第2类U型装配线平衡问题的双阶段蚁群算法

郑巧仙[1] 何国良[2] 李 明[3] 唐秋华[4]

(湖北大学计算机与信息工程学院 武汉 430062)[1] (武汉大学计算机学院 武汉 430072)[2]

(武汉科技大学理学院 武汉 430081)[3] (武汉科技大学机械自动化学院 武汉 430081)[4]

摘 要 针对电子、汽车等行业中普遍存在的第2类U型装配线平衡问题(UALBP-2),提出了一种双阶段蚁群算法。强调全局搜索的第一阶段算法利用探路蚁,根据操作选择和分配策略以及迭代压缩机制快速得到问题的较优解,减小搜索空间;注重局部搜索的第二阶段算法利用搜索蚁,根据所提的信息素减小更新策略在包含最优解且不断减小的搜索空间中搜索各工位的不同精英负载,基于精英复制策略利用精英蚁将其组合为问题的可行解。对18个标杆算例的33个实例的求解结果验证了所提算法的有效性和稳定性。

关键词 U型装配线平衡问题,双阶段蚁群算法,组合优化

中图法分类号 TP311,TP18 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.06.034

Two Stage Ant Colony Optimization for Type 2 of U-shaped Assembly Line Balancing Problem

ZHENG Qiao-xian[1] HE Guo-liang[2] LI Ming[3] TANG Qiu-hua[4]

(College of Computer and Information Engineering, Hubei University, Wuhan 430062, China)[1]

(School of Computer, Wuhan University, Wuhan 430072, China)[2]

(College of Science, Wuhan University of Science and Technology, Wuhan 430081, China)[3]

(College of Machinery and Automation, Wuhan University of Science and Technology, Wuhan 430081, China)[4]

Abstract A two stage ant colony optimization for the type 2 of U-shaped assembly line balancing problem (UALBP-2) was proposed, which is widespread in the electronics and automobile industry. In the first stage algorithm with the high capability of global search, a better feasible solution is obtained by the scout ants according to the task selection strategy, the task assignment strategy and the iteration compress mechanism. The search space is decreased according to the solution. In the second stage algorithm with the high capability of local search, different elite station loads are searched by the pathfinding ants according to the update strategy of decreasing pheromones. The elite station loads of every station are grouped together into the feasible solutions of UALBP-2 by the elite ants according to the elite copy strategy. The computational results of 33 instances from 18 benchmark examples verify the effectiveness and the stability of the proposed algorithm.

Keywords U-shaped assembly line balancing problem, Two stage ant colony optimization, Combinatorial optimization

1 前言

装配是电子、汽车等加工业的核心制造工艺。装配时,未完成产品在传送带上被连续地从一个工位传送至下一个工位,每个工位需在限定的时间(称为节拍)内重复地完成几项操作。操作的作业时间称为操作时间,各个工位上的操作时间之和称为该工位的工位时间,分配至某个工位上的操作称为该工位的工位负载。工艺和逻辑关系要求部分操作之间需按一定的先后顺序作业,称为优先关系。

图1为装配操作间的一个优先关系图。图中共有10项操作,其序号由圆圈中的数字标识,其操作时间由圆圈上方的数值表示,操作间的直接优先关系由箭头的方向确定。

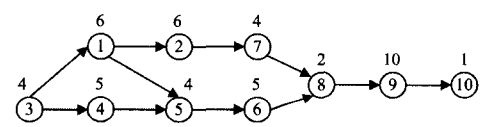


图1 优先关系图

装配线中,如果入口和出口被设计在同一个位置(见图2),

到稿日期:2016-05-05 返修日期:2016-07-09 本文受湖北省教育厅科学技术研究项目(D20161104),武汉科技大学青年科技骨干基金(2015XZ031)资助。

郑巧仙(1978—),女,博士,副教授,主要研究方向为调度优化问题的智能算法,E-mail:lmzqx@163.com;何国良(1974—),男,博士后,副教授,主要研究方向为数据挖掘、智能信息处理;李 明(1976—),男,博士,副教授,主要研究方向为调度优化问题建模、启发式算法设计;唐秋华(1970—),女,博士,教授,主要研究方向为现代制造系统、制造业信息化。

则该装配线称为 U 型装配线。在 U 型装配线中，同一个工位可含有两类操作：入口操作和出口操作，即所有前序操作都已经被分配至前序或当前工位的操作和所有后继操作都已经被分配至前序或当前工位的操作。与传统的直线型布局相比，U 型布局更紧凑，柔性更好，装配效率更高，更有利于采用精益生产模式组织生产[1]。

图 2 就图 1 中的数据给出了 U 型装配线的一种布局。图 2 中，10 项操作被分配至 3 个工位，其中工位 I 含有操作 3，10，9，8；工位 II 含有操作 4，1，6；工位 III 含有操作 2，5，7。在工位 I 中，操作 3 为入口操作，其余操作为出口操作；而在工位 III 中，所有操作既是入口操作也是出口操作。

