求解置换流水车间调度问题的混合蚁群算法

周 鹏[1,2]
ZHOU Peng[1,2]

1.湖北汽车工业学院 计算机系,湖北 十堰 442002
2.西北工业大学 计算机学院,西安 710072
1.Department of Computer Science,Hubei University of Automotive Technology,Shiyan,Hubei 442002,China
2.School of Computer Science,Northwestern Polytechnical University,Xi'an 710072,China
E-mail:pengpeng1193@163.com

ZHOU Peng.Hybrid ant colony algorithm for permutation flow shop scheduling problem.Computer Engineering and Applications,2009,45(17):191-193.

Abstract: To the problem that Max-Min Ant System (MMAS) plunges into local best situation easily when it is applied to Permutation Flow Shop Scheduling Problem(PFSP),a hybrid ant colony algorithm is proposed by incorporating the pheromone matrix mutating and restarting rule suggested in Best-Worst Ant System(BWAS).To introduce diversity in the search process,the pheromone matrix suffers mutations and a restart when the search process stops.Contrastive experiments on benchmark problems show that the hybrid algorithm has a better capability of global optimization than traditional ant colony algorithm.
Key words: permutation flow shop scheduling problem;ant system;pheromone mutatio

摘 要:针对最大—最小蚂蚁系统在解决置换流水车间调度问题时易陷入局部最优的问题,引入最好—最差蚂蚁系统中的信息素变异和重置规则,提出了一种混合蚁群算法。使信息素矩阵变异并在搜索过程停滞时重置信息素矩阵以在搜索过程中引入多样性。在基准问题集上的对比实验表明,该算法比传统的蚁群算法具有更好的搜索全局最优解的能力。
关键词:置换流水车间调度问题;蚂蚁系统;信息素变异
DOI:10.3778/j.issn.1002-8331.2009.17.058 **文章编号**:1002-8331(2009)17-0191-03 **文献标识码**:A **中图分类号**:TP18

蚁群优化(Ant Colony Optimization,ACO)算法是一种元启发式算法,成功地解决了多种组合优化问题[1]。置换流车间调度问题(Permutation Flow Shop Scheduling Problem,PFSP)是一个著名的NP-难问题,1998年Stützle首次将名为最大-最小蚂蚁系统(Max-Min Ant System,MMAS)的ACO算法用于PFSP上[2],此后,其他一些ACO算法也被用于PFSP。如Ying和Liao使用了蚁群系统[3](Ant Colony System,ACS),Rajendran和Ziegler提出了一种信息素总和规则,并在此基础上提出名为M-MMAS和PACO的两种ACO算法[4]。这些方法只在迭代以来最优路径上增加信息素,这样虽然加快了收敛速度,但同时也会导致算法陷入局部最优。Cordón等针对旅行商问题(traveling salesman problems,TSP)提出了一种最好-最差蚂蚁系统(best-worst ant system,BWAS),其中使用了一种信息素矩阵变异和重置机制,可增强算法跳出局部最优解的能力[5]。虽然BWAS尚未被用于PFSP上,但从它在TSP的表现上来看,其用于跳出局部最优的机制在PFSP上也可能是有效的。结合MMAS和BWAS各自的优点,提出了一种混合蚁群算法BW-MMAS,并在典型问题(benchmark)集上进行了测试。

1 置换流车间调度问题

PFSP可简要描述为[6]:n个工件在m台机器上进行加工,每个工件在各机器上加工顺序相同且在每台机器上仅加工一次,每台机器加工的各工件的顺序相同且在某一时刻只能够加工一个工件,各工件在各机器上所需的加工时间已知,要求得到某项指标最优。

令t_{ij}为工件i在机器j上的加工时间,T_i为工件i的加工完毕时间。不失一般性,假设各工件按机器1至m的顺序进行加工,令$\pi=(\sigma_1,\sigma_2,\cdots\sigma_n)$为所有工件的一个排列(置换),$T_{ij}$为工件$i$的工序$j$在机器$j$上的加工完成时间。

$$\pi^*=\arg\{f(\pi)=T_{\sigma_n,m}\}\rightarrow\min \quad (1)$$

式(1)中给出的指标为找到一个排列π^*,以最小化最大完工时间(makespan)。本文讨论的指标均为makespan指标。

2 MMAS和BWAS

2.1 MMAS[2]

2.1.1 初始解的获得

利用Nawaz等在1983年提出的NEH启发式方法得到初

基金项目:湖北省教育厅科研项目(No.B20082304)。
作者简介:周鹏(1975-),男,博士研究生,讲师,主研方向:人工智能,多媒体智能决策。
收稿日期:2009-01-22 **修回日期**:2009-02-16

