# 一种求解置换 Flow Shop 调度问题的 DRPFSP 算法

魏嘉银 秦永彬 许道云

(贵州大学计算机科学与技术学院 贵阳 550025)

**摘 要** 针对置换 Flow Shop 调度问题,在对经典启发式算法进行研究的基础上,提出了一种用于求解此类问题的 DRPFSP 算法。算法首先对加工时间矩阵 $A$ 进行数据标准化处理;然后通过引入一个概率矩阵 $P_{2\times m}$ 和相应的降维函数 $f_p(A)=PA$,将含有 $m$ 台机器的原问题转化为含 2 台机器的新问题;再运用 Johnson 算法对新问题进行求解得到一个调度序列 $\pi^0$;最后结合插入邻域快速评价法对 $\pi^0$ 进行处理以获得原问题的一个调度方案 $\pi$。实验结果表明,相对于经典的启发式算法,DRPFSP 算法能更有效地对置换 Flow Shop 调度问题进行求解。

**关键词** 置换 Flow Shop 调度问题,数据标准化,降维

**中图法分类号** TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.7.015

## DRPFSP Algorithm for Solving Permutation Flow Shop Scheduling Problem

WEI Jia-yin QIN Yong-bin XU Dao-yun

(Department of Computer Science & Technology, Guizhou University, Guiyang 550025, China)

**Abstract** For the permutation Flow Shop scheduling problem, a new algorithm named DRPFSP, which is based on the study of the classic heuristic algorithms, was proposed in this paper. The algorithm normalizes the matrix $A$ of processing times firstly. Secondly, it transforms the original problem containing $m$ machines into a new problem containing 2 machines by introducing a probability matrix $P_{2\times m}$ and a corresponding dimension reduction function $f_p(A)=PA$. Thirdly, it uses the Johnson algorithm to solve the new problem and finds a scheduling sequence $\pi^0$. Finally, it processes $\pi^0$ with the insert neighborhood fast evaluation method to obtain a scheduling scheme $\pi$ for the original problem. The experiment results show that, compared with the classical heuristic algorithms, DRPFSP algorithm is more effective for the permutation Flow Shop scheduling problem.

**Keywords** Permutation Flow Shop scheduling problem, Data normalization, Dimensionality reduction

## 1 引言

Flow Shop 调度问题(Flow-shop Scheduling Problem, FSP)是许多实际流水线生产调度问题的简化模型,亦是研究最为广泛的生产调度问题之一,具有很强的工程应用背景。置换 Flow Shop 调度问题是约定每台机器上所加工的各工件的顺序相同的一类 Flow Shop 调度问题。具有 $m$ 台机器、每个工件有 $m$ 道工序、各工序分别在不同机器上加工并且工件的加工过程不允许中断、以最小化最大完成时间为优化目标的置换 Flow Shop 调度问题,可表示为 $F_m|prmu|C_{\max}$。已有研究证明 $F_m|prmu|C_{\max}$($m\geqslant 3$)为 NP-hard 问题[1],至今没有一个具有多项式计算复杂度的全局优化算法。

自 1954 年 Johnson 提出双机 Flow Shop 调度问题以来,许多学者对该问题进行了深入研究,提出了若干精确调度算法。虽然精确算法可以保证求解结果的最优性,但此类算法所需的计算时间会随着问题规模的增大而呈指数级增长,所以它仅适合于处理小规模的调度问题。因此,对于 Flow Shop 调度问题求解算法的设计主要是从启发式算法、智能优化算法或混合算法这 3 个方面来考虑。启发式算法通过利用面向问题的特定知识和经验产生调度方案,能够在较短的时间内得到次优解。经典的用于求解 Flow Shop 调度问题的启发式算法有 Johnson 算法[2](可在 $O(n\log n)$ 时间内求得 $F_2|prmu|C_{\max}$ 的最优解)、CDS 算法[3]、Palmer 算法[4]、Gupta 算法[5]、RA 算法[6]、NEH 算法[7]、NEH_KK 算法[8,9]和 FRB1—FRB5 算法[10]等。其中 NEH 算法是最有效的启发式算法之一,在该算法的基础上产生了众多相应的改进算法,FRB5 算法具有最好的优化质量,但其所花费的计算时间较长。智能优化算法是通过模拟某些自然现象或过程而建立起来的,目前可用于求解 Flow Shop 调度问题的智能优化算法有 GA 算法[11]、TS 算法[12]、SA 算法[13]、ACO 算法[14]和 PSO 算法[15]等。混合算法是通过利用不同优化机制、优化行为和优化结构的互补性,合理融合各类算法而得到的组合算法。

本文在对经典启发式算法进行研究的基础上,提出一种用于求解置换 Flow Shop 调度问题的 DRPFSP 算法。该算

到稿日期:2014-06-13 返修日期:2014-09-26 本文受国家自然科学基金(60863005,61262006),贵州省科学技术基金(黔科合 J 字[2012]2125 号),贵州省科技厅制造业信息化项目(黔科合 GY(2011)3074)资助。

魏嘉银(1986—),男,博士生,CCF 会员,主要研究方向为算法设计与分析、算法机制设计;秦永彬(1980—),男,副教授,硕士生导师,主要研究方向为智慧计算与智能计算、大数据管理与应用;许道云(1959—),男,教授,博士生导师,主要研究方向为可计算性与计算复杂性、算法设计与分析等,E-mail:dyxu@gzu.edu.cn(通信作者)。

法通过引入一个 $2\times m$ 的概率矩阵 $P$ 并定义相应的降维函数，将含 $m$ 台机器的调度问题降维为一个含 2 台虚拟机器的调度问题，然后对降维后的问题运用 Johnson 算法得到[$n$]上的一个调度序列 $\pi^0$，最后再结合插入邻域快速评价法对 $\pi^0$ 进行处理得到最终的一个序 $\pi$，并以此作为算法所得的调度方案。仿真实验表明，DRPFSP 算法可以有效地运用于置换 Flow Shop 调度问题的求解。

## 2 问题描述

置换 Flow Shop 调度问题是指：假定有 $n$ 个待加工工件，其工件指标集[$n$]=$\{1,2,\cdots,n\}$，每个加工工件需要在 $m$ 台机器 $M_1,M_2,\cdots,M_m$ 上顺序加工，假定 $m$ 维正实数组($a_{i1},a_{i2},\cdots,a_{im}$)为工件 $i$ 在机器序列($M_1,M_2,\cdots,M_m$)上对应的加工时间；要求在每台机器加工的各工件的顺序相同且工件的加工过程不允许中断的约定下，求一个[$n$]上的排序 $\pi$，使得按排序 $\pi$ 处理 $n$ 个待加工工件的最大完成时间(makespan，亦称调度时间跨度)最短。

