DOI：10.3724/SP.J.1219.2012.00514

瓶颈指向的启发式算法求解混合流水车间调度问题

屈国强 [1,2]

（1. 北京科技大学经济管理学院，北京 100083； 2. 河南理工大学经济管理学院，河南 焦作 454000）

摘 要：针对以最小化时间表长为目标的复杂混合流水车间调度问题，提出了一种将机器布局和工件加工时间特征紧密结合的启发式算法．首先，充分利用各阶段平均机器负荷一般不相等的特点确定瓶颈阶段，构建初始工件排序．其次，针对在瓶颈阶段前加工时间较短而瓶颈阶段后加工时间相对较长的工件，在第 1 阶段优先开始加工．同时，在瓶颈阶段前的每一个阶段，每当有工件等待加工或同时完工时，优先选择瓶颈阶段前剩余加工时间最短的工件加工；在瓶颈阶段以及瓶颈阶段之后，则优先选择这台机器后剩余加工时间最长的工件加工．最后，采用工件交换和插入操作改进初始调度．用 Carlier 和 Neron 的 Benchmark 算例测试提出的启发式算法．将计算结果与 NEH 启发式算法进行了比较，平均偏差降低了 0.055 5%，表明这个启发式算法是有效的．

关键词：混合流水车间；调度；瓶颈；启发式；时间表长

中图分类号：TP278 **文献标识码**：A **文章编号**：1002-0411(2012)-04-0514-08

Bottleneck Focused Heuristic Algorithm for Hybrid Flow Shop Scheduling Problem

QU Guoqiang[1,2]

(1. *School of Economics and Management*, *University of Science and Technology Beijing*, *Beijing* 100083, *China*;
2. *School of Economics and Management*, *Henan Polytechnic University*, *Jiaozuo* 454000, *China*)

Abstract: For the complex shop scheduling problem of hybrid flow shop with makespan minimization criterion, a heuristic algorithm cooperating the characters of machine configuration and job processing time intensively is proposed. Firstly, the bottleneck stage is identified using the character that the average processing time is not equal at every stage, and then an initial job sequence is constructed. Secondly, a job is given higher priority at the first stage when it takes shorter processing time before the bottleneck stage and longer processing time after the bottleneck stage. Meanwhile, at every stage before the bottleneck, if there are jobs waiting for processing or completed simultaneously, the priority is given to the job with shortest remain processing time before the bottleneck; at or after the bottleneck stage, it is given to the job with longest remaining processing time. Finally, pairwise interchange and insert operation are used to improve the initial scheduling. The performance of the proposed heuristic algorithm is tested using Carlier and Neron's benchmark problems. The computational results show that compared with the well-known NEH heuristic algorithm, the average deviation is reduced 0.055 5%, which verifies the effectiveness of the proposed heuristic algorithm.

Keywords: hybrid flow shop; scheduling; bottleneck; heuristic; makespan

1 引言（Introduction）

由于扩展生产能力、平衡机器负荷及同时使用效率不等的机器等实际生产需要，生产线上增加并行机器的情况日益涌现，引起了众多研究人员的关注．混合流水车间（hybrid flow shop，HFS），在理论上是一个典型的有并行机处理同样操作的流水车间（flow shop，FS），在实践中具有广泛的工程应用背景．如在钢铁生产企业中，生产工艺一般可以分为炼钢、精炼和连铸三个阶段，每个阶段又有多台并行机器, 就是一个典型的 HFS[1]．

文 [2] 从计算复杂性、调度目标和求解方法三个方面对早期（1995 年前后）的 HFS 研究进行了综述，指出大多数 HFS 调度问题是非确定性多项式（NP）难题．基于启发式的指派规则在现实中应用广泛．文 [3] 以最小化时间表长和平均流程时间为目标函数，对求解加工时不可中断的同速机 HFS 的精确算法（分支定界和约束传播）进行了综述，分别给出了基于工件和基于机器的下界．文 [4-6] 分别从启发式算法的集成算法与分解算法分类，精确算法、启发式算法和元启发式算法分类及研究趋势、生产系统和求解方法等角度对 HFS 调度问题进行了非常详尽的综述．

通讯作者：屈国强，guoqiangqu@163.com 收稿/录用/修回：2011-04-14/2011-07-22/2012-05-02

HFS 调度问题求解算法可分为精确算法、启发式算法和元启发式算法三种类型. 精确算法如分支定界、约束规划等具有理论研究上的意义，计算量较大，适用于小规模问题；元启发式算法，如遗传算法、禁忌搜索、模拟退火等，研究较多，适用于中大规模问题，耗时较长；启发式算法，如 NEH 算法 [7]、Palmer 算法 [8]、CDS 算法 [9] 等，虽不能保证得到最优解，但耗时较少，应用广泛，成为研究的一个热点.

针对 HFS 调度问题难于求解的特性，启发式算法在调度效果与计算时间之间做出折中，以较小的计算量获得近似最优解. 因其具有易于实现、计算复杂度低等优点，许多年来一直受到学者们的广泛关注. 自 1954 年 Johnson 对 2 阶段的排列排序流水车间调度问题提出了一个简便而又相当巧妙的多项式最优算法——Johnson 规则——以后，许多研究人员尝试扩展 Johnson 规则来求解 HFS 调度问题. 文 [10] 指出许多研究借鉴了 Johnson 规则，提出用求解 FS 调度问题的启发式算法求解 HFS 调度问题，并且对 HFS 调度启发式算法进行了总结. 文 [11] 提出循环启发式、基于旅行商问题可以看作改进的 NEH 插入启发式，并提出 2 种扩展 Johnson 规则的启发式算法，求解安装时间依赖于顺序的柔性流水线调度问题，计算结果表明扩展 Johnson 规则的启发式算法求解效果较好. 文 [12] 通过不断删除中间加工阶段，逐渐构造一个 2 阶段工件加工时间，利用 Johnson 规则构建工件初始排序.

