文章编号: 1001-0920(2008)08-0850-07

求解作业排序问题的一种改进修复约束满足算法

上官春霞[1], 周　泓[1], 师瑞峰[2], 吴　旸[1]
(1. 北京航空航天大学 经济管理学院, 北京 100191; 2. 华北电力大学 自动化系, 北京 102206)

摘　要: 修复约束满足算法(修复法)是在完整初始解的基础上不断对变量进行修复,最终得到可行解. 对此,提出一种求解 flow shop 排序问题的改进修复法(IRCS_WT),通过采用新的变量表达方式,设计了一种以启发式优化规则为指导的变量选择算法(LWT),并采用一种变量互换算法(LTEE)保证算法的全局搜索性能. 将新算法应用于 31 个标准算例,与传统算法及遗传算法的优化结果进行比较,结果表明在相同运算时间下改进算法具有明显的优越性.

关键词: 约束满足; 修复法; flow shop 排序问题; 加权总延误
中图分类号: TP391; TP301.6　　**文献标识码**: A

Improved repair-based constraint satisfaction method for flow shop scheduling

SHANGGUAN Chun-xia[1], *ZHOU Hong*[1], *SHI Rui-feng*[2], *WU Yang*[1]
(1. School of Economics and Management, Beihang University, Beijing 100191, China; 2. Department of Automation, North China Electric Power University, Beijing 102206, China. Correspondent: SHANGGUAN Chun-xia, E-mail: sgchunxia@126.com)

Abstract: Repair-based constraint satisfaction method is one of the two main categories of constraint satisfaction (CS) algorithms. It starts with an arbitrary completed solution, and then resolves conflicts by local repair. An improved repair-based constraint satisfaction (IRCS_WT) method for flow shop scheduling is proposed. In IRCS_WT, a heuristic rule of largest weighted tardiness priority (LWT) is proposed for variables ordering procedure, and a new heuristic of largest tardy and early variables exchanging (LTEE) is designed to escape from local optima. Computational experiments of 31 flow shop scheduling problems are conducted to compare the IRCS_WT with other two heuristic methods, extended NEH and filtered beam search (FBS), as well as genetic algorithm (GA). The results show that IRCS_WT significantly outperforms the competitors with the same running time.

Key words: Constraint satisfaction; Repair-based method; Flow shop scheduling; Total weighted tardiness

1 引　　言

约束满足算法可分为构造法和修复法两类[1-3]. 构造法以回溯算法为基础,通过不断扩充一致的部分解得到一致的完整解,该方法适用于中小规模问题[2,3]. 修复法(也称修复约束满足算法)则首先提出一个可能含有冲突的完整解;然后不断地修复某些变量的赋值以减少冲突;最终得到一个无冲突的可行解[4,5]. 此方法通常对大规模问题非常有效[6].

1992 年,Minton 等人受人工神经网络的启发,最先提出一种修复算法以求解约束满足问题. 该算法分为两个阶段:首先随机构造一个不可行的完整解;然后用最小冲突启发式算法对该完整解中的不一致变量逐一进行修复以减少冲突,直至找到可行解[7]. 该算法效率很高,但不能保证解的全局性及算法的完整性. 之后,20 世纪 90 年代涌现了大量关于修复算法的研究成果,包括算法改进以及应用研究. 对修复算法的改进研究主要集中在如何提高搜索效率以及如何跳出局部极值方面. 1993 年,Morris 提出一种跳出局部极值的方法(BA). BA 在修复阶段使用 next-descent 为选中变量赋值,若修复陷入局部极值,BA 记录此极值中的 no-good,并增加其权重,此时当前解不再是局部极值[8]. 同年,文献[2]研究了 GENET 方法. 1994 年,Yugami 等人提出用值传播方法来改进修复法,从而使其能够跳出局部极

收稿日期: 2007-05-22; **修回日期**: 2007-10-31.
基金项目: 国家自然科学基金项目(70771003,70521001); 新世纪优秀人才支持计划项目(NCT040175).
作者简介: 上官春霞(1980—),女,山西阳城人,博士生,从事生产计划优化、约束满足算法等研究; 周泓(1965—),男,山东潍坊人,教授,博士生导师,从事生产系统、物流系统的优化与仿真等研究.

