文章编号: 1001-0920(2006)04-0376-05

实时无等待 HFS 调度的一种拉格朗日松弛算法

轩 华[a], 唐立新[b]
(东北大学 a. 教育部暨辽宁省流程工业综合自动化重点实验室, b. 物流优化与控制研究所, 沈阳 110004)

摘 要: 研究了实时无等待 HFS 调度问题, 并建立一个整数规划模型, 提出运用拉格朗日松弛算法来求解. 在此算法中, 常采用次梯度方法更新拉格朗日乘子, 但它随着迭代数的增加收敛速度会减慢, 因此设计了一个改进的 bundle 方法, 将以前的次梯度累积到 bundle 中, 以获得一个更好的乘子更新方向. 仿真实验表明, 与次梯度方法相比, 所设计的 bundle 法不仅在较少的迭代数内得到了更快的收敛速度而且改进了优化性能, 对于大规模问题效果更为显著.

关键词: 实时混合流水车间调度; 无等待; 整数规划; 拉格朗日松弛; 改进的 bundle 法
中图分类号: TP278 **文献标识码**: A

Lagrangian Relaxation Algorithm for Real-time Hybrid Flow-shop Scheduling with No-wait in Process

XUAN Hua[a], *TANG Li-xin*[b]
(a. Key Laboratory of Process Industry Automation of Ministry of Education, b. The Logistics Institute, Northeastern University, Shenyang 110004, China. Correspondent: TANG Li-xin, E-mail: qhjytlx@mail.neu.edu.cn)

Abstract: The no-wait hybrid flow shop scheduling problem in a real-time environment is formulated as an integer programming model which has been proven NP-hard. A solution methodology based on Lagrangian relaxation is presented. In this method, the subgradient algorithm is commonly used to update Lagrange multipliers. However, the zigzagging behavior of subgradient optimization motivates the development of an improved bundle approach that accumulates the past subgradients in a bundle to achieve a better direction. Testing results show that the designed bundle approach provides a faster convergence and a better performance within less iteration, especially for large-scale problems, comparing with subgradient method.

Key words: Real-time hybrid flow shop scheduling; No-wait; Integer programming; Lagrangian relaxation; Improved bundle approach

1 引 言

混合流水车间(HFS)调度是流程工业中常出现的问题, 如钢铁业、化工业和制药业等. 在这些工业中, 生产的进行不仅要保证逻辑结果的有效性而且要使得到的结果满足时间要求. 生产调度必须满足时间要求, 这是因为只有在满足任务截止时间的前提下才能构造出实时系统的可行解. 目前关于实时 HFS 调度问题的研究还较少.

无等待要求出现在流程工业主要是由生产技术本身或存储能力引起的, 它要求物料在前一操作完成后直接传送到下一机器上, 这时, 为了衔接相邻两个操作以保证加工的连续性, 工件的操作可能会延迟. 就钢铁生产而言, 在连铸-热直轧过程, 经加热器加热板坯边缘后热板坯直接传送到热轧厂而不需经过加热炉加热[1], 这个过程不仅要考虑铁水流动的及时性也对实时操作有高度的要求. 炉次的等待时间会导致其温度降低, 从而需要再加热, 因此熔化的铁水必须在冷却之前连续地进行一系列操作, 以避

收稿日期: 2005-03-07; **修回日期**: 2005-05-23.
基金项目: 国家杰出青年科学基金项目(70425003); 国家自然科学基金项目(70171030, 60274049); 高等学校优秀青年教师教学科研奖励计划基金项目(教育司[2002]383).
作者简介: 轩华(1979—), 女, 河南睢县人, 博士生, 从事生产调度与计划等研究; 唐立新(1966—), 男, 黑龙江兰西人, 教授, 博士生导师, 从事生产调度、智能优化和物流管理等研究.

免降低钢的成分.

