doi: 10.12068/j.issn.1005-3026.2018.12.002

缓冲区间有限条件下的作业车间调度方法

曾程宽, 刘士新
(东北大学 信息科学与工程学院, 辽宁 沈阳 110819)

摘 要: 针对缓冲区间有限条件下的作业车间调度问题,以最小化 make-span 为目标建立了非线性混合整数规划模型,提出了基于邻域搜索的两阶段算法对问题进行求解. 算法的第一阶段为迅速找到可行解,第二阶段为基于非连通图,通过邻域搜索对得到的可行解进行优化. 针对 benchmark 算例进行测试并与已有的算法进行对比,验证了算法的有效性. 对比分析发现,如果工件的加工时间符合均匀分布,当缓冲区间容量与工件数量的比例达到 20% ,缓冲区间大小对调度结果的影响将会迅速变小.

关 键 词: 作业车间调度;缓冲区间有限;非连通图;均匀分布
中图分类号: TP 301.6 **文献标志码**: A **文章编号**: 1005-3026(2018)12-1679-06

Job Shop Scheduling Problem with Limited Output Buffer

ZENG Cheng-kuan[1], *LIU Shi-xin*[1]
(School of Information Science & Engineering, Northeastern University, Shenyang 110819, China. Corresponding author: ZENG Cheng-kuan, E-mail: 956721427@qq.com)

Abstract: The job shop problem with limited output buffers (JS-LOB) was addressed with the objective of minimizing the process make-span. An integer nonlinear mathematical programming (INLP) model was proposed to describe this problem. Based on the model, a two-stage algorithm consisting of obtaining feasible solutions and a local search was proposed to solve the JS-LOB problem. The operator in local search was a neighborhood structure based on a disjunctive graph model. Computational results were presented for a set of benchmark tests, some of which were enlarged by different proportions between the capacity of the buffer and the number of jobs. The results show the effectiveness of the proposed algorithm through comparing with other exist algorithms and indicate when the processing time of the job conforms to a uniform distribution, and when the proportion between the capacity of the buffer and the number of jobs is larger than 20% , the influence of the buffer will become very small.

Key words: job shop scheduling; limited output buffer; disjunctive graph; uniform distribution

作业车间(job shop)调度是最经典的调度问题之一,随着生产系统的发展,对于作业车间调度问题考虑的因素也变得越来越多. 本文研究设备缓冲区间有限下的作业车间调度问题.

所谓缓冲区间,是指工件在当前设备完成加工后,停留在当前设备等待下一道工序生产期间所占用的等待空间[1]. 对于传统作业车间调度问题,每台设备的缓冲区间被视为容量无限[2]. 对于本文考虑设备缓冲区间有限的情境,若工件在当前设备加工时,此设备的缓冲区间已被占满,且当前工序完成后不能马上进入下一工序,需要在当前设备上继续等待,届时工件只能停留在当前设备的加工区间上,使得后续工件无法在此设备上进行加工. 在考虑设备缓冲区间有限的情境下,调度问题的复杂性将大大加强,绝大多数相关研究均针对流水车间调度问题[3-8],并提出一系列算法进行求解,包括:免疫算法[3]、混沌和声搜索算法[4]、离散差分进化算法[5]、混合遗传算法[6]、

收稿日期: 2017-09-20
基金项目: 国家自然科学基金资助项目(71702028,61573089); 中国博士后科学基金资助项目(2017M621154); 国家重点研发计划项目(2017YFB0306401); 国家自然科学基金创新群体项目(71621061).
作者简介: 曾程宽(1988-), 男, 辽宁沈阳人, 东北大学讲师, 博士; 刘士新(1968-), 男, 辽宁调兵山人,东北大学教授,博士生导师.

混合进化算法[7]、离散人工蜂群算法[8]等,文献[9]更将设备缓冲区间有限下的流水车间调度问题与批处理问题相结合. 而在作业车间调度方面,已有研究仅针对问题的特征进行了分析,缺少完整的求解方法. 本文针对此问题,建立数学模型进行描述,提出基于邻域搜索的两阶段算法进行求解,并通过对比试验证明算法的有效性.