基于瓶颈分析思想构建启发式算法已经用于求解许多车间调度问题，以求解作业车间调度问题的转移瓶颈启发式算法（SBH）最具代表性 [13]. 文 [14] 对于现有瓶颈识别方法进行分析，提出用正交实验识别作业车间瓶颈. 对于 FS 调度问题，文 [15] 区分瓶颈机器前后，计算工件的优先级后采用不同的指派规则求解. 文 [16] 对瓶颈阶段的机器建立带有到达时间和传递时间约束的单机调度模型并进行优化求解，通过不断修正工件到达瓶颈阶段机器时间和传递时间建立非瓶颈阶段机器的调度. 对于 HFS 调度问题，文 [17] 通过计算工件在瓶颈阶段的上下游阶段时间之和，基于 Johnson 规则的思想确定一个工件排序，用迭代插入的方式寻优求解；文 [18] 以求解工件总延迟为目标函数，首先用瓶颈阶段前的上游阶段工件加工时间之和作为工件到达瓶颈阶段的时间，然后聚焦于瓶颈阶段，用求解并行机（PM）调度问题的方法构建瓶颈阶段机器初始调度，最后迭代使用后向调度和前向调度确定工件到达瓶颈阶段机器的时间后采用不同的指派规则求解. 文 [19] 采用模拟退火改进第 1 阶段工件排序后，用最先可用机器（FAM）优先规则选择机器，最大剩余加工时间（LRPT）优先规则选择工件，数值计算表明负荷不等的机器布局效果最好，原因在于存在瓶颈阶段. 文 [4] 也指出基于阶段的分解方法中，首先考虑哪一个阶段依赖于 HFS 的特点. 如果能够确认一个瓶颈阶段，首先集中考虑瓶颈阶段是有利的；若不能，常用的方法是从第 1 阶段开始.

在前文研究的基础上，综合考虑了瓶颈阶段机器前后工件加工时间特点构建工件排序，使一些工件尽快达到瓶颈阶段机器开始加工，与另外一些工件在瓶颈阶段机器加工后尽快离开下游机器之间取得平衡；根据机器加工状态，扩大寻优空间，生成工件的排列排序或非排列排序初始调度；通过工件成对交换和插入进行邻域搜索寻求改进初始调度. 将本文算法应用于 Carlier 和 Neron 提出的 Benchmark 算例 [20]，实验结果验证了算法的有效性.

2　问题描述（Description of the problem）

为了叙述方便，引入如下符号：工件集合 J，$J=\{1,2,\cdots,n\}$；工件编号 j，$j\in J$；阶段集合 I，$I=\{1,2,\cdots,s\}$；阶段编号 i，$i\in I$；阶段 i 上的平行同速机数 $M^{(i)}$；机器编号 m，$m=1,2,\cdots,M^{(i)}$；工件 j 在阶段 i 上的加工时间 p_{ij}. 决策变量包括：阶段 i 上的机器 m 所加工的工件有序集 π_{im}，$\pi_{im}=\{\pi_{im}(1),\pi_{im}(2),\cdots,\pi_{im}(n_{im})\}$，其中 $n_{im}=|\pi_{im}|$；工件的机器指派和加工顺序 π，$\pi=\bigcup_{i=1}^{s}\{\pi_{i1},\pi_{i2},\cdots,\pi_{iM}^{(i)}\}$；工件 j 在阶段 i 上的开工时间 t_{ij}；工件 j 在阶段 i 上的完工时间 c_{ij}；最大完工时间 $C_{\max}$.

HFS 调度问题可以描述为 n 个工件要在 s 个阶段的流水车间上加工，其中阶段 i 具有 $M^{(i)}$ 台同速平行机，且至少有 1 个阶段存在 2 台以上的平行机，并满足以下基本假定与约束：(1) 在开始调度时，机器与工件均处于可用状态；(2) 每台机器在某一时刻最多只能加工 1 个工件；(3) 每个工件在某一时刻最多只能被 1 台机器加工；(4) 工件在某台机器上开始加工后，不允许中断，即 $c_{ij}=t_{ij}+p_{ij}$；(5) 相邻阶段之间具有容量无限的缓冲区；(6) 工件 j 在阶段 i 上的加工时间 p_{ij} 已知；(7) 机器调整时间和工件传送时间忽略不计.

研究的目标是在满足各项假定与约束关系的条件下，把工件 j 指派到阶段 i 加工的机器 m 上，对同一台机器上加工的工件进行排序，即确定一个工件指派和排序方案 π，并确定工件 j 在阶段 i 上

的开工时间 t_{ij} 和完工时间 c_{ij}，以最小化最大的时间表长 $C_{\max}$. 采用 Graham 等提出并由 Vignie 等改进的标准三元组 $\alpha|\beta|\gamma$ 表示法 [21]，将此问题记为 $FHs,((PM^{(i)})_{i=1}^{s})\|C_{\max}$，其中 FHs 表示由 s 个阶段构成的混合流水车间，$PM^{(i)}$ 表示阶段 i 中有 $M^{(i)}$ 台同速平行机.

3 算法思路及实现步骤（The basic idea and the steps of the algorithm）

3.1 算法思路

约束理论（TOC）指出：瓶颈决定了整条生产线的性能，可以通过充分利用瓶颈资源来提高整个生产系统的性能. 瓶颈是指实际生产能力小于或等于生产负荷的资源，它限制了整个生产系统的产出速度.

对于 HFS，各阶段的机器数可能是不同的，各阶段工件的加工时间也可能是不等的. 各阶段机器数相同或各阶段工件加工时间相等的情况较少. 因此，在 HFS 中可能存在一个阶段，其平均机器加工负荷最大，容易发生拥塞现象，成为 HFS 制约 $C_{\max}$ 的主要因素之一. 因此，选择平均机器加工负荷最大的阶段作为 HFS 的瓶颈阶段. 瓶颈阶段之前或之后的阶段分别称为上游或下游阶段. 与负荷相对较小的阶段相比较，瓶颈阶段损失的时间对目标函数的影响更严重.

文 [22] 扩展了 5 种求解 FS 调度问题的启发式方法来求解 HFS 调度问题，采用回归分析的方法研究了问题的结构特点（阶段数、各阶段并行机数、工件数、工件加工时间等）与采用不同启发式算法求解 HFS 调度问题效果之间的关系，研究表明其主要的影响因素是工件特点、工件数、机器阶段和各阶段并行机数，并且发现与启发式规则相比，问题结构特点对于 HFS 调度问题求解效果影响较大. 因此，若能够在算法中充分考虑加工机器的布局以及工件在各阶段加工时间的特点，优先考虑瓶颈阶段的工件加工顺序，应该可以提高算法的求解效率和质量.

求解 $F2\|C_{\max}$ 的 Johnson 规则充分利用了工件在 2 台机器上加工时间不等的特点，取得了明显的效果. Johnson 规则的基本思想是把工件分成 2 个部分，将能够快速地通过前一台机器的工件放在工件排序的前部，将能够快速地通过后一台机器的工件放在工件排序的后部，使工件能够快速开始和快速通过 [12]. Palmer 算法基于 Johnson 规则的思想，按照机器的前后顺序，通过计算斜率指数对工件进行排序，加工时间趋于增加（减少）的工件有较高（低）的优先级. CDS 算法也是借鉴了 Johnson 规则的思想进行工件排序. 文 [4] 指出：最小化 $C_{\max}$ 时，经常使用扩展的 Johnson 规则来构建 HFS 第 1 阶段的工件排序.