当 $m=2$ 时，著名的 Johnson 算法给出了该问题的一个时间复杂度为 O($n\log n$)的最优求解方案。Johnson 算法本质上是找给定加工工件集上的 Johnson 序。Johnson 序由如下方法产生：对于给定的 $n$ 个时间对$\{(a_1,b_1),(a_2,b_2),\cdots,(a_n,b_n)\}$，将[$n$]划分为两个集合 $N_<=\{i\in[n]:a_i<b_i\}$，$N_\geqslant=\{i\in[n]:a_i\geqslant b_i\}$。在 $N_<$ 中按 $a_i$ 上升排序，得到 $N_<$ 中指标的一个排序 $\pi_<$；在 $N_\geqslant$ 中按 $b_i$ 下降排序，得到 $N_\geqslant$ 中指标的一个排序 $\pi_\geqslant$。令 $\pi=\pi_<\circ\pi_\geqslant$ 得到[$n$]上的一个排序，称为 Johnson 序。

经典的 CDS 算法和 RA 算法均是通过设定一个降维矩阵 $P$ 将 $F_m\mid prmu\mid C_{\max}$ 转换为 $F_2\mid prmu\mid C_{\max}$，然后再运用 Johnson 算法进行求解以得到原问题的解。例如：当 $m=5$ 时，CDS 算法和 RA 算法所选择的降维矩阵 $P$ 分别如下：

$$P_{CDS}=\left\{\begin{bmatrix}1&0&0&0&0\\0&0&0&0&1\end{bmatrix},\begin{bmatrix}1&1&0&0&0\\0&0&0&1&1\end{bmatrix},\begin{bmatrix}1&1&1&0&0\\0&0&1&1&1\end{bmatrix},\begin{bmatrix}1&1&1&1&0\\0&1&1&1&1\end{bmatrix}\right\}$$

$$P_{RA}=\begin{bmatrix}5&4&3&2&1\\1&2&3&4&5\end{bmatrix}$$

显然，CDS 算法和 RA 算法所选取的降维矩阵只与问题中的机器数 $m$ 有关，但均未充分运用各工件在机器上的加工时间数据，这使得算法往往无法得到一个很好的解。

为避免 CDS 算法和 RA 算法所存在的问题，本文提出一种算法，该算法以各工件在机器上的加工时间数据作为参照，选择相应的概率矩阵作为降维矩阵来将 $F_m\mid prmu\mid C_{\max}$ 问题转换为一个新的 $F_2\mid prmu\mid C_{\max}$ 问题，然后再运用 Johnson 算法对新问题进行求解以得到一个初始调度顺序 $\pi^0$，最后再结合插入邻域快速评价法对 $\pi^0$ 进行处理得到最终的一个序 $\pi$，并以此作为各个工件的加工顺序。

## 3 数学模型

### 3.1 初始模型建立

对于一个给定的有 $m$ 台机器 $n$ 个工件的置换 Flow Shop 调度问题，其加工时间矩阵为：

$$A=\begin{pmatrix}a_{11}&a_{12}&\cdots&a_{1n}\\a_{21}&a_{22}&\cdots&a_{2n}\\\vdots&\vdots&\ddots&\vdots\\a_{m1}&a_{m2}&\cdots&a_{mn}\end{pmatrix}$$

选定一个概率矩阵(视为变量矩阵，其值的设定将在 3.2 节中详细分析)

$$P=\begin{pmatrix}p_{11}&p_{12}&\cdots&p_{1m}\\p_{21}&p_{22}&\cdots&p_{2m}\end{pmatrix}$$

其中，$0\leqslant p_{ij}\leqslant 1$，$\sum_{j=1}^{m}p_{ij}=1$，$i=1,2$；$j=1,2,\cdots,m$。

(1)定义降维函数 $f_P:R^m\to R^2$。从而，

$$f_P(A)=PA=\begin{pmatrix}d_1&d_2&\cdots&d_n\\e_1&e_2&\cdots&e_n\end{pmatrix}$$

其中，$d_j=\sum_{l=1}^{m}p_{1l}a_{lj}$，$e_j=\sum_{l=1}^{m}p_{2l}a_{lj}$，$j=1,2,\cdots,n$。

(2)由矩阵 $\begin{pmatrix}d_1&d_2&\cdots&d_n\\e_1&e_2&\cdots&e_n\end{pmatrix}$，运用 Johnson 算法得到[$n$]上的一个 Johnson 序 $\pi$。调整顺序后得到如下矩阵：

$$\begin{pmatrix}d_1&d_2&\cdots&d_n\\e_1&e_2&\cdots&e_n\end{pmatrix}^{\pi}=\begin{pmatrix}d_{\pi(1)}&d_{\pi(2)}&\cdots&d_{\pi(n)}\\e_{\pi(1)}&e_{\pi(2)}&\cdots&e_{\pi(n)}\end{pmatrix}$$

(3)定义目标函数 $g(P,A)$。

对于矩阵 $\begin{pmatrix}d_{\pi(1)}&d_{\pi(2)}&\cdots&d_{\pi(n)}\\e_{\pi(1)}&e_{\pi(2)}&\cdots&e_{\pi(n)}\end{pmatrix}$，定义一个空闲时间向量($\delta_1,\delta_2,\cdots,\delta_n$)，其中

$$\delta_1=d_{\pi(1)}$$

$$\delta_2=\max\{0,d_{\pi(2)}-e_{\pi(1)}\}$$

$$\delta_3=\max\{0,\sum_{l=2}^{3}(d_{\pi(l)}-e_{\pi(l-1)})-\sum_{l=2}^{3-1}\delta_l\}$$

$$\vdots$$

$$\delta_i=\max\{0,\sum_{l=2}^{i}(d_{\pi(l)}-e_{\pi(l-1)})-\sum_{l=2}^{i-1}\delta_l\},\ i=4,\cdots,n$$