1 数学模型

在作业车间中,有 n 个工件 $J=\{1,2,\cdots,n\}$ 和 m 台设备 $M=\{1,2,\cdots,m\}$. 每个工件的加工路线固定,包括多道工序. 在任何时刻,每台设备只能同时加工一个工件,设备 k 的缓冲区间最多能同时容纳 B_k 个工件. 假设所有的设备具有相同的缓冲区间. 每个工件的最后一道工序完成加工后,被视为马上离开生产系统,不再需要占用缓冲区间. P_{ij} 为第 i 个工件第 j 道工序的加工时间,$S_{ij}(C_{ij})$ 和 IB_{ij} 为需要决策的变量,$S_{ij}(C_{ij})$ 为第 i 个工件第 j 道工序的开始(结束)加工的时间,IB_{ij} 为第 i 个工件第 j 道工序进入设备缓冲区间的时间. 问题的目标是寻找一个合理的调度方案,使得全体工件的完工时间最小.

$$\mathrm{Min}\ \max\{C_{ij}\}\ . \tag{1}$$

$$S_{ij}+P_{ij}=C_{ij},\ \forall i,j\ . \tag{2}$$

$$\mathrm{IB}_{ij}\geqslant C_{ij},\ \forall i,j\ . \tag{3}$$

$$S_{ij}\geqslant \mathrm{IB}_{i(j-1)},\ \forall i,j=2,\cdots,n\ . \tag{4}$$

$$\left.\begin{aligned}
&\alpha_{ijk}\alpha_{i'j'k}(S_{ij}-\mathrm{IB}_{i'j'})(S_{i'j'}-\mathrm{IB}_{ij})\leqslant 0,\ i\neq i',j,j'=1,2,\cdots,n-1,\ \forall k,\\
&\alpha_{ijk}\alpha_{i'j'k}(S_{ij}-\mathrm{IB}_{i'j'})(S_{i'j'}-C_{ij})\leqslant 0,\ i\neq i',j=n,j'=1,2,\cdots,n-1,\ \forall k,\\
&\alpha_{ijk}\alpha_{i'j'k}(S_{ij}-C_{i'j'})(S_{i'j'}-C_{ij})\leqslant 0,\ i\neq i',j=n,j'=n,\ \forall k.
\end{aligned}\right\} \tag{5}$$

$$\begin{aligned}
&\alpha_{i_1j_1k}\alpha_{i_2j_2k}\cdots\alpha_{i_{N+1}j_{N+1}k}(\max\{\mathrm{IB}_{i_1j_1},\mathrm{IB}_{i_2j_2},\cdots,\mathrm{IB}_{i_{N+1}j_{N+1}}\}-\\
&\quad\min\{S_{i_1(j_1+1)},S_{i_2(j_2+1)},\cdots,S_{i_{N+1}(j_{N+1}+1)}\})\geqslant 0,\\
&i_1\neq i_2\neq\cdots\neq i_{N+1},j_1,j_2,\cdots,j_{N+1}=1,2,\cdots,n-1,\ \forall k.
\end{aligned} \tag{6}$$

$$\alpha_{ijk}=\begin{cases}1, & \text{如果工序 } O_{ij} \text{需要设备 } k \text{ 加工;}\\ 0, & \text{否则.}\end{cases} \tag{7}$$

式(1)为模型的目标函数:最小化全体工件的完工时间;式(2)表明自某道工序开始加工起,直至此工序完成加工前不能被中断;式(3)表明工件在当前加工设备完成后,可以进入设备的缓冲区间等待下一道工序的加工;式(4)表明当下一道工序设备可用时,工件可以立刻从当前设备缓冲区间离开;式(5)表明任何设备在任何时刻都不能同时加工多个工件;式(6)给出了工件进入设备缓冲区间在时间上的先后关系.

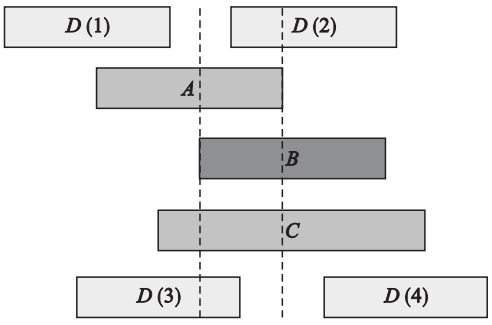

图 1 缓冲区间内工件在时间上的排列关系
Fig. 1 Relations between operations in the buffer

针对考虑设备缓冲区间的作业车间调度问题,本文求解的思路是首先快速寻找到一个可行解,然后针对得到的可行解进行优化. 首先通过求解一个小型算例分析解的特征. 算例的具体信息如表 1 所示. 基于上文提出的模型,使用 Lingo 求解此算例,假设设备缓冲区间的容量为 1,得到最终调度方案甘特图如图 2 所示.

表 1 工件负载矩阵
Table 1 Job machine load matrix

工件	工序加工时间		
工件 1	*M*3:1	*M*1:3	*M*2:6
工件 2	*M*2:8	*M*3:4	*M*1:10
工件 3	*M*3:5	*M*1:4	*M*2:8

由图 2 可知,如果将缓冲区间看作一种特殊的设备,对于每个工件来说,相当于处在无等待的约束下得到的调度方案. 无等待约束,即任何工件前一道工序完成后,后一道工序必须马上开始. 对于存在无等待约束条件下的作业车间调度问题,求解的关键是给出一个合理的加工顺序[10]. 本文使用 NEH 算法[10],主要步骤如下.

步骤 1 计算每个工件的加工时间总和,依照降序排列,得到排序 $\pi=\{\pi(1),\pi(2),\cdots,\pi(n)\}$;

步骤 2 选择排序中最前面的两个工件进行单独排序调度,选择一个结果最好的顺序,作为当前的暂定顺序;

步骤 3 对于剩余的工件 $\pi(j)$,$j=3,4,\cdots,n$,每次将一个新工件插入现有的排序,保持现有的排序不动,新工件找到一个合适的位置插入,使

得插入后形成新排序的调度结果为最佳,得到的排序为新的暂定排序. 以此类推,得到 n 个工件最终的排序.

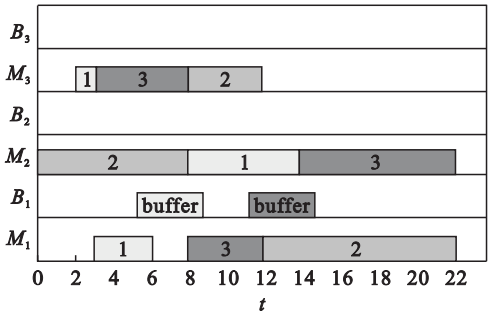

图 2　使用 Lingo 得到的调度方案甘特图

Fig. 2　Gantt chart of scheduling result obtained by Lingo

根据以上给出的顺序,提出以下缓冲区间调度机制:当某道工序完成当前的加工,如果它下一道工序的设备是可用的,将其直接转向下一台设备;如果下一台设备处于繁忙状态且当前设备的缓冲区间尚未被占满,工件则转入当前设备缓冲区间等待;否则,工件只能继续停留在当前设备的加工区间进行等待.

综上,快速得到可行解的步骤如下.

步骤 1　根据 NEH 方法得到工件的顺序,按照顺序依次进行调度;

步骤 2　从零时刻检测个各个工序的状态,如果有工序完成当前加工,记录当前时间点,转入步骤 3,否则检测下一时间点直至有工序完成当前加工,转入步骤 3;

步骤 3　如果当前完成工序为某个工件的最后一道工序,转入步骤 4,否则转入步骤 5;

步骤 4　如果所有工件均完成加工,算法结束,否则返回步骤 2;

步骤 5　判断工件当前停留在设备的加工区间或是缓冲区间,转入步骤 6;

步骤 6　如果下一台设备处于繁忙状态并且工件处于设备的加工区间,转入步骤 7,如果工件处于设备的缓冲区间,转入步骤 8,否则转入步骤 9;

步骤 7　如果当前设备的缓冲区间没有被占满,将当前工序移至缓冲区间等待下一台设备,更新设备状态,返回步骤 2,否则只能继续停留在当前设备的加工区间继续等待,返回步骤 2;

步骤 8　如果当前工件需要继续在缓冲区间内等待下一设备,返回步骤 2,否则转入步骤 9;

步骤 9　将工序移向下一台设备进行加工,更新设备状态和工序状态,返回步骤 2.

2　基于非连通图的邻域搜索

在得到可行解后,本文提出邻域搜索机制对其进一步深入优化,防止其陷入局部最优.

对于不同的调度方案,各道工序在设备的缓冲区间的等待时间不同. 因此,要通过优化尽量减少或消除工序在缓冲区间的等待,减少整体工件最终的完工时间. 本文借鉴非连通图(disjunctive graph)在作业车间调度方面已有的研究,通过得到可行解的关键路径,改变工序之间的位置得到新的邻域,来减少工序在缓冲区间内的等待时间. 下面举例说明如何基于关键路径改变工序顺序得到新的邻域,更多关于非连通图的信息详见文献[11]. 对于表 1 中的算例,如果按照 1 -2 -3 的工件顺序进行加工,得到的调度方案如图 3 所示.

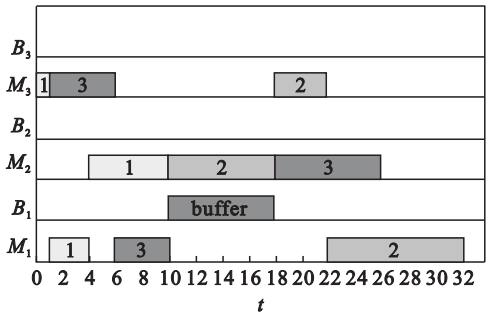

图 3　基于 1 -2 -3 顺序得到的调度方案甘特图

Fig. 3　Gantt chart of scheduling result based on sequence 1 -2 -3

图 3 中,虚线的部分将构成关键路径,在关键路径上,O_{32}在设备 1 的缓冲区间内占用大量的等待时间,通过改变其后续工序所在设备上的加工顺序来减少等待时间. 本文改变 O_{33}所在设备 2 上工序的加工顺序,交换 O_{13}和 O_{21}的顺序,得到新的调度方案如图 4 所示.

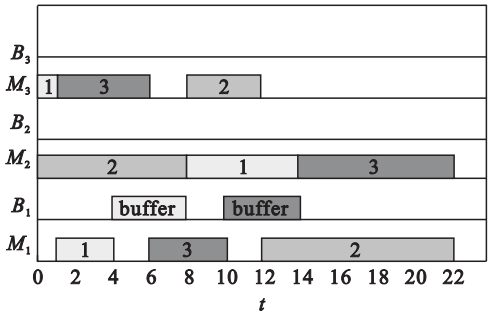

图 4　改进后新调度方案的甘特图

Fig. 4　Gantt chart of scheduling result after improving

上面的例子说明,通过寻找关键路径,改变工序在设备上的加工顺序,能够减少在设备缓冲区

间的等待时间,进而减少最终的完工时间 make-span. 据此,本文提出基于邻域搜索的两阶段算法,步骤如下.

步骤 1 参数设置:初始可行解的数量 K,最大迭代次数 GENNO,建立集合 s 和 $s1$,Iter = 1;

步骤 2 得到 K 个初始解,转入步骤 3;

步骤 3 针对每个可行解,计算其关键路径,清空集合 s,选出关键路径中需要在设备缓冲区间进行等待的工序,将它们的等待时间按降序排列,依次加入集合 s,转入步骤 4;

步骤 4 选定集合 s 中第一个元素对应关键工序路径中的工序 O_{ij},找到工序 $O_{i(j+1)}$ 对应的设备,确定当前调度方案中,在此设备上加工排序在 $O_{i(j+1)}$ 前的工序,尝试改变它们现有的顺序,得到新的邻域,并删除集合 s 中的第 1 个元素. 如果得到邻域的完工时间 make-span 比现有的方案少,用得到的邻域更新现有方案,返回步骤 3,如果此时集合 s 是空集,转至步骤 5,否则重复步骤 4;

步骤 5 如果当前所有的初始解均完成了邻域搜索优化过程,将当前所有解中最优的一个加入集合 $s1$,Iter = Iter + 1,如果 Iter > GENNO,转至步骤 6,否则针对现有可行解中某一设备上的加工顺序,进行两两交换,形成新的可行解,返回步骤 3;

步骤 6 选择集合 $s1$ 中的最优解作为最终的调度方案,算法结束.

讨论算法的计算复杂度. 在寻找初始可行解阶段,假设初始解的个数为 A,每个初始解中包含的工件数量为 B,每个工件需要经过 C 个道次的设备进行各个工序. 在寻找每个可行解的过程中,需要针对每个工件每个道次的工序进行依次调度. 由于设备的缓冲区间大小有限,对于每个工件的每道工序,需要从 0 时刻起,判断每一个时间点上设备的状态,能否进行其对应工序. 因此,如果第 i 个工件的第 j 道工序的完工时间为 C_{ij},那么得到第 i 个工件的第 j 道工序的调度方案需要循环计算的次数为 C_{ij},因此,得到第 i 个工件调度方案需要的循环计算次数为 $C_{i1} + C_{i2} + \cdots + C_{ic}$,由此可以得到每个工件调度方案的计算复杂度为 $O(C)$,若算例包含 B 个工件,则得到算例的调度方案(一个可行解)的计算复杂度为 $O(BC)$,以此类推,得到 A 个初始可行解的计算复杂度为 $O(ABC)$. 在基于非连通图的邻域搜索阶段,在每次搜索中,通过改变关键路径中关键块内关键工序的顺序得到新的邻域(可行解),并根据新的排序得到新的调度方案,其计算方式及过程与初始解相同,假设每个可行解进行邻域搜索的次数为 Q,则邻域搜索阶段的计算复杂度为 $O(QABC)$,综上所述,本文提出的基于邻域搜索的两阶段算法的计算复杂度为 $O(QABC)$.

3 实验结果与比较

为测试算法的性能,本文选用标准的作业车间 benchmark 算例进行测试. 算例 La01 ~ La20 来自文献[12].

实验参数设置:对于每组算例,生成初始解个数为 200,邻域搜索最大迭代次数 GENNO 为 100.

首先尝试使用 Lingo 对问题进行求解. 即使对于小规模算例,在计算 4 h 后尚不能得到可行解,说明 Lingo 不适合求解此类问题,也证明了此问题的复杂性.

使用本文提出的基于邻域搜索的两阶段算法求解算例,结果如表 2 所示. 10 × 5 表示算例中包含 10 个工件,每个工件均含有 5 道工序. 表 2 中信息为不同规模算例在设备缓冲区间容量不同情境下的工件完工时间 make-span. 若工件数量为 20,每台设备缓冲区间最多可同时容纳 4 个工件,此时的比例为 20% . 0% 为无缓冲区间的状态,其结果来自文献[13]. 当比例为 100% 时,等同于经典的作业车间(job shop,简称 JS)调度问题. BKS 为现有在 JS 研究中得到的最优解,Gap 为本文两阶段算法在 100% 比例下得到结果与 BKS 的偏差.

从表 2 可以看出,对于经典的 JS 问题,本文提出的算法也能够得到理想的结果. 对于小规模算例,即使在比例仅为 20% 时,本算法依然能够得到全局最优解,例如算例 La01,La05,充分证明了本文提出两阶段算法对于此问题求解的有效性.

为了进一步分析本文提出的两阶段算法,本文尝试与其他算法进行比较. 由于现有研究缺少针对本文提出问题的求解算法,只能与求解缓冲区间有限下流水车间(flow shop)调度问题的方法相比较. 分别选取来自文献[3 – 6]中的算法,计算在不同缓冲区间容量下的完工时间. 选取 4 种算法中最好的结果,如表 3 所示.