始解,这样可保证 MMAS 的解不弱于 NEH。NEH 方法的基本思想是[6]赋予总加工时间越长的工件越大的在排列中的插入优先权,然后将前两个工件进行最优调度,进而依次将剩余工件逐一插入到已调度的工件排列中的某个位置,使得调度指标最小,直到所有工件调度完毕,从而得到一个次优调度。

2.1.2 节点选择

引入虚工件 0 作为蚂蚁的出发节点,信息素矩阵中的元素 τ_{ij} 表示工件 i 在排列中位置 j 上的期望程度。为选择位置 j 上的工件 i,蚂蚁以概率 P_0 进行最优可能选择,即在未排序的工件中选择 τ_{ij} 最大者置于位置 j 上。以概率 $1-P_0$ 进行按式(2)的概率分布选择位置 j 上的工件 i,p_{ij} 表示工件 i 在位置 j 上的概率。

$$p_{ij}=\begin{cases}\dfrac{\tau_{ij}}{\sum_{i\text{为未排序的工件}}\tau_{ij}} & \text{若工件 } i \text{ 未排序}\\ 0 & \text{否则}\end{cases} \tag{2}$$

为减少选择时间,式(2)中 i 的范围可缩小到未排序节点集中尺寸为 $cand$ 的一个候选解集中。

2.1.3 局部搜索和信息素更新

使用 first-improvement 插入方法进行局部搜索,即每当某只蚂蚁选择了位置 j 上的工件 σ_j 时,将该工件插入到该蚂蚁的局部路径中位置 1 到 $j-1$ 上以寻找是否存在得到改善的局部路径,如存在,则以改善程度最大的局部路径作该蚂蚁当前的局部路径。由于局部搜索的计算代价较大,MMAS 只采用一只蚂蚁。

对信息素矩阵中所有元素 τ_{ij},信息素更新公式为式(3):

$$\tau_{ij}^{new}=\rho\tau_{ij}^{old}+\Delta\tau_{ij} \tag{3}$$

式(3)中,ρ 为信息素残留率,$1-\rho$ 为信息素蒸发率,$0\leqslant\rho\leqslant1$。令 C_{best} 为迭代以来最优解(置换),$|C_{best}|$为 C_{best} 的 makespan 值。如果 C_{best} 中位置 j 上工件不为 i,则 $\Delta\tau_{ij}$ 为 0,否则 $\Delta\tau_{ij}=1/|C_{best}|$。信息素取值区间为$[\tau_{min},\tau_{max}]$。

2.2 BWAS

BWAS[5]最初是针对 TSP 问题提出的,经文献检索尚未发现将之用于 PFSP 上。BWAS 中采用了不同于 MMAS 和 ACS 的信息素更新、变异和重置机制。

信息素更新:对迭代以来最优蚂蚁进行信息素正增强,对本次迭代中最差的蚂蚁进行信息素负增强。

信息素变异:以概率 P_m 进行信息素变异,在搜索早期,信息素变异比较小,在搜索后期,当搜索区域越来越集中时,增大信息变异量以扩大搜索区域,从而达到增强找到全局最优解的能力的目的。

$$\tau_{ij}^{new}=\begin{cases}\tau_{ij}^{new}+mut(it,\tau_{thresold}) & a=0\\ \tau_{ij}^{new}-mut(it,\tau_{thresold}) & a=1\end{cases} \tag{4}$$

式(4)中,$a\in\{0,1\}$为一随机数,it 为当代迭代次数,$\tau_{threshold}$ 为至今最优路径上信息素的均值,mut 为一变异算子。针对 PFSP 的具体 mut 和 $\tau_{threshold}$ 在第 3 章中引入。

信息素矩阵重置:当算法陷入局部最优时,进行信息素矩阵重置。通过本次迭代最优解 C_{lb} 和迭代以来最优解 C_{best} 的相似程度来决定重置时机。

3 BW-MMAS

结合 MMAS 初始解的质量好、收敛速度快和 BWAS 跳出局部最优能力强的优点,提出一种混合算法BW-MMAS。

3.1 BW-MMAS 的要点

3.1.1 初始解的获得

使用 NEH 能快速获得次优调度,该次优调度与实际最优调度较为接近,在该初始解的基础上进行进一步搜索,可快速得到质量更好的次优或最优解。BW-MMAS 和 MMAS 一样使用 NEH 方法来获得初始解,将之作为初始 C_{best},并初始化 $\tau_{max}=1/[(1-\rho)*|C_{best}|]$,$\tau_{min}=\tau_{max}/5$。

3.1.2 节点选择和局部搜索

MMAS 中采用期望度来选择节点。由于算法的随机特性,即使工件 i 具有较高的信息素值 τ_{ij},但是算法却有可能将一个信息素值 τ_{hj} 较低的工件 h 分配到了位置 j 上,这种情况有可能导致非常差的调度的出现[1]。PFSP 中希望在这种情况发生时,i 应该有一个较高的概率被分配到 j 邻近的位置上,Rajendran 等[4]提出的信息素总和原则巧妙的解决了这一问题。BW-MMAS 中,采用信息素总和原则来进行节点选择。