令 $g(P,A)=\sum_{i=1}^{n}(\delta_i+e_{\pi(i)})=\sum_{i=1}^{n}(\delta_i+e_i)$。

(4)与给定置换 Flow Shop 调度问题相应的数据模型可以表示如下：

$$\min\ g(P,A)=\sum_{i=1}^{n}(\delta_i+e_i)\qquad(1)$$

$$\text{s.t. }\delta_1=d_{\pi(1)}$$

$$\delta_2=\max\{0,d_{\pi(2)}-e_{\pi(1)}\}$$

$$\delta_i=\max\{0,\sum_{l=2}^{i}(d_{\pi(l)}-e_{\pi(l-1)})-\sum_{l=2}^{i-1}\delta_l\},\ i=3,\cdots,n$$

$$d_j=\sum_{l=1}^{m}p_{1l}a_{lj},\ e_j=\sum_{l=1}^{m}p_{2l}a_{lj},\ j=1,2,\cdots,n$$

$$0\leqslant p_{ij}\leqslant 1,\ \sum_{j=1}^{m}p_{ij}=1,\ i=1,2$$

由于该模型中目标函数 $g(P,A)$ 与空闲时间向量($\delta_1,\delta_2,\cdots,\delta_n$)有关，而约束条件中的 $\delta_i$ 是一个与 Johnson 序 $\pi$ 有关的 max 函数，因此，该模型的目标函数和约束条件均是非线性的，这便导致无法运用线性规划的思想来求解该模型以确定最优的概率矩阵 $P^*$ 并最终获得问题的调度顺序 $\pi$。为了实现运用上述降维思想对置换 Flow Shop 调度问题进行求解的目标，下面考虑将上述模型化简为一个易于求解的 LP 优化问题。

### 3.2 模型化简

显然上述模型的求解难点在于其中的空闲时间 $\delta_i$ 是一个与 Johnson 序 $\pi$ 有关的 max 函数，其是非线性的。因此，首

先采用准空闲时间 $\delta_i'$（$\delta_i'$可以取负值）来替代上述模型中的空闲时间 $\delta_i$ 以将相应的约束条件转换为一个线性函数。本文所采用的准空闲时间 $\delta_i'$的取值设定如下：

对于上述加工时间矩阵 $A$ 和概率矩阵 $P$，令与之对相应的准空闲时间 $\delta_i'$的取值如下：

$$\delta_1' = d_{\pi(1)} > 0$$

$$\delta_2' = d_{\pi(2)} - e_{\pi(1)}$$

$$\begin{aligned}\delta_3' &= (d_{\pi(2)} + d_{\pi(3)}) - (e_{\pi(1)} + e_{\pi(2)} + \delta_2') \\ &= (d_{\pi(2)} + d_{\pi(3)}) - [e_{\pi(1)} + e_{\pi(2)} + (d_{\pi(2)} - e_{\pi(1)})] \\ &= d_{\pi(3)} - e_{\pi(2)}\end{aligned}$$

$$\vdots$$

$$\begin{aligned}\delta_i' &= \sum_{l=2}^{i}(d_{\pi(l)} - e_{\pi(l-1)}) - \sum_{l=2}^{i-1}\delta_l' \\ &= d_{\pi(i)} - e_{\pi(i-1)}, i = 4,5,\cdots,n\end{aligned}$$

则与 $\delta_i'$相对应的目标函数 $g(P,A) = \sum_{i=1}^{n}(\delta_i' + e_{\pi(i)}) = \sum_{i=1}^{n} d_{\pi(i)} + e_{\pi(n)} = \sum_{i=1}^{n} d_i + e_{\pi(n)}$。

由于 $g(P,A)$中的项 $e_{\pi(n)}$同 Johnson 序 $\pi$ 有关，使得模型中概率矩阵 $P$ 的求解依赖于 Johnson 序 $\pi$，而 $\pi$ 是运用 Johnson 算法对由 $P$ 所导出的嵌入函数 $f_P(A)$所表示的 $F_2 \mid prmu \mid C_{\max}$进行求解所得，这便导致该求解过程产生了一个圈。为了打破该圈，设与 $\delta_i'$相对应的目标函数 $g(P,A) = \sum_{i=1}^{n} d_i$。与此同时，限定概率矩阵 $P$ 除了一般的概率矩阵约束之外，其取值还需满足如下条件：

$$\forall\, p_{1j} > 0, p_{2j} = 0; \forall\, p_{1j} = 0, p_{2j} = \frac{1}{m - |\{p_{1j} \mid p_{1j} > 0\}|}$$

其中，$|\{p_{1j} \mid p_{1j} > 0\}|$是指 $p_{1j}$中非负元素的个数。加入这两个约束，主要是确保在计算降维函数 $f_p(A) = PA$ 并依据 $f_p(A)$求初始排序时，能够使用全部的加工时间信息而不至于造成信息的大量损耗，从而使得算法能够求得一个更好的解。

至此完成了对模型(1)的化简，并得到了用于求解置换 Flow Shop 调度问题的与准空闲时间 $\delta_i'$相对应的 LP 优化模型，如下所示：

$$\min\ g(P,A) = \sum_{i=1}^{n} d_i \tag{2}$$

$$\text{s.t. } d_i = \sum_{l=1}^{m} p_{1l} a_{li}, i = 1,2,\cdots,n$$

$$e_i = \sum_{l=1}^{m} p_{2l} a_{li}, i = 1,2,\cdots,n$$

$$\sum_{j=1}^{m} p_{ij} = 1, i = 1,2$$

$$p_{2j} = 0, \forall\, p_{1j} > 0$$

$$p_{2j} = \frac{1}{m - |\{p_{1j} \mid p_{1j} > 0\}|}, \forall\, p_{1j} = 0$$

$$0 \leqslant p_{ij} \leqslant 1, i = \{1,2\}, j = \{1,2,\cdots,n\}$$

## 4 求解置换 Flow Shop 调度问题的 DRPFSP 算法

### 4.1 数据标准化处理

从化简后的模型容易得知，概率矩阵 $P$ 的作用相当于是对问题中的各个工件在各机器上的处理时间作加权处理。然而，由于流水车间调度问题中的各工件的各道工序之间和各工件之间所需的处理时间均有可能存在较大的差异，因此，其取值有可能对所求得的概率矩阵 $P$ 产生影响并进一步影响算法的结果。为了消除这一影响，可以运用数据的标准化技术对待加工工件的时间矩阵 $A = (a_{ij})_{m\times n}(a_{ij} > 0)$进行预处理。

数据标准化技术是将数据按比例缩放，使之落入一个小的特定区间，以去除数据的单位限制，将其转化为无量纲的纯数值，便于不同单位或量级的指标能够进行比较和加权。其中最典型的就是数据的归一化处理，即将数据统一映射到[0,1]区间上，常见的数据归一化的方法有 min-max 标准化（亦称离差标准化）、log 函数转换、atan 函数转换和 z-score 标准化（亦称标准差标准化）。

本文采用 min-max 标准化方法对问题中的加工时间矩阵 $A = (a_{ij})_{m\times n}(a_{ij} > 0)$进行预处理。min-max 标准化方法是对原始数据进行线性变换，使结果落到[0,1]区间，转换函数如下：

$$a'_{ij} = \frac{a_{ij} - Ai_{\min}}{Ai_{\max} - Ai_{\min}}, i = 1,2,\cdots,m; j = 1,2,\cdots,n$$