由于无等待HFS 是NP hard 的, 因此本文研究的更复杂问题也是NP hard 的. 一些文献对相关问题进行了阐述, 文献[2]综述了一些无等待和阻塞调度问题的计算复杂性, 并描述了一些调度方法的应用; 文献[3]探讨了制造系统中无等待流水车间的调度算法的性能, 对于每个中心有任意台机器的流水车间, 提供一个启发式算法, 并能保证其上界的性能; 文献[4～7]探索了求解HFS 调度问题的不同方法, 包括启发式、分支定界、禁忌搜索, 以最小化makespan、平均流水时间、总完成时间或平均完成时间; 文献[8]将炼钢过程归结为HFS 结构, 建立了整数规划模型, 并提出了拉格朗日松弛算法进行求解.

从上述文献成果可以看出, 关于实时无等待HFS 调度问题的研究很少, 并且目标是带权重的完成时间之和, 也很少出现在HFS 的文献中. 本文设计了适用于混合流水车间的改进的bundle 法.

2　问题描述

本文所研究的问题描述如下: N 个工件沿着相同的方向经过 J 个阶段进行加工, 每个阶段 j 有 M_j 台同构机, 一台机器一次至多加工一个工件, 而一个工件任何时候至多在一台机器上加工. 工件的加工是没有优先级的, 即一旦工件的一个工序开始加工, 则此工序不允许中断. 每个工件 i 都有一个权重 w_i 和截止时间 $\overline{d_i}$, 在阶段 j 的加工时间为 p_{ij}, 且各个工序要保证其连续性, 即无等待. 假设所考虑的时间水平 K 足够长以使所有的工件都能完成加工. 在工件加工顺序给定的情况下, 就可计算工件在各个阶段的完成时间 C_{ij}, 目标是最小化加权完成时间 $\sum w_i C_{iJ}$. 钢铁业中一个HFS 示意图如图 1 所示. 为简单起见, 假定所有工件都在时刻 0 可利用.

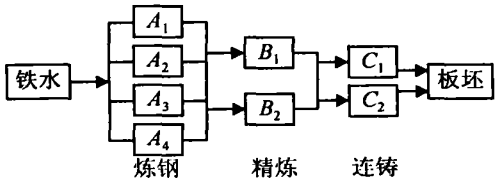


图 1　HFS 示意图

数学表达式中用到 3 个变量: 二进制变量 δ_{ijk}, 时间变量 B_{ij} 和 C_{ij}. 如果工件 i 在时刻 k 正在第 j 个阶段加工, 则二进制变量 δ_{ijk} 为 1, 否则为 0; 时间变量 B_{ij} 为工件 i 在第 j 阶段的开始时间, 表示一个时间点('$B_{ij} = k$' 表示操作在时刻 k 的起始点开始加工); C_{ij} 为工件 i 在第 j 阶段的完成时间, 表示一个时间点('$C_{ij} = k$' 表示操作在时刻 k 的末端完成加工).

利用静态和离散时间的整数规划模型可描述实时无等待HFS 调度问题如下:

$$P: \min_{\{B_{ij}\}} \sum_{i=1}^{N} w_i C_{iJ}. \tag{1}$$

其中

$$\sum_{i=1} \delta_{ijk} \leq M_j,$$
$$j = 1, \cdots, J, k = 1, \cdots, K. \tag{2}$$
$$C_{ij} - B_{ij} + 1 = p_{ij},$$
$$i = 1, \cdots, N, j = 1, \cdots, J. \tag{3}$$
$$B_{ij} = B_{i,j-1} + p_{i,j-1},$$
$$i = 1, \cdots, N; j = 2, \cdots, J. \tag{4}$$
$$C_{i,J} \leq \overline{d_i}, i = 1, \cdots, N. \tag{5}$$
$$\delta_{ijk} \in \{0, 1\}, i = 1, \cdots, N,$$
$$j = 1, \cdots, J, k = 1, \cdots, K. \tag{6}$$
$$C_{ij} \in \{1, 2, \cdots, \overline{d_i}\},$$
$$i = 1, \cdots, N, j = 1, \cdots, J. \tag{7}$$

式(1) 表示目标函数是总权重完成时间最小化; 约束(2) 表示在时刻 k 第 j 阶段正在加工的工件数必须不超过那个时刻可利用的机器数; 约束(3) 说明了操作一旦开始就不允许中断; 约束(4) 描述了操作间的无等待策略; 约束(5) 描述了完成时间要求, 即所有工件必须在它们各自的截止时间内完成; 约束(6) 和(7) 定义了决策变量的取值范围.

3　拉格朗日松弛算法

拉格朗日松弛算法是求解复杂问题的一个有效工具, 它的优点在于能够通过松弛某些耦合约束将原问题分解成较易求解的小问题. 松弛后的对偶函数的目标值 Z^{LB} 是原问题的下界, 由于松弛问题的解总是与不可行调度相关, 因此利用启发式把不可行调度转换成可行调度, 可行解的目标值 Z^{UB} 为原问题提供了一个上界, 解的质量通过相对对偶间隙 $(Z^{UB} - Z^{LB})/Z^{LB}$ 来衡量.

3.1　拉格朗日松弛

从第 2 节模型可以看到只有约束(2) 耦合不同的工件, 引入拉格朗日乘子向量 u, 把约束(2) 松弛到目标函数, 形成下面的松弛问题:

$$\min_{\{B_{ij}\}} J_{LR}, \text{ with } J_{LR} =$$
$$\left\{ \sum_{i=1}^{N} w_i C_{iJ} + \sum_{j=1}^{J} \sum_{k=1}^{K} u_{jk} \left(\sum_{i=1}^{N} \delta_{ijk} - M_j \right) \right\}. \tag{8}$$

并满足约束(3)～(7) 和

$$u_{jk} \geq 0, j = 1, 2, \cdots, J, k = 1, 2, \cdots, K. \tag{9}$$

则拉格朗日对偶问题是

$$\max_{\{u_{jk}\}} L(u), \text{ with } L(u) \equiv$$

$$\left\{ - \sum_{j=1}^{J}\sum_{k=1}^{K} u_{jk}M_j + \min_{\{B_{ij}\}} \sum_{i=1}^{N} \left(w_i C_{iJ} + \sum_{j=1}^{J}\sum_{k=1}^{K} u_{jk}\delta_{ijk} \right) \right\}. \tag{10}$$

并满足约束(3)～(7)和(9).

分解后的工件级子问题是

$$\min_{\{B_{ij}\}} L_i, \text{ with } L_i \equiv \left\{ w_i C_{iJ} + \sum_{j=1}^{J}\sum_{k=1}^{K} u_{jk}\delta_{ijk} \right\}, \tag{11}$$

并满足约束(3)～(7)和(9). 求解上述子问题时, 表达式中的乘子$\{u_{jk}\}$已知, 且工件i也是给定的.

利用拉格朗日松弛求解本文的调度问题包括求解工件级子问题、构造可行解和更新乘子. 这里主要研究乘子更新策略.

3.2 求解工件级子问题

对每个工件i, 采用枚举法来求解每个工件的开始时间. 注意到一旦得到B_{i1}的值, 就可根据式(4)推断出工件在后来阶段的开始时间. 因此枚举B_{i1}的每个可能值, 并计算L_i直到得到最优值(L_i的最小值). 在求解子问题时, 工件i和乘子$\{u_{jk}\}$是已知的.

3.3 构造可行解

因为对偶问题得到的解可能违反约束(2), 因此它的解对于原问题一般是不可行的. 这时, 需要利用当前信息构造可行解, 以获得原问题的上界. 根据文献[9]提出的列表调度概念, 结合插入和邻域交换的思想提出了一个两阶段启发式. 首先, 按照对偶解中工件各自开始时间的增序排列, 构造出一个列表, 当机器在时间B_{i1}和$(B_{i1} + \sum_j p_{ij} - 1)$之间可以利用时, 把工件分配到相应的机器上. 否则, 就要延迟工件直到机器可利用.