值[9]. 1995～1998年,Richards等人提出了learning-by-merging方法,对no-good进行记录,以避免陷入局部极值,并提高算法效率. 该方法对人工智能中的SAT等问题非常有效[10]. 1997年,田盛丰将构造法中的前向检验算法与修复法相结合,提出了一种改进的修复算法,并对规模分别为20,40,60和80的N皇后问题进行了实验. 结果表明,该算法在运行效率上有明显的改善[1]. 1998年,Wallace等人提出了一种可以跳出局部极值的改进修复法. 同样采用两个阶段,但在修复阶段定义了参数α:对某一选中变量进行修复时,以α%概率在其值域中随机选择赋值,以(1 - α%)概率按Minton提出的最速下降法为其选择使冲突最小的赋值. 这一改进使修复算法效率和效果得到提高[11].

此外,很多启发式方法也被引入修复算法以克服局部极值问题. 如模拟退火和禁忌搜索[12,13]. 2005年,Gersman等人将修复法与SVM方法相结合,对资源约束下的项目排序问题进行实验分析,结果表明算法效果有所提高,但时间性能有所下降[14].

目前,对修复法的研究多数集中在人工智能领域,如N皇后问题、SAT问题等;而对于组合优化领域的经典问题,如flow shop和job shop排序问题则关注较少,这也阻碍了修复法进一步的推广应用. 本文针对flow shop问题建立了一种改进的修复法,以尝试其在排序领域的应用,取得了满意的效果.

2　问题定义

一般flow shop问题可描述为:n项工艺路线相同的待加工作业$J = \{1,2,\cdots,n\}$,通过m台不同的机器$M = \{1,2,\cdots,m\}$进行处理,各项作业在每台机器上加工且仅加工一次. 排列型flow shop问题是指所有作业以相同的次序依次通过机器$M = [1,2,\cdots,m]$加工,即仅考虑作业排列顺序对结果的影响. 一般flow shop问题的解空间规模为$(n!)^m$,而排列型flow shop问题解空间规模为$n!$[15].

排列型flow shop问题是实际生产中常见的一种生产形式,如生产流水线等. 实际生产中,延迟交货通常会使企业蒙受包括罚金以及客户满意度下降等损失,因而避免延误成为生产计划中的主要经济目标之一. 本文研究目标函数为最小化加权总延误的flow shop问题. 按照约束满足问题(CSP)的描述方式,一个CSP问题由(X,D,C)三元组表示. 其中:X为变量集合,D为各变量值域的集合,C则定义了变量之间的约束关系. 该问题可描述为如下带有目标函数的CSP模型:

$$\min_{FS} \sum_{j=1}^{n} w_j T_j, \tag{1}$$

$$\text{s.t. } t_{ij}^S + p_{ij} \le t_{(i+1)j}^S,$$

$$i = 1,2,\cdots,m-1,\ j = 1,2,\cdots,n, \tag{2}$$

$$t_{ij}^S + p_{ij} \le t_{ik}^S + G*(1 - x_{jk}),$$

$$i = 1,2,\cdots,m,\ j,k = 1,2,\cdots,n, \tag{3}$$

$$t_{mj}^S + p_{mj} \le d_j,\ j = 1,2,\cdots,n, \tag{4}$$

$$t_{ij}^S \ge 0,\ i = 1,2,\cdots,m,\ j = 1,2,\cdots,n, \tag{5}$$

$$x_{jk} = \{0,1\},\ j,k = 1,2,\cdots,n. \tag{6}$$

以o_{ij}表示作业j在机器i上第i道工序的加工操作,模型中的参数及变量符号含义如下:

问题参数:n为作业数,m为机器数,FS为可行排序集合,p_{ij}为操作o_{ij}的加工时间,w_j为作业j的权重,d_j为作业j的交货期,G为任意大的正数.

决策变量:t_{ij}^S为操作o_{ij}的开工时间,x_{jk}为序变量,有

$$x_{jk} = \begin{cases} 1, & \text{若某一可行序中 } k \text{ 紧接在 } j \text{ 之后;} \\ 0, & \text{否则.} \end{cases}$$

模型中,式(1)表示优化的目标函数为加权总延误,其中$T_j = \max(0, t_{mj}^S + p_{mj} - d_j)$为作业$j$的延误时间. 求解过程中,式(2)为硬约束,表明所有作业的工艺路线均遵从机器1～m的顺序;硬约束式(3)定义了机器的能力约束,即在某一个时刻,机器i上最多只能有一项操作在加工;式(4)为软约束,可以不满足,不满足时称为存在"冲突".

根据文献[15]中关于排序问题的复杂性分类可知,以加权总延误为目标的flow shop问题也是NP难问题.