其中，$Ai_{\min} = \min_{j=1,2,\cdots,n}\{a_{ij}\}$，$Ai_{\max} = \max_{j=1,2,\cdots,n}\{a_{ij}\}$，$i = 1,2,\cdots,m$。

例如，对于如下加工时间矩阵

$$A = (a_{ij})_{2\times 5} = \begin{bmatrix} 5 & 4 & 3 & 7 & 3 \\ 4 & 5 & 2 & 3 & 8 \end{bmatrix}$$

进行 min-max 标准化处理后的结果为：

$$A' = (a'_{ij})_{2\times 5} = \begin{bmatrix} 1/2 & 1/4 & 0 & 1 & 0 \\ 1/3 & 1/2 & 0 & 1/6 & 1 \end{bmatrix}$$

### 4.2 插入邻域快速评价法

插入操作是对工件排序进行局部搜索的常用操作，即在工件排序 $\pi$ 中，随机选择两个不同的位置 $k$ 和 $k'$，把位置 $k$ 上的工件插入到位置 $k'$。称一次插入操作为一次插入移动，所得新排序为原排序的一个邻居，原排序的所有邻居构成其插入领域。例如：$\pi = \{1,2,3\}$，则 $\pi$ 的插入邻域为{{1,3,2},{3,1,2},{3,2,1},{2,1,3},{2,3,1}}。显然，对于含有 $n$ 个工件的一个排列 $\pi$，其插入领域的规模为 $(n-1)^2$，若是使用一般的算法逐一计算所有邻居的最大完成时间，其时间复杂度为 $O(mn^3)$。为了提高评价的效率，Taillard E 在文献[16]中结合最大完成时间的 3 种计算方法提出了用于计算 $F_m \mid prmu \mid C_{\max}$的插入邻域快速评价算法，如算法 1 所示。

**算法 1** 插入邻域快速评价算法（本文中简记为 INFE）

Input：n 个工件的加工时间矩阵 $A = (a_{ij})_{m\times n}(a_{ij} > 0)$，排序 $\pi^0 = \{\pi^0(1), \pi^0(2), \cdots, \pi^0(n)\}$

Output：排序 $\pi$

1. 令 $\pi \leftarrow \pi^0$，$MinC_{\max} \leftarrow C_{\max}(\pi^0)$。
2. for k←1 to n do
3. 从 $\pi^0$ 中移除工件 $\pi^0(k)$，令 $\pi' = \pi^0 \backslash \{\pi^0(k)\} = \{\pi'(1), \pi'(2), \cdots, \pi'(n-1)\}$。
4. 由前向计算法求得排序 $\pi'$中各工序的最早开工时间 $s'_{i,j}$和完成时间 $c'_{i,j}$。
5. 由反向计算法求得排序 $\pi'$中各工序的最迟完成时间 $c'^{b}_{i,j}$和开工时间 $s'^{b}_{i,j}$。
6. for l←1 to n do
7. if l≠k then
8. 将 $\pi^0(k)$插入到 $\pi'$的位置 l，记 $\pi'' = \{\pi'(1), \pi'(2), \cdots, \pi'(l-1), \pi^0(k), \pi'(l), \cdots, \pi'(n-1)\}$。
9. if l=1 then
10. 令 $c''_{0,l} = 0$，计算 $s''_{i,l} = c''_{i-1,l}, c''_{i,l} = s''_{i,l} + p_{i,\pi(k)}, i = 1,2,\cdots,m$。
11. else
12. 令 $c''_{0,l} = c'_{i-1,l}$，计算 $s''_{i,l} = \max\{c'_{i,l-1}, c''_{i-1,l}\}, c''_{i,l} =$

$s''_{i,l}+p_{i,\pi(k)}$，i=1,2,…,m。

13. end
14. if l<n then
15. $C_{max}(\pi'')=\max_{i=1,2,\cdots,m}\{c''_{i,l}+s'^{b}_{i,l}\}$
16. else
17. $C_{max}(\pi'')=c''_{m,l}$
18. end
19. if $C_{max}(\pi'')<MinC_{max}$ then
20. $\pi\leftarrow\pi''$, $MinC_{max}\leftarrow C_{max}(\pi'')$
21. end
22. end
23. end
24. end
25. return π

很容易验证插入邻域快速评价算法的时间复杂度为 $O(mn^2)$，运用该算法能够有效地提高插入邻域搜索过程的执行效率。本文在设计算法的过程中借鉴了 FRB5 算法的思想，而 FRB5 算法中非常关键的一步就是插入邻域搜索，因此，在本文所设计的算法中将采用插入邻域快速评价算法以提高算法的执行效率。

### 4.3 DRPFSP 算法

在本文所建立的数学模型的基础上，结合运用数据标准化处理方法和插入邻域快速评价法，提出一种用于求解置换 Flow Shop 调度问题的降维算法（Dimensionality Reduction Algorithm for Permutation Flow-shop Scheduling Problem，简称为 DRPFSP 算法），如算法 2 所示。

**算法 2** DRPFSP 算法

Input：n 个工件的加工时间矩阵 $A=(a_{ij})_{m\times n}(a_{ij}>0)$，参数 h

Output：概率矩阵 $P^*$，排序 π 和相应的 $C_{max}$

1. if m=2 then
2. $P^*\leftarrow\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$
3. 调用 Johnson 算法获得[n]上的一个排序 π 和相应的 $C_{max}$。
4. return $P^*$，π and $C_{max}$。
5. end
6. for i←1 to m do
7. $Ai_{min}\leftarrow\min_{j=1,2,\cdots,n}\{a_{ij}\}$，$Ai_{max}\leftarrow\max_{j=1,2,\cdots,n}\{a_{ij}\}$
8. for j←1 to n do
9. $a'_{ij}\leftarrow(a_{ij}-Ai_{min})/(Ai_{max}-Ai_{min})$
10. end
11. end
12. 以 $A'=(a'_{ij})_{m\times n}$作为加工时间矩阵，对模型(2)调用 LP 算法求得概率矩阵 $P^*$。
13. 令 $f_{P^*}(A')\leftarrow P^*A'$
14. 依据 $f_{P^*}(A')$求[n]上的 Johnson 序 $\pi^0=\{\pi^0(1),\pi^0(2),\cdots,\pi^0(n)\}$。
15. 令 π 为两个部分排序$\{\pi^0(1),\pi^0(2)\}$和$\{\pi^0(2),\pi^0(1)\}$中具有较小最大完成时间的一个排序。
16. for k←3 to n do
17. 取出 $\pi^0$ 的第 k 个工件 $\pi^0(k)$，将其插入到 π 的所有可能位置，共得到 k 个部分排序，评价所得到的各排序，令 π 为产生最小最大完成时间的排序。
18. if k>h then
19. $\pi=INFE(A,\pi)$
20. end
21. end
22. $C_{max}\leftarrow C_{max}(A,\pi)$
23. return $P^*$，π and $C_{max}$