表 2　设备缓冲区间容积与工件数量不同比例下的完工时间

Table 2　Result for instances under different proportions between the buffer and jobs

算例	规模	设备缓冲区间容积与工件数量的比例/%							BKS	偏差
		0	10	20	30	50	80	100		
La01	10×5	832	780	666	666	666	666	666	666	0
La02	10×5	793	776	683	677	665	661	656	655	0.15
La03	10×5	747	697	617	617	613	604	597	597	0
La04	10×5	769	723	606	602	595	590	590	590	0
La05	10×5	698	658	593	593	593	593	593	593	0
La06	15×5	1 180	1 026	926	926	926	926	926	926	0
La07	15×5	1 091	970	893	890	890	890	890	890	0
La08	15×5	1 125	976	880	874	874	868	864	863	0.11
La09	15×5	1 223	1 060	951	951	951	951	951	951	0
La10	15×5	1 203	1 033	958	958	958	958	958	958	0
La11	20×5	1 584	1 318	1 222	1 222	1 222	1 222	1 222	1 222	0
La12	20×5	1 391	1 205	1 040	1 039	1 039	1 039	1 039	1 039	0
La13	20×5	1 548	1 327	1 150	1 150	1 150	1 150	1 150	1 150	0
La14	20×5	1 620	1 391	1 292	1 292	1 292	1 292	1 292	1 292	0
La15	20×5	1 650	1 414	1 229	1 222	1 214	1 207	1 207	1 207	0
La16	10×10	1 142	1 128	979	972	968	964	958	945	1.38
La17	10×10	1 026	941	806	803	795	795	793	784	1.15
La18	10×10	1 078	1 009	876	869	860	858	854	848	0.07
La19	10×10	1 093	995	877	875	866	860	857	842	1.78
La20	10×10	1 154	1 066	921	912	917	915	914	902	1.33

表 3　对比算法求得算例在不同比例下的完工时间

Table 3　Result for instances obtained by compared algorithms

算例	规模	设备缓冲区间容积与工件数量的比例/%					
		10	20	30	50	80	100
La01	10×5	784	704	692	688	684	679
La02	10×5	784	715	704	696	690	681
La03	10×5	707	654	646	641	638	632
La04	10×5	724	652	633	625	620	615
La05	10×5	663	625	623	617	613	609
La06	15×5	1 039	967	960	955	953	946
La07	15×5	984	941	925	923	918	914
La08	15×5	987	936	921	911	907	889
La09	15×5	1 064	1 022	984	973	964	957
La10	15×5	1 039	1 018	987	982	972	969
La11	20×5	1 328	1 294	1 282	1 269	1 258	1 255
La12	20×5	1 221	1 144	1 092	1 073	1 066	1 063
La13	20×5	1 335	1 271	1 235	1 221	1 208	1 196
La14	20×5	1 401	1 384	1 359	1 354	1 349	1 342
La15	20×5	1 428	1 343	1 282	1 272	1 259	1 254
La16	10×10	1 143	1 087	1 034	1 021	1 008	996
La17	10×10	957	904	869	864	855	849
La18	10×10	1 024	972	954	931	924	916
La19	10×10	1 013	977	942	937	924	921
La20	10×10	1 084	1 033	994	986	979	972

针对每组算例，计算表 3 与表 2 中结果的偏差. 从上述结果可以得到，本文提出的两阶段算法得出的结果均好于比较算法，平均偏差分别为 1.04%，8.03%，5.54%，4.93%，4.41% 和 3.85%. 在比例大于 20% 时尤为明显，再次验证了本文提出算法的有效性.

接下来分析讨论算法的稳定性. 针对每组算例，经过多次计算，选取其得到的最优解与最差解，分别如表 2，表 4 所示，并计算表 4 与表 2 中结果的偏差. 从上述结果可以得到，对于不同比例下的同组算例，在多次计算下得到的最差解与最优解的平均偏差分别为 2.21%，2.99%，2.92%，2.93%，2.92% 和 2.76%，充分验证了本文提出算法的稳定性.

缓冲区间容量大小对调度结果的影响根据表 2 的结果可知，以 100% 比例状态下的 make-span 值为基准，计算各比例状态下 make-span 值的偏差，结果如表 5 所示.

从表 5 可知，如果工件的加工时间符合均匀分布，当缓冲区间容量与工件数量的比例达到 20%，缓冲区间大小对调度结果的影响将会迅速变小. 故 20% 可被视为缓冲区间有限情况下的设置参考点.