3　基于修复的约束满足算法

3.1　基本算法

修复法的核心是在不一致的完整解中随机选择冲突变量,以尽量减少冲突的原则为该变量赋值. 基本的修复法分为两个阶段,其基本过程如下:

1) 初始化. 随机为变量进行赋值,得到可行/不可行的完整初始解.

2) 修复. 将所有变量分为两个集合:Done和Left,Done包含所有无冲突的变量,Left包含有冲突的变量. 每次迭代在Left集合中随机选择冲突变量,在不与Done中变量冲突的前提下以最小冲突法为该变量赋值.

3.2　改进的加权总延误修复法

为了将修复法用于优化目标为加权总延误的flow shop排序问题,本文提出一种改进的加权总延误修复法(IRCS_WT).

变量的表示方法对算法设计很重要,文献[7]和文献[12]中的修复法均使用t_{ij}^S作为排序问题的

变量,其值域由操作的最早开始时间与最晚开始时间确定上下界. 针对排列型 flow shop 问题的特点——所有作业均以同样的序通过所有机器,本文用序位作为问题的变量,可以大大减少变量个数,从而降低算法的复杂性. 具体来讲,每一个作业 j 对应一个变量 s_j,以表示作业 j 在整个序中的位置. s_j 的值域为 $D_j = [1, n]$,为了保证解的可行性,所有变量的取值均不重复. 当需要调整某一变量的取值时,按照调整变量优先的原则,将其他发生重复取值的变量取值进行顺推(向前或向后),直至无重复值发生. 顺推的方法是,当调整变量的取值由小变大时,发生重复的变量取值向前顺推,依次将其取值减 1;当调整变量的取值由大变小时,重复变量的取值向后顺推,依次将其取值增 1.

例如,当 $n = 5$ 时,问题含 5 个变量:s_1, s_2, s_3, s_4, s_5,每一变量的值域均为 $D_j = \{1,2,3,4,5\}$. 当 5 个变量分别取不同的值,如 $s_1 = 1, s_2 = 2, s_3 = 3, s_4 = 4, s_5 = 5$ 时,它们构成了一个可行解 $FS = \{J_1, J_2, J_3, J_4, J_5\}$. 当某一变量的值需要调整时,如 s_4 的取值需要由 $s_4 = 4$ 调整为 $s_4 = 2$,则依次将 s_2 和 s_3 的取值分别后推为 $s_2 = 3, s_3 = 4$,s_1 和 s_5 取值保持不变,此时各变量的取值构成新的可行序 $FS' = \{J_1, J_4, J_2, J_3, J_5\}$.

针对前面的模型,上述变量表示方法可以确保硬约束(2)和(3)总能满足,因此计算冲突时可以不必考虑. 软约束(4)成为计算变量冲突情况的主要约束,可用变量对约束(4)的加权违反程度来计算变量冲突情况,目标函数则用于评价解的好坏.

IRCS_WT 算法流程如图 1 所示.

3.2.1 初始化方法:启发式 ENEH

基本修复算法的初始化是通过为变量随机赋值来获得,这使得算法初始解随机性很大,无法保证算法的搜索效率. 对此,本文通过对 NEH 启发式[17]的拓展,提出一种基于启发式 ENEH 的初始化方法以提高算法的初始解质量.

NEH 启发式方法是针对最小化总完工时间(makespan)排序问题的,其分为两个步骤:

1) 为每一作业 $j \in J$,计算 $P_j = \sum_{i}^{m} p_{ij}$,按照 P_j 不增的顺序排列所有作业,所得序记为 S_P,令结果序 $S = \Phi$;

2) 若 S_P 非空,则重复如下操作:从 S_P 中取出排在首位的作业 j,将其试验性地插入序 S 中的所有可能位置,选取最小 makespan 的序替换 S,令 $S_P = S_P - \{j\}$.

本文提出的 ENEH 算法以最小化加权总延误为优化目标,其算法步骤可描述为:

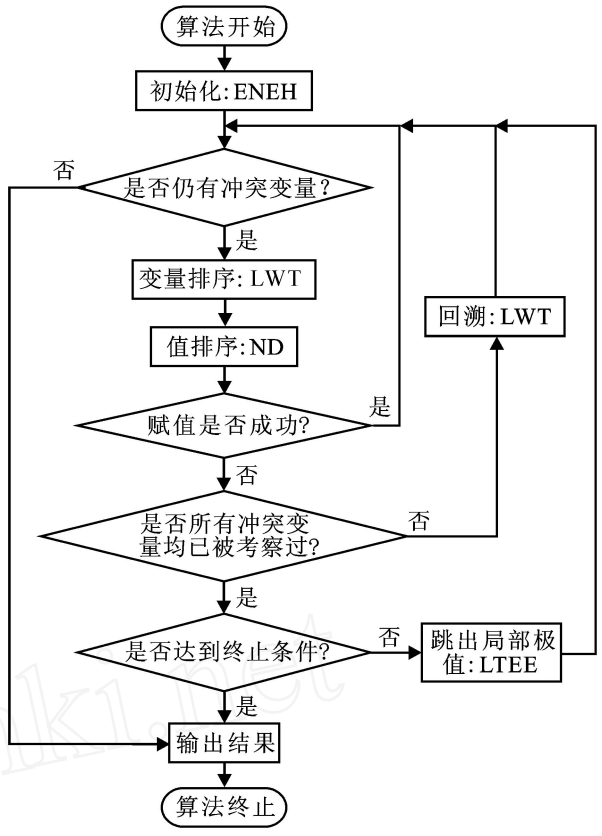

图 1 IRCS_WT 算法流程图

1) 对所有 $j \in J$,计算 $\text{index}_j = d_j / w_j$,按照 index_j 不减的顺序排列所有作业,所得序记为 S_P,令结果序 $S = \Phi$;

2) 若 S_P 非空,则重复如下操作:从 S_P 中取出排在首位的作业 j,将其试验性地插入序 S 中的所有可能位置,选取最小加权总延误 $\sum_{j}^{n} w_j T_j$ 的序替换 S,令 $S_P = S_P - \{j\}$.

得到初始序之后,分析比较每一作业的实际完工时间和交货期,若发生延误,表明违反了约束(4),则将该作业对应的变量 s_j 作为冲突变量放入 Left 集合;否则放入 Done 集合.

3.2.2 变量排序方法:最大加权延误优先原则

基本修复法在选择变量时,随机选择冲突变量进行调整. 针对 flow shop 问题的特点,本文采用最大加权延误优先原则(LWT)选择变量:

1) 对所有 $s_j \in$ Left,计算对应作业的加权延误量 $WT_j = w_j * \max(t_{mj}^{S} + p_{mj} - d_j, 0)$,按照 WT_j 不增原则排列 Left 中的变量;

2) 选取 Left 中排在首位的变量作为调整变量.

LWT 原则寻找对减小总冲突贡献最大的变量进行调整,可以最大限度地改善当前解的冲突情况.

3.2.3　值排序方法(ND)和回溯

基本修复法在为调整变量重新选择赋值时选用最速下降法,即将所有可能赋值进行对比,选取冲突最小的值,这必然导致计算量过大. 本文采用随机方式为变量赋值,只要所产生的目标值(总冲突)小于原赋值,就接受该值作为调整变量的新值.

1) 对选定的变量 $s_j = v_j$,从其值域中随机挑选替换值 $v_j' \neq v_j$;

2) 令 $s_j = v_j'$,同时按照调整变量优先的原则调整其他重复变量,形成新的序 FS';

3) 若由新赋值 v_j'构成的序 FS'产生的总冲突(即目标值)比原序 FS 导致的冲突量小,则接受值 v_j'作为变量 s_j 的值,赋值成功,并按照各变量的新值更新 Left 和 Done;否则,撤销 $s_j = v_j'$及其他调整操作,返回步骤 1) 重新选择不重复的新赋值进行尝试.

在值排序阶段,若变量值域中的所有可能赋值都不能使冲突变小,则需要进行回溯,重新选择变量. 此时仍按照 LWT 原则选取排在下一位的变量进行调整.

3.2.4　跳出局部极值算法:最大延误-提前变量互换法

如果 Left 中所有的变量值都没有被调整,表明当前解已无法通过已定义的方式改进,则算法陷入了局部极值,达到局部最优. 此时就需要跳出局部极值,本文提出了最大延误-提前变量互换方法(LTEE)以帮助算法跳出局部极值:

1) 依照 LWT 原则对 Left 集合中的变量进行排序;

2) 对所有 Done 集合中的变量 $s_j \in$Done,计算加权提前时间 $WE_j = w_j * \max(d_j - t_{mj}^S + p_{mj}, 0)$,按照 WE_j 不增原则排列 Done 中的变量;

3) 分别选取 Left 和 Done 中排在首位的变量,交换它们的取值,从而得到新的排序.

LTEE 启发式将加权延误最大和加权提前最大的作业序位进行交换,得到的新排序不再是局部极值,并且使目标变坏的程度尽可能小. 用这一新排序作为初始解,修复法可以重新开始迭代.

4　数值实验分析

为了分析 IRCS_WT 算法对 flow shop 算例优化的有效性,本文选取 OR_library 发布的 31 个标准 flow shop 算例对 IRCS_WT 算法进行验证,包括 8 个 CAR 系列问题,2 个 HEL 系列问题,以及 21 个 REC 系列问题. 本文提到的算法均使用 C++ 编程语言实现,所有计算均在配置为 Pentium 3.2 GHz 处理器、1 GB 内存的 PC 机上运行,统计分析使用了 SPSS 软件.

由于 OR_library 公布的标准算例不包含交货期 d_j 和权重 w_j 的数据,因此本文参照文献[18]和文献[19]的方法,采用基于 TWK 思想计算各算例中作业的交货期,即

$$d_j = k * \sum_i^m p_{ij},\ j = 1,2,\cdots,n. \qquad (7)$$

其中:k 为确定误工比例的常数,不失一般性,此处取 $k = 1.5$;根据文献[19]的研究,权重数据 W_j 从均匀分布的 $U[1,10]$ 中随机生成.

4.1　确定 IRCS_WT 算法参数

IRCS_WT 算法采用迭代方式,因而算法的停止条件设定为迭代次数,即 IT. 为确定 IT 较佳的取值,首先通过仿真试验分析了 IT 对算法性能的影响. 为了不同算例之间进行比较和统计分析的方便,按下式定义的偏差量 DEV 对解的优劣进行评价:

$$\mathrm{DEV} = \frac{\text{当前值} - \text{已知最优值}}{\text{已知最优值}} \times 100\ \%. \qquad (8)$$

为了检验参数 IT 对算法结果的影响,设定 $IT = [1,20]$,分别用 IT 的 20 个取值对 31 个算例各自独立运行 30 次.

从图 2 所示参数 IT 的 95 % 置信区间图直观分析,IT 取值小则计算效率高,但结果较差;IT 取值增大可以搜索更多的邻域,得到更好的结果,但运算时间会相应增加.

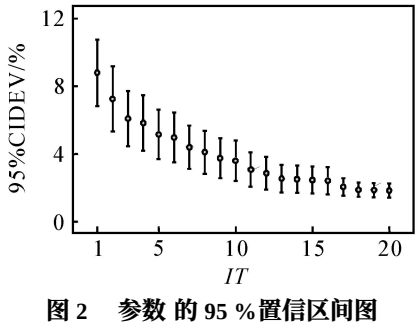

图 2　参数 的 95 % 置信区间图

首先用方差分析(ANOVA)检验选取不同参数值所得到结果之间差异是否显著. 分析结果如表 1 所示,从表中可以得出,IT 的不同取值所得到的结果之间在 0.1 % 的置信水平上存在着显著的差异.

为了具体分析差异所在,并找出一个合适的参数取值,对计算结果进行了事后比较分析. 表 2 截取了事后比较部分结果($IT = 9 \sim 12$),从表中可以看出,当 $IT \geq 11$ 时,其结果显著优于较小取值所得结果,且与更大取值之间的结果没有显著差异,因而取 $IT = 11$,以兼顾效率与效果.

4.2　时间性能

算法的时间性能与变量数目密切相关,相关分析的结果表明,IRCS_WT 的运行时间与其变量数 n

表 1 方差分析结果

	Sum of Squares	Degress of Freedom	Mean Square	F Statistic	Level of Significance
Between Groups	22 410.5	19	1 179.502	10.3	0.000
Within Groups	706 755.4	6 180	114.362		
Total	729 166.0	6 199			

表 2 多重比较结果(IT = 9 ～ 12)

IT (I)	*IT* (J)	Mean Difference (I-J)	*IT* (I)	*IT* (J)	Mean Difference (I-J)	*IT* (I)	*IT* (J)	Mean Difference (I-J)	*IT* (I)	*IT* (J)	Mean Difference (I-J)
9	1	- 5.05 * *	10	1	- 5.19 * *	11	1	- 5.72 * *	12	1	- 5.93 * *
	2	- 3.50 * *		2	- 3.64 * *		2	- 4.17 * *		2	- 4.38 * *
	3	- 2.34 * *		3	- 2.48 * *		3	- 3.01 * *		3	- 3.22 * *
	4	- 2.08 *		4	- 2.23 * *		4	- 2.76 * *		4	- 2.97 * *
	5	- 1.40		5	- 1.54		5	- 2.07 *		5	- 2.28 * *
	6	- 1.22		6	- 1.37		6	- 1.90 *		6	- 2.11 *
	7	- 0.65		7	- 0.79		7	- 1.32		7	- 1.53
	8	- 0.36		8	- 0.51		8	- 1.03		8	- 1.25
	10	0.15		9	- 0.15		9	- 0.67		9	- 0.89
	11	0.67		11	0.52		10	- 0.52		10	- 0.74
	12	0.89		12	0.74		12	0.21		11	- 0.21
	13	1.20		13	1.05		13	0.53		13	0.31
	14	1.24		14	1.09		14	0.57		14	0.35
	15	1.29		15	1.14		15	0.62		15	0.40
	16	1.33		16	1.18		16	0.66		16	0.44
	17	1.68 *		17	1.54		17	1.01		17	0.80
	18	1.86 *		18	1.72 *		18	1.19		18	0.98
	19	1.88 *		19	1.73 *		19	1.20		19	0.99
	20	1.91 *		20	1.76 *		20	1.23		20	1.02

注: * 表示平均差异在 5 % 置信水平上显著; * * 表示平均差异在 1 % 置信水平上显著.

显著正相关(见表3). 因m仅对解的评价过程有影响,而对搜索阶段不发挥作用,故其对运行时间的影响不显著.

IT = 11 时, IRCS_WT 在 $n \leq 30$ 的算例上平均运行时间在 5 s 之内; 在 $n = 50$ 的算例上平均运行时间在 20 s 之内; 在 $n = 75$ 的算例上平均运行时间在 50 ～ 70 s 之间; 在 $n = 100$ 的算例上平均运行时间约为 190 s.

表 3 平均运行时间(IT = 11) 与 n 和 m 的相关系数

Pearson Correlation	*n*	*m*
Average Time	0.858 * *	0.316

注: * * 表示相关系数平均差异在 1 % 的置信水平上显著.

4.3 对比分析

为了对 IRCS_WT 算法效果进行验证,本文选择 ENEH,过滤束搜索(FBS) 以及遗传算法(GA) 作为对比算法.

4.3.1 对比算法参数设定

FBS 的参数根据文献[20,21] 的研究结果选择 beamwidth = 5, filterwidth = 5, 局部和全局评价函数均使用 WEDD 规则. GA 的参数与所解决的问题密切相关,因此,本文根据对少量算例的试运算结果设定交叉概率为 0.8, 变异概率为 0.1; 群体规模设为 100, 遗传算子分别使用 PMX 交叉算子和移位变异[16,22]; 为了方便比较,终止条件设为与 IRCS_WT 相同的运行时间,同时为了保证 GA 结果的最优性, GA 进化代数最少为 400.

4.3.2 结果分析

ENEH 和 FBS 算法均为确定性算法, 而 IRCS_WT 和 GA 均含有随机因素,因此,采用后两种算法对每一算例分别独立运行 30 次,得出其最好、最差和平均的结果,如表 4 所示. 通过对表中数据进行直观比较可得出以下 3 点结论:

1) 表 4 的最末一行计算了 4 种算法在 31 个算例上所得 DEV 结果的平均值, 从中可以看出, IRCS_WT 的最好、最差和平均的总平均结果均小于其他 3 种算法; ENEH 和 FBS 的平均结果很接近,而在不同算例上各有优势.

2) 在 4 种算法中, IRCS_WT 的最好结果在 30 个算例上(96.8 %) 取得了最小的 DEV,仅在一个算例上大于 GA 的最好结果.