例 1　对于如下加工时间矩阵

$$A=(a_{ij})_{2\times5}=\begin{bmatrix}5 & 4 & 3 & 7 & 3\\ 4 & 5 & 2 & 3 & 8\end{bmatrix}$$

由于机器数 $m=2$，算法直接调用 Johnson 算法并返回如下结果：

$$P^*=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix},\pi=\{5,2,1,4,3\},C_{max}=25$$

例 2　对于如下加工时间矩阵

$$A=(a_{ij})_{3\times6}=\begin{bmatrix}19 & 44 & 85 & 59 & 87 & 51\\ 46 & 63 & 56 & 68 & 66 & 4\\ 65 & 12 & 98 & 25 & 53 & 63\end{bmatrix}$$

由于机器数 $m=3$，算法将先对其进行数据标准化处理得到新的加工时间矩阵：

$$A'=(a'_{ij})_{3\times6}=\begin{bmatrix}0 & \frac{25}{68} & \frac{33}{34} & \frac{10}{17} & 1 & \frac{8}{17}\\ \frac{21}{32} & \frac{59}{64} & \frac{13}{16} & 1 & \frac{31}{32} & 1\\ \frac{53}{86} & 0 & 1 & \frac{13}{86} & \frac{41}{86} & \frac{51}{86}\end{bmatrix}$$

然后再调用 LP 算法求得概率矩阵：

$$P^*=\begin{bmatrix}0 & 0 & 1\\ 0.5 & 0.5 & 0\end{bmatrix}$$

接着依据 $f_{P^*}(A')=P^*A'$ 调用 Johnson 算法求得[n]上的 Johnson 序 $\pi^0=\{2,4,5,3,1,6\}$，最后再结合插入邻域搜索算法对 $\pi^0$ 进行调整，可以求得：

$\pi=\{1,3,5,2,6,4\}$，$C_{max}=438$

根据算法所求得的工件加工顺序可以很容易地绘制出相应的甘特图，与例 1 和例 2 的调度结果相对应的甘特图分别如图 1、图 2 所示。

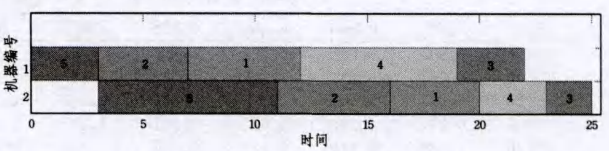


图 1　与例 1 的调度结果相对应的甘特图

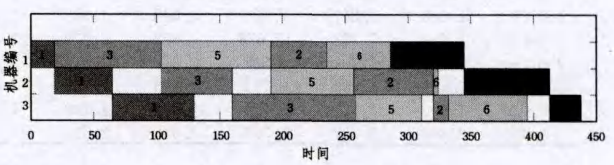


图 2　与例 2 的调度结果相对应的甘特图

很容易验证 DRPFSP 算法与 FRB5 算法具有相同的时间复杂度，均为 $O(n^3m)$。由于 DRPFSP 算法是在 $k>h$ 时才执行插入邻域搜索算法，因此在理论上 DRPFSP 算法的执行效率会比 FRB5 算法更高。

## 5　实验对比

为了验证本文所提出算法的正确性与有效性，在 CPU 为 Intel(R) Core(TM) i5-3470 3.20GHz、内存为 8GB 的 PC 机上，用 Matlab 实现本文提出的 DRPFSP 算法和 CDS、Palmer、Gupta、RACS、RAES、Raj、NEH、NEHT、$NEH_{kk1}$、$FRB4_{12}$ 与

FRB5 这 11 个典型的算法，并采用 Taillard benchmark 问题作为测试数据集，对所选的 12 个算法进行相关的实验验证。

对于 Taillard 典型问题的每个测试实例，运行各个算法，获得各算法的解及运行时间，然后按问题规模进行分组，对算法所得平均相对偏差、运行时间和算法所得最优值的个数等信息进行比较。算法所得解的相对偏差计算公式为：

$$RPI(c_i)=\frac{c_i-c^*}{c^*}\times 100\%$$

式中，$c_i$ 为算法 $i$ 的输出结果，$i\in S$，$S=\{$CDS，Palmer，Gupta，RACS，RAES，Raj，NEH，NEHT，$NEH_{KK1}$，$FRB4_{12}$，FRB5，$DRPFSP_{h=n/2}\}$；$c^*$ 为 Taillard 典型问题的最优解或上界。按照问题规模 $n\times m$，统计各算法所得解的平均相对偏差与平均运行时间，所得结果如表 1、表 2 所列。进一步地，令 $c^*=\min(c_i)$，计算各算法所得解的平均相对偏差和最优值的个数，所得结果如表 3、表 4 所列。

表 1　算法所得解的平均相对偏差(以已知最优解或上界作参照)

| 问题 | CDS | Palmer | Gupta | RACS | RAES | Raj | NEH | NEHT | $NEH_{KK1}$ | $FRB4_{12}$ | FRB5 | $DRPFSP_{n/2}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20×5 | 12.232 | 14.646 | 18.581 | 12.291 | 11.970 | 12.104 | 11.922 | 11.922 | 11.922 | 11.213 | 11.213 | 11.520 |
| 20×10 | 13.180 | 20.486 | 23.803 | 15.992 | 13.472 | 13.030 | 11.325 | 11.325 | 11.460 | 10.238 | 10.238 | 10.226 |
| 20×20 | 6.084 | 8.792 | 15.931 | 7.564 | 6.241 | 6.258 | 1.992 | 1.992 | 1.971 | 0.446 | 0.123 | 0.072 |
| 50×5 | 8.901 | 11.042 | 14.694 | 10.505 | 9.848 | 10.435 | 8.914 | 8.914 | 8.914 | 8.659 | 8.659 | 8.777 |
| 50×10 | 16.983 | 21.215 | 25.269 | 18.689 | 17.009 | 17.579 | 15.636 | 15.636 | 15.636 | 14.879 | 14.879 | 15.235 |
| 50×20 | 16.948 | 21.630 | 26.286 | 17.870 | 16.952 | 16.215 | 13.173 | 13.173 | 13.461 | 12.057 | 11.981 | 11.613 |
| 100×5 | 7.242 | 8.666 | 13.712 | 7.619 | 7.619 | 7.612 | 7.280 | 7.280 | 7.280 | 7.159 | 7.159 | 7.298 |
| 100×10 | 12.476 | 14.945 | 18.946 | 14.081 | 12.850 | 13.433 | 11.563 | 11.563 | 11.563 | 10.850 | 10.850 | 11.069 |
| 100×20 | 16.751 | 20.727 | 24.609 | 17.851 | 15.828 | 16.656 | 14.159 | 14.159 | 14.205 | 13.071 | 12.937 | 12.677 |
| 200×10 | 9.759 | 12.805 | 14.669 | 11.259 | 10.646 | 10.464 | 9.430 | 9.430 | 9.444 | 8.839 | 8.839 | 8.949 |
| 200×20 | 15.564 | 18.751 | 22.609 | 17.404 | 15.651 | 15.817 | 13.871 | 13.871 | 13.876 | 13.485 | 13.295 | 12.794 |
| 500×20 | 11.648 | 13.681 | 16.093 | 12.418 | 11.674 | 11.787 | 10.515 | 10.515 | 10.508 | 9.899 | 9.968 | 9.974 |
| 平均 | 12.314 | 15.615 | 19.600 | 13.629 | 12.480 | 12.616 | 10.815 | 10.815 | 10.853 | 10.066 | 10.012 | 10.017 |

表 2　算法的运行时间

| 问题 | CDS | Palmer | Gupta | RACS | RAES | Raj | NEH | NEHT | $NEH_{KK1}$ | $FRB4_{12}$ | FRB5 | $DRPFSP_{n/2}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20×5 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.001 | 0.001 |
| 20×10 | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.001 | 0.001 | 0.000 | 0.001 | 0.003 | 0.003 | 0.003 |
| 20×20 | 0.000 | 0.000 | 0.000 | 0.001 | 0.002 | 0.003 | 0.004 | 0.001 | 0.005 | 0.014 | 0.017 | 0.015 |
| 50×5 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.001 | 0.001 |
| 50×10 | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.001 | 0.001 | 0.000 | 0.001 | 0.003 | 0.003 | 0.003 |
| 50×20 | 0.000 | 0.000 | 0.000 | 0.001 | 0.002 | 0.005 | 0.008 | 0.002 | 0.008 | 0.020 | 0.023 | 0.021 |
| 100×5 | 0.000 | 0.000 | 0.000 | 0.001 | 0.001 | 0.000 | 0.000 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 |
| 100×10 | 0.000 | 0.000 | 0.000 | 0.001 | 0.002 | 0.002 | 0.002 | 0.001 | 0.002 | 0.005 | 0.005 | 0.005 |
| 100×20 | 0.000 | 0.000 | 0.000 | 0.002 | 0.006 | 0.008 | 0.014 | 0.004 | 0.014 | 0.030 | 0.036 | 0.034 |
| 200×10 | 0.002 | 0.001 | 0.001 | 0.001 | 0.002 | 0.003 | 0.004 | 0.002 | 0.004 | 0.010 | 0.008 | 0.008 |
| 200×20 | 0.002 | 0.001 | 0.001 | 0.004 | 0.013 | 0.015 | 0.025 | 0.005 | 0.025 | 0.050 | 0.059 | 0.055 |
| 500×20 | 0.002 | 0.002 | 0.002 | 0.008 | 0.028 | 0.035 | 0.059 | 0.013 | 0.060 | 0.114 | 0.152 | 0.125 |
| 平均 | 0.000 | 0.000 | 0.000 | 0.002 | 0.005 | 0.006 | 0.010 | 0.002 | 0.010 | 0.021 | 0.026 | 0.023 |

表 3　算法所得解的平均相对偏差(以所选 12 个算法所得最小解作参照)

| 问题 | CDS | Palmer | Gupta | RACS | RAES | Raj | NEH | NEHT | $NEH_{KK1}$ | $FRB4_{12}$ | FRB5 | $DRPFSP_{n/2}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20×5 | 0.903 | 3.097 | 6.590 | 0.997 | 0.706 | 0.784 | 0.643 | 0.643 | 0.643 | 0.000 | 0.000 | 0.266 |
| 20×10 | 3.092 | 9.733 | 12.725 | 5.642 | 3.347 | 2.933 | 1.400 | 1.400 | 1.524 | 0.410 | 0.410 | 0.400 |
| 20×20 | 6.377 | 9.060 | 16.240 | 7.886 | 6.542 | 6.578 | 2.279 | 2.279 | 2.254 | 0.729 | 0.408 | 0.349 |
| 50×5 | 0.271 | 2.234 | 5.622 | 1.731 | 1.120 | 1.684 | 0.274 | 0.274 | 0.274 | 0.039 | 0.039 | 0.147 |
| 50×10 | 1.949 | 5.644 | 9.178 | 3.437 | 1.979 | 2.478 | 0.774 | 0.774 | 0.774 | 0.119 | 0.119 | 0.429 |
| 50×20 | 4.997 | 9.191 | 13.372 | 5.823 | 5.001 | 4.339 | 1.613 | 1.613 | 1.872 | 0.611 | 0.543 | 0.211 |
| 100×5 | 0.077 | 1.392 | 6.128 | 0.425 | 0.425 | 0.419 | 0.111 | 0.111 | 0.111 | 0.000 | 0.000 | 0.130 |
| 100×10 | 1.643 | 3.867 | 7.484 | 3.098 | 1.988 | 2.504 | 0.816 | 0.816 | 0.816 | 0.175 | 0.175 | 0.364 |
| 100×20 | 3.693 | 7.223 | 10.676 | 4.672 | 2.872 | 3.610 | 1.389 | 1.389 | 1.430 | 0.422 | 0.303 | 0.072 |
| 200×10 | 0.984 | 3.792 | 5.508 | 2.373 | 1.811 | 1.635 | 0.681 | 0.681 | 0.694 | 0.139 | 0.139 | 0.237 |
| 200×20 | 2.464 | 5.290 | 8.711 | 4.093 | 2.542 | 2.690 | 0.963 | 0.963 | 0.968 | 0.620 | 0.451 | 0.008 |
| 500×20 | 1.679 | 3.527 | 5.724 | 2.381 | 1.702 | 1.806 | 0.645 | 0.645 | 0.640 | 0.084 | 0.147 | 0.153 |
| 平均 | 2.344 | 5.337 | 8.997 | 3.546 | 2.503 | 2.621 | 0.966 | 0.966 | 1.000 | 0.279 | 0.228 | 0.231 |

