文章编号:1671-4598(2020)11-0216-06 DOI:10.16526/j.cnki.11—4762/tp.2020.11.044 中图分类号:TU756 文献标识码:A

基于改进迭代贪婪算法的预制构件调度研究

陈竑翰，熊福力，曹劲松，李 志
（西安建筑科技大学 信息与控制工程学院，西安 710055）

摘要：迭代贪婪算法是一种具有较强局部搜索能力的元启发式算法，但由于传统迭代贪婪算法搜索范围过大，搜索效率有限，为了进一步提升传统迭代贪婪算法的搜索能力，考虑到阈值接受算法具有能缩小搜索范围的特点，提出了一种改进的迭代贪婪算法解决流水车间预制生产的订单接受与调度问题；该改进算法是在破坏原调度序列后加入一种基于构造启发式规则的重建策略，并结合阈值接受算法的自适应接受准则用以跳出局部最优；经大量仿真实验结果显示，与传统迭代贪婪算法、禁忌搜索算法以及遗传算法对比，改进的迭代贪婪算法具有更好的求解质量和鲁棒性。

关键词：迭代贪婪算法；阈值接受算法；流水车间；订单接受与调度

Research on Scheduling of Prefabricated Components Based on Modified Iterative Greedy Algorithm

Chen Honghan, Xiong Fuli, Cao Jinsong, Li Zhi
(College of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710055, China)

Abstract: Iterative greedy (IG) algorithm is a meta-heuristic algorithm with strong local search ability, but due to the excessive search range of traditional iterative greedy algorithm and limited search efficiency, in order to further improve the search ability of traditional iterative greedy algorithm, considering the threshold acceptance algorithm has the characteristics of narrowing the search range, an improved iterative greedy algorithm is proposed to solve the problem of order acceptance and scheduling for prefabricated production in flow shop. The improved algorithm is to add a reconstruction strategy based on constructing heuristic rules after destroying the original scheduling sequence, and combined with the adaptive acceptance criterion of the threshold acceptance algorithm to jump out of the local optimum. A large number of simulation experiments show that the improved iterative greedy algorithm has better solution quality and robustness compared with the traditional iterative greedy algorithm, tabu search (TS) algorithm and genetic algorithm (GA).

Keywords: iterative greedy algorithm; threshold acceptance algorithm; flow shop; order acceptance and scheduling

0 引言

在混凝土预制构件生产的系统中，利润最大化是企业的首要目标。决策者需要根据两个重要的约束条件，即企业生产能力约束和交货期约束，在两个约束条件下会迫使企业来决定是否接受或拒绝哪些候选订单以及如何对订单进行安排。这种问题被称为订单接受与调度问题（order acceptance and scheduling，OAS）[1]。

有许多精确的方法被用来解决小规模的OAS问题，例如分枝定界算法[2]和动态规划算法[3]。目前已经证实流水车间的OAS问题属于NP-hard问题[4]，所以当问题规模过大时为了在合理的计算时间内获得近似最优解，一些学者提出了有效的启发式算法[5]和元启发式算法[6-7]进行求解。郑凡等[8]针对订单接收的流水车间调度问题，提出了一种并行变邻域搜索算法，采用双串表示方法、新型的邻域结构和并行搜索机制解决了该问题。Wang等人[9]考虑了模具制造、预制构件储存和运输的影响，以最小化延迟和早期惩罚的总成本，并通过GA解决了这一调度问题。Prata[10]考虑了铸梁模具有效生产能力的约束条件，并采用整数线性规划方法来最小化订单生产损失。Ko等人[11]讨论了相邻工序之间带缓冲区容量的预制调度问题，并通过GA进行求解。

目前在预制构件的生产中还未对OAS问题进行过研究。因此，本文以最大化总净收益为目标，提出了一个预制流水车间的OAS模型。针对此模型本文设计了一种改进的IG算法。在改进的IG算法中，在重建过程之前加入了一种构造启发式的规则，并且结合阈值接受算法设计了一种自适应接受准则。最后将改进的IG算法与禁忌搜索算法、遗传算法以及经典IG算法进行了对比分析。结果表明改进的IG算法能获得更好的效果。

收稿日期:2020-04-16； **修回日期**:2020-05-12。
基金项目:国家自然科学基金项目(61473216)；陕西省教育厅科学研究计划项目(17JK0459)；西安建筑科技大学基础研究项目(ZR18049)；陕西省自然科学面上项目(2020JM-489)。
作者简介:陈竑翰(1994-)，男，重庆人，硕士研究生，主要从事智能制造、优化算法、人工智能方向的研究。
通讯作者:熊福力(1974-)，男，黑龙江肇东人，硕士生导师，副教授，主要从事人工智能与系统优化、生产计划与调度优化、智能建筑方向的研究。

1　问题描述

预制供应链环境下的预制过程主要由九个工序组成：(S1) 模具制造；(S2) 模具组装；(S3) 钢筋预埋；(S4) 混凝土浇筑；(S5) 蒸汽养护；(S6) 脱模；(S7) 整理精修；(S8) 存储；(S9) 运输。根据生产的工艺特征分为并行工序和串行工序，其中 S5、S8 和 S9 为并行工序，可同时处理多个工件，其余则为串行工序。在预制生产环境中，J 个独立订单在工厂加工生产，每个订单 j 都有一个对应的交货日期 $\overline{d}_j$ 和一个强制交货期 $\overline{\overline{d}}_j$。针对每个工序 s，订单 j 都有一个对应的处理时间 $p_{j,s}$，每个订单都必须按顺序分九个工序处理，在预制供应链环境下，通常会由于企业有限的生产资源和较紧迫的交货期使得订单延期交付，从而产生巨大的惩罚成本。

在生产过程中需满足以下条件：1) 相邻工序之间工件的安装时间和运输时间可忽略不计；2) 每道工序一次最多只能处理一个工件；3) 每个工件一次最多只能在一道工序上处理；4) 工件在工序上处理完成之前不能被其他工件抢占。

2　生产调度模型

若订单 j 被接受，且其完工时间 Cj 小于等于交货期 $\overline{d}_j$，则订单收益为 Q_j；若 Cj 大于交货期 $\overline{d}_j$，小于强制交货期 $\overline{\overline{d}}_j$，则产生拖期惩罚 w_j。当 Cj 大于等于 $\overline{\overline{d}}_j$ 时，Q_j 等于 0。预制构件流水车间 OAS 问题的目标函数为：

$$f = \max \sum_{j \in J} x_j (Q_j - w_j T_j) \tag{1}$$

订单 j 的拖期时间 T_j 和完工时间 C_j 可由以下约束计算。

$$T_j \leq (\overline{\overline{d}}_j - \overline{d}_j) x_j, \forall j \tag{2}$$

$$T_j \geq C_{[k]} - \overline{d}_j + \Omega(y_{j,[k]} - 1), \forall j,k \tag{3}$$

$$\sum_{k \in J} y_{j,[k]} = x_j, \forall j \tag{4}$$

$$\sum_{j \in J} y_{j,[k]} \leq 1, \forall k \tag{5}$$

$$C_{[k],s} = \max\{C_{[k-1],s}, C_{[k],s-1}\} + \sum_{j \in J} y_{j,[k]} p_{j,s}$$

$$\forall k, \quad \forall s \in \{1,2,3,4,6,7\} \tag{6}$$

$$C_{[k],s} = C_{[k],s-1} + \sum_{j \in J} y_{j,[k]} p_{j,s}, \forall k$$

$$\forall s \in \{5,8,9\} \tag{7}$$

$$x_j \in \{0, 1\} \forall j \tag{8}$$

$$y_{j,[k]} \in \{0, 1\} \forall j,k \tag{9}$$

其中：式 (2) 和 (3) 表示每个订单的拖期时间的约束，式中 Ω 是一个大数；式 (4) 确保将每个接受的订单分配到一个位置，式中 $[k]$ 表示订单序列的位置指标；式 (5) 表示每个位置只能分配一个订单；式 (6) 表示预制供应链环境下连续工序的完工时间，式中 $C_{[k]}$ 是指在第 k 个位置上订单的完工时间，$p_{j,s}$ 是指订单 j 在工序 s 上的处理时间；式 (7) 表示预制供应链环境下并行工序的完工时间。式 (8) 和式 (9) 表示变量 x_j 和 $y_{j,[k]}$ 均为二进制变量，若订单 j 被接受，x_j 取 1，否则为 0；若订单 j 分配到了订单序列第 k 个位置，$y_{j,[k]}$ 取 1，否则为 0。

3　改进迭代贪婪算法

迭代贪婪 (iterative greedy，IG) 算法最初是由 Ruiz[12] 提出的，是对贪婪搜索方法的一种扩展。经典 IG 算法主要由两个主要阶段构成，一个是可行解的破坏阶段，另一个是可行解的重建阶段。在破坏阶段中，从完整的可行解中删除一些元素。然后在重建阶段通过应用一些贪婪规则重新构成一个完整的可行解。最后对完整的可行解进行选择性接受。由于其在调度问题上良好的寻优能力，目前它已在许多领域得到了广泛的应用。本文在结合预制构件订单接受与调度问题的基础上提出了一种结合构造式启发规则的带阈值接受的迭代贪婪算法 (iterative greedy with threshold acceptance，IGTA)，以下各节详细介绍了 IGTA 算法的主要过程。

3.1　编码方式

本文将预制生产 OAS 问题的解采用实数编码方式对个体进行编码，对于每个个体表示为 $\pi = (\pi_1, \pi_2, \ldots, \pi_j)$，其中 π_j 表示的是序列中在第 j 个位置上的订单，每个位置上的数字在区间 $[1, J]$ 上随机产生且不重复。为保证可比性，本文中提出的 IGTA 算法以及仿真实验中的对比算法中的初始解均是随机产生的。

3.2　基于构造启发式的破坏－重建策略

在经典 IG 算法中的破坏－重建阶段中分为两个主要过程，首先在破坏过程中，从当前解中随机选择 g 个订单并将其移除，然后可以得到两组序列，一组是由剩余的订单组成的部分序列，另外一组是有移除的订单组成的部分序列，其中，移除订单数即破坏因子 g 通常在 (2，3，...，8) 中取值；其次在重建过程中，将移除订单组成的部分序列从左至右逐一插入所有的位置，并进行比较，最终将订单插入是目标值增加最多的位置，从而可以得到一个当前最优解。

从经典 IG 算法中可以看出，重建过程直接对剩余订单序列进行插入操作，并未充分利用剩余订单集合中交货期、处理时间以及最大净收益等信息，很可能导致求解质量下降。因此，为了在调度序列重建过程中有效改善目标值，本文结合预制构件订单接受与调度问题数据信息，提出了一种基于构造启发式规则的重建操作策略，如图 1 所示，首先利用公式 (10) 在剩余订单集合中对 ξ_j 进行递减排序，然后再从移除订单集合中逐个抽出订单，并插入到使得总净利润最小的剩余订单序列位置，直到移除订单集合中没有订单。

$$\xi_j = Q_j / (\overline{\overline{d}}_j + p_j) \tag{10}$$

其中：Q_j 表示订单 j 的最大净收益，$\overline{\overline{d}}_j$ 表示订单 j 的交货截止日期，p_j 表示订单 j 在所有工序上的处理时间之和。

3.3　局部邻域搜索

本文中的邻域搜索方法采用是一种插入式的局部邻域

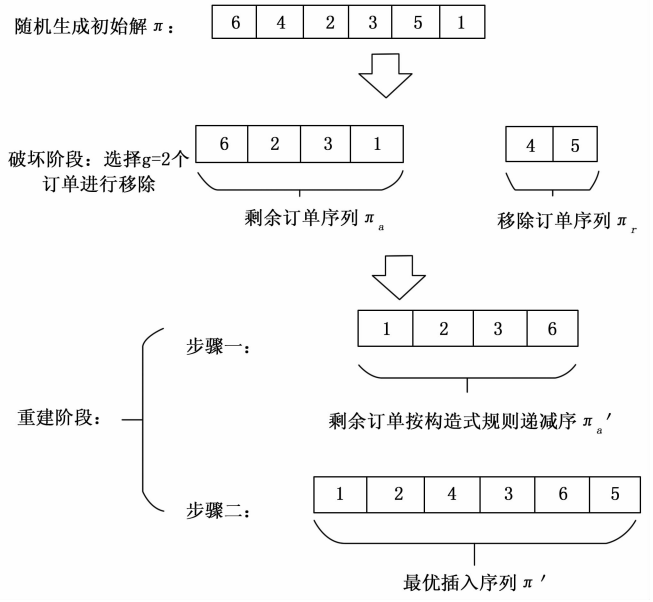

图 1 基于构造启发式规则的破坏－重建策略图

搜索方法。其基本思想是：每次从当前解中随机地选择一个订单，将订单从左至右逐一试插，最终将订单插入是目标值增加最多的位置。如果通过邻域搜索找到的新解优于当前解，则对当前解进行替换并继续搜索，否则就结束搜索。

3.4 阈值接受准则

经典 IG 算法中的接受准则是基于模拟退火算法中的按概率接受新解作为当前解。为了帮助 IG 算法能够拥有更好扰动性能，并能更加容易跳出局部最优，本文通过结合一种阈值接受算法，提出了一种新的阈值接受准则。阈值接受算法是对模拟退火算法的改进，它是使用阈值对整个求解过程进行控制。阈值接受准则能够使 IG 算法在一定范围内接受稍差的解，从而使算法跳出局部最优值，与模拟退火算法中的接受准则相比，阈值接受准则能进行更小范围内的搜索[13]。

在本文中，为了综合考虑订单规模和优化效率，初始阈值设置为 $T_0=n^2$，其中 n 表示订单数；阈值衰减系数 α 是随着迭代次数 Iter 不断进行变化的一个自适应值，由式（11）确定。

$$\alpha=\frac{T_0-1}{Iter\times T_0} \tag{11}$$

3.5 终止条件

本文中提出的 IGTA 算法为了综合考虑问题的规模，且保证算法充分收敛情况下，设定终止条件为 $10*n^2$ 毫秒的程序运行时间。其中，n 为待决策订单的数量。

IGTA 算法的伪代码如算法 1 所示。

算法 1：IGTA 算法伪代码

1：输入初始可行解 π，初始目标值 f（π）

2：设置算法参数，其中包括：移除订单数 g 以及初始化阈值 T_0；

3：设定初始解为当前最好解 $\pi^*\leftarrow\pi$

4：while 不满足结束条件 do

5：令 π’ ← π，flag＝1，Iter＝1，利用式（11）计算 α；

6：for i＝1 to g do

7：从 π’ 中随机选择一个订单移除，并将其置入移除订单集，组成剩余订单序列 π_a 和移除订单序列 π_r；

8：end for

9：for i＝1 to g do

10：通过公式（10）中的构造式规则对剩余订单序列 π_a 进行递减排序优化得到 π_a’；

11：从移除订单序列 π_r 中选择一个订单将其插入 π_a’ 中所有可能位置中能使目标增加最多的位置，得到最优插入序列 π’；

12：end for

13：while flag＝1 do

14：flag＝0

15：for i＝1 to n do

16：从 π’ 中不重复地随机移除一个订单并将其插入 π’ 所有剩余位置中的最优位置，得到邻域搜索解 π’’；

17：if f（π’’）＞f（π’）then

18：π’ ←π’’，flag＝1；

19：end if

20：end for

21：end while

22：if f（π’’）＞f（π）then

23：π ←π’’

24：if f（π）＞f（π＊）then

25：$\pi^*\leftarrow\pi$

26：end if

27：elseif（f（π）－f（π’’））＜α×T_0then

28：π←π’’

29：end if

30：end while

31：return π^*

4 实验结果及分析

本节根据所研究问题的特点设计了仿真实验，分析了本文所提出算法对不同问题规模求解的效果，并通过与禁忌搜索算法（tabu search，TS）、遗传算法（genetic algorithm，GA）以及经典 IG 算法进行比较，对算法的鲁棒性和求解质量给出分析结果。

4.1 测试实例

为了验证本文所提出的改进迭代贪婪算法对预制供应链环境下订单接受与调度问题的求解效果，本文首先生成了一系列测试实例。基于文献［14］提出的问题测试实例生成方法，文中考虑了实际问题的特点对问题特征参数进行调整，生成了 3 种规模的测试实例。例如，本文选择的

延迟因子 τ 与交货期范围因子 R 均为 0.3，0.5，0.9，针对每一组延迟因子与交货期范围因子的组合，共生成了 9 个问题测试实例，因此每种规模问题均生成共计 9×3＝27 个测试实例。其中，每个测试实例中各订单 i 的交货期 d_i 在区间 $[P_T(1-\tau-R/2), P_T(1-\tau+R/2)]$ 内均匀产生，订单 i 的交货截止日期 $\overline{d}_i=\overline{d}_i+RP_i$，其中 P_T 是所有订单的总处理时间，P_i 是订单 i 的总处理时间。同时，为了检验算法对于不同问题规模下的求解效果，待决策的订单数量 n 分别取 20，40 和 60，生成了小、中、大 3 种规模的问题实例。表 1 中展示了本文采用的 10 种订单类型在各道工序上的处理时间以及各类型订单能获得的最大净收益。

表 1　生产数据

类型	各阶段处理时间（小时）									净收益
编号	S1	S2	S3	S4	S5	S6	S7	S8	S9	Q
1	11.0	1.5	2.0	0.5	8.0	1.0	0.5	10.0	1.5	300.0
2	11.0	1.0	2.0	0.4	8.0	1.0	0.5	10.0	1.5	250.0
3	10.0	1.0	1.5	0.5	8.0	0.5	0.5	10.0	1.0	280.0
4	8.0	0.5	1.0	0.3	8.0	0.3	0.5	10.0	1.5	400.0
5	4.0	1.0	0.8	1.0	8.0	1.5	0.5	10.0	1.5	420.0
6	8.0	0.5	2.0	0.4	8.0	0.5	0.5	10.0	1.5	200.0
7	5.0	1.5	2.0	0.5	8.0	1.0	0.4	10.0	0.5	280.0
8	5.0	0.5	2.0	0.3	8.0	0.6	0.3	10.0	1.5	280.0
9	8.0	1.5	1.8	1.2	8.0	1.5	1.5	10.0	1.0	260.0
10	4.0	4.0	0.5	0.6	8.0	0.5	0.5	10.0	2.0	450.0

4.2　仿真结果与分析

所有实验均通过 Matlab 2017b 编程实现，并在计算机配置为 Microsoft Windows 10，处理器为 Intel Core i5－6300HQ CPU @ 2.3 GHz，8 GB RAM 的个人电脑上运行。本文针对订单 n 为 20，40 和 60 的 3 种小、中、大规模的问题测试实例在实验中使用了禁忌搜索算法，遗传算法以及经典 IG 算法和本文所提出的 IGTA 算法进行对比，它们将在每个问题实例下进行 30 次测试。总计运行 3 240（27×30×4）次。在本文中 GA 算法中的种群数量，交叉率和变异率分别选取为 $P_s=100$，$P_c=0.8$，$P_m=0.02$；TS 算法中的禁忌长度和邻域大小分别取 $(n(n-1)/2)^{1/2}$ 和 $2n$，其中 n 表示待决策订单的数量；经典 IG 算法中的参数分别取 $g=4$，$T=0.4$；本文中的 IGTA 算法中的阈值和破坏因子分别取 $T_0=n^2$，$g=4$。

具体来说，针对每个测试的问题实例，分别比较 4 种算法在不同实例上的求解效果，文中用最优目标均值（AVG）和最大值（MAX）来对算法的求解质量进行评估，以及使用标准差（STD）来评价算法的鲁棒性。此外对于某个问题规模下的测试实例 l，考虑到算法会重复运行 M 次，本文定义了一个平均相对百分偏差（average relative percentage deviation，ARPD）来对各个规模下不同算法的性能进行评估，计算公式如（12）：

$$ARPD_{alg}=\frac{1}{M\times L}\sum_{l=1}^{L}\sum_{m=1}^{M}\frac{TNR_{best}(l)-TNR_{alg}(l,m)}{TNR_{best}(l)}\times 100\% \tag{12}$$

其中：$TNR_{alg}(l,m)$ 表示对于给定算法 alg 在实例 l 下运行第 m 次所获得的目标值，$TNR_{best}(l)$ 表示在实例 l 下所有实验得到的最优目标值，L 表示同规模问题下的测试实例之和。

在问题规模 $n=20$ 情况下的仿真实验结果如表 2 所示。

由表 2 所示的实验结果比较分析可知：本文所提出的 IGTA 算法表现较好，在最优值的寻找方面均能找到不差于其余 3 种对比算法的解，在均值和标准差方面 IGTA 算法与经典 IG 算法求解效果相近。

在问题规模 $n=40$ 和 $n=60$ 情况下的仿真实验结果如表 3 和表 4 所示。由表 3 和表 4 中所列出的数据可以明显的看出随着订单规模的增加，改进的 IG 算法无论是在最优值的寻找还是整体均值的计算都能找到 4 种算法中最好的解。除此之外，从上表的 STD 对比中可以看出，本文提出的 IGTA 算法始终保持着比较稳定的状态，由此说明 IGTA 算法相比于其余 3 种算法具有更好的鲁棒性。

为了更加直观的展示 4 种算法在不同订单规模下的求

表 2　$n=20$ 时 4 种算法的性能比较

$n=20$	τ	0.3			0.5			0.9		
	R	0.3	0.5	0.9	0.3	0.5	0.9	0.3	0.5	0.9
IGTA	*MAX*	5 627.2	5 690.0	5 690.0	4 741.3	5 108.8	5 690.0	2 004.5	2 919.2	4 198.4
	AVG	5 627.2	5 690.0	5 690.0	4 714.4	5 091.9	5 690.0	2 004.5	2 889.9	4 190.7
	STD	0.0	0.0	0.0	28.5	6.0	0.0	0.0	25.4	12.0
IG	*MAX*	5 627.2	5 690.0	5 690.0	4 741.3	5 108.8	5 690.0	2 004.5	2 919.2	4 198.4
	AVG	5 627.2	5 690.0	5 690.0	4 718.2	5 080.8	5 690.0	2 004.5	2 844.2	4 193.2
	STD	0.0	0.0	0.0	38.2	20.5	0.0	0.0	55.1	10.0
TS	*MAX*	5 627.2	5 690.0	5 690.0	4 640.0	5 067.4	5 690.0	2 004.5	2 919.2	4 131.3
	AVG	5 487.7	5 398.2	5 690.0	4 554.7	4 774.0	5 639.1	2 004.5	2 918.7	4 066.5
	STD	102.1	174.3	0.0	93.5	130.7	56.1	0.0	1.2	45.8
GA	*MAX*	5 491.1	5 275.4	5 416.1	4 540.0	4 470.0	4 761.6	1 887.3	2 475.3	3 748.1
	AVG	5 389.5	5 107.3	5 214.1	4 236.6	4 271.2	4 563.3	1 849.5	2 347.9	3 557.3
	STD	84.4	108.9	125.3	140.0	143.5	123.1	27.6	124.3	135.9

表 3 $n=40$ 时 4 种算法的性能比较

$n=40$	τ	0.3			0.5			0.9		
	R	0.3	0.5	0.9	0.3	0.5	0.9	0.3	0.5	0.9
IGTA	*MAX*	12 520.0	12 520.0	12 520.0	11 037.4	12 270.0	12 520.0	5 720.0	7 693.2	10 081.8
	AVG	12 520.0	12 520.0	12 520.0	10 911.7	12 018.4	12 520.0	5 491.2	7 438.1	10 007.0
	STD	0.0	0.0	0.0	90.5	75.6	0.0	133.7	144.6	46.9
IG	*MAX*	12 520.0	12 520.0	12 520.0	11 001.6	12 270.0	12 520.0	5 660.0	7 643.5	9 821.3
	AVG	12 520.0	12 520.0	12 520.0	10 913.3	11 935.0	12 490.0	5 431.6	7 433.8	9 539.9
	STD	0.0	0.0	0.0	78.6	207.8	94.9	155.4	224.9	171.4
TS	*MAX*	12 520.0	12 520.0	12 035.1	10 450.0	10 850.0	11 290.0	5 540.0	7 422.3	8 291.5
	AVG	12 454.0	12 408.3	11 631.1	9 936.0	10 349.6	10 436.7	5 380.0	7 123.5	8 006.1
	STD	107.5	185.1	198.4	355.1	416.0	404.2	137.9	220.7	228.3
GA	*MAX*	12 520.0	12 260.0	11 185.9	9 700.0	9 860.0	9 937.2	4 641.8	5 903.3	7 970.0
	AVG	12 322.2	12 123.6	10 998.2	9 321.1	9 497.3	9 716.1	4 205.2	5 512.0	7 360.7
	STD	133.2	127.5	126.5	205.3	170.9	152.7	280.8	321.6	372.8

表 4 $n=60$ 时 4 种算法的性能比较

$n=60$	τ	0.3			0.5			0.9		
	R	0.3	0.5	0.9	0.3	0.5	0.9	0.3	0.5	0.9
IGTA	*MAX*	19 460.0	19 460.0	19 460.0	17 830.0	19 051.5	18 270.0	9 120.0	11 318.1	16 320.9
	AVG	19 460.0	19 460.0	19 460.0	17 646.7	18 572.9	18 158.1	8 756.3	10 798.4	16 092.6
	STD	0.0	0.0	0.0	159.3	193.0	74.7	244.3	232.6	146.6
IG	*MAX*	19 460.0	19 460.0	19460.0	17 778.6	18 950.0	18 387.1	8 928.1	11 167.3	16 140.0
	AVG	19 440.0	19 432.0	19 460.0	17 357.3	18 423.0	18 002.9	8 550.3	10 727.3	15 923.2
	STD	63.2	88.5	0.0	365.4	311.9	282.5	344.3	337.3	198.7
TS	*MAX*	19 460.0	19 460.0	18 576.5	16 420.0	14 650.0	14 330.0	7 515.6	10 646.0	14 080.0
	AVG	19 037.7	18 651.3	17 413.3	15 796.5	14 328.5	13 958.8	7 083.1	9 691.6	13 448.9
	STD	401.2	473.6	546.2	353.1	231.3	366.5	244.7	447.2	464.4
GA	*MAX*	19 460.0	18 570.5	17 415.1	15 600.0	14 389.4	13 670.0	7 040.0	8 690.0	12 700.0
	AVG	19 087.9	18 340.6	16 710.6	14 924.9	14 059.9	13 196.7	6 593.2	7 862.1	12 117.2
	STD	174.6	134.5	372.8	370.6	228.1	299.2	316.3	532.2	416.8

解效果对比，本文通过提出一种平均相对偏差的评价指标可以更加清楚的看到在不同问题规模下的 4 种算法的差异，4 种对比算法的总体 ARPD 对比柱状图如图 2 所示。

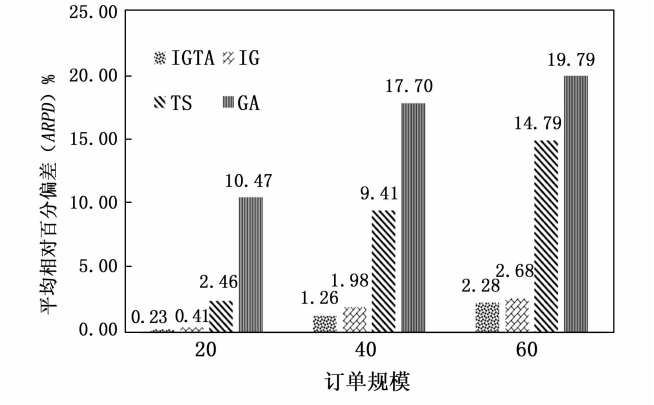

图 2 4 种算法在不同规模下的 ARPD 对比图

从图 2 中可以看出当订单规模为 20 的时候所有算法的 ARPD 值均很小，随着问题规模的增大，IGTA 算法的 ARPD 值呈现出递增趋势。且在哪种订单规模下，统计上 IGTA 算法的 ARPD 值都是最小的，由此我们可以得出 IGTA 算法在小、中、大规模问题下的求解质量均优于其余 3 种对比算法。

5 结束语

本文针对预制流水车间的订单接受与调度问题，构建了线性整数规划模型，并通过提出一种改进的迭代贪婪算法来求解这一问题，并通过计算仿真的方式与经典 IG，TS 以及 GA 算法进行对比。结果通过最优目标值、目标均值、标准差以及平均相对百分偏差，4 个性能评价指标进行对比分析，表明了本文提出的 IGTA 算法有良好的求解效果。下一步的研究可以对 IGTA 算法中的局部搜索作进一步改良以开发效率更高的预制流水车间的订单接受与调度模型的元启发式算法；同时，在未来的研究中可以将本文方法推广到汽车制造，钢生产等按订单进行生产的行业中。

参考文献：

[1] Slotnick S A, Morton T. Order acceptance with weighted tardiness [J]. Computers and Operations Research, 2007, 34 (10): 3029 - 3042.

[2] Wang X, Xie X, Cheng T C E. Order acceptance and scheduling in a two— machine flowshop [J]. International Journal of Production Economics, 2013, 141: 366 - 376.

[3] Gordon V S, Strusevich V A. Single machine scheduling and due date assignment with positionally dependent processing times [J]. European Journal of Operational Research, 2009, 198: 57 - 62.

[4] Ghosh J B. Job selection in a heavily loaded shop [J]. Computers & Operations Research, 1997, 24: 141 - 145.

[5] Xiao Y Y, Zhang R Q, Zhao Q H, Kaku I. Permutation flow shop scheduling with order acceptance and weighted tardiness [J]. Applied Mathematics and Computation, 2012, 218: 7911 - 7926.

[6] Lin S W, Ying K C. Increasing the total net revenue for single machine order acceptance and scheduling problems using an artificial bee colony algorithm [J]. Journal of the Operational Research Society, 2013, 64: 293 - 311.

[7] Wang X, Xie X, Cheng T C E. A modified artificial bee colony algorithm for order acceptance in two—machine flow shops [J]. International Journal of Production Economics, 2013, 141: 14 - 23.

[8] 郑　凡，雷德明．并行变邻域搜索下的订单接收与流水车间调度 [J]．武汉理工大学学报（信息与管理工程版），2015，37 (4)：519—523.

[9] Wang Z J, Hu H. Improved precast production — scheduling model considering the whole supply chain [J]. Journal of Computing in Civil Engineering, 2017, 31 (4): 667.

[10] Prata B D A, Neto A R P, Sales CJDM. An integer linear programming model for the multiperiod production planning of precast concrete beams [J]. Journal of Construction Engineering & Management, 2015, 141 (10): 991.

[11] Ko C H, Wang S F. Precast production scheduling using multi —objective genetic algorithms [J]. Expert Systems with Applications, 2011, 38 (7): 8293 - 8302.

[12] Ruiz R, Stutzle T. A simple and effffective iterated greedy algorithm for the permutation flowshop scheduling problem [J]. European Journal of Operational Research, 2007, 177: 2033 - 2049.

[13] 潘全科，谢圣献，张亚卿，等．解决无等待流水线调度问题的新算法 [J]．机械科学与技术，2006 (12)：1487 - 1490.

[14] Chaurasia S N, Singh A. Hybrid evolutionary approaches for the single machine order acceptance and scheduling problem [J]. Applied Soft Computing, 2017, 52: 725 - 747.

(上接第 215 页)

5　结束语

基于模型的系统工程方法能有效解决基于文档的设计手段和流程产生的风险和问题。本文研究了使用较少的 MBSE 方法论——基于封装 SysML 建模语言的 ARCADIA 方法，并基于该方法论的基本思想和开发流程，结合运载火箭电气系统的研制特点，获得了针对运载火箭地面测试时正常箭上供配电场景的能源子系统的架构模型，详细描述了系统分析、逻辑架构及物理架构的过程，并与目前较为主流的 Harmony 系统工程方法进行对比，针对运载火箭电气系统设计，ARCADIA 方法建模在系统架构和系统运行方面有着较好的优势，对运载火箭专业设计师本身要求也较低，比较适合解决系统层面的问题，为后续在电气系统中开展应用 MBSE 方法提供了参考。

参考文献：

[1] 朱　静，杨　晖，高亚辉，等．基于模型的系统工程概述 [J]．航空发动机，2016，42 (4)：12 - 16.

[2] 李　强，胡元威，董余红，等．基于模块的运载火箭电气系统匹配验证仿真 [J]．计算机测量与控制，2019，27 (2)：256 - 259.

[3] 薛　威，贾超群，李　雯，等．基于 MBSE 在航空电子通信系统中的应用 [J]．电子科技，2016，29 (5)：45 - 48.

[4] Holt J, Perry S. SysML for systems engineering [M]. Institution of Engineering and Technology, 2008.

[5] Weilkiens T. Systems engineering with SysML/UML: modeling, analysis, design [M]. Morgan Kaufmann OMG Press/Elsevier, 2008.

[6] Sanford F, Alan M, Rick S. OMG Systems Modeling Language (OMG SysML) Tutorial [A]. INCOSE International Symposium [C]. 2006: 1731 - 1862.

[7] 鞠文煜，付　昕．Arcadia 建模方法与 SysML 建模方法比较研究 [J]，民用飞机设计与研究，2018 (3)：92 - 96.

[8] Bonnet S. Acquisition and early evaluation of architecture with arcadia and capella [A]. [S.I]: NASA JPL MBSE Syposim [C]. 2016.

[9] US Department of Defense Deputy Chief Information Officer. DoD architecture framework Version 2.02 [S]. [S.I]: US Department of Defense Deputy Chief Information Officer, 2010.

[10] 李浩敏．基于模型的飞机系统架构设计综述 [J]，民用飞机设计与研究，2017 (3)：17 - 20.

[11] Jean Luc Voirin, Stephane Bonