求解多Hoist循环调度问题的帝国主义竞争算法

毛永年[1,2,3],唐秋华[2,3],张利平[2,3]
(1.遵义师范学院工学院,贵州 遵义 563000;2.武汉科技大学冶金装备及其控制教育部重点实验室,湖北 武汉 430081;
3.武汉科技大学机械传动与制造工程湖北省重点实验室,湖北 武汉 430081)

摘 要:自动化的电镀生产线通常设计有多台受计算机控制的Hoist(物料搬运设备),以便生产大批具有柔性制造特征的产品。基于搬运作业的最小时间间隔法,构建了多Hoist循环调度问题的混合整数线性规划模型,并首次使用基于群智能的元启发式算法(帝国主义竞争算法)求解该问题。借鉴遗传算法的进化机制,分别对搬运作业的优先关系序列、Hoist的分配序列进行不同的交叉、变异操作以实现帝国主义竞争算法的同化过程。针对种群进化过程中产生的大量不可行解,提出基于Hoist分配的不可行解修复策略以修复搬运作业优先关系。最后,基于标杆案例和随机案例,分别与专业优化软件CPLEX以及遗传算法进行对比,测试结果验证了所提出的方法的有效性。

关键词:帝国主义竞争算法;多Hoist循环调度;柔性制造;修复策略
中图分类号:TH16;F224 文献标识码:A 文章编号:1001-3997(2020)05-0054-05
DOI:10.19356/j.cnki.1001-3997.2020.05.013

Multi-Hoist Cyclic Scheduling Based on Imperialist Competitive Algorithm

MAO Yong-nian[1,2,3], TANG Qiu-hua[2,3], ZHANG Li-ping[2,3]
(1.College of Engineering and Technology, Zunyi Normal University, Guizhou Zunyi 563000, China;2.Key Laboratory of Metallurgical Equipment and Control Technology, Wuhan University of Science and Technology, Hubei Wuhan 430081, China;3.Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Hubei Wuhan University of Science and Technology, Hubei Wuhan 430081, China)

Abstract:*Automated electroplating production lines are usually designed with multiple hoists which are controlled by computer,so as to produce a large number of products with flexible manufacturing characteristics. Based on the method of minimum time interval between handling tasks, a mixed integer linear programming model for multi-hoist cyclic scheduling is constructed.And for the first time,a meta- heuristic algorithm based on swarm intelligence (imperialist competitive algorithm,* ICA) *is utilized to solve this problem. Assimilation stage of* ICA*is performedby implementing different crossover and mutation operations on priority sequence of handling tasksand hoist sequence, which is brought fromgenetic algorithm* (GA).*A repair strategy is proposed to repair unfeasible priority relationship. At last, based on benchmark and randomly generated instance, the effectiveness of the proposed method is demonstrated by comparing the results of* CPLEX *and* GA.

Key Words:**Imperialist Competitive Algorithm; Multi-Hoist Cyclic Scheduling; Flexible Manufacturing; Repair Strategy**

1 引言

Hoist调度问题[1]来源于以生产印刷电路板(Printed Circuit Board,PCB)为代表的电镀生产线。

印刷电路板(PCB)是计算机、手机、以及几乎所有自动化设备的重要元器件。在制造PCB的电镀工艺环节,工件(半成品)需要依次经历多个处理槽以完成必备的工艺要求。工件在这些处理槽中的加工过程具有一定的柔性,即工件的加工时间可以在给定的上下限范围内变动。工件的这种允许变动的加工时间范围被称为时间窗口。受计算机编程控制的Hoist必须在给定的时间窗口内,将工件从当前处理槽转移到其下一工序所在的处理槽;否则,工件将会因为制造缺陷而产生质量问题。为了便于管理,自动化的电镀生产线常采用循环生产模式,即Hoist在每一个生产循环内执行相同的搬运作业序列以完成大批量工件的生产。因此,规划与调度Hoist搬运作业对系统的生产效率、产品质量有重要影响。为了克服由运输设备(Hoist)造成的生产瓶颈,同一条轨道上通常安装有多台Hoist以协同完成电镀生产线上的运输作业。典型的多Hoist电镀生产线布局,如图1所示。图中,处理槽呈线性

来稿日期:2019-12-17
基金项目:中国博士后科学基金资助项目(2013M542073)
作者简介:毛永年,(1988-),男,湖北孝感人,博士研究生,主要研究方向:运筹学与运作管理,生产系统工程;
唐秋华,(1970-),女,湖北利川人,博士研究生,教授,主要研究方向:现代制造系统、制造业信息化、工业工程与管理

排列,且工件的入口和出口在同一个位置,在处理槽上方的轨道上,多台 Hoist 同时保持运行。

虽然,多台 Hoist 协同作业可以显著提高电镀生产线的生产效率[2-3],但同时也增加了调度 Hoist 搬运作业的难度。首先,相比于基本的单 Hoist 调度问题[1],多 Hoist 调度不仅要决策搬运作业的执行时间,同时还要决策 Hoist 的分配,从而使得问题规模变得更为庞大。此外,由于多台 Hoist 在同一条轨道上运行,必须处理它们之间潜在的碰撞冲突,从而增加了调度的复杂性。

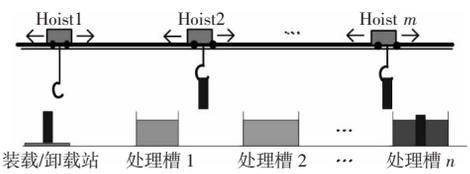

图 1 多 Hoist 电镀生产线示例

Fig.1 The Example of Electroplating Production Line with Multiple Hoists

文献[3]首次提出了多 Hoist 循环调度问题的混合整数线性规划(MILP)模型。针对该模型中 Hoist 搬运作业不能跨越周期的缺陷,文献[4]提出了一个改进的 MILP 模型以完善文献[3]的模型。针对具有双向工件运输流的电镀生产线,文献[2]分析了多种 Hoist 避碰条件,提出了该问题的 MILP 模型以及基于混合整数规划的分支定界算法。此外,基于启发式的求解方法[5-6]被提出以求解针对只有两台 Hoist 的循环调度问题。由于时间窗口约束使得该类问题的可行解空间十分有限,研制针对此类问题的元启发式算法变得极为困难。现有的相关研究[7-9]仅考虑一个运输资源(Hoist/Robot)的情形。针对多 Hoist 调度,文献[10]将电镀生产线划分为若干个互不重叠的区域, 每台 Hoist 只负责一个区域内的搬运作业,并使用模拟退火算法求解最佳的区域划分方法。然而,由于区域划分的方法限制了 Hoist 可能的移动区域,因而该方法不具有最优性。此外,区域划分的方法不适合的具有双向工件流的电镀生产线,如图 1 所示。

首次提出使用基于群智能的元启发式算法(帝国主义竞争算法)求解考虑具有重叠区域的多 Hoist/Robot 循环调度问题。首先, 基于搬运作业的最小时间间隔法建立该问题的 MILP 模型。考虑到此类问题的可行解空间极为有限这一特点[8],在设计算法时,提出在给定 Hoist 分配条件下的启发式目标函数以引导种群向有利方向进化。同时,为了提高算法搜索效率和求解质量,针对种群进化过程中产生的大量不可行解,提出基于 Hoist 分配的可行解修复策略以修复搬运作业的优先关系。最后,与专业优化软件 CPLEX 以及遗传算法的对比测试结果验证了所提出的方法的有效性。

2 问题描述和数学模型

研究图 1 所示的电镀生产线, 包括 m 台 Hoist、n 个处理槽。其中装载站 0 和卸载站(n+1)同在一个位置。Hoist 的编号从左到右依次是 1、2、…、m。每个工件自装载站进入系统,依次经历从处理槽 1 到处理槽 n,最后被 Hoist 运输到卸载站。装载/卸载站的容量设为无限大。处理槽 1 至 n 的容量为 1 个工件。由于处理槽之间没有缓冲设施,工件在当前处理槽完成加工后(必须满足时间窗口约束),须立即被 Hoist 运输到下一个工序所在的处理槽。

文献[2]基于搬运作业的最小时间间隔法构建了具有无碰撞约束的最优化 MILP 模型,为了研究方便,对该模型进行了简化处理,并使用如下定义:

搬运作业 i—Hoist 将工件从处理槽 i 上取出,然后将其搬运到处理槽 i+1,最后将其卸载到该处理槽内的全部过程,$0\leqslant i\leqslant n$;

L_i—工件在处理槽 i 上的最小处理时间,$0\leqslant i\leqslant n$;

U_i—工件在处理槽 i 上的最大处理时间,$0\leqslant i\leqslant n$;

d_i—执行搬运作业 i 所需的时间,$0\leqslant i\leqslant n$;

α_{ij}^{h}—搬运作业 i 的优先级高于搬运作业 j 时,执行此两项搬运作业的最小时间间隔。其中 $h=q-p$,p、q 分别为执行搬运作业 i 和 j 的 Hoist 编号。

M—足够大的正数。

该问题的决策变量为:

T—周期长度;

t_i—执行搬运作业 i 的开始时间,$0\leqslant i\leqslant n$;

$$z_{ik}=\begin{cases}1 \text{ 如果搬运作业 } i \text{ 由 Hoist } k \text{ 执行}\\ 0 \text{ 否则}\end{cases};$$

$0\leqslant i\leqslant n$,$1\leqslant k\leqslant m$;

$$y_{ij}=\begin{cases}1 \text{ 如果 } t_j-t_i\geqslant\alpha_{ij}^{h}\\ 0 \text{ 如果 } t_i-t_j\geqslant\alpha_{ji}^{-h}\end{cases};0\leqslant i,j\leqslant n。$$

文献[2]模型的简化版可以表示为:

$$\text{s.t.:}\sum_{k=1}^{m}z_{ik}=1,0\leqslant i\leqslant n \tag{1}$$

$$y_{ij}+y_{ji}=1,0\leqslant i<j\leqslant n \tag{2}$$

$$t_i-t_{i-1}\geqslant d_{i-1}+L_i-(1-y_{ij})M,1\leqslant i\leqslant n \tag{3}$$

$$t_i-t_{i-1}\leqslant d_{i-1}+U_i+(1-y_{ij})M,1\leqslant i\leqslant n \tag{4}$$

$$(t_i+T)-t_{i-1}\geqslant d_{i-1}+L_i-y_{ij}M,1\leqslant i\leqslant n \tag{5}$$

$$(t_i+T)-t_{i-1}\leqslant d_{i-1}+U_i+y_{ij}M,1\leqslant i\leqslant n \tag{6}$$

$$t_j-t_i\geqslant\alpha_{ij}^{q-p}-(3-z_{ip}-z_{jp}-y_{ij})M,0\leqslant i<j\leqslant n,1\leqslant p,q\leqslant m \tag{7}$$

$$t_i-t_j\geqslant\alpha_{ij}^{p-q}-(2-z_{ip}-z_{jp}+y_{ij})M,0\leqslant i<j\leqslant n,1\leqslant p,q\leqslant m \tag{8}$$

$$(t_j+T)-t_i\geqslant\alpha_{ij}^{q-p}-(2-z_{ip}-z_{jp})M,0\leqslant i,j\leqslant n,i\neq j,1\leqslant p,q\leqslant m \tag{9}$$

$$0\leqslant t_i<T,1\leqslant i\leqslant n \tag{10}$$

模型的目标函数是最小化周期长度 T。式(1)表示任何一项搬运作业只能分配给一台 Hoist。式(2)强调任意两个搬运作业之间只有唯一一种优先关系。式(3)~式(6)为时间窗口约束。当 $y_{i+1,i}=1$ 时,式(3)~式(4)强调工件的加工时长不小于允许的最小加工时间 L_i 同时不大于允许的最大加工时间 U_i。当 $y_{i+1,i}=0$ 时,工件在处理槽 i 上的加工跨越了两个周期,那么工件的实际加工时长为(t_i+T)-(t_{i-1}+d_{i-1})。式(5)~式(6)表示,当 $y_{i+1,i}=0$ 时,工件的加工时长同样不小于允许的最小加工时间 L_i,同时不大于允许的最大加工时间 U_i。式(7)~式(9)是与 Hoist 移动轨迹相关的约束。假设 Hoist p 执行搬运作业 i、Hoist q 执行搬运作业 j (即 $z_{ip}=z_{jq}=1$),式(7)表示当搬运作业 i 的优先级高于搬运作业 j 时(即 $y_{ij}=1$),搬运作业 i、j 的开始时间间隔 t_j-t_i 不小于参数 α_{ij}^{q-p},从而保证执行搬运

作业 i、j 的 Hoist 有足够的时间来完成相应的移动或者避免潜在的碰撞冲突。当 y_{ij}=0 时，式(8)与式(7)的情形相反。式(9)保证 Hoist 移动轨迹在两个周期之间的连续性，即：在完成上一周期内的搬运作业后，每一台 Hoist 都有足够的时间回到下一个周期的起始位置。不失一般性，式(10)表示任何搬运作业的开始时间必须在一个周期[0,T)的调度域内。

3 改进的帝国主义竞争算法

帝国主义竞争算法(Imperialist Competitive Algorithm，ICA)[11]是一种受社会行为启发的群智能随机搜索算法，该算法在调度及优化等领域得到广泛应用[12]。与遗传算法[13]相比，ICA 的多种群进化机制使得算法不易陷入局部最优。利用该特点，同时将遗传算法(GA)的交叉、变异操作引入到 ICA 的同化机制中，使得改进的帝国主义竞争算法同时具有 GA 的特征。

3.1 编码方式

染色体编码由两部分组成，分别用序列 μ=$\{h_0,h_1,\cdots,h_n\}$ 与 λ=$\{x_0,x_1,\cdots,x_n\}$ 序列表示种群中任意一条染色体。序列 μ 中的第 i 个基因 h_i 对应执行搬运作业 i 的 Hoist 编号；序列 λ 中的第 i 个基因 x_i 所拥有的搬运作业优先级为 i。序列 λ 中的搬运作业越靠前，其优先级越高。用 8 项搬运作业为例说明了此编码方式，如图 2 所示。

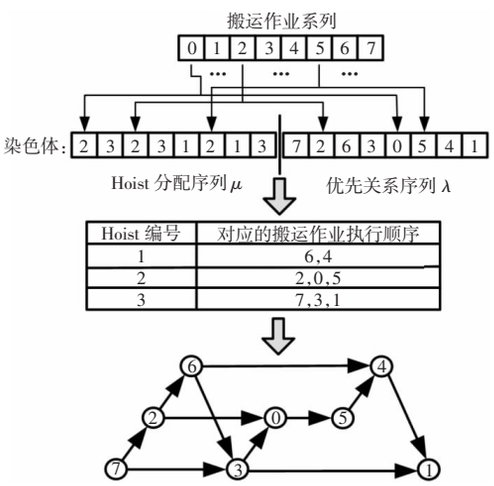

图 2 编码示例

Fig.2 An Example of Coding

在图 2 中，由于基因“2”分别在序列 μ 中的第 0、2、5 个位置出现，因此，编号为 0、2、5 的搬运作业分配给 2 号 Hoist 执行。其次，由于此三项任务在序列 λ 中出现的顺序为 2、0、5，因此，2 号 Hoist 按照此顺序依次执行这三项搬运任务。同理，可获得其它 Hoist 的搬运作业以及排序。此外，根据搬运作业在序列 λ 中出现的先后顺序，也可获得由不同 Hoist 执行的搬运作业之间的优先关系。

3.2 启发式评价函数

所研究问题的标准目标函数为：

$$f_1=T \tag{11}$$

当搬运作业优先关系序列 λ 和 Hoist 分配序列 μ 为已知时。式(1)~式(10)则可转换为如下形式：

$$t_j-t_i\geqslant a+b\cdot T,0\leqslant i,j\leqslant n \tag{12}$$

式中：a—实数，b 的取值为−1、0 或者 1。

事实上，式(12)是一类特殊的线性规划问题，可以用赋权有向图来描述[7-8]。文献[14]提出了计算复杂度为 $o(n^2m\log n)$ 的多项式算法以求解此问题，此处 n 为图中节点数量，m 为图中弧的数量。为了提高算法运行效率，在使用文献[14]中的多项式算法求解染色体(u,λ)的最短周期 T 之前，可先对染色体(u,λ)进行可行性判断。如果该染色体不可行，则可直接设置 $f_1=T^0$，其中 $T^0=\sum_{i=1}^{n}(d_{i-1}+U_i)$。

染色体(u,λ)的不可行主要由Hoist 移动能力不能满足工件的加工时间窗口所导致。当 $y_{i-1,i}$=1 时，若序列$\{x_0,\cdots,i\text{–}1\rightarrow\cdots\rightarrow i,\cdots,x_n\}$ 中从搬运作业 i–1 到搬运作业 i 的优先关系序列不可行，则染色体(u,λ)一定不可行；当 $y_{i-1,i}$=0 时，若序列$\{i\text{–}1\rightarrow\cdots x_n\,|x_0\rightarrow\cdots\rightarrow i\}$ 中从搬运作业 i–1 到搬运作业 i 的优先关系序列不可行，染色体(u,λ)也一定不可行。为了方便描述，定义序列 S^j 为序列 λ 中分别以搬运作业 i–1 开始、搬运作业 i 结束的一串有序优先关系子序列。由于一个循环序列 λ 包含 n 个 S^j，因此，任意一个子序列不可行都会导致整条染色体不可行。交叉、变异操作会产生大量的不可行解，这些不可行解隐含了种群的进化信息，为了区分这些不可行解，记参数 ω 为所有不可行子序列 S 的个数之和，并采用式(13)计算不可行染色体的适应度值。

$$f_2=\omega^3 \tag{13}$$

为了统一染色体适应度值的计算，采用公式(14)来计算任意一条染色体的适应度值。

$$F=T+\omega^3 \tag{14}$$

对于序列 S^j，可采用的算法来判断其可行性，如图 3 所示。

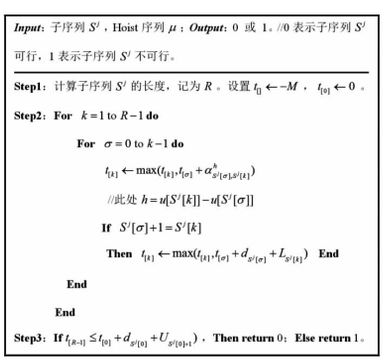

图 3 子序列的可行性判断

Fig.3 Feasibility Judgment of Subsequence

3.3 同化与竞争

同化、竞争是 ICA 的两个进化机制。在 ICA 中，帝国和其所属的殖民地可以看成一个独立的种群。种群中适应度值最好的一条染色体(解)被视为帝国，其它染色体都为其殖民地。每个帝国(种群)的进化是通过其内部的同化机制实现的。而帝国之间争夺殖民地的过程被成为竞争，该过程体现了帝国(种群)之间的信息交流。在标准的 ICA 中，同化是所属殖民地全部向帝国靠近的过程。引入遗传算法的进化机制实现此过程。具体的，利用锦标赛选择法进行选择操作，针对搬运作业的优先关系序列 λ 使用双点交叉操作，如图 4 所示。针对与之对应的 Hoist 分配序列 μ 使用单点交换交叉操作，如图 5 所示。针对序列 λ 的变异操作使用 Swap 变异，即随机选择两个基因位置并交换其基因，针对序列 μ 则使用随机变异。

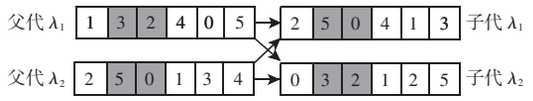

图 4 针对序列 λ 的双点交叉操作

Fig.4 Two-Point Crossover for Sequence λ

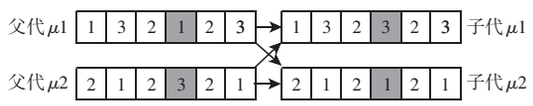

图 5 针对序列 μ 的交换交叉操作

Fig.5 Swap Crossover for Sequence μ

使用标准的 ICA 竞争机制，即势力最大的帝国夺取势力最小的帝国中最弱的殖民地个体，因此，各帝国殖民地的规模在竞争中动态变化。考虑到所研问题可行解空间极少这一特点，设置一个参数 $\rho(0<\rho<1)$，ICA 在每次迭代中以小于 ρ 的概率进行竞争操作，从而适当控制殖民地掠夺这一过程，以尽可能的维持种群的多样性。

3.4 给定 Hosit 分配时的不可行解修复策略

由于算法进化过程会产生大量的不可行解，针对不可行解的修复策略对提高算法的运行效率十分关键。在给定 Hosit 分配时，仅仅修复序列 λ 的某个特定子序列 S^j 往往会导致序列 λ 的其它子序列不可行，从而影响到修复策略的实际效果。这里提出，在修复不可行子片段 S^j 时，采用联动机制，即在修复 S^j 子片段后同时修复与其相关的上游子片段 S^{j-1} 或者下游子片段 S^{j+1}，并根据搜索进程不断更新修复目标 S，直到满足终止条件。同时，使用方向禁忌表 v 记录序列 λ 中各基因的移动方向，从而避免迂回搜索。针对序列 λ 的修复策略具体列出，如图 6 所示。

Input：染色体 (u,λ)，修复次数 η；***Output***：染色体 (u,λ')。

Step1：设置向量 v[]←0；参数 $obj\leftarrow1$，$g1\leftarrow0$，$g2\leftarrow0$，$p1\leftarrow0$，$p2\leftarrow0$，$n\leftarrow0$。

Step2：如果 $n>\eta$，则进入 **Step7**；否则 $n\leftarrow n+1$，进入 **Step3**。

Step3：//找到需要修复的子片段 S^j。

探测序列 λ 的每一个子序列 S，找到最大不可行子片段 S^j。分别设置 $g1$、$g2$ 为片段 S^j 的第一个元素以及最后一个元素，同时设置 $p1$、$p2$ 分别为 $g1$、$g2$ 在序列 λ 中的位置。

Step4：//确定需要移动的目标基因 obj，若 $obj\leftarrow1$，表示移动基因 $g1$；否则移动基因 $g2$。

If v[g1]!=-1 且 v[g2]=1，**then** $obj\leftarrow1$，v[g1]←1；

Else if v[g1]=-1 且 v[g2]!=1，**then** $obj\leftarrow0$ v[g2]←-1；

Else if v[g1]=-1 且 v[g2]=1，**then** 进入 **Step7**；

Else

如果 *Rand*()<0.5，则 $obj\leftarrow1$，v[g1]←1；否则 $obj\leftarrow0$，v[g2]←-1。

End

Step5：//移动目标基因。

如果 $obj=1$，则在序列 λ 中顺时针移动基因 $g1$ 直到子序列 S^{g2}（以 $g1$ 为首的子序列）为可行；否则，在序列 λ 中逆时针移动 $g2$ 直到 S^{g2} 为可行。分别用 $p1$、$p2$ 记录 $g1$、$g2$ 在序列 λ 中的位置。

Step6：//探测上游或下游序列是否可行。

If $obj=1$，**then** 如果序列 S^{g2} 可行，则进入 **Step2**；否则，设置 $g1\leftarrow g1-1$，$g2\leftarrow g1+1$，$obj\leftarrow1$，v[g1]←1，进入 **Step5**。**Else** 如果序列 S^{g2+1} 可行，则进入 **Step2**；否则，设置 $g1\leftarrow g2$，$g2\leftarrow g1+1$，$obj\leftarrow0$，v[g2]←-1，进入 **Step5**。**End**

Step7：输出染色体 (λ,u)，停止。

图 6 不可行解修复策略

Fig.6 Repair Strategy for Infeasible Solution

3.5 改进的帝国主义竞争算法步骤

综上所述，设计的帝国主义竞争算法步骤如下：

Step1：设置种群总规模 N_{pop}、帝国主义国家数量 N_{imp}、竞争概率 ρ、交叉概率 P_c、变异概率 P_m、最大修复循环次数 η。

Step2：随机初始化种群，平均分配殖民地。

Step3：对帝国 $C_i(1\leqslant i\leqslant N_{imp})$ 进行评价，保存帝国 C_i 新产生的最好解。如果帝国 C_i 没有新的最好解产生，则应用精英保留策略，即：用帝国 C_i 的历史最好解随机替换 C_i 的一个殖民地。

Step4：更新全局最优解。如果满足终止条件则算法停止；否则，进入下一步。

Step5：对帝国 $C_i(1\leqslant i\leqslant N_{imp})$ 进行同化、竞争操作。

Step6：对种群中的不可行解进行可行性修复。跳转到 Step3。

4 算法验证

为了验证算法的性能，在 CPU 为 3.2GHz 的 PC 机上使用 C++语言对改进的 ICA 进行编码。同时，将遗传算法（GA）作为对照项。GA 使用与 ICA 相同的交叉、变异操作。此外，在 Microsoft Visual Studio 2010 软件平台下调用专业优化软件 IBM ILOG CPLEX（版本 12.5）求解 MILP 模型以获得案例的最优解。

标杆案例 P&U[1]的测试结果，如表 1 所示。该案例的参数设置与文献[2]相同。ICA 的相关参数设置为：$N_{pop}=60$，$N_{imp}=3$，$\rho=0.1$，$P_c=0.9$，$P_m=0.1$，$\eta=3$。针对 GA，除了设置 $N_{imp}=1$ 以外，其它参数设置与 ICA 相同。ICA 和 GA 的终止条件为：全局最优解连续 200 代不更新则终止。

表 1 标杆案例 P&U 的测试结果

Tab.1 The Result of Benchmark Instance P&U

m	最好解			运行时间/s		
	CPLEX	GA	ICA	CPLEX	GA	ICA
1	514	514	514	0.28	1.0	1.7
2	236	238.5	236	12.3	15	25
3	196	196	196	88.4	4.9	3.1

从表 1 可以看出，GA 和 ICA 在较快的时间内给出了标杆案例在 m=1 和 m=3 时的最优解；当 m=2 时，ICA 给出了最优解而 GA 给出了近优解。

为了进一步验证算法的性能，使用 CPLEX、GA、ICA 求解了问题规模 n 分别为 10、12、14、16、18 时的随机案例。随机案例的设计方法与文献[2]相同。ICA 的相关参数设置为：$N_{pop}=100$，$N_{imp}=4$，$\rho=0.1$，$P_c=0.9$，$P_m=0.1$，$\eta=5$。针对 GA，除了设置 $N_{imp}=1$ 以外，其它参数则与 ICA 相同。终止条件为：迭代次数达到 200（n-5）代终止。针对每一个问题规模，30 项随机案例求解结果的平均值，如图 7、表 2 所示。

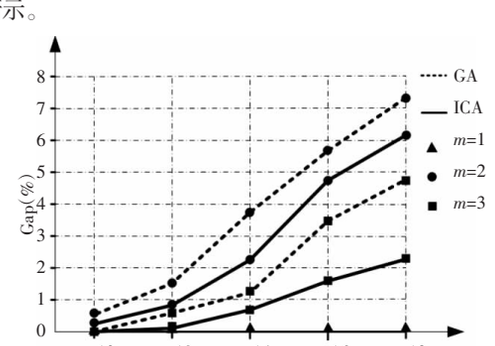

图 7 GA 和 ICA 所获得的最好解与最优解的平均百分偏差

Fig.7 The Mean Gap(%) of Solusion by GA and ICA Compared with the Optimal Solution

从图 7 可以看出，当 m=1 时，GA 和 ICA 每次都能获得最优解。这表明所提出的算法针对带时间窗口的单 Hoist 循环调度问题十分有效。当 m=2、3 时，GA 和 ICA 所获得的结果与最优解有一定的偏差，ICA 的结果要优于 GA。此外，针对同样的问题规模，算法的偏差并没有因为 Hoist 数量的持续增加而扩大。相反，m=3 时算法所获得的结果要明显比 m=2 时更接近最优解。

表 2 随机案例的平均运行时间/s
Tab.2 The Mean CPU Time(s) for the Randomly Generated Instance

n	m	ICA	GA	CPLEX
10	1	6.10	5.30	0.10
	2	10.7	9.20	0.55
	3	9.28	7.41	3.78
12	1	7.44	6.17	0.23
	2	17.6	14.2	1.67
	3	16.9	10.4	15.3
14	1	11.7	9.49	0.25
	2	29.5	24.7	14.6
	3	26.9	19.8	60.6
16	1	18.3	14.9	0.37
	2	65.4	49.3	257
	3	51.2	42.9	214
18	1	27.8	20.2	0.64
	2	98.7	102	1425
	3	84.4	92.1	852

另一方面，从表 2 所给出的运行时间来看，ICA 的求解时间要略大于 GA 的求解时间。此外，在求解小规模问题时，CPLEX 表现出了较快的求解速度。但是，由于问题的解空间随着 m 或 n 的增大而快速扩大，当 m 与 n 增长到一定程度时，CPLEX 的求解时间则要远远大于群智能算法(ICA/GA)所需要的运行时间。因此，ICA/GA 在求解中等或者大规模问题时展现出了较强的优势。

5 结论

首次提出基于群智能的元启发式算法(ICA)求解带时间窗口的多 Hoist 循环调度问题。将 Hoist 搬运作业的优先关系以搬运作业序列的方式来表达，以方便处理 Hoist 无碰撞约束。提出了在给定 Hoist 分配条件下的启发式目标函数，同时，针对种群进化过程中产生的大量不可行解，提出了搬运作业优先关系的修复策略以提高算法的搜索效率以及求解精度。最后，基于标杆案例和随机案例，分别与专业优化软件 CPLEX 以及遗传算法进行对比，测试结果验证了所提出的方法的有效性。

参考文献

[1] Phillips L W，Unger P S.Mathematical programming solution of a hoist scheduling program[J].AIIE Transactions，1976，8(2)：219–225.

[2] Jiang Yun，Liu Ji-yin.A new model and an efficient branch-and-bound solution for cyclic multi-hoist scheduling[J].IIETransactions，2014，46(3)：249–262.

[3] Leung J M Y，Zhang G，Yang X.Optimal cyclic multi-hoist scheduling：a mixed integer programming approach[J].Operations Research，2004，52(6)：965–976.

[4] Che A，Lei W，Feng J.An improved mixed integer programming approach for multi-hoist cyclic scheduling problem[J].IEEE Transactions on Automation Science & Engineering，2014，11(1)：302–309.

[5] Liu J.A heuristic algorithm for the two-hoist cyclic scheduling problem with overlapping hoist coverage ranges[J].IIE Transactions，2008，40(8)：782–794.

[6] Chtourou S，Manier M A，Loukil T.A hybrid algorithm for the cyclic hoist scheduling problem with two transportation resources[J].Computers & Industrial Engineering，2013，65(3)：426–437.

[7] 晏鹏宇，车阿大，李鹏.具有柔性加工时间的搬运机器人制造单元调度问题改进遗传算法[J].计算机集成制造系统，2010，16(2)：404–413.
(Yan Peng-yu，Che A-da，Li Peng.Improved genetic algorithm for robotic cell scheduling problem with flexible processing times[J].Computer Integrated Manufacturing Systems，2010，16(2)：404–410.)

[8] 王跃岗，车阿大.基于混合量子进化算法的自动化制造单元调度[J].计算机集成制造系统，2013，19(9)：2193–2201.
(Wang Yue-gang，Che A-da.Robotic cells scheduling based on hybrid quantum evolutionary algorithm [J].Computer Integrated Manufacturing Systems，2013，19(9)：2193–2201.)

[9] Lim J M.A genetic algorithm for a single hoist scheduling in the printed-circuit-board electroplating Line[J].Computer and Industrial Engineering，1997，33(3)：789–792.

[10] 杨广文，鞠大鹏，郑纬民.利用模拟退火技术求解多 Hoist 调度问题[J].软件学报，2001，12(1)：11–17.
(Yang Wen-guang，Ju Da-peng，Zheng Wei-min.Solving multiple hoist scheduling problems by use of simulated annealing[J].Journal of Software，2001，12(1)：11–17.)

[11] Esmaeil AG，Caro L.Imperialist competitive algorithm：an algorithm for optimization inspired by imperialistic competition[C]//Evolutionary Computation，2007.CEC 2007.IEEE Congress on.IEEE，2008：4661–4667.

[12] 陈志楚，李聪，张超勇.基于帝国主义竞争算法的切削参数优化[J].制造业自动化，2012，34(24)：10–15.
(Chen Zhi-chu，Li Cong，Zhang Chao-yong.Cutting parameters optimization based on imperialist competitive algorithm[J].Manufacturing Automation，2012，34(24)：10–15.)

[13] 师名林.基于遗传算法的欠驱动双足机器人步态优化设计[J].机械设计与制造，2017(6)：225–229.
(Shi Ming-lin.Gait optimization design of under-actuated biped robot based on genetic algorithm[J].Machinery Design & Manufacture，2017(6)：225–229.)

[14] Kats V，Levner E.Cyclic routing algorithms in graphs：performance analysis and applications to robot scheduling[J].Computers & Industrial Engineering，2011，61(2)：279–288.