表 4　算法所得最优值的个数

| 问题 | CDS | Palmer | Gupta | RACS | RAES | Raj | NEH | NEHT | $NEH_{KK1}$ | $FRB4_{12}$ | FRB5 | $DRPFSP_{n/2}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20×5 | 5 | 1 | 1 | 6 | 7 | 6 | 5 | 5 | 5 | 10 | 10 | 9 |
| 20×10 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 5 | 5 | 6 |
| 20×20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 |
| 50×5 | 8 | 2 | 0 | 2 | 4 | 3 | 7 | 7 | 7 | 9 | 9 | 8 |
| 50×10 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 7 | 7 | 5 |
| 50×20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 7 |
| 100×5 | 8 | 3 | 0 | 6 | 6 | 5 | 9 | 9 | 9 | 10 | 10 | 7 |
| 100×10 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 6 | 6 | 4 |
| 100×20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 7 |
| 200×10 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 1 | 7 | 7 | 6 |
| 200×20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 9 |
| 500×20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 4 |
| 合计 | 21 | 6 | 1 | 14 | 17 | 15 | 29 | 29 | 28 | 65 | 67 | 77 |

分析表1－表4，可得出如下结论。

(1)就平均相对偏差而言，各算法的表现从好到差依次为：FRB5＞DRPFSP＞$FRB4_{12}$＞NEH＞NEHT＞$NEH_{KK1}$＞CDS＞RAES＞Raj＞RACS＞Palmer＞Gupta。由此可见，本文所提出的DRPFSP算法和FRB5算法表现最好。虽然DRPFSP算法的总平均相对偏差略高于FRB5算法，但是对于某些组的测试实例，DRPFSP算法所得解的平均相对偏差优于FRB5算法。因此，总体而言本文所提出的DRPFSP算法对于Taillard典型问题的求解是有效的。

(2)就运行时间而言，本文所提出的DRPFSP算法和FRB5算法所用时间最长，总的平均时间分别为0.023s和0.026s。在各测试实例分组中DRPFSP算法的平均执行时间均比FRB5算法短，这验证了算法分析中所得的DRPFSP算法的执行效率比FRB5算法高的结论。另外，虽然DRPFSP算法和FRB5算法所需运行时间较长，但即使是对于最大规模为500×20的Taillard典型问题，其平均执行时间分别为0.125s和0.152s，这样的运行时间在实际生产调度中是可以接受的。

(3)从表4可以看出，本文提出的DRPFSP算法所取得的最优值个数最多，这进一步表明DRPFSP算法对于Taillard典型问题的求解是有效的。

各算法用于求解Taillard典型问题时的平均运行时间和所得解的平均相对偏差如图3、图4所示。

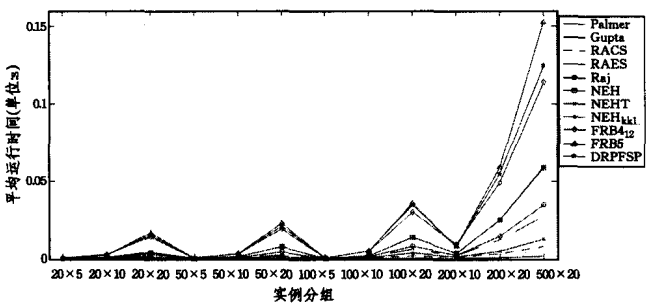


图3　平均运行时间对比(CDS算法除外)

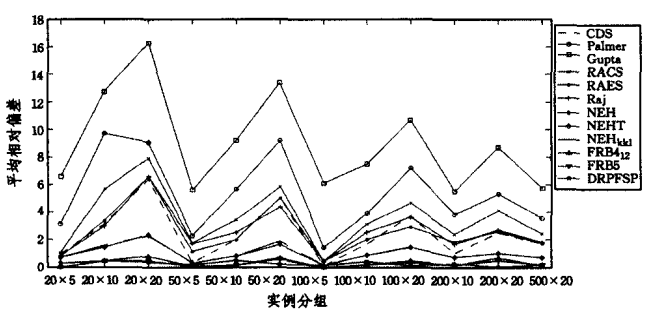


图4　平均相对偏差对比

从图3可以看出，本文所提出的DRPFSP算法的平均运行时间少于FRB5算法，图4则反映出DRPFSP算法所得解的平均相对偏差同FRB5算法是非常接近的，这进一步证明了DRPFSP算法对置换Flow Shop调度问题的求解是有效的。

综上所述，在所选的12个算法中，本文提出的DRPFSP算法和FRB5算法对于置换Flow Shop调度问题的求解性能是最好的。而且在每组测试实例中，DRPFSP算法均可以在更短的时间内求得与FRB5算法相差无几的解。因此，本文提出的DRPFSP算法可以有效地应用于置换Flow Shop调度问题的求解。

**结束语**　本文针对置换Flow Shop调度问题，在对经典的启发式算法进行深入研究的基础上，提出一种求解置换Flow Shop调度问题的DRPFSP算法。该算法的基本思想是通过引入一个$2\times m$的概率矩阵$P$并定义相应的降维函数$f_p(A)=PA$，以将$m$维欧氏空间$R^m$上的加工时间矩阵降维至二维欧氏空间$R^2$，运用Johnson算法对降维所得的$R^2$上的加工时间矩阵进行求解得到$[n]$上的一个序$\pi^0$，然后再结合插入邻域搜索算法的思想，对$\pi^0$进行重新插入和搜索以改进结果，从而获得一个最终的调度顺序$\pi$。在算法执行时，首先对所给的加工时间数据进行了数据标准化处理以消除各数据的量级差距，从而确保算法能够得到一个更好的解。

本文选择11个经典的调度算法和所提出的DRPFSP算法在相同的实验环境下运用各算法对Taillard典型问题进行求解，并从算法所得解的平均相对偏差、运行时间和取得最优值个数等方面来比较各算法的性能。实验结果表明，本文提出的DRPFSP算法在对Taillard典型问题的求解中的表现良好，其可以有效地应用于对置换Flow Shop调度问题的求解。

由于在提出的模型中，对概率矩阵中的元素$P_{2j}$的取值设定还相对简单，若进一步分析其值与问题中的加工时间矩阵之间的关系，是否能够改进算法的解？另一方面，如果将本文的算法与智能优化算法相结合，对算法的解和执行效率将有何影响？这都将是我们进一步研究的方向。

## 参考文献


[1] Garey M R, Johnson D S, Sethi R. The Complexity of Flowshop and Jobshop Scheduling [J]. Mathematics of Operations Research, 1976, 1(2): 117-129

[2] Johnson S M. Optimal two-and three-stage production schedules with setup times included [J]. Naval Research Logistics Quarterly, 1954, 1: 61-68