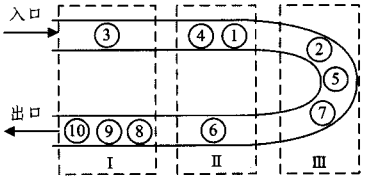


图 2　U 型装配布局示意图

装配线平衡问题（Assembly Line Balancing Problem，ALBP）是指将存在优先关系约束的操作分配至预先设定的工位，使得某些目标达到最优的一系列优化问题。常见的优化目标有两类：1）给定装配线的节拍，最小化其工位个数；2）给定装配线的工位个数，最小化其节拍[2]。相应的 U 型装配线平衡问题（the U-shaped Assembly Line Balancing Problem，UALBP）分别称为第 1 类 U 型装配线平衡问题（UALBP-1）和第 2 类 U 型装配线平衡问题（UALBP-2）。两类问题均需满足以下约束：1）每项操作都必须被分配且仅被分配至一个工位；2）每个工位的工位时间不能超过装配线的节拍；3）如果操作 i_1 是操作 i_2 的前序，则当 i_2 是入口操作时，i_1 也必须是入口操作，当 i_1 是出口操作时，i_2 也必须是出口操作。两类问题都是组合优化中的 NP 难问题。

U 型装配线平衡问题由 Miltenburg 和 Wijngaard 于 1994 年首次建立模型[3]，目前已有多种算法求解 UALBP-1。Shwetank 等提出了一种可提高劳动生产率的启发式算法[4]；Toksarı 等结合问题的现场需求，对带有学习效果的 UALBP，提出了一种多项式算法[5]；Özcan 和 Toklu 针对 UALBP，提出了一种融合了学习机制和模拟退火策略的改进启发式算法[6]；Kim 等针对融合了平衡和排序要求的混合 UALBP，提出了一种内共生进化算法[7]；Nourmohammadi 等针对多目标 UALBP，提出了一种帝国主义竞争算法[8]。虽然已有较多算法求解 UALBP-1，但求解 UALBP-2 的算法还很少。

作为一种群智能进化算法，蚁群算法具有自组织性、并行性、正反馈性等优良特性，在求解调度优化问题时表现卓越[9-10]。已有多名学者应用蚁群算法求解 ALBP。Ihsan 等针对单产品 UALBP 提出了一种蚁群算法[11]；Adil B 和 Türkay 将已有的 COMSOAL 和位阶权重法融入蚁群算法，对直线型和 U 型 ALBP 进行了求解[12]；郑巧仙等对第 2 类单边[13]和双边[10] ALBP 分别提出了一种改进的蚁群算法，有效地对问题进行了求解；张则强等对第 1 类 ALBP 提出了一种改进的蚁群算法[14]。上述蚁群算法致力于对所有工位进行全局搜索，即各个蚂蚁对所有工位进行操作分配，然后得到多个可行解，释放信息素，进入下一次遍历搜索。此搜索方式主要关注问题的整体解，而对各个工位负载的局部搜索较差，所以当优化效果达到一定程度后，难以进一步优化。

Zheng 等针对第 2 类单边装配线平衡问题，提出了一种面向工位的蚁群算法[15]。该算法致力于各个工位的局部搜索，通过局部优化实现全局优化，这种思想能够有效、稳定地求解装配线平衡问题。但是该算法存在蚂蚁的探索能力较弱、收敛速度较慢等不足。

本文以第 2 类 U 型装配线平衡问题为研究对象，将 Zheng 等所提算法的思想进行进一步改进，提出了一种新颖的蚁群算法：双阶段蚁群算法。其中第一阶段蚁群算法侧重于全局搜索，可快速缩小算法的有效搜索空间；第二阶段蚁群算法强调局部搜索，强化搜索各工位的优良负载，提高解的求解质量。最后利用标杆算例，验证所提算法的有效性。

2　数学模型

记 $I=\{i\mid i=1,2,\cdots,n\}$ 为问题的操作集合；t_i 为操作 i 的操作时间。$J=\{j\mid j=1,2,\cdots,m\}$ 为工位集合；ST_j（$j\in J$）为工位 j 的工位时间。P 为直接优先关系矩阵 $P=(P_{i_1,i_2})_{n\times n}$，其中 $P_{i_1,i_2}=1$，如果操作 i_2 是 i_1 的直接前序；否则，$P_{i_1,i_2}=0$；Q 为优先关系矩阵，$Q=(Q_{i_1,i_2})_{n\times n}$，其中 $Q_{i_1,i_2}=1$，如果操作 i_2 是 i_1 的前序；否则 $Q_{i_1,i_2}=0$，$i_1,i_2\in I$。C 为装配线的节拍；$\bar{C}$ 为装配线的理想节拍，$\bar{C}=\lceil\frac{1}{m}\sum_{i\in I}t_i\rceil$。$K$ 为装配线入口操作和出口操作的记号集合，$K=\{k\mid k=1,2\}$，其中 $k=1$ 表示操作为入口操作，$k=2$ 表示操作为出口操作。定义 0-1 变量 x_{ijk}，当操作 i 作为 k 种类型的操作被分配至工位 j 时，$x_{ijk}=1$；否则 $x_{ijk}=0$，$i\in I,j\in J,k\in K$。