大多数情况下, 根据此启发式都能构造可行解, 否则引入插入的概念形成可行解. 令i表示顺序中工件的位置标号, 假设列表中前$i-1$个工件已经分配并满足所有的约束, 当第$i(i = 2, \cdots, N)$个工件在满足约束(2)～(4), (6)和(7)的情况下安排时, 违反了截止日期, 此时剩余的工件就不需分配了, 因为此顺序并不能构造出可行解, 这时将工件i插入到工件$i-k(k = 1, 2, \cdots, i-1)$的紧前面, 保持其他工件不变, 如果变化后的部分调度能够满足所有约束, 则继续按照原列表分配剩余工件, 否则继续此插入过程, 直到形成可行解.

为了进一步降低原问题的上界, 当迭代数达到一个给定值时, 程序进入第二阶段: 从上述步骤得到的可行解(作为初始解)开始, 进行邻域交换.

4 基于 bundle 法的乘子更新策略

求解拉格朗日对偶问题常用的方法是次梯度优化, 其优点是在开始迭代数内很容易执行且收敛速度快, 不过随着迭代数的增加, 收敛速度会减慢. 次梯度方法的大多数改进形式仅利用当前次梯度或当前次梯度与先前方向的凸组合计算新的方向. 虽然改进的次梯度方法能改进性能, 但其质量涉及到一些关键参数的选择, 且不同的问题下参数的设置也需要调整, 并不能得到适合所有问题的一组合理的设置, 因此这类方法对于"灵活"的参数选择更有效. 但 bundle 方法避免了这种情况, 它以分离的方式累积以前获得的次梯度, 与次梯度方法相比, 其优势在于收敛性更快、对参数的依赖性更少, 然而执行这种算法是一个复杂的过程. 在此过程中求解二次规划和线性搜索是耗时的, 因此它的主要瓶颈在于计算下一搜索方向.

本文基于次梯度法和 bundle 法的优点设计了改进的 bundle 方法, 此思想首先在文献[10]提到并应用于车辆路径问题, 由 2 个阶段组成.

第 1 阶段中, 根据次梯度方法在开始迭代时收敛性快的特点在一定的迭代数内执行次梯度优化, 乘子根据下式进行更新:

$$u^{n+1} = u^n + s^n g^n. \tag{12}$$

其中: n是迭代标号; g^n是第n次迭代中L的次梯度, $g^n = \sum_{i=1} \delta_{ijk} - M_j$; s^n是第n次迭代的步长, 给定

$$s^n = \lambda \frac{L^* - L^n}{\| g^n \|^2}, \quad 0 < \lambda < 2, \tag{13}$$

L^*是最优值, 通过迄今为止得到的最好的可行解来估计; L^n是第n次迭代中L的值.

第 2 阶段中, 程序从当前得到的最好解开始执行 bundle 算法. bundle 算法采用了ϵ次微分的概念

$$\partial_\epsilon L(u) \equiv \{ g \in R^m \mid L(u') \leq L(u) + g^{\mathrm{T}}(u' - u) + \epsilon, \forall u' \in R^m \}. \tag{14}$$

一般地, $\partial_\epsilon L(u)$由 bundle 元素的凸组合来近似[15], 即

$$P = \left\{ g \mid g = \sum_i \alpha_i g_i, \sum_i \alpha_i = 1, \sum_i \alpha_i e_i \leq \epsilon, \alpha_i \geq 0 \right\}. \tag{15}$$

其中e_i是线性误差, $e_i = L(u_i) + g_i^{\mathrm{T}}(u - u_i) - L(u)$. P中具有最小范数的一个方向认为是下一搜索方向, 即求解下面问题以得到新的方向:

$$\min \| \sum_i \alpha_i g_i \|, \quad \text{s.t.} \sum_i \alpha_i = 1, \sum_i \alpha_i e_i \leq \epsilon, \alpha_i \geq 0. \tag{16}$$

根据上述二次规划问题计算出方向后, 尝试沿着这个方向进行线性搜索, 如果函数值增加得足够

大, 则更新乘子到新的一点, 这就产生了一个严重步, 否则产生了空步, 两种情况下, 新得到的次梯度都要添加到 bundle 中, 以计算新的方向

在算法执行过程中, 采用 IBM OSL 库的二次规划求解器快速求解式(16). 为了简化求解线性搜索的复杂性, 应用了另外一种方法[11], 它试着逐步减少当前迭代和最优解之间的距离, 而不是对每一个严重步目标函数至少增加 ϵ 如果根据式(12) 更新乘子, 且

$$s^n = \lambda(L^* - L^n)/\| g^n \|^2,$$
$$g^n \in P, 0 < \lambda < 1,$$

则乘子逐步接近最优的 u^*.

每次迭代中添加一个次梯度到 bundle 中, 且设置最大 bundle 规模为 50(最小为 2), 这是为了避免存储空间超过限制 当 bundle 规模达到最大值时, 具有最大线性化误差 e 的次梯度从 bundle 中删去 当 $\| \sum_i \alpha_i g_i \|$ 小于一个很小的数或者达到最大迭代数时程序停止

5　实验结果

为说明设计的 bundle 方法的性能, 进行实验测试, 并显示它与次梯度方法相比的计算结果 程序由 C 语言编写, 在 P4 2.4 G, 512M 内存的机器上测试

所有的测试数据都是随机产生的, 工件数在 {20, 40, 60, 100} 中选择, 阶段数和每阶段的机器数都设为 2, 3 和 4 为简单起见, 假定每个阶段的机器数相同, 然而, 提出的方法也能解决不同阶段具有不同机器数的问题 上述参数的组合产生了 36 个不同的问题, 对每种问题, 随机产生 10 个不同的事例, 因此, 共产生 360 个问题用于实验 加工时间和权值在[1, 10] 满足均匀分布, 且是正整数 工件 i 的截止时间根据其总加工时间和所有工件的平均加工时间随机产生 基本时间单位是 min, 计划水平设为 $\sum_{i=1}^{N}\sum_{j=1}^{J} p_{ij}$ min.

为公平比较这两种拉格朗日松弛算法, 采用相同的最大迭代数 1 000 作为停止准则 定义 LR_{SG} 为采用次梯度方法更新乘子的拉格朗日松弛, LR_B 为结合改进的 bundle 法的拉格朗日松弛 表 1 比较了这两种拉格朗日松弛算法, 其性能通过对偶间隙和迭代数来估计, 图 2 显示了不同规模下对偶间隙随迭代数的变化(阶段数和每阶段的机器数都为 3).

从表 1 和图 2 可以看出, 与 LR_{SG} 相比, LR_B 不仅性能好而且收敛速度快, 在较少迭代数内, 平均对偶间隙降低了 0.44%. 随着问题规模的增大, 改进的