[3] Campbell H G, Dudek R A, Smith M L. A heuristic algorithm for the n-job, m-machine sequencing problem [J]. Management Science, 1970, 16(10): 630-637

[4] Palmer D. Sequencing jobs through a multi-stage process in the minimum total time-a quick method of obtaining a near optimum [J]. Operational Research Quarterly, 1965, 16(1): 101-107

[5] Gupta J N. A functional heuristic algorithm for the flowshop scheduling problem [J]. Operational Research Quarterly, 1971, 22(1): 39-47

[6] Dannenbring D G. An evaluation of flow shop sequencing heuristics [J]. Management Science, 1977, 23(11): 1174-1182

[7] Nawaz M, Enscore J E E, Ham I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem [J]. OMEGA-the International Journal of Management Science, 1983, 11(1): 91-95

[8] Kalczynski P J, Kamburowski J. On the NEH heuristic for minimizing the makespan in permutation flow shops [J]. OMEGA-the International Journal of Management Science, 2007, 35: 53-60


(下转第107页)

会话。第二种是类缺陷攻击，在 A 作为发起者完成身份验证后，攻击者转发了一条以前截获的信息{Na+1，Na_}Kab，这时 A 接收到这条信息，以为是新的共享密钥和新的随机数，没有做类型检测就确认了交换，把 Na+1 当做新的共享密钥，把 Na_当做了新的随机数。

**结束语** 模型检测技术在网络协议的安全验证中起了重要的作用，由于模型检测的高度自动化和自动生成攻击路径等特点，使得模型检测方法被人们所熟知。但是由于网络技术的发展，协议运行的网络环境也十分复杂，协议参与者们可能会遇到多协议并行运行的情况；协议开发者为了使协议能够有更多的功能和更加强大的安全性，将协议设计得也更加复杂；这便给分析协议的学者们带来了更多的挑战，在模型检测的过程中，可能会忽略多轮协议并行运行的情况，使得并行会话中存在的攻击路径不能被发现；更加复杂的协议内容，使得建立的模型十分复杂，且容易发生状态爆炸问题[10,11]。本文所用的组合身份建模方法解决了两轮协议会话并行运行过程中的模型检测问题，并且这种建模方法简单明了，建模步骤清晰，代码可读性强，为以后复杂环境下协议的模型检测研究提供了参考。未来所要研究的是完善建模方法，使其应用在更加复杂的协议运行环境（三轮并行会话及以上）之中，并且拓展模型检测技术的应用领域[12]，增加模型检测技术的实用性。

## 参 考 文 献


[1] Burrows M, Abadi M, Needham R M. A logic of authentication [J]. Series A, Mathematical and Physical Sciences, 1989, 426 (1871): 233-271

[2] Lowe G. Some new attacks upon security protocols[C]// CSFW, 1996. 1996: 162-169

[3] 周清雷，赵琳，赵东明. 基于串空间模型的 Andrew RPC 协议的分析与验证[J]. 计算机工程与应用，2007，43(13)：153-155
Zhou Qing-lei, Zhao Lin, Zhao Dong-ming. Analysis and verification of Andrew RPC protocol based on strand spaces[J]. Computer Engineering and Applications, 2007, 43(13): 153-155

[4] Holzmann G J. The model checker SPIN[J]. IEEE Transactions on software engineering, 1997, 23(5): 279-295

[5] 吴昌，肖美华，罗敏，等. 安全协议验证模型的高效自动生成[J]. 计算机工程与应用，2010，46(2)：79-82
Wu Chang, Xiao Mei-hua, Luo Min, et al. Effective automatic generation of verification model on security protocol[J]. Computer Engineering and Applications, 2010, 46(2): 79-82

[6] Maggi P, Sisto R. Using SPIN to verify security properties of cryptographic protocols [M] // Model Checking Software. Springer Berlin Heidelberg, 2002: 187-204

[7] Krawczyk U, Sapiecha P. Effective reduction of cryptographic protocols specification for model-checking with Spin[J]. Annales UMCS, Informatica, 2011, 11(3): 27-40

[8] Ruys T C, Holzmann G J. Advanced spin tutorial[M]// Model Checking Software. Springer Berlin Heidelberg, 2004: 304-305

[9] Dolev D, Yao A C. On the security of public key protocols[J]. IEEE Transactions on Information Theory, 1983, 29(2): 198-208

[10] 侯刚，周宽久，勇嘉伟，等. 模型检测中状态爆炸问题研究综述[J]. 计算机科学，2013，40(z6)：77-86，111
Hou Gang, Zhou Kuan-jiu, Yong Jia-wei, et al. Survey of State Explosion Problem in Model Checking[J]. Computer Science, 2013, 40(z6): 77-86, 111

[11] 李兴锋，张新常，杨美红，等. 基于 SPIN 的模块化模型检测方法研究[J]. 电子与信息学报，2011，33(4)：902-907
Li Xing-feng, Zhang Xin-chang, Yang Mei-hong, et al. Study on Modularized Model Checking Method Based on SPIN[J]. Journal of Electronics & Information Technology, 2011, 33(4): 902-907

[12] Yamada Y, Wasaki K. Automatic generation of SPIN model checking code from UML activity diagram and its application to Web application design[C]// 2011 7th International Conference on Digital Content, Multimedia Technology and its Applications (IDCTA). IEEE, 2011: 139-144


(上接第 73 页)


[9] Kalczynski P J, Kamburowski J. An improved NEH heuristic to minimize makespan in permutation flow shops [J]. Computers & Operations Research, 2008, 35(9): 3001-3008

[10] Farahmand R S, Ruiz R, Boroojerdian N. New high performing heuristics for minimizing makespan in permutation flowshops [J]. OMEGA-the International Jouranal of Management Science, 2009, 37: 331-345

[11] Sioud A, Gravel M, Gagne C. A genetic algorithm for solving a hybrid flexible flowshop with sequence dependent setup times [C] // 2013 IEEE Congress on Evolutionary Computation (CEC). 2013: 2512-2516

[12] Wang X, Tang L. A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers [J]. Computers & Operations Research, 2009, 36(3): 907-918

[13] Lin S W, Ying K C. Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems [J]. International Journal of Production Research, 2009, 47(5): 1411-1424

[14] Yagmahan B, Yenisey M M. A multi-objective ant colony system algorithm for flow shop scheduling problem[J]. Expert Systems with Applications, 2010, 37(2): 1361-1368

[15] Liao C J, Tjandradjaja E, Chung T P. An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem [J]. Applied Soft Computing, 2012, 12 (6): 1755-1764

[16] Taillard E. Some effective heuristic methods for the flowshop sequencing problem [J]. European Journal of Operational Research, 1990, 47: 67-74