假设参与分配的操作时间是确定且相互独立的；操作间的优先关系是给定的；整个进程中不存在并行工位和工作缓冲区；不考虑操作员的步行时间且不存在不确定性因素。则在给定工位个数的前提下，以最小化装配线的节拍为优化目标，建立 UALBP-2 的数学模型：

$$\min C$$

$$\text{s.t.}\sum_{i\in I}\sum_{k\in K}t_i\cdot x_{ijk}\leqslant C,j=1,2,\cdots,m \tag{1}$$

$$\sum_{k\in K}\sum_{j\in J}x_{ijk}=1,i=1,2,\cdots,n \tag{2}$$

$$\sum_{j_2\in J}x_{i_2j_21}\leqslant\sum_{j_1\in J}x_{i_1j_11},\text{if } P_{i_2,i_1}=1 \tag{3}$$

$$\sum_{j_1\in J}j_1\cdot x_{i_1j_11}\leqslant\sum_{j_2\in J}j_2\cdot x_{i_2j_21}+M\sum_{j_3\in J}j_3\cdot x_{i_2j_32},\ \text{if } P_{i_2,i_1}=1 \tag{4}$$

$$\sum_{j_1\in J}x_{i_1j_12}\leqslant\sum_{j_2\in J}x_{i_2j_22},\text{if } P_{i_2,i_1}=1 \tag{5}$$

$$\sum_{j_2\in J}j_2\cdot x_{i_2j_22}\leqslant\sum_{j_1\in J}j_1\cdot x_{i_1j_12}+M\sum_{j_3\in J}j_3\cdot x_{i_1j_31},\ \text{if } P_{i_2,i_1}=1 \tag{6}$$

$$x_{ijk}\in\{0,1\},i\in I,j\in J,k\in K \tag{7}$$

其中，式(1)表示各个工位的工位时间不能超过装配线的节拍 C；式(2)表示每项操作都必须作为一种类型的操作(出口操作或入口操作)被分且仅能分至一个工位。假设操作 i_1 为操作 i_2 的直接前序，式(3)－式(6)为其相应的优先关系约束，依次表示：如果 i_2 为入口操作，则 i_1 必须为入口操作；如果 i_1，i_2 都为入口操作，则它们所在工位的序号 j_1，j_2 应满足 $j_1 \leqslant j_2$；如果 i_1 为出口操作，则 i_2 必须为出口操作；如果 i_1，j 都为出口操作，则它们所在工位的序号 j_1，j_2 应满足 $j_1 \geqslant j_2$。式中，M 为一个较大的常数。

3 双阶段蚁群算法

双阶段蚁群算法借助 3 类蚂蚁对问题进行求解，它们分别称作探路蚁、搜索蚁和精英蚁。其中探路蚁用于获得问题的较优可行解，从而得到较小的工位时间变化区间 $[C_{\min}, C_{\max}]$，减小问题的搜索空间；搜索蚁用于搜索各个工位；满足定界条件式(8)的不同工位负载称为精英工位负载。

$$C_{\min} \leqslant ST_j \leqslant C_{\max} \tag{8}$$

精英蚁用于保留前面工位的精英负载至下一个工位。

3.1 算法流程

算法共分两个阶段。第一阶段利用少量探路蚁，基于 3 种操作选择策略、操作分配策略[2]和迭代压缩机制[15]快速得到 UALBP-2 的一个较优可行解，减小第二阶段算法的搜索空间。其主要流程如下：

```
1.  初始化
2.  for 遍历搜索次数 nc=1 to NC
3.    for 探路蚁 s=1 to S
4.      for 工位 j=1 to m-1
5.        基于操作分配策略和 3 种操作选择策略分配操作至工位 j
6.      end for
7.      将剩余操作分配至工位 m
8.    end for
9.    if 所得最好解优于当前最好解
10.     更新当前最好解以及工位时间的上界和下界
11.       if 当前最好节拍等于理想节拍
12.       输出最好解及其节拍，双阶段蚁群算法结束
13.       end if
14.     else
15.     第一阶段算法结束.
16.   end if
17. end for
18. 第一阶段算法结束，转入第二阶段算法.
```

第二阶段算法利用较多的搜索蚁在含有最优解的不断减小的可行解空间中依次对各个工位搜索其不同精英负载，然后由精英蚁根据精英复制策略将其组合成问题的可行解。其主要流程如下：

```
1.  初始化
2.  for 遍历搜索次数 nc1=1 to NC1 do
3.    for 工位 j=1 to m-1 do
4.      for 当前工位搜索次数 mc=1 to MC do
5.        for 搜索蚁 k1=1 to s1 do
6.        基于操作分配策略和操作选择策略分配操作至工位 j
7.        if 该工位负载为未记录精英负载且已有精英负载个数少于
          精英蚁的个数，do
8.          保存该精英负载，更新所记录精英负载的个数
9.        else
10.         跳出当前循环
11.         end if
12.       end for
13.       if 精英负载的个数少于精英蚁的个数，do
14.         实施信息素更新策略
15.       else
16.         跳出当前循环
17.       end if
18.     end for
19.     if 精英负载的个数少于精英蚁的个数 do
20.       实施精英复制策略
21.     end if
22.   end for
23.   各搜索蚁将剩余操作分配至其最后工位，计算最好解及其节拍
24.   if 满足迭代压缩机制的实施条件，do
25.     更新当前最好解以及工位时间的上界和下界
26.     if 当前最好节拍等于理想节拍
27.       输出最好解及其节拍，总算法结束
28.     end if
29.   else
30.     输出当前最好解及其节拍，算法结束
31.   end if
32. end for
33. 输出当前最好解及其节拍，算法结束
```