表 1　两种拉格朗日松弛算法的比较结果

问题	问题结构 工件数×阶段数×机器数	对偶间隙/%		迭代数	
		LR_{SG}	LR_B	LR_{SG}	LR_B
1	20×2×2	3.03	2.74	1 000	1 000
2	20×2×3	1.08	1.10	853	797
3	20×2×4	0.61	0.58	449	390
4	20×3×2	5.11	4.72	1 000	1 000
5	20×3×3	2.35	2.07	1 000	1 000
6	20×3×4	0.83	0.87	698	698
7	20×4×2	6.56	6.36	1 000	1 000
8	20×4×3	2.97	2.81	1 000	1 000
9	20×4×4	0.81	0.73	563	560
10	40×2×2	4.42	3.80	1 000	1 000
11	40×2×3	1.93	1.94	1 000	1 000
12	40×2×4	1.14	1.13	924	924
13	40×3×2	8.91	8.01	1 000	1 000
14	40×3×3	4.23	4.19	1 000	1 000
15	40×3×4	1.98	1.82	1 000	1 000
16	40×4×2	13.35	12.21	1 000	1 000
17	40×4×3	6.01	5.69	1 000	1 000
18	40×4×4	2.89	2.54	1 000	1 000
19	60×2×2	5.10	4.69	1 000	1 000
20	60×2×3	3.26	3.08	1 000	1 000
21	60×2×4	1.78	1.72	1 000	1 000
22	60×3×2	11.19	10.41	1 000	1 000
23	60×3×3	7.34	6.34	1 000	1 000
24	60×3×4	3.92	3.52	1 000	1 000
25	60×4×2	16.81	15.32	1 000	1 000
26	60×4×3	9.40	8.59	1 000	1 000
27	60×4×4	5.42	5.29	1 000	1 000
28	100×2×2	6.99	6.00	1 000	1 000
29	100×2×3	4.34	3.99	1 000	1 000
30	100×2×4	3.06	2.85	1 000	1 000
31	100×3×2	13.37	13.07	1 000	1 000
32	100×3×3	9.53	8.38	1 000	1 000
33	100×3×4	6.23	5.91	1 000	1 000
34	100×4×2	20.23	18.97	1 000	1 000
35	100×4×3	13.29	12.36	1 000	1 000
36	100×4×4	8.48	8.30	1 000	1 000
平均		6.05	5.61	958	955

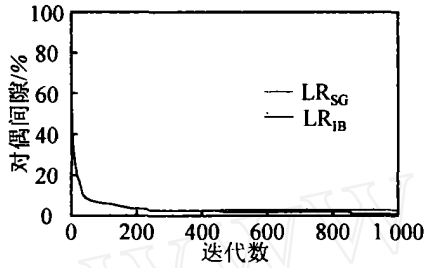
bundle 算法性能更好, 如果工件比较少, LR_B 在较少的迭代数内得到的结果与 LR_{SG} 类似, 因此改进的 bundle 方法对于求解大规模的调度问题更有效 同时也测试了两种方法的运行时间, LR_{SG} 所使用的 CPU 时间(平均为 7.23 s) 要比 LR_B (89.58 s) 少得多, 这是由改进的 bundle 算法本身的耗时性决定的, 它不能与次梯度算法相比拟, 仅从这点考虑可能会得到 LR_{SG} 比 LR_B 性能好的结论, 但这些结果是在相同迭代数 1 000 的停止准则下得到的, 当运行 LR_{SG} 使其计算时间与 LR_B 相同时, LR_B 得到的对偶间隙要优于 LR_{SG}, 从解的质量和收敛速度方面来讲, LR_B 还是很有竞争力的, 同时得到以下结论:

1) 当工件数增加时, 对偶间隙和迭代数都增

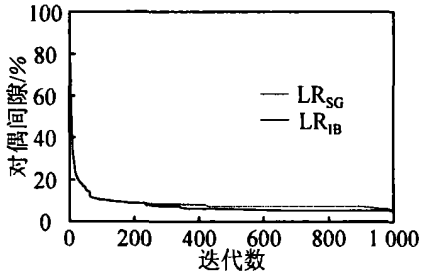
加, 这是因为工件数的增加使得所调度的任务增多, 从而使问题变得更复杂

2) 当机器数增多时, 解的质量变得更好, 迭代数也降低了, 这是因为随着机器数的增加, 工件之间对机器的竞争相对减少, 从而使问题变得简单一些

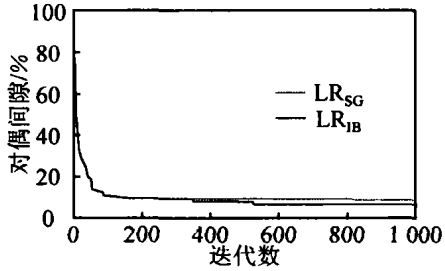
3) 当阶段上并行机数减少时, 即出现瓶颈现象的可能性越大时, 截止时间越紧则产生不可行解的可能性越大