表 4　4 种算法 DEV 结果比较　　%

	ENEH	FBS	IRCS_WT			GA		
			最好	最差	平均	最好	最差	平均
car1	1.85	5.29	0.00	0.00	0.00	1.16	25.95	12.23
car2	2.22	15.98	0.00	2.22	0.22	0.00	22.24	7.33
car3	3.15	3.15	0.00	0.00	0.00	0.55	24.03	9.15
car4	5.41	9.58	0.00	2.47	0.78	3.78	14.36	9.45
car5	23.96	36.75	5.56	5.56	5.56	1.22	36.52	15.95
car6	21.13	0.00	0.00	21.13	6.34	0.00	28.11	9.12
car7	98.08	54.12	0.00	98.08	29.60	0.00	49.91	9.33
car8	28.66	0.00	0.00	0.00	0.00	0.00	22.73	4.01
hel1	2.40	11.58	0.34	0.51	0.38	14.12	22.04	16.79
hel2	11.87	9.06	1.24	9.02	2.84	11.25	30.94	20.99
rec01	8.43	5.90	0.00	0.72	0.13	11.55	34.26	18.96
rec03	14.10	13.44	0.00	3.35	0.80	1.97	37.67	20.93
rec05	14.63	2.28	0.40	2.53	1.32	9.33	29.29	16.18
rec07	5.94	13.38	0.00	3.12	1.11	8.56	30.60	19.66
rec09	9.56	4.89	0.00	7.04	3.79	11.52	28.95	20.11
rec11	9.74	12.88	0.00	4.44	1.01	12.50	32.66	20.76
rec13	15.80	37.00	0.00	6.10	3.00	18.92	50.41	30.18
rec15	7.29	20.56	0.00	5.33	4.04	14.85	52.10	32.24
rec17	14.08	18.00	2.86	11.82	7.49	14.75	47.28	31.15
rec19	2.57	6.81	0.01	1.49	1.10	13.48	29.73	21.71
rec21	9.48	11.06	0.08	4.55	2.87	19.64	38.71	28.64
rec23	5.49	14.78	0.00	0.65	0.39	18.47	34.00	27.97
rec25	18.30	15.43	4.69	9.99	7.60	19.99	49.82	33.70
rec27	16.93	5.74	0.00	1.89	1.05	25.69	55.02	37.88
rec29	12.07	20.21	0.00	5.04	3.06	23.61	38.98	30.82
rec31	6.07	10.51	0.17	2.89	1.40	11.47	21.76	17.08
rec33	10.94	10.86	1.12	4.80	2.28	10.66	28.07	20.82
rec35	11.62	12.02	0.84	5.31	2.12	15.97	26.72	20.42
rec37	7.43	14.81	0.39	2.80	1.59	17.35	25.60	22.60
rec39	5.89	12.21	0.43	2.57	1.75	14.97	25.39	19.34
rec41	5.06	12.10	0.27	2.78	1.87	17.90	31.80	23.47
Average	13.23	13.56	0.59	7.36	3.08	11.14	33.09	20.29

表 5　IRCS_WT 和 GA 三组结果配对 T 检验

IRCS-WT-GA	Mean/ %	Standard Deviation	Standard Error Mean	95 % Confidence Interval of the Difference		t Statistic	Level of Significance (2-tailed)
				Upper/ %	Lower/ %		
最好	- 10.54	0.077 3	0.013 9	- 13.38	- 7.71	- 7.595	0.000
最差	- 25.72	0.167 4	0.030 1	- 31.86	- 19.59	- 8.558	0.000
平均	- 17.21	0.103 2	0.018 5	- 20.99	- 13.43	- 9.286	0.000

3) 在对 IRCS_WT 与 GA 的结果进行比较时,将两种算法的最好、最差和平均结果分别对比:对于最好结果,IRCS_WT 在 30 个算例上(96.8 %) 不劣于 GA 的相应结果;对于最差结果,IRCS_WT 在 30 个算例上 (96.8 %) 不劣于 GA;对于平均结果,IRCS_WT 同样在 30 个算例上 (96.8 %) 不劣于 GA.

为了进一步检验 IRCS_WT 和 GA 结果之间的差异性,采用配对 T 检验分别对最好、最差和平均结果进行了统计分析,结果如表 5 所示. 从检验结果可以得出如下结论:在 3 组比较中,IRCS_WT 的结果都显著小于 GA 结果,并且差值的 95 % 置信区间的上下界都小于 0.

5　结　论

本文针对 flow shop 排序问题提出了一种改进的修复约束满足算法(修复法). 改进算法 IRCS_WT 引入扩展的 NEH 启发式(ENEH)为算法产生初始解,使新算法的初始解得到改善;在新的变量表示方式下,提出了一种采用最大加权延误优先(LWT)规则选择变量的方法,有效提高了算法的搜索效率. 此外,本文还提出一种最大延误-提前变量互换法(LTEE),该启发式不仅能够有效跳出局部

极值,而且还能减小新起点变坏的程度.

对于 IRCS_WT 算法性能至关重要的参数 IT,本文通过仿真实验及统计方法确定了相应的参数值,可以在计算效率和算法结果之间取得平衡.为检验算法性能,将 IRCS_WT 的运算结果与 ENEH, FBS 和 GA 进行了对比,研究结果表明 IRCS_WT 具有明显的优势.

作为一种非系统性(non-systematic)算法,IRCS_WT 仍面临搜索效率和局部极值两大问题,搜索效率的提高有赖于变量排序和值排序启发式的改进,这一点可以借鉴更多结构化方法中的启发式;而跳出局部极值的算法也有待进一步研究.