3.2 操作选择策略

算法为工位 $j(j=1,2,\cdots,m-1)$ 选择操作时，首先生成候选操作集合 CA，然后基于候选操作的位阶权重[16] ξ_i 和信息素 $\tau_i(i \in CA)$ 选择操作。其中候选操作集合为：

$$CA = CA\_f \cup CA\_b \tag{9}$$

其中，$CA\_f = \{i \mid \sum_{i_1 \in I} P_{i,i_1} = \sum_{i_2 \in V} P_{i,i_2}, i \in I\}$ 和 $CA\_b = \{i \mid \sum_{i_1 \in I} P_{i_1,i} = \sum_{i_2 \in V} P_{i_2,i}, i \in I\}$ 分别为入口和出口候选操作集合。候选操作 i 的位阶权重为：

$$\xi_i = \begin{cases} t_i + \sum_{i' \in UV} Q_{i',i} \cdot t_{i'}, & \text{if } i \in CA\_f \\ t_i + \sum_{i' \in UV} Q_{i,i'} \cdot t_{i'}, & \text{if } i \in CA\_b \end{cases} \tag{10}$$

候选操作 i 的选择概率为：

$$p_i = \frac{\xi_i^{\alpha} \tau_i^{\beta}}{\sum_{i \in CA} \xi_i^{\alpha} \tau_i^{\beta}}, i \in CA \tag{11}$$

其中，τ_i 为操作 i 的信息素，α 和 β 分别为位阶权重和信息素的指数权重。

第一阶段算法中信息素 τ_i 均恒取为 1，且分别利用贪心、随机和伪随机 3 种策略选择操作。贪心策略选择位阶权重最大的候选操作，随机策略随机选择操作，伪随机策略根据随机

数 $r(0\leqslant r<1)$的取值确定选择策略。当 $r\geqslant 0.5$ 时采用贪心策略，否则采用随机策略。

第二阶段算法利用 3.4 节的方法更新信息素。选择操作时，如果已分配操作已作为精英负载中的一部分被保存，则该精英负载中其他操作的选择概率在式(11)的基础上再减半，以减小得到此精英负载的概率。

3.3　迭代压缩机制

算法采用文献[15]的迭代压缩机制更新各工位时间的变化区间$[C_{\min},C_{\max}]$。对工位时间的上界 $C_{\max}$，算法在初次遍历搜索时按有别于文献[14]的无参数方法确定其初始值。具体如下：

Step1　蚂蚁为工位 1($j=1$)分配操作时，分配工位时间最接近理想节拍$\bar{C}$的候选操作组合至工位，计算工位时间 ST_j，令 $j=2$。

Step2　蚂蚁为工位 j 分配操作时，令 $C_{\max}=\max\{\max_{j'=1,\cdots,j-1} ST_{j'},\bar{C}\}$，$C_{\min}=\min\{2\bar{C}-C_{\max},\bar{C}-1\}$，然后根据文献[2]的操作分配策略分配操作至工位 j，并计算工位时间 ST_j，令$j=j+1$。

Step3　重复 Step2，直至 $j=m$ 时搜索结束，得到工位时间上界 $C_{\max}$的初始值。

3.4　信息素的更新策略

利用搜索蚁对各工位进行搜索时，为了得到多个不同精英负载，算法采用了信息素减少更新策略，以增强蚂蚁的全局探索能力。

对于工位 $j(j=1,\cdots,m-1)$，记 s_1 只搜索蚁在第 mc 次迭代搜索所得的负载中操作 $i(i\in I)$的频数为 H_i。操作 i 被分配至当前工位的次数越多，即 H_i 越大，所减少的信息素应越多；对同一项操作，随着 H_i 的变大，其信息素减少的幅度应变小，故定义操作 i 与当前工位间的信息素减少量为以下等比函数，即：

$$q+q^2+\cdots+q^{H_i}=\frac{q(1-q^{H_i})}{1-q}$$

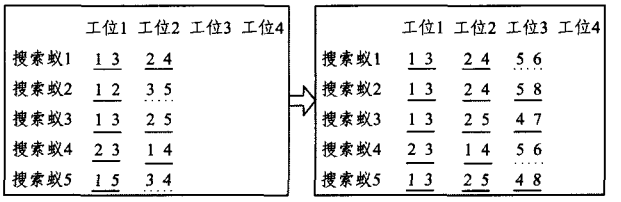
其中，$q(0<q<1)$为操作 i 随着搜索蚁所分配个数的增加其信息素减少的比例，在此比例下，信息素的最大减少量为 $\frac{q}{1-q}$。于是得操作 i 的信息素更新策略如下：

$$\tau_i\leftarrow[1-\frac{q(1-q^{H_i})}{1-q}]\tau_i \tag{12}$$

3.5　精英复制策略

记精英蚁的个数为 s_2。对工位 j 的搜索结束后，如果所得不同精英负载的个数也为 s_2，则搜索蚁基于这些精英负载直接对工位 $j+1$ 进行搜索；否则，精英蚁将所得的精英负载(假设为 $s_3(s_3<s_2)$个)及其前序工位的负载称为精英负载组合，实施均匀随机复制策略，即用这 s_3 个精英负载组合随机、均匀替换 s_2-s_3 个非精英负载组合，再利用搜索蚁对工位 $j+1$实施搜索，该策略称为精英复制策略。

示例：假设问题给定 4 个工位，算法利用 5 只搜索蚁进行搜索，则实施精英复制策略的主要思想如图 3 所示。图 3 中，搜索蚁 k 在工位 j 处的线为实线表示该工位的负载为精英负载，虚线表示为非精英负载，线上的数字为分配至此工位的操作序号，工位下面为空表示该工位还没有分配操作。



(a)　　(b)

图 3　精英复制策略示意图

在图 3(a)中，各个搜索蚁都在工位 1 得到精英负载，但在工位 2 中，搜索蚁 2 和搜索蚁 5 得到的负载为非精英负载。精英复制策略在搜索蚁 1、搜索蚁 3、搜索蚁 4 的 3 个精英负载组合中随机复制两个用于替换搜索蚁 2 和搜索蚁 5 的原负载组合，见图 3(b)(图中搜索蚁 2 和搜索蚁 5 分别采用搜索蚁 1 和搜索蚁 3 的负载组合)。所得的 5 个精英负载组合由精英蚁传递给对下一工位进行搜索的搜索蚁。

4　实验算例

为了检验算法所提选择策略、第一阶段算法以及整体算法的有效性，对 18 个标杆算例进行了求解。所有实验均在一台配置为英特尔 i5-337U @1.80GHz 双核处理器、4GB 内存的笔记本电脑上基于 MATLAB 软件进行。

第一阶段算法含有 5 个参数，分别为探路蚁的个数 S、最大搜索次数 NC、信息素初值 τ_0 以及位阶权重和信息素在选择概率中的权重指数 α 和 β。由于本阶段算法主要用于快速缩小问题的搜索空间，因此取 3 种选择策略的探路蚁个数依次为 1,5,5。在搜索过程中，当所有探路蚁都无法满足式(8)的解时，本阶段算法结束。此设定一般可使得算法在到达最大搜索次数前终止，故可取最大搜索次数为一个较大值，实验中被设定为 100。为加快搜索速度，本阶段算法都取各信息素值，且两个权重指数 α 和 β 都为 1。

第二阶段算法含有 8 个参数，依次为全局最大搜索次数 NC_1、局部最大搜索次数 MC、搜索蚁的个数 s_1、精英蚁的个数 s_2、信息素初值 τ_0、位阶权重和信息素的权重指数 α 和 β 以及信息素的减少比例 q。基于相同理由，取 NC_1 为 100。对各个工位进行搜索时，所得精英负载的个数与 $MC\times s_1$ 的取值相关，而精英蚁的个数与搜索速度和精度密切相关。大量参数实验表明，当问题的操作个数小于 30 时，s_1 取为操作的个数；否则 s_1 取 30，而 MC 取 5，$s_2=s_1$ 时，可保证算法以较快的速度求得问题精度较高的解。考虑到信息素初值对算法的影响较小，而位阶权重和信息素在选择概率中的地位相当，对这 3 个参数分别取值为理想节拍$\bar{C}$，1 和 1。依据参数实验的结果，将参数 q 取为 0.25。

4.1　选择策略测试实验

第一阶段算法引入了 3 种选择策略，分别应用这 3 种策

略对表 1 中的 33 个实例求解 100 次，计算其节拍的平均值并进行比较，以检验各策略的性能，并从中选择最优。

统计所得的 33 个平均值可得，贪心策略占优的有 15 个，占 45.5%；伪随机策略和随机策略占优的分别比较 9 个和 7 个，分别占 27.3%和 21.2%。显然贪心策略比较占优，但不占绝对优势。因为 3 种策略的计算时间都很短，且都不占绝对优势，所以在第一阶段算法中可同时采用 3 种策略，再从中择优，以提高算法求解不同问题的求解精度。

4.2 第一阶段算法的性能测试实验

为了检验第一阶段算法在双阶段蚁群算法中的性能，将第一阶段算法运行 100 次，双阶段蚁群算法运行 10 次，并比较其平均结果，如表 1 所列。

表 1 第一阶段算法和双阶段蚁群算法的求解结果比较

算例	m	理想节拍	第一阶段算法			双阶段算法				
			平均节拍	均方差	平均 CPU/s	平均节拍	均方差	平均 CPU/s	最好节拍	相对误差/%
Bowmax-8	3	26	26	0	0.044	26	0	0.213	26	0
Jackson-11	3	16	16	0	0.031	16	0	0.058	16	0
Mitchell-21	5	21	21	0	0.106	21	0	0.303	21	0
Rosenberg-25	5	25	25	0	0.109	25	0	0.274	25	0
	7	18	18	0	0.100	18	0	0.235	18	0
Heskiaoff-28	7	147	147	0	0.307	147	0	0.669	147	0
	9	114	114	0	0.412	114	0	0.933	114	0
Buxey-29	6	54	54.65	0.479	0.221	54	0	18.089	54	0
	8	41	41	0	0.195	41	0	0.463	41	0
Sawyer-30	6	54	54.54	0.500	0.239	54	0	34.520	54	0
	8	41	41	0	0.220	41	0	0.487	41	0
Lutz1-32	6	2366	2389	10.207	0.399	2366.4	1.450	99.348	2366	0
	9	1592	1610	8.270	0.392	1598.4	3.980	81.397	1596	0.25
Gunther-35	6	81	81	0	0.271	81	0	0.751	81	0
	8	61	61.40	0.492	0.452	61	0	18.281	61	0
Kilbridge&-Wester-45	7	79	79.30	0.460	0.449	79	0	27.302	79	0
	9	62	62	0	0.414	62	0	1.013	62	0
Hahm-53	8	1754	1796	6.433	0.722	1783.6	9.070	74.418	1775	1.20
	9	1559	1746	45.265	0.497	1759.2	33.322	14.895	1694	8.66
Warnecke-58	13	120	121.17	0.493	0.821	120	0	119.350	120	0
	17	92	93.88	0.433	1.135	92	0	170.868	92	0
Tonge-70	10	351	352.88	0.455	1.669	351	0	357.627	351	0
	16	220	222.88	0.551	2.182	220.6	0.516	420.109	220	0
Lutz2-89	13	38	38	0	0.645	38	0	1.483	38	0
	21	24	24	0	0.938	24	0	2.053	24	0
Mukherjee-94	14	301	302.74	0.440	3.510	301	0	551.708	301	0
	24	176	177.96	0.196	5.146	178	0	11.609	178	1.14
Arcus2-111	9	16711	16715	0.641	7.294	16711.7	0.816	4421.288	16711	0
	25	6016	6069.40	8.038	67.974	6064.3	5.907	355.674	6053	0.62
Bartholdi-148	7	805	805.41	0.494	9.151	805	0	465.846	805	0
	14	403	403.09	0.287	17.870	403	0	16.687	403	0
Scholl-297	21	3317	3327.90	1.495	100.520	3318.6	2.119	9480.200	3317	0
	50	1394	1410.40	1.454	367.390	1398.2	1.398	10535.800	1397	0.22

在表 1 中，第 1—3 列分别给出了各实例的来源、工位个数以及理想节拍，第 4—6 列和第 7—9 列分别为两算法求解所得的平均节拍、节拍的均方差和平均计算时间(s)。

由表 1 可知，对于 33 个实例，第一阶段算法可求得 13 个理想节拍，占 39.4%，所有平均节拍与理想节拍相对误差的平均值为 0.87%。在 100 次运算所得节拍的 33 个均方差中，低于 1 的有 26 个，占 78.8%。在计算时间方面，除了求得理想节拍的 13 个实例外，其他实例的平均计算时间为29.40s，仅占双阶段蚁群算法(1363.75s)的 2.16%，其中最小比例仅为 0.16%，而且这一比例随着规模的增加有进一步减小的趋势，这说明第一阶段算法的计算时间占总计算时间的比例很小。上述统计结果表明第一阶段算法能够快速、稳定地求得问题的较高精度的可行解，但其进一步优化的难度较大。

4.3 算法有效性测试实验

在双阶段蚁群算法中，第一阶段算法需要 S 只探路蚁遍历搜索 NC 次，每次每只蚂蚁需将 n 项操作分配至 m 个工位，故其计算复杂度为 $O(S \cdot NC \cdot m \cdot n)$；第二阶段算法需遍历搜索 NC_1 次，每次将 n 项操作分配至 m 个工位，而每个工位需安排 s_1 只搜索蚁搜索 MC 次，故其计算复杂度为 $O(NC_1 \cdot m \cdot n \cdot MC \cdot s_1)$。由于第一阶段算法所需的探路蚁很少，可忽略不计，于是所提算法的计算复杂度近似为 $O(NC_1 \cdot m \cdot n \cdot MC \cdot s_1)$。在原始蚁群算法[13]中，设其所需蚂蚁数为 s_3，最大遍历搜索次数仍为 NC_1，则其计算复杂度为 $O(NC_1 \cdot m \cdot n \cdot s_3)$，显然当 $s_3 = MC \cdot s_1$ 时，两算法的计算复杂度相当。然而，在双阶段蚁群算法中，第一阶段算法可较大幅度地缩小算法的搜索空间，第二阶段算法在此搜索空间中专注于局部搜索，故其计算精度较高；而原始蚁群算法则不具备此优势。

为了检验双阶段蚁群算法的有效性，对表 2 中 4 个算例的 7 个实例分别应用该算法和文献[13]中的原始蚁群算法求

解 10 次，求解所得的平均节拍、平均 CPU 和最好节拍见表 2 的第 5—10 列。表中第 4 列为文献[13]所提蚁群算法求解直线型第 2 类 ALBP(ALBP-2)的最好节拍，第 11 列为两算法求解最好节拍的相对误差。

由第 11 列的结果可知，对于 UALBP-2，双阶段蚁群算法的精度明显高于原始蚁群算法，前者在计算时间方面也较为占优。由第 4 和第 7 列的结果可知，对同一个实例，UALBP-2 的结果优于 ALBP-2，即 U 型装配具有更高的平衡率。

表 2　原始蚁群算法和双阶段蚁群算法的结果比较

算例	m	理想节拍	文献最好节拍	原始蚁群算法			双阶段算法			
				平均节拍	平均 CPU/s	最好节拍	平均节拍	平均 CPU/s	最好节拍	相对误差/%
Sawyer-30	8	41	41	41.8	91.20	41	41.0	0.72	41	2.38
	5	65	65	65.6	86.46	65	65.0	0.56	65	1.52
Kilbridge & Wester-45	10	56	57	57.0	145.12	57	56.0	1.42	56	1.75
	6	92	92	92.5	194.89	92	92.0	29.50	92	1.08
Tonge-70	11	320	323	322.0	554.30	321	320.0	184.42	320	0.31
	8	439	444	440.5	4111.81	440	439.0	202.31	439	0.23
Arcus2-111	9	16723	16725	16721.7	519.72	16716	16711.7	4421.29	16711	0.03

为了进一步验证双阶段蚁群算法求解 UALBP-2 的有效性，对表 1 中的 33 个实例进行了求解，求解 10 次所得的最好节拍见表 2 第 10 列。将所得最好节拍与理想节拍进行比较，其相对误差见表 2 第 11 列。

由表 1 的第 7 列可知，算法所得的平均节拍中有 23 个为理想节拍，占总数的 69.7%；而由第 10 列可知，算法所得的最好节拍中有 27 个为理想节拍，占总数的比例上升至 81.8%，较第一阶段算法提高了 42.4%。在其他 6 个实例中，虽然其最好节拍与理想节拍的最大相对误差为 8.66%，但其最大相对误差的平均值仅为 2.01%，且有 32 个低于 1.2%，这表明第二阶段算法有较强的跳出局部最优的能力，从而验证了其有效性。此外，由第 8 列可知，除了实例 Hahm-53 外，求解其他实例所得 10 个节拍的均方差都较小，这说明算法有较好的稳定性。

由表 1 的第 9 列的计算时间容易看出，算法求解小规模问题的时间很短，但随着问题规模的增大，计算时间明显呈现增长趋势，特别是对最大算例 Scholl-297 的求解时间超过两个小时。实验中发现，对于大规模实例 Scholl-297，若减少搜索蚁的个数，虽然平均节拍稍有增加，但是计算时间会大幅度减少。因此对大规模实例进行求解时，如果求解时间充足，可取搜索蚁个数为 30；否则可减少此个数，以缩短计算时间。

结束语　本文针对装配平衡调度中的 UALBP-2 提出了一种双阶段蚁群算法。算法中，少量探路蚁根据操作选择策略、操作分配策略和迭代压缩机制得到了问题的较优解，且对工位时间的变化范围进行定界，快速缩小了算法的搜索空间；然后搜索蚁基于新定义的信息素减少更新策略搜索各个工位中的精英负载，由精英蚁利用精英复制策略将所得的各个工位精英复制组合得到问题的解。对大量标杆算例的求解结果表明，所提算法可在较短时间内稳定地求得问题的高精度解。此外，大量的测试实验验证了第一阶段算法全局搜索和第二阶段算法局部搜索的有效性。

虽然算法的求解精度较高，但其计算复杂度也较高，对大规模问题的求解时间较长。在后续研究中，将结合 UALBP 的特性对双阶段蚁群算法进行进一步改进，在保证算法精度的前提下加快算法的搜索速度。此外，还可以将所提算法应用于其他装配线平衡问题，如双边装配线平衡问题和混合装配线平衡问题。

参 考 文 献

[1] HADI G, KÜRSAT A, CEVRIYE G, et al. A shortest route formulation of simple U-type assembly line balancing problem[J]. Applied Mathematical Modeling, 2005, 29(4): 373-380.

[2] LI M, TANG Q H, ZHENG Q X, et al. Improved rules combination algorithm of the type 2 assembly line balancing problem [J]. Computer Integrated Manufacturing Systems, 2015, 21(1): 88-93. (in Chinese)
李明，唐秋华，郑巧仙，等. 第 2 类装配平衡问题的改进多规则组合优化算法[J]. 计算机集成制造系统，2015，21(1)：88-93.

[3] MILTENBURG G J, WIJNGAARD J. The U-line balancing problem[J]. Management Science, 1994, 40(10): 1378-1388.

[4] SHWETANK A, RAJEEV J, MISHRA P K, et al. A heuristic approach for U-shaped assembly line balancing to improve labor productivity[J]. Computer & Industrial Engineering, 2013, 64(4): 895-901.

[5] TOKSARI M D, İSLEYEN S K, GÜNER E, et al. Simple and U-type assembly line balancing problems with a learning effect [J]. Applied Mathematical Modeling, 2008, 32(12): 2954-2961.

[6] ÖZCAN U, TOKLU B. A new hybrid improvement heuristic approach to simple straight and U-type assembly line balancing problems[J]. Journal of Intelligent Manufacturing, 2009, 20(1): 123-136.

[7] KIM Y K, KIM Y J, KIM Y. An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines[J]. European Journal of Operation Research, 2006, 168(3): 838-852.

[8] NOURMOHAMMADI A, ZANDIEH M, TAVAKKOLI-MOGHADDAM R. An imperialist competitive algorithm for multi-objective U-type assembly line design[J]. Journal of Computational Science, 2013, 4(5): 393-400.

[9] WEI X M. Application of mind evolution based ant colony algorithm in typical production scheduling[J]. Computer Science, 2013, 40(7): 236-238, 257. (in Chinese)

(下转第 225 页)

运行效果，从而极大地影响了算法的实用性。本文将均匀设计方法引入到差分进化算法的参数设定中，试验点均匀分布的理念有效地避免了算法最终陷入局部最优解的局面，这是一种科学、有效的算法参数设定手段。本文针对 3 种不同类型的标准测试函数，将基本差分进化算法的参数设定问题描述为多因素多水平的均匀试验设计，利用均匀设计表来安排实验，经过多次运行寻找到了适合 3 种函数的最优参数组合 $F=0.2, CR=0.9, NP=180$ 以及 $F=0.2, CR=0.9, NP=300$。最优参数组合下的平均全局最优解为 4.3215，非常接近理论最优解，说明利用均匀设计的方法对算法参数进行设定具有可行性、有效性，达到了令人满意的效果，这为算法应用于目标函数为单峰函数、多峰函数、病态函数这 3 种不同类型函数的最优化问题提供了参考。

参 考 文 献

[1] BIN G, ZHU J H, LANG W C. Research on Improved Differential Evolution Algorithm based on Hybrid Multi-strategy and its application[J]. International Journal of Hybrid Information Technology, 2015, 8(4): 147-156.

[2] FENG Q X, LIU S Y, TANG G Q, et al. Orthogonal Differential Evolution Algorithm for Solving System of Equations[J]. Computer Science, 2012, 39(5): 187-190. (in Chinese)
封全喜，刘三阳，唐国强，等. 求解方程组的正交差分进化算法[J]. 计算机科学，2012，39(5)：187-190.

[3] LI R P, FENG D, OUYANG H B, et al. An Improved Differential Evolution Algorithm Reliability Problems[J]. Journal of Northeastern University (Natural Science), 2012, 33(2): 182-186. (in Chinese)
李若平，冯达，欧阳海滨，等. 改进差分进化算法在系统可靠性问题中的应用[J]. 东北大学学报(自然科学版)，2012，33(2)：182-186.

[4] LI Z W, ZHOU X G, ZHANG G J. Dynamic Adaptive Differential Evolution Algorithm [J]. Computer Science, 2015, 42(6A): 52-57. (in Chinese)
李章维，周晓根，张贵军. 一种动态自适应差分进化算法[J]. 计算机科学，2015，42(6A)：52-57.

[5] LEE C H, KUO C T, CHANG H H. Performance enhancement of the differential evolution algorithm using local search and a self-adaptive scaling factor[J]. International Journal of Innovative Computing, Information and Control, 2012, 8(4): 2665-2679.

[6] ZHONG J H, ZHANG J. Adaptive Multi-objective Differential Evolution with Stochastic Coding Strategy [C]//GECCO. 2011: 665-672.

[7] 方开泰，马长兴. 正交与均匀试验设计[M]. 北京：科学出版社，2001.

[8] FANG K T. Uniform Design [J]. Tactical Missile Technology, 1994, 3(2): 56-69. (in Chinese)
方开泰. 均匀设计[J]. 战术导弹技术，1994，3(2)：56-69.

[9] HUANG Y Q, LIANG C Y, ZHANG X D. Parameter Establishment of an Ant System Based on Uniform Design [J]. Control and Decision, 2006, 21(1): 93-96. (in Chinese)
黄永青，梁昌勇，张祥德. 基于均匀设计的蚁群算法参数设定[J]. 控制与决策，2006，21(1)：93-96.

[10] HE T K, WANG F L, ZHANG C M. Parameter Establishment of Genetic Algorithm Based on Uniform Design[J]. Journal of Northeastern University (Natural Science), 2003, 24(5): 419-411. (in Chinese)
何太阔，王福利，张春梅. 基于均匀设计的遗传算法参数设定[J]. 东北大学报(自然科学版)，2003，24(5)：419-411.

[11] LOCATELLI M, MAISCHBERGER M, SCHOEN F. Differential Evolution Algorithm Based on Local Search [J]. Computers & Operations Search, 2014, 43: 169-180.

[12] STORN R, PRICE K. Differential Evolution. A simple and Efficient Heuristic for Global Optimization over Continuous Spaces [J]. Journal of Global Optimization, 1997, 11(4): 341-359.

(上接第 211 页)

魏先民. 基于思维进化的蚁群算法在典型生产调度中的应用[J]. 计算机科学，2013，40(7)：236-238，257.

[10] ZHENG Q X, LI M, LI Y X, et al. An Improved Colony Optimization for Two-Sided Assembly Line Balancing Problem[J]. Acta Electronica Sinica, 2014, 42(5): 841-845. (in Chinese)
郑巧仙，李明，李元香，等. 求解双边装配平衡问题的改进蚁群算法[J]. 电子学报，2014，42(5)：841-845.

[11] IHSAN S, ERDAL E, ARDA A. Ant colony optimization for the single model U-type assembly line balancing problem[J]. International Journal of Production Economics, 2009, 120(2): 287-300.

[12] ADIL B, TÜRKAY D. Simple and U-type assembly line balancing by using an ant colony based algorithm[J]. Mathematical and Computational Applications, 2009, 14(1): 1-12.

[13] ZHENG Q X, LI Y X, LI M, et al. Ant colony optimization for type-II. assembly line balancing problem[J]. Computer Integrated Manufacturing Systems, 2012, 18(5): 999-1005. (in Chinese)
郑巧仙，李元香，李明，等. 面向第 II 类装配平衡问题的蚁群算法[J]. 计算机集成制造系统，2012，18(5)：999-1005.

[14] ZHANG Z Q, CHENG W M, ZHONG B, et al. Improved colony optimization for assemblyline balancing problem[J]. Computer Integrated Manufacturing Systems, 2007, 13(8): 1632-1637. (in Chinese)
张则强，程文明，钟斌，等. 求解装配平衡问题的一种改进蚁群算法[J]. 计算机集成制造系统，2007，13(8)：1632-1637.

[15] ZHENG Q, LI M, LI Y, et al. Station ant colony optimization for the type 2 assembly line balancing problem[J]. International Journal of Advanced Manufacturing Technology, 2013, 66(9): 1859-1870.

[16] HELGESON W B, BIRNIE D P. Assembly line balancing using the ranked positional weight technique[J]. The Journal of Industrial Engineering, 1961, 12(6): 394-398.