当要选择位置 j 上的工件时,定义工件 i 的信息素总和 Γ_{ij} 为式(5),以概率 P_0 在未排序的工件中选择信息素总和 Γ_{ij} 最大的工件 i 放置在位置 j 上。以概率 $1-P_0$ 按式(6)中定义的概率 p_{ij} 来选择位置 j 上的工件 i,对于 n 个工件的 PFSP,BW-MMAS 采用了和 MMAS 一样的 $P_0=(n-4)/n$。

$$\Gamma_{ij}=\sum_{k=1}^{j}\tau_{ik} \tag{5}$$

$$p_{ij}=\begin{cases}\dfrac{\Gamma_{ij}}{\sum_{i\text{为未排序的工件}}\Gamma_{ij}} & \text{若工件 } i \text{ 未排序}\\ 0 & \text{否则}\end{cases} \tag{6}$$

BW-MMAS 采用和 MMAS 相同的 first-improvement 插入方法来进行局部搜索,也只采用一只蚂蚁。

3.1.3 信息素变化

BW-MMAS 采用和 BWAS 类似的信息素变化机制。对 n 个工件的 PFSP,信息素变化步骤如下:

(1)如果 C_{lb} 优于 C_{best},则令 $C_{best}=C_{lb}$,更新 $\tau_{max}=1/[(1-\rho)*|C_{best}|]$,$\tau_{min}=\tau_{max}/5$,并转(2)。否则,依次比较 C_{lb} 和 C_{best} 各个位置上的工件是否相同,相同工件数计为 n'。如 $n'/n\geqslant P_n$,则重置信息素矩阵中所有元素为 τ_{max},转(4);如 $n'/n<P_n$,转(2)。

(2)首先按式(3)更新信息素矩阵中所有信息素。然后按 $j=1,2,\cdots,n$,依次比较 C_{lb} 中位置 j 上工件 i 是否等于 C_{best} 中位置 j 上工件 i',如果 $i\neq i'$,则令 $\tau_{ij}^{new}=\rho\tau_{ij}^{old}$。转(3)。

(3)对信息素矩阵中每个信息素 τ_{ij},都以概率 P_m 按式(4)变异,并将 τ_{ij} 限制在区间$[\tau_{min},\tau_{max}]$内。转(4)。$\tau_{threshold}$ 和 mut 分别表示为式(7)(8)。式(8)中,it_r 为上次信息素矩阵重置时的迭代代数,Nit 为算法设定的总迭代次数,σ 为一系数。

$$\tau_{threshold}=\frac{\sum_{C_{best}\text{中位置 }j\text{ 上工件为 }i}\tau_{ij}}{n} \tag{7}$$

$$mut(it,\tau_{threshold})=\frac{it-it_r}{Nit-it_r}*\sigma*\tau_{threshold} \tag{8}$$

(4)完成本次迭代中信息素变化过程

3.2 BW-MMAS 算法伪码

```
BW-MMAS
  begin
  按 3.1.1 节中的 NEH 算法求出初始解,计算出 τmax 和 τmin,并初
```

始化信息素矩阵。

Repeat *Nit* 次
begin
初始化蚂蚁路径,其中仅含一个虚节点 0。
for(k=1;$k\leqslant n$;k++)
begin
按 3.1.2 节中方法寻找位置 k 上工件。
end.
按 3.1.3 中步骤进行信息素矩阵的重置、更新、变异。
达到预设终止条件,则算法终止。
end.
end.

4　实验结果

将 BW-MMAS、MMAS[2]、ACS[3]用于名为 Rec 类的 benchmark 问题上,进行比较实验。在数据集中的每个子问题上,三种算法各执行 5 次,取 5 次实验中最好的结果作为该算法在该子问题上的实验结果,每个子问题上,各迭代 2 500 次。MMAS 和 BW-MMAS 的公共参数选取为 ρ=0.75,$cand$=5。BW-MMAS 特有参数 P_n=0.95,P_m=0.3,σ=4。ACS 中蚂蚁数为 20,全局和局部蒸发系数 α=ρ=0.1,$q0$=0.1,β=2。三种算法均用 Visual C#2008 实现,并在配有 3 G 内存,2.4 GHz 的 Intel 双核 CPU 的计算机上运行。