受此启发，在 HFS 中：上游阶段加工时间较短而下游阶段加工时间相对较长的工件，经过上游阶段加工能够较快到达瓶颈阶段，因此在第 1 阶段应优先开始加工；而上游阶段加工时间较长而下游阶段加工时间相对较短的工件，经过上游阶段加工需要相对较长时间才能到达瓶颈阶段，因此在第 1 阶段应相对稍后开始加工. 这样可以使瓶颈阶段机器尽早开始加工，尽可能减小瓶颈阶段机器间歇等待时间，并且使工件在快速通过上游阶段，与快速离开下游阶段之间取得平衡. 同时，在上游阶段机器，若有数个工件等待加工或同时完工，优先选择能够尽快到达瓶颈阶段机器开始加工的工件，以使瓶颈阶段机器间歇等待时间最小；若是瓶颈阶段及下游阶段机器，优先选择这个阶段后剩余加工时间最长的工件，使下游阶段机器尽可能处于加工状态，减少下游阶段机器间歇等待时间. 若有机器空闲，工件到达后马上加工. 经过这样的调整，产生的工件排序可能是非排列排序的，可行解空间就扩大了. 由于假设在 HFS 中每个阶段的并行机都是同速机，所以采用 FAM 规则来选择某一个阶段的加工机器.

这种思路考虑了在瓶颈阶段的上游及下游阶段工件加工时间特点，也考虑了在上游或下游阶段有数个工件等待加工或同时完工的工件加工时间特点，但没有具体考虑工件在各个阶段加工时间之间的关系及相互影响. 文 [23] 指出由于 FS 调度问题的复杂性，单靠给定一个规则，然后按其一次排序是不可能得到很好的解的. 而 HFS 是 FS 的扩展，比 FS 问题更为复杂，这个结论对于 HFS 问题也是成立的. 因此，对于某一个加工阶段的同一台机器以及不同机器加工的工件，采用工件成对交换（pairwise interchange）的方式进行邻域搜索，然后通过工件插入（insert）的方式，改变邻域结构，扩大搜索范围，再次进行邻域搜索以求改进.

3.2 算法步骤

为简便起见，称本文算法为 BFH（bottleneck focused heuristic），具体步骤为：

步骤 1 识别瓶颈阶段. 计算每一个阶段 i 的平均机器加工负荷 l_i：$l_i=\sum_{j=1}^{n}p_{ij}/M^{(i)}$，取最大的 l_i 所在的阶段为瓶颈阶段 b. 若有多个阶段的 l_i 相等，则位于 HFS 后面的阶段为瓶颈阶段 b.

步骤 2　初始工件排序. (1) 若 $b=1$，计算 $\sum_{i=2}^{m} p_{ij}$ 后按非增顺序排序，以 π 表示. (2) 若 $b=m$，计算 $\sum_{i=1}^{m-1} p_{ij}$ 后按非降顺序排序，以 π 表示. (3) 若 $1<b<m$，计算 $\sum_{i=1}^{b-1} p_{ij}$ 后按非降顺序排序，以 π_1 表示；计算 $\sum_{i=b+1}^{m} p_{ij}$ 后按非增顺序排序，以 π_2 表示. 用 π 表示包含 n 个可填充位置的有序集合，置 $\pi=\varnothing$. 把 π_1 中最前面的工件分配给 π 中第 1 个没有填充的位置；把 π_2 中最后面的工件分配给 π 中最后 1 个没有填充的位置. 若有工件已经分配给 π，则从 π_1 和 π_2 中删除，直到 n 个工件分配完毕.

步骤 3　生成初始的工件的机器指派和加工顺序 π'_{im}.

步骤 3.1　生成第 1 阶段中工件的机器指派和加工顺序. 取 π 中前 $M^{(1)}$ 个工件依次排在第 1 阶段 $M^{(1)}$ 台机器上加工，每当有工件加工完毕，则取 π 中最前面的工件指派到该机器上加工，直到全部工件指派完毕.

步骤 3.2　在上游阶段 i' 中，若有机器空闲，工件到达后立即加工；若机器繁忙，有数个工件等待加工或同时完工，则对工件 j'，计算 $\sum_{i=i'}^{b-1} pi'j'$，取 $\min\left(\sum_{i=i'}^{b-1} pi'j'\right)$ 的工件优先加工.

步骤 3.3　在瓶颈及下游阶段 i'' 中，若有机器空闲，工件到达后立即加工；若机器繁忙，有数个工件等待加工或同时完工，对这些工件 j''，计算 $\sum_{i=i''}^{m} pi''j''$，取 $\max\left(\sum_{i=i''}^{m} pi''j''\right)$ 的工件优先加工.

步骤 4　改进初始的工件的机器指派和加工顺序 π'.

步骤 4.1　对于初始的工件的机器指派和加工顺序 π'，计算 $C_{\max}(\pi')$；阶段计数器 $i=0$.

步骤 4.2　令 $i=i+1$.

步骤 4.3　对于第 i 阶段中工件的机器指派和加工顺序，首先采用同一台机器上以及同一阶段不同机器上工件成对交换进行邻域搜索，直到没有改进为止；然后采用同一台机器上，以及同一阶段不同机器上工件插入方式进行邻域搜索，同样直到没有改进为止. 记最小 $C_{\max}$ 的一个工件指派和加工顺序为 π''_{im}. 若 $C_{\max}(\pi''_{im})<C_{\max}(\pi'_{im})$，则 $\pi'_{im}=\pi''_{im}$.

步骤 4.4　若 $i<s$，转步骤 4.2；否则 $\pi_{im}=\pi'_{im}$，输出调度方案，结束.

本文以 Carlier 和 Neron 提出的 Benchmark 算例集中的算例 j10c5a5 为例来说明算法 BFH 的实现过程. 算例 j10c5a5 有 10 个工件及 5 个加工阶段，加工机器在各阶段分布情况是 3-3-1-3-3，工件在各阶段加工时间见表 1，其中第 3 阶段平均机器负荷最大，是瓶颈阶段. BFH 计算的工件排序 $\pi_1=\{5,10,8,1,4,3,9,6,2,7\}$，$\pi_2=\{10,2,1,6,5,8,3,7,9,4\}$，工件初始排序 $\pi=\{5,10,8,1,6,2,3,7,9,4\}$，计算过程见表 2. 在第 3 阶段，当工件 10 加工完毕时，工件 1、3、6 和 8 同时处于等待加工状态，这 4 个工件在瓶颈阶段 3 后的加工时间之和分别是 24、16、21 和 17，其中工件 1 在后续阶段加工时间之和最大. 因此，此时选择工件 1 紧接在工件 10 之后加工，其余类似.

表 1　算例 j10c5a5 的加工时间

Tab.1　Processing time of the instance j10c5a5

j	1	2	3	4	5	6	7	8	9	10
p_{1j}	12	14	6	3	3	6	11	6	8	3
p_{2j}	3	12	13	12	6	15	15	6	11	8
p_{3j}	14	13	6	13	11	11	10	11	5	8
p_{4j}	14	14	6	4	14	7	8	10	10	13
p_{5j}	10	10	10	7	7	14	7	7	4	15

表 2　算例 j10c5a5 的初始工件排序 π 构建过程

Tab.2　The initial job sequence π constructing process of the instance j10c5a5

工件排序 π_1	工件排序 π_2	工件分配	初始工件排序 π
5,10,8,1,4,3,9,6,2,7	10,2,1,6,5,8,3,7,9,4	工件 5	5,*,*,*,*,*,*,*,*,*
10,8,1,4,3,9,6,2,7	10,2,1,6,8,3,7,9,4	工件 4	5,*,*,*,*,*,*,*,*,4
10,8,1,3,9,6,2,7	10,2,1,6,8,3,7,9	工件 10	5,10,*,*,*,*,*,*,*,4
8,1,3,9,6,2,7	2,1,6,8,3,7,9	工件 9	5,10,*,*,*,*,*,*,9,4
8,1,3,6,2,7	2,1,6,8,3,7	工件 8	5,10,8,*,*,*,*,*,9,4
1,3,6,2,7	2,1,6,3,7	工件 7	5,10,8,*,*,*,*,7,9,4
1,3,6,2	2,1,6,3	工件 1	5,10,8,1,*,*,*,7,9,4
3,6,2	2,6,3	工件 3	5,10,8,1,*,*,3,7,9,4
6,2	2,6	工件 6	5,10,8,1,6,*,3,7,9,4
2	2	工件 2	5,10,8,1,6,2,3,7,9,4

对于这个算例，没有经过步骤 4 而直接求得了最优解，$C_{\max}=122$，调度方案见图 1. 用 NEH 算法求得的工件排序是 $\pi=\{4,8,5,3,10,6,1,2,7,9\}$，$C_{\max}=125$，调度方案见图 2.

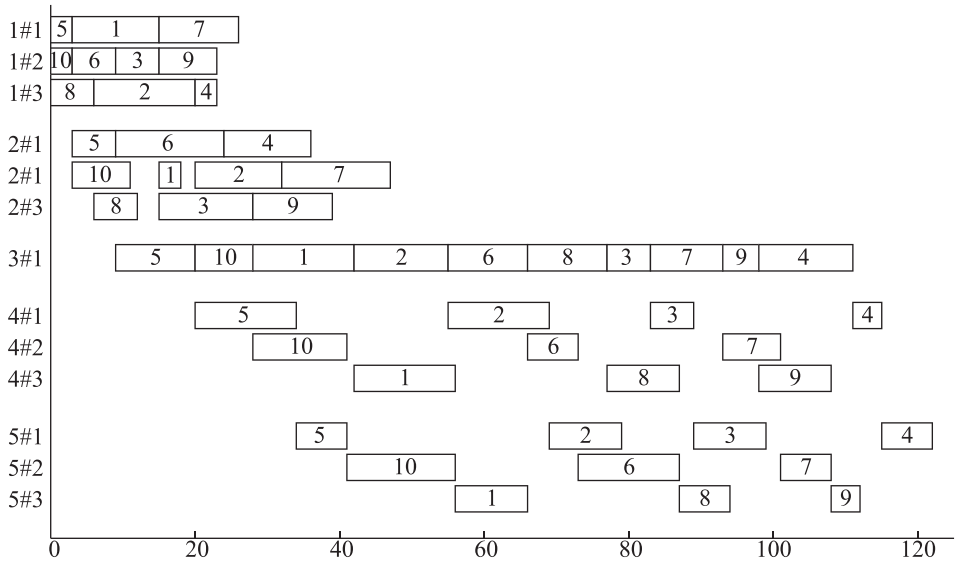

图 1 采用 BFH 算法求解算例 j10c5a5 的甘特图（$C_{\max}=122$）

Fig.1 Gantt chart of the instance j10c5a5 solved by BFH ($C_{\max}=122$)

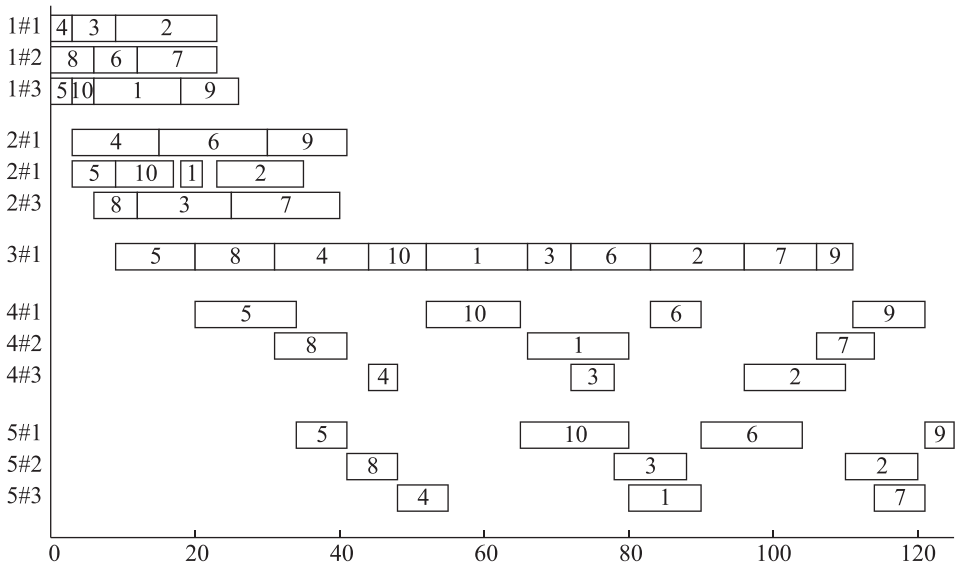

图 2 采用 NEH 算法求解算例 j10c5a5 的甘特图（$C_{\max}=125$）

Fig.2 Gantt chart of the instance j10c5a5 solved by NEH ($C_{\max}=125$)

4 数据实验（Data experiment）

4.1 测试数据

采用 MATLAB7.6 在 Pentium4/CPU3.00GHz/RAM512MB 的机器上编程实现上述算法.

测试数据来自 Carlier 和 Neron 提出的 Benchmark 算例，这些算例的规模从 10 个工件、5 个阶段变化到 15 个工件、10 个阶段，工件在各阶段的加工时间 $p_{ij}\in(3,20)$. 根据工件数、阶段数及每个阶段的平行同速机数不同，这 77 个算例被分为 13 组，有 a、b、c 和 d 四种类型. a 类和 b 类的算例都有一个阶段并且仅有一台机器，而其它阶段都有 3 台机器；c 类的算例的中间阶段有 2 台机器，其它阶段都有 3 台机器；d 类的算例的各阶段都有 3 台机器（唯一的一个例外是算例 j15c5d1）. 例如 j10c5a3 表示 10 个工件、5 个阶段，a 类中的第 3 个算例. 文 [24] 利用分支定界法来求解这些算例，j10c10c1 根据求解难度将这些算例分成易解和难解两组，易解算例包括 6 个 j10c10c* 类，以及所有的 23 个 a 类和 24 个 b 类的算例，共计 53 个算例；难解算例包括 6 个 j10c5c* 类、6 个 j10c5d* 类、6 个 j15c5c* 类和 6 个

j15c5d* 类的算例，共计 24 个算例，其中 * 代表 6 个 j10c10c 类算例的尾数，即 1、2、3、4、5、6.

目前主要采用精确算法（如分支定界、基于约束传播和树搜索的限制深度的差异搜索）和元启发式算法（如正逆序结合的遗传算法、蚁群优化、人工免疫系统）等算法求解这些算例，以衡量和比较各种求解 HFS 调度问题算法的效果，例如文 [25] 采用了人工免疫系统求解此算例集，并给出了已知最好下界 LB'. 本文也采用了此已知最好下界 LB' 来衡量所提出的启发式算法的求解效果. 研究表明，以最小化 $C_{\max}$ 为目标求解 HFS 时，几种常用启发式算法以 NEH 表现最好 [6]. 因此，选择 NEH 作为对比算法. 与相关文献类似，通过以下 3 个指标衡量算法的有效性：(1) 最优比例，指算法求得已知最好下界的算例的个数占此类型的算例总数的百分比；(2) 偏差 d，指算法求得的 $C_{\max}$ 与已知最好下界 LB' 的偏差，$d=((C_{\max}-LB')/LB')\times 100$；(3) 耗时 s，指 CPU 的计算时间，以 s 为单位.

4.2　实验结果与比较

对此 53 个易解、24 个难解共计 4 种类型的 77 个算例，采用本文提出的 BFH 与 NEH 进行计算，计算结果见表 3 和表 4.

表 3　NEH 与 BFH 算法计算结果——易解算例

Tab.3　Computational results of NEH and BFH algorithms – Easy instance

序号	算例	LB'	NEH			BFH		
			$C_{\max}$	d	s	$C_{\max}$	d	s
1	j10c5a2	88	88	0	0.108 2	88	0	0.006 4
2	j10c5a3	117	117	0	0.107 5	117	0	0.006 6
3	j10c5a4	121	121	0	0.107 7	121	0	0.006 2
4	j10c5a5	122	125	2.459 0	0.107 1	122	0	0.006 8
5	j10c5a6	110	115	4.545 5	0.104 4	110	0	0.006 5
6	j10c5b1	130	130	0	0.116 9	130	0	0.005 9
7	j10c5b2	107	107	0	0.099 7	107	0	0.005 4
8	j10c5b3	109	110	0.917 4	0.097 4	109	0	0.005 2
9	j10c5b4	122	122	0	0.100 7	122	0	0.005 67
10	j10c5b5	153	153	0	0.101 9	153	0	0.005
11	j10c5b6	115	115	0	0.099 2	115	0	0.005 4
12	j10c10a1	139	148	6.478 2	0.215 1	139	0	0.018 5
13	j10c10a2	158	164	3.797 5	0.207 5	158	0	0.0178
14	j10c10a3	148	148	0	0.202 8	148	0	0.013 6
15	j10c10a4	149	161	8.053 7	0.204 3	149	0	0.014 7
16	j10c10a5	148	154	4.054 1	0.207 0	148	0	0.012 4
17	j10c10a6	146	148	1.369 9	0.210 4	146	0	0.016 7
18	j10c10b1	163	163	0	0.403 3	163	0	0.009 4
19	j10c10b2	157	157	0	0.479 3	157	0	0.009 3
20	j10c10b3	169	169	0	0.400 8	169	0	0.009 6
21	j10c10b4	159	159	0	0.405 0	159	0	0.009 84
22	j10c10b5	165	165	0	0.382 2	165	0	0.009 5
23	j10c10b6	165	165	0	0.369 1	165	0	0.009 7
24	j15c5a1	178	181	1.685 4	0.3876	178	0	0.009 2
25	j15c5a2	165	165	0	0.377 3	165	0	0.009 3
26	j15c5a3	130	130	0	0.391 8	130	0	0.009 8
27	j15c5a4	156	156	0	0.375 3	156	0	0.009 5
28	j15c5a5	164	174	6.097 6	0.376 6	164	0	0.009 5
29	j15c5a6	178	178	0	0.385 4	178	0	0.009 2
30	j15c5b1	170	170	0	0.360 3	170	0	0.007 5
31	j15c5b2	152	152	0	0.358 9	152	0	0.007 4
32	j15c5b3	157	157	0	0.350 6	157	0	0.007 3
33	j15c5b4	147	147	0	0.349 5	147	0	0.007 5
34	j15c5b5	166	166	0	0.348 3	166	0	0.007 7
35	j15c5b6	175	175	0	0.347 6	175	0	0.007 4
36	j15c10a1	236	236	0	1.436 3	236	0	0.017 8
37	j15c10a2	200	202	1.000 0	1.3547	201	0.5000	0.013 6
38	j15c10a3	198	198	0	1.326 2	198	0	0.018 6
39	j15c10a4	225	225	0	1.344 5	225	0	0.013 5
40	j15c10a5	182	182	0	1.341 7	182	0	0.011 5
41	j15c10a6	200	200	0	1.343 9	200	0	0.016 8
42	j15c10b1	222	222	0	0.775 0	224	0.900 9	0.013 6
43	j15c10b2	187	187	0	0.756 6	187	0	0.015 4
44	j15c10b3	222	222	0	0.775 8	222	0	0.017 4
45	j15c10b4	221	221	0	0.742 3	221	0	0.018 3
46	j15c10b5	200	202	1.000 0	0.751 7	202	1.000 0	0.014 5
47	j15c10b6	219	219	0	0.757 5	219	0	0.016 4
48	j10c10c1	113	122	7.964 6	0.213 9	120	6.194 7	0.013 6
49	j10c10c2	116	123	6.034 5	0.209 0	120	3.448 3	0.016 3
50	j10c10c3	98	120	22.449 0	0.211 3	123	25.510 2	0.019 4
51	j10c10c4	103	126	22.330 1	0.209 6	128	24.271 8	0.016 2
52	j10c10c5	121	131	8.264 5	0.223 6	140	15.702 5	0.013 6
53	j10c10c6	97	106	9.278 4	0.215 1	117	20.618 6	0.013 3

