 | **1 850** | 0.0 | 0.0 |
| la33 | 30×10 | 1 719 | 10.1 | 1 719 | **1 719** | 0.0 | 0.0 |
| la34 | 30×10 | 1 721 | 10.1 | 1 753 | **1 721** | 1.9 | 0.0 |
| la35 | 30×10 | 1 888 | 10.1 | 1 888 | **1 888** | 0.0 | 0.0 |

**References**:


[1] Yahyaoui A, Fnaiech F. Recent trends in intelligent job shop scheduling. In: Proc. of the 2006 1st IEEE Int'l Conf. on E-Learning in Industrial Electronics. 2006. 191−195. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4152793

[2] Watanabe M, Ida K, Gen M. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. Computers and Industrial Engineering, 2005,48(4):743−752. [doi: 10.1016/j.cie.2004.12.008]

[3] Krishna K, Ganeshan K, Ram DJ. Distributed simulated annealing algorithms for job shop scheduling. IEEE Trans. on Systems, Man and Cybernetics, 1995,25(7):1102−1109. [doi: 10.1109/21.391290]

[4] Wan GH, Wan F. Job shop scheduling by taboo search with fuzzy reasoning. In: Proc. of the IEEE Int'l Conf. on Systems, Man and Cybernetics. Washington: IEEE, 2003. 1566−1570. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1244635&userTyp=inst

[5] Li JP, Uwe A. Explicit learning: An effort towards human scheduling algorithms. In: Proc. of the 1st Multidisciplinary Int'l Conf. on Scheduling: Theory and Applications. 2003. 240−241. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.6.3123

[6] Chen X, Kong QS, Wu QD. Hybrid algorithm for job-shop scheduling problem. In: Proc. of the 4th World Congress on Intelligent Control and Automation. Shanghai: IEEE, 2002. 1739−1743. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1021380

[7] Cantu-Paz E. Markov chain models of parallel genetic algorithms. IEEE Trans. on Evolutionary Computation, 2000,4(3):216−226. [doi: 10.1109/4235.873233]

[8] Salwani A, Uwe A, Edmund B, Aniza D, Qu R. Investigating a hybrid metaheuristic for job shop rescheduling. In: Randall M, Abbass HA, Wiles J, eds. Proc. of the 3rd Australian Conf. on Artificial Life. Berlin: Springer-Verlag, 2007. 357−368.

[9] Chen H, Flann NS, Watson DW. Parallel genetic simulated annealing: A massively parallel SIMD algorithm. IEEE Trans. on Parallel and Distributed Systems, 1998,9(2):126−136. [doi: 10.1109/71.663870]

[10] Essafi I, Mati Y, Dauzère-Pérès S. A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Computers and Operations Research, 2008,35(8):2599−2616.

[11] Liand KH, Yao X, Newton C. Lamarckian evolution in global optimization. In: Proc. of the 26th Annual Conf. of the IEEE on Industrial Electronics Society. Nagoya: IEEE, 2000. 2975−2980. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=972471

[12] Geng SH. Job shop scheduling problem resolving based on natural computation [MS. Thesis]. Xi'an: Xidian University, 2005 (in Chinese with English abstract).

[13] Binato S, Hery WJ, Loewenstern DM, Resende MGC. A GRASP for job shop scheduling. AT&T Labs Research Technical Report: 00.6.1, Kluwer Academic Publishers, 2000. 58−79.

**附中文参考文献**:

[12] 耿树会.基于自然计算的作业车间调度问题求解[硕士学位论文].西安:西安电子科技大学,2005.

**夏柱昌**(1983－),男,辽宁鞍山人,硕士,主要研究领域为自然计算及其应用.

**公茂果**(1979－),男,博士,副教授,CCF 高级会员,主要研究领域为进化计算,人工免疫,智能图像处理,模式识别.

**刘芳**(1963－),女,教授,博士生导师,CCF 高级会员,主要研究领域为网络智能信息处理,智能图像处理,模式识别.

**咸玉涛**(1981－),男,博士,讲师,主要研究领域为并行计算,人工免疫.