表 4 每组算例多次计算得到的最差解
Table 4 Worst solution obtained by each instance

算例	规模	设备缓冲区间容积与工件数量的比例/%					
		10	20	30	50	80	100
La01	10×5	792	680	680	678	677	675
La02	10×5	789	694	693	693	691	688
La03	10×5	711	638	638	635	630	625
La04	10×5	734	621	620	617	615	611
La05	10×5	669	608	605	604	601	600
La06	15×5	1 049	945	941	939	936	933
La07	15×5	991	916	912	908	905	903
La08	15×5	986	902	899	895	894	891
La09	15×5	1 088	974	969	965	964	960
La10	15×5	1 061	981	977	972	969	966
La11	20×5	1 342	1 258	1 252	1 248	1 246	1 241
La12	20×5	1 227	1 068	1 064	1 057	1 055	1 049
La13	20×5	1 365	1 183	1 176	1 171	1 169	1 165
La14	20×5	1 423	1 342	1 341	1 338	1 335	1 335
La15	20×5	1 442	1 268	1 261	1 257	1 251	1 250
La16	10×10	1 158	1 012	1 004	999	996	994
La17	10×10	973	841	838	836	835	832
La18	10×10	1 035	908	904	902	898	894
La19	10×10	1 027	923	918	916	912	908
La20	10×10	1 091	953	948	943	941	936

表 5 不同缓冲区间容积下完工时间的偏差比例
Table 5 Result for deviations between the current proportion and 100% proportion

算例	规模	设备缓冲区间容积与工件数量的比例/%					
		0	10	20	30	50	80
La01~05	10×5	23.29	16.65	1.61	1.29	0.93	0.44
La06~10	15×5	26.79	10.33	0.37	0.16	0.16	0.07
La11~15	20×5	32.04	12.81	0.38	0.25	0.12	0
La16~20	10×10	25.72	17.46	1.9	1.4	0.68	0.36

4 结 语

本文针对考虑缓冲区间有限的作业车间调度问题,提出了基于邻域搜索的两阶段算法,包括快速得到可行解和基于非连通图的优化. 算法能够针对不同缓冲区间容积的情境进行求解,并通过与已有算法进行对比,验证了算法的有效性. 最后分析了缓冲区间的容量大小对调度完工时间 make-span 的影响.

参考文献:

[1] Brucker P, Heitmann S, Hurink J, et al. Job-shop scheduling with limited capacity buffers[J]. *OR Spectrum*, 2006, 28(2): 151–176.

[2] Lacomme P, Larabi M, Tchernev N. Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles[J]. *International Journal of Production Economics*, 2013, 143(1): 24–34.

[3] Hsieh Y C, You P S, Liou C D. A note of using effective immune based approach for the flow shop scheduling with buffers[J]. *Applied Mathematics and Computation*, 2009, 215(5): 1984–1989.

[4] Pan Q K, Wang L, Gao L. A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers[J]. *Applied Soft Computing*, 2011, 11(8): 5270–5280.

[5] Pan Q K, Wang L, Gao L, et al. An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers[J]. *Information Sciences*, 2011, 181(3): 668–685.

[6] Wang L, Zhang L, Zheng D Z. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers[J]. *Computers & Operations Research*, 2006, 33(10): 2960–2971.

[7] Xie Z P, Zhang C Y, Shao X Y, et al. Flow shop scheduling with limited buffers based on memetic algorithm[J]. *Computer Integrated Manufacturing Systems*, 2015, 21(5): 1253–1261.

[8] Pan Q K, Tasgetiren M F, Suganthan P N, et al. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem[J]. *Information Sciences*, 2011, 181(12): 2455–2468.

[9] Zhang C, Shi Z S, Huang Z W, et al. Flow shop scheduling with a batch processor and limited buffer[J]. *International Journal of Production Research*, 2017, 55(11): 3217–3233.

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

[11] Zeng C K, Tang J F, Yan C J. Job-shop cell-scheduling problem with inter-cell moves and automated guided vehicles[J]. *Journal of Intelligent Manufacturing*, 2015, 26(5): 845–859.

[12] Lawrence S. Supplement to resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques[R]. Pittsburgh: Carnegie Mellon University, 1984.

[13] Gröflin H, Klinkert A. A new neighborhood and tabu search for the blocking job shop[J]. *Discrete Applied Mathematics*, 2009, 157(17): 3643–3655.