首先，从机器布局的角度进行比较. 从表 5 可以看出，对于 a 类的 23 个算例，NEH 和 BFH 分别有 13 个和 22 个算例求得了已知下界，最优比例分别达到了 56.521 7%和 95.652 2%，平均偏差为 1.719 1%和 0.021 7%，表明求解效果非常好，并且在最优比例和平均偏差这两个指标上，BFH 远远超过了 NEH. 对于 b 类的 24 个算例，NEH 和 BFH 都有 22 个算例

求得了已知下界，最优比例都是 91.666 7%，平均偏差分别是 0.079 9%和 0.079 2%，表明求解效果非常好，在最优比例和平均偏差这两个指标上，BFH 与 NEH 相当. 对于 c 类的 18 个算例，NEH 和 BFH 平均偏差分别是 8.690 6%和 9.137 2%，且没有一个算例求得已知下界，表明求解效果不好. 对于 d 类的 12 个算例，NEH 和 BFH 平均偏差分别是 12.408 3%和 14.638 3%，都是仅有一个算例求得已知下界，表明求解效果也是不好的. 同时，从最优比例和平均偏差这两个指标来看，a 类和 b 类的算例求解效果都是优于 c 类和 d 类的算例.

表 4 NEH 与 BFH 算法计算结果——难解算例

Tab.4 Computational results of NEH and BFH algorithms – Hard instance

序号	算例	LB'	NEH			BFH		
			$C_{\max}$	d	s	$C_{\max}$	d	s
1	j10c5c1	68	71	4.411 8	0.111 4	73	7.352 9	0.007 3
2	j10c5c2	74	77	4.054 1	0.112 7	75	1.351 4	0.006 5
3	j10c5c3	71	75	5.633 8	0.116 2	75	5.633 8	0.006 2
4	j10c5c4	66	70	6.060 1	0.111 9	71	7.575 8	0.006 1
5	j10c5c5	78	82	5.128 2	0.111 2	79	1.282 1	0.006 8
6	j10c5c6	69	73	5.797 1	0.112 7	74	7.246 4	0.006 5
7	j15c5c1	85	91	7.058 8	0.417 0	93	9.411 8	0.009 3
8	j15c5c2	90	99	10.000 0	0.438 5	97	7.777 8	0.009 6
9	j15c5c3	87	97	11.494 3	0.447 2	89	2.298 9	0.009 2
10	j15c5c4	89	96	7.865 2	0.406 7	95	6.741 6	0.009 6
11	j15c5c5	73	79	8.219 2	0.398 3	81	10.958 9	0.009 7
12	j15c5c6	91	95	4.395 6	0.401 1	92	1.098 9	0.009 6
13	j10c5d1	66	72	9.090 9	0.183 5	68	3.030 3	0.006 6
14	j10c5d2	73	77	5.479 5	0.251 2	79	8.219 2	0.006 3
15	j10c5d3	64	67	4.687 5	0.161 9	69	7.812 5	0.006 8
16	j10c5d4	70	75	7.142 9	0.177 4	75	7.142 9	0.006 4
17	j10c5d5	66	72	9.090 9	0.229 6	71	7.575 8	0.006 1
18	j10c5d6	62	67	8.064 5	0.193 1	67	8.064 5	0.006 0
19	j15c5d1	167	167	0	0.373 6	167	0	0.008 3
20	j15c5d2	82	92	12.195 1	0.487 8	98	19.512 2	0.009 4
21	j15c5d3	77	88	14.285 7	0.499 3	90	16.883 1	0.009 1
22	j15c5d4	61	89	45.901 6	0.646 7	94	54.098 4	0.009 6
23	j15c5d5	67	84	25.373 1	0.697 1	85	26.865 7	0.009 3
24	j15c5d6	79	85	7.594 9	0.6438	92	16.455 7	0.009 9

其次，从算例的性质是易解或难解的角度进行比较. 从表 6 可以看出，对于 53 个易解算例，NEH 与 BFH 分别有 35 个和 44 个算例求得了已知下界，最优比例分别达到了 66.037 7%和 83.018 9%，平均偏差分别是 2.222 3%和 1.851 8%，表明求解效果非常好，并且在最优比例和平均偏差这两个指标上，BFH 都超过了 NEH；对于 24 个难解算例，NEH 和 BFH 都仅有 1 个算例求得了已知下界，最优比例都是 4.1667%，平均偏差分别是 9.542 7%和 10.182 9%，表明求解效果不好. 显然，易解算例的求解效果是优于难解算例的.

表 5 NEH 与 BFH 算法的计算结果比较 (1)

Tab.5 Comparison between the results of NEH and BFH algorithms (1)

算例类型		a 类	b 类	c 类	d 类
最优比例 /%	NEH	56.521 7	91.666 7	0	8.333 3
	BFH	95.652 2	91.666 7	0	8.333 3
平均偏差 /%	NEH	1.719 1	0.079 9	8.690 6	12.408 3
	BFH	0.021 7	0.079 2	9.137 2	14.638 3
平均耗时 /s	NEH	0.531 4	0.405 4	0.248 2	0.378 8
	BFH	0.011 9	0.009 6	0.010 5	0.007 8

表 6 NEH 与 BFH 计算结果比较 (2)

Tab.6 Comparison between the results of NEH and BFH (2)

算例性质	最优比例 /%		平均偏差 /%		平均耗时 /s	
	NEH	BFH	NEH	BFH	NEH	BFH
易解	66.037 7	83.018 9	2.222 3	1.851 8	0.438 4	0.013 2
难解	4.166 7	4.166 7	9.542 7	10.182 9	0.322 1	0.0079
总体	46.753 2	58.441 6	4.504 0	4.448 5	0.402 1	0.011 5

再次，从最优比例和平均偏差这两个指标上进行比较. 从表 6 可以看出，NEH 与 BFH 的求解比率分别是 46.753 2%和 58.441 6%，BFH 高出 NEH 的比例是 11.688 4%；NEH 与 BFH 的平均偏差分别是 4.504 0%和 4.448 5%，BFH 低于 NEH 的比例是 0.055 5%. 也就是说，无论从最优比率还是从平均偏差来看，BFH 都是优于 NEH 的.