Rec 类问题含有 7 个规模的 21 个子问题,问题规模(工件数/机器数)从 20/5 到 75/20。

从图 1 中可以看出,随着迭代次数的增加,MMAS 陷入了局部最优,而 BW-MMAS 仍在试图扩展搜索范围,并且找到了比 MMAS 更好的解。

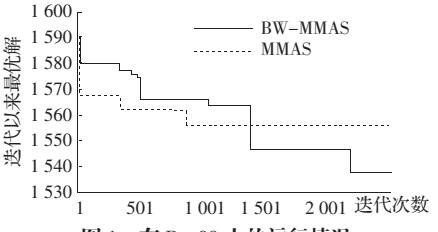

图 1　在 Rec09 上的运行情况

Rec 类问题上的实验结果见表 1。引入相对误差百分率(relative percentage deviation,RPD)来衡量解的质量。RPD=100*(C_g-C_b)/C_b,C_g 为实验中求得的最好解,C_b 为该问题的已知最优解(Rec37-Red41 上为最优解的上界),C_b 可在文献[6]中查得。

从表 1 中可以看出,BW-MMAS 求得的解的质量要优于 MMAS,远优于 ACS。在 21 个子问题中,BW-MMAS 在 17 个子问题上取得了优于另两种算法的解,在 5 个子问题上取得了最优解。从运行过程来看,多数子问题上在取得同等质量解的前提下,BW-MMAS 取得解的时间要小于另两种算法。

5　结语

在 MMAS 的基础上通过引入信息素变化机制,提出了一种混合算法 BW-MMAS,增强了 MMAS 的全局搜索能力。由于这两种算法的局部搜索计算代价较大,使得当问题规模较大时,计算时间大量增加。如何改进局部搜索算法的效率,在保证解的质量的前提下结合 PFSP 的 block 性质,对劣解空间进行剪枝处理,从而缩小搜索空间,提高计算速度,是下一步的一个研究方向。此外,与目前在 PFSP 上最好的高性能禁忌算法相比,ACO 的性能还有一定差距,尚未达到一流算法的水平[1]。将 ACO 与其他算法相结合,是另一个研究方向。

表 1　Rec 类问题上运行结果

问题	BW-MMAS		MMAS		ACS	
	解	RPD	解	RPD	解	RPD
Rec01	1 250	0.24	1 249	0.16	1 271	1.92
Rec03	1 109[ab]	0.00	1 110	0.09	1 111	0.18
Rec05	1 242[ab]	0.00	1 243	0.08	1 266	1.93
Rec07	1 568[a]	0.13	1 584	1.15	1 697	8.37
Rec09	1 538[a]	0.07	1 557	1.30	1 611	4.81
Rec11	1 431[ab]	0.00	1 436	0.35	1 452	1.47
Rec13	1 939[a]	0.47	1 957	1.40	1 969	2.02
Rec15	1 954[a]	0.21	1 960	0.51	1 993	2.21
Rec17	1 902[ab]	0.00	1 935	1.74	2 004	5.36
Rec19	2 122[a]	1.39	2 127	1.62	2 163	3.34
Rec21	2 046[a]	1.44	2 050	1.64	2 054	1.83
Rec23	2 042	1.54	2 027	0.80	2 091	3.98
Rec25	2 576	2.51	2 567	2.15	2 622	4.34
Rec27	2 420[a]	1.98	2 421	2.02	2 461	3.71
Rec29	2 340[a]	2.32	2 354	2.93	2 378	3.98
Rec31	3 130[a]	2.79	3 140	3.12	3 174	4.24
Rec33	3 140[a]	0.83	3 151	1.19	3 163	1.57
Rec35	3 277[b]	0.00	3 277	0.00	3 355	2.38
Rec37	5 160[a]	4.22	5 182	4.67	5 366	8.38
Rec39	5 230[a]	2.81	5 249	3.18	5 429	6.72
Rec41	5 173[a]	4.29	5 226	5.36	5 317	7.20
average		1.30		1.69		3.81

注:[a] 优于 MMAS 和 ACS 的解;[b] 达到最优解

参考文献:

[1] Dorigo M,Stützle T.蚁群优化[M].张军,胡晓敏,罗旭耀,译.北京:清华大学出版社,2007.
[2] Stützle T.An ant approach for the flow shop problem[C]//Aachen:Proceedings of the sixth European Congress on intelligent Techniques and Soft Computing,1998,3:1560-1564.
[3] Ying K C,Liao C J.An ant colony system for permutation flow-shop sequencing[J].Computers & Operations Research,2004,31(5):791-801.
[4] Rajendran C,Ziegler H.Ant-colony algorithms for permutation flow-shop scheduling to minimize makespan/total flowtime of jobs[J].European Journal of Operational Research,2004,155(2):426-438.
[5] Cordón O,de Viana,Herrera F,et al.Analysis of the best-worst ant system and its variants on the TSP[J].Mathware and Soft computing,2002,9(2-3):177-192.
[6] 王凌.车间调度及其遗传算法[M].北京:清华大学出版社,2003.