参考文献(References)

[1] 田盛丰. 一种基于修改的约束满足算法[J]. 计算机研究与发展, 1997, 34(2): 93-98.
(Tian S F. A repair-based algorithm for constraint satisfaction[J]. Computer Research & Development, 1997, 34(2): 93-98.)

[2] Tsang E. Foundations of constraint satisfaction[M]. San Diego: Academic Press, 1993.

[3] Dechter R, Jeavons P, Rossi F, et al. Constraint processing[M]. San Francisco: Morgan Kaufmann Publishers, 2003.

[4] 段黎明, 陈进, 刘飞. 基于约束分析的 Job Shop 调度算法的综述[J]. 重庆大学学报, 1988, 21(1): 133-138.
(Duan L M, Chen J, Liu F. The algorithms of constraint-based job shop scheduling[J]. J of Chongqing University, 1988, 21(1): 133-138.)

[5] 郭冬芬, 李铁克. 基于约束满足的车间调度算法综述[J]. 计算机集成制造系统——CIMS, 2007, 13(1): 117-125.
(Guo D F, Li T K. Constraint-based algorithm for job shop scheduling[J]. Computer Integrated Manufacturing Systems, 2007, 13(1): 117-125.)

[6] Russel S, Norvig P. Artificial intelligence: A modern approach[M]. 2nd ed. New Jersey: Pearson Education, 2002.

[7] Minton S, Johnston M D, Philips A B, et al. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems[J]. Artificial Intelligence, 1992, 58(1-3): 161-205.

[8] Morris P. The breakout method for escaping from local minima[C]. Proc of the 11th National Conf on Artificial Intelligence. Washington, 1993: 40-45.

[9] Yugami N, Ohta Y, Hara H. Improving repair-based constraint satisfaction methods by value propagation[C]. Proc of the 12th National Conf on Artificial Intelligence. Sna Jose, 1994: 344-349.

[10] Richards E T, Richards B. Non-systematic search and learning: An empirical study[C]. Proc of the 4th Int Conf on Principles and Practice of Constraint Programming —CP'98. Pisa, 1998: 370-384.

[11] Wallace R J, Freuder E C. Stable solutions for dynamic constraint satisfaction problems[C]. Proc of the 4th Int Conf on Principles and Practice of Constraint Programming —CP'98. Pisa, 1998: 447-461.

[12] Wang J. Constraint-based schedule repair for product development projects with time-limited constraints[J]. Int J of Production Economics, 2005, 95(3): 399-414.

[13] Glover F. Tabu search: Part I[J]. ORSA J on Computing, 1989, 1(3): 190-206.

[14] Gersmann K, Hammer B. Improving iterative repair strategies for scheduling with the SVM[J]. Neurocomputing, 2005, 63: 271-292.

[15] Pinedo M. Scheduling: Theory, algorithms, and systems[M]. 2nd ed. New Jersey: Prentice-Hall, 2002.

[16] 师瑞峰, 周泓, 上官春霞. 混合递进多目标进化算法及其在 flow shop 排序中的应用[J]. 系统工程理论与实践, 2006, 26(8): 101-108.
(Shi R F, Zhou H, Shangguan C X. A hybrid escalating multi-objective evolutionary algorithm with its application to flow shop problems[J]. Systems Engineering-theory & Practice, 2006, 26(8): 101-108.)

[17] Taillard E. Some efficient heuristic methods for the flow shop sequencing problem[J]. European J of Operational Research, 1990, 47(1): 65-74.

[18] Baker K R. Sequencing rules and due-date assignments in a job shop[J]. Management Science, 1984, 30(9): 1093-1104.

[19] Congram R K, Potts C N, van de Velde S. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem[J]. INFORMS J on Computing, 2002, 14(1): 52-67.

[20] Sabuncuoglu I, Bayiz M. Job shop scheduling with beam search[J]. European J of Operational Research, 1999, 118(2): 390-412.

[21] 上官春霞, 周泓, 师瑞峰. 带部分回溯的过滤束搜索算法及其在 Job Shop 问题中的应用[J]. 系统工程理论与实践, 2007, 27(1): 143-151.
(Shangguan C X, Zhou H, Shi R F. Filtered beam search algorithm with partial backtracking and its application to job shop scheduling[J]. Systems Engineering-theory & Practice, 2007, 27(1): 143-151.)

[22] Gen M, Cheng R W. Genetic algorithms and engineering optimization[M]. New York: Wiley-Interscience, 1999.