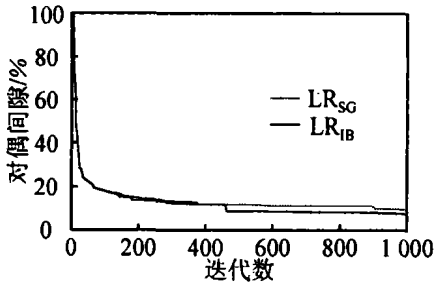
(a) 20 个工件



(b) 40 个工件



(c) 60 个工件



(d) 100 个工件

图 2 不同工件数下的收敛速度

6 结 论

本文针对实时无等待 HFS 调度建立了整数规划模型, 目标是在满足一些实际特征下(截至时间要求和优先级约束)最小化总加权完成时间. 对此问题, 采用了基于改进的 bundle 法的拉格朗日松弛来求解, 实验结果说明了此方法的有效性, 尤其在求解大规模问题方面. 与次梯度方法相比, 收敛速度更快, 性能更好. 此算法还可以修改以求解其他具有类似特征的 HFS, 且可进一步探讨启发式的改进, 以确保在每个问题中可行解的构造

参考文献(References)

[1] Tang L X, Liu J Y, Rong A Y, et al. A Review of Planning and Scheduling Systems and Methods for Integrated Steel Production[J]. *European J of Operational Research*, 2001, 133(1): 1-20.

[2] Hall N G, Sriskandarajah C. A Survey of Machine Scheduling Problems with Blocking and No-wait in Process[J]. *Operations Research*, 1996, 44(3): 510-525.

[3] Sriskandarajah C. The Performance of Scheduling Algorithms for No-wait Flowshops with Parallel Machines [J]. *European J of Operational Research*, 1993, 70(3): 365-378.

[4] Brah S A, Loo L L. Heuristics for Scheduling in a Flowshop with Multiple Processors[J]. *European J of Operational Research*, 1999, 113(1): 113-122.

[5] Moursli O, Pochet Y. A Branch-and-bound Algorithm for the Hybrid Flowshop [J]. *Int J of Production Economics*, 2000, 64(1- 3): 113-125.

[6] Wardono B, Fathi Y. A Tabu Search Algorithm for Multi-stage Parallel Machine Problem with Limited Buffer Capacities [J]. *European J of Operational Research*, 2004, 155(2): 380-401.

[7] Al-Anzi F, Allahverdi A. Flexible Multistage Parallel-server Scheduling for Internet Service Architecture[J]. *Int J of Parallel and Distributed Systems and Networks*, 2002, 5(3): 134-142.

[8] Tang L X, Luh P B, Liu J Y, et al. Steel-making Process Scheduling Using Lagrangian Relaxation[J]. *Int J of Production Research*, 2002, 40 (1): 55-70.

[9] Luh P B, Hoitomt D J, Max E, et al. Schedule Generation and Reconfiguration for Parallel Machines[J]. *IEEE Trans on Robotics and Automation*, 1990, 6(6): 687-696.

[10] Kohl N, Madsen O B G. An Optimization Algorithm for the Vehicle Routing Problem with Time Windows Based on Lagrangian Relaxation [J]. *Operations Research*, 1997, 45(3): 395-406.

[11] Zhao X, Luh P B. New Bundle Methods for Solving Lagrangian Relaxation Dual Problems [J]. *J of Optimization Theory and Application*, 2002, 113(2): 373-397.