最后，从耗时上进行比较. 从表 5 和表 6 可以看出，无论是从算例的机器布局，还是从算例的性质是易解或难解的角度来看，NEH 耗时均比 BFH 长. 从总体上来看，NEH 的平均耗时是 0.402 1 s，BFH 则是 0.011 5 s. 从表 7 可以看出，当各阶段机器布局相同且工件从 10 个增加到 15 个时，NEH 的平均耗时从 0.186 0 s 增加到 0.613 8 s，BFH 则从 0.006 9 s 增加到 0.010 2 s，BFH 平均耗时的增加幅度是小于 NEH 的.

究其原因，一方面，a 类和 b 类的算例都是易解的算例，c 类和 d 类的算例都是比较难解的算例. 另一方面，总计 47 个机器布局为 a 类和 b 类的算例中，NEH 和 BFH 分别有 35 个和 44 个算例求得了已知下界；总计 18 个机器布局为 c 类的算例中，NEH

和 BFH 都没有求得算例的已知下界；总计 12 个机器布局为 d 类的算例中，NEH 和 BFH 都没有求得算例的已知下界，唯一的一个例外是算例 j15c5d1，机器分布是 3-3-2-3-1，NEH 和 BFH 都求得了这个算例的已知下界. 这表明对于 NEH 和 BFH 而言，不同的机器布局对 HFS 调度问题的求解效果有非常重要的影响.

表 7　NEH 与 BFH 算法计算结果比较 (3)——平均耗时（单位：s）

Tab.7　Comparison between the results of NEH and BFH algorithms (3) – Average running time

算例	$n=10$		算例	$n=15$	
	NEH	BFH		NEH	BFH
j10c5a*	0.108 8	0.006 5	j15c5a*	0.382 3	0.009 4
j10c5b*	0.102 6	0.005 4	j15c5b*	0.352 5	0.007 5
j10c10b*	0.406 6	0.009 6	j15c10a*	1.357 9	0.015 3
j10c5c*	0.112 7	0.006 6	j15c5c*	0.418 1	0.009 5
j10c5d*	0.199 5	0.006 4	j15c5d*	0.558 1	0.009 3
平均	0.186 0	0.006 9	平均	0.613 83	0.010 2

5　结束语（Conclusion）

混合流水车间是现实生活中一种具有并行生产设施的流水线布局的高度抽象，具有重要的理论研究价值和广泛的工程应用背景. 本文研究了以最小化时间表长为目标的混合流水车间调度问题，其求解性质是 NP 难的. 本文提出了一种将机器布局和工件加工时间特征紧密结合的启发式算法. 首先，充分利用各阶段平均机器负荷一般不相等的特点构建初始工件排序. 其次，根据工件的加工阶段及其在瓶颈阶段前后的加工时间特点，计算工件加工的优先级后构建初始调度. 最后，采用工件交换和插入操作改进初始调度. 用 Carlier 和 Neron 提出的 Benchmark 算例仿真，结果表明最优求解比率比 NEH 高 11.688 4%，平均偏差比 NEH 低 0.055 5%，计算时间比 NEH 也有较大幅度的降低. 但对于难解算例，求解比率非常低，平均偏差也较大，尤其是算例的最大偏差来自本文提出的算法，表明这个算法具有一定的局限性. 进一步工作将加强对混合流水车间的性质及高效启发式算法的研究.

参考文献（References）

[1] 李铁克，苏志雄. 炼钢连铸生产调度问题的两阶段遗传算法 [J]. 中国管理科学，2009，17(5)：68-74.
Li T K, Su Z X. Two-stage genetic algorithm for SM-CC production scheduling[J]. Chinese Journal of Management Science, 2009, 17(5): 68-74.

[2] Linn R, Zhang W. Hybrid flow shop scheduling: A survey[J]. Computer & Industrial Engineering, 1999, 37(1/2): 57-61.

[3] Kis T, Pesch E. A review of exact solution methods for the non-preemptive multiprocessor flowshop problem[J]. European Journal of Operational Research, 2005, 164(3): 592-608.

[4] Quadt D, Kuhn H. A taxonomy of flexible flow line scheduling procedures[J]. European Journal of Operational Research, 2007, 178(3): 686-698.

[5] Ruiz R, Antoonio J, Rodriguez V. The hybrid flow shop scheduling problem[J]. European Journal of Operational Research, 2009, 5(1): 1-18.

[6] Ribas I, Listen R, Framinan J M. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective[J]. Computer & Operations Research, 2010, 37(8): 1439-1454.

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

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

[9] 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): B-630-B-637.

[10] Guinet A. Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time[J]. International Journal of Production Research, 1996, 34(6): 1643-1654.

[11] Kurz M E, Askin R G. Comparing scheduling rules for flexible flow lines[J]. International Journal of Production Economics, 2003, 85(3): 371-388.

[12] Thornton H W, Hunsucker J L. A new heuristic for minimal makespan in flow shops with multiple processors and no intermediate storage[J]. European Journal of Operational Research, 2004, 152(1): 96-114.

[13] 李黎，成晔，袁守华. 基于瓶颈分析的优先权调度算法研究 [J]. 计算机集成制造系统，2005，11(2)：247-250.
Li L, Cheng Y, Yuan S H. Research on priority scheduling algorithm based on bottleneck analysis[J]. Computer Integrated Manufacturing Systems, 2005, 11(2): 247-250.

[14] 翟颖妮，孙树栋，王军强，等. 基于正交试验的作业车间瓶颈识别方法 [J]. 计算机集成制造系统，2010，16(9)：1945-1952.
Zhai Y N, Sun S D, Wang J Q, et al. Bottleneck detection method based on orthogonal experiment for job shop[J]. Computer Integrated Manufacturing Systems, 2010, 16(9): 1945-1952.

[15] Rajendran C, Alicke K. Dispatching in flowshops with bottleneck machines[J]. Computer & Industrial Engineering, 2007, 52(1): 89-106.

[16] 左燕，谷寒雨，席裕庚. 大规模流水线调度的瓶颈分解算法研究 [J]. 控制与决策，2006，21(4)：425-429.
Zuo Y, Gu H Y, Xi Y G. Bottleneck-based decomposition algorithms for large-scale flow shop scheduling problems[J]. Control and Decision, 2006, 21(4): 425-429.

[17] Chen C L, Chen C L. A bottleneck-based heuristic for minimizing makespan in a flexible flow line with unrelated parallel machines[J]. Computer & Operations Research, 2009, 36(11): 3073-3081.

[18] Lee G C, Kim Y D, Choi S W. Bottleneck-focused scheduling for a hybrid flowshop[J]. International Journal of Production Research, 2004, 42(1): 165-181.

（下转第 528 页）

[2] Hespanha J P, Naghshtabrizi P, Xu Y G. A survey of recent results in networked control systems[J]. Proceedings of the IEEE, 2007, 95(1): 138-162.

[3] Lagkas T D, Papadimitriou G I, Nicopolitidis P, et al. Priority-oriented adaptive control with QoS guarantee for wireless LANs[J]. IEEE Transactions on Vehicular Technology, 2007, 56(4): 1761-1772.

[4] Panousopoulou A, Nikolakopoulos G, Tzes A, et al. Recent trends on QoS for wireless networked controlled systems[J]. Mediterranean Journal of Computers and Networks, 2006, 2(1): 31-40.

[5] 李祖欣，王万良，雷必成，等. 网络控制系统中基于模糊反馈的消息调度 [J]. 自动化学报，2007，33(11)：1229-1232.

Li Z X, Wang W L, Lei B C, et al. Message scheduling based on fuzzy feedback in networked control systems[J]. Acta Automatica Sinica, 2007, 33(11): 1229-1232.

[6] Colandairaj J, Irwin G W, Scanlon W G. Wireless networked control systems with QoS-based sampling[J]. IET Control Theory & Applications, 2007, 1(1): 430-438.

[7] Liu Q W, Wang X, Giannakis G B. A cross-layer scheduling algorithm with QoS support in wireless networks[J]. IEEE Transactions on Vehicular Technology, 2006, 55(3): 839-847.

[8] Tomovic R, Bekey G. Adaptive sampling based on amplitude sensitivity[J]. IEEE Transactions on Automatic Control, 1966, 11(2): 282-284.

[9] Otanez P G, Moyne J R, Tilbury D M. Using deadbands to reduce communication in networked control systems[C]// American Control Conference. Piscataway, NJ, USA: IEEE, 2002: 3015-3020.

[10] 汤贤铭，钱凯，俞金寿. 网络控制系统动态死区反馈调度 [J]. 华东理工大学学报：自然科学版，2007，33(5)：716-721.

Tang X M, Qian K, Yu J S. Dynamic deadband feedback scheduling in networked control systems[J]. Journal of East China University of Science and Technology: Natural Science Edition, 2007, 33(5): 716-721.

[11] McKernan A D, Irwin G W, Colandairaj J, et al. Event-based sampling for wireless network control systems with QoS[C]// American Control Conference. Piscataway, NJ, USA: IEEE, 2010: 1841-1846.

[12] 陈惠英，王万良，李祖欣. 基于误差阈值的网络控制系统通信调度 [J]. 信息与控制，2009，38(5)：580-584,590.

Chen H Y, Wang W L, Li Z X. Communication scheduling of networked control system based on error threshold[J]. Information and Control, 2009, 38(5): 580-584,590.

[13] Zhao Y B, Liu G P, Rees D. Packet-based deadband control for Internet-based networked control systems[J]. IEEE Transactions on Control Systems Technology, 2010, 18(5): 1057-1067.

[14] 俞立. 自动控制原理 [M]. 北京：清华大学出版社，2002：158-212.

Yu L. Principles of automatic control[M]. Beijing: Tsinghua University Press, 2002: 158-212.

[15] Yue D, Peng C, Tang G Y. Guaranteed cost control of linear systems over networks with state and input quantisations[J]. IEE Proceedings: Control Theory and Applications, 2006, 153(6): 658-664.

[16] El Ghaoui L, Oustry F, AitRami M. A cone complementarity linearization algorithm for static output-feedback and related problems[J]. IEEE Transactions on Automation Control, 1997, 42(8): 1171-1176.

[17] Gao Z N, Chen Q W, Hu W L. A new experimental platform for networked control systems based on CAN and switched-Ethernet[J]. Information Technology Journal, 2011, 10(1): 219-230.

作者简介：

高政南（1981–），男，博士生. 研究领域为网络控制系统.

谢蓉华（1975–），女，博士生，讲师. 研究领域为网络化控制，非线性控制，伺服控制.

陈庆伟（1963–），男，博士，教授. 研究领域为智能控制，非线性控制，网络化控制.

（上接第 521 页）

[19] Jin Z H, Yang Z, Takahiro I. Metaheuristic algorithms for the multistage hybrid flow shop scheduling problem[J]. International Journal of Production Economics, 2006, 100(2): 322-334.

[20] Carlier J, Neron E. An exact method for solving the multi-processor flow-shop[J]. Rario-Recherché Operationnelle, 2000, 34(1): 1-25.

[21] Graham R L, Lawler E L, Lenstra J K, et al. Optimization and approximation in deterministic sequencing and scheduling: A survey[J]. Annals of Discrete Mathematics, 1979, 5(2): 287-326.

[22] Brah S A, Loo L L. Heuristics for scheduling in a flow shop with multiple processor[J]. European Journal of Operational Research, 1999, 113(1): 113-122.

[23] 韦有双，杨湘龙，冯允成. 一种新的求解 flow shop 问题的启发式算法 [J]. 系统工程理论与实践，2000，20(9)：41-47.

Wei Y S, Yang X L, Feng Y C. Heuristic algorithms for flow shop scheduling problem[J]. Systems Engineering – Theory & Practice, 2000, 20(9): 41-47.

[24] Neron E, Baptiste P, Gupta J N D. Solving hybrid flow shop problem using energetic reasoning and global operations[J]. Omega, 2001, 29(6): 501-511.

[25] Engin O, Döyen A. A new approach to solve hybrid flow shop scheduling problems by artificial systems[J]. Future Generation Computer Systems, 2004, 20(6): 1083-1095.

作者简介：

屈国强（1970–），男，博士生. 研究领域为生产计划与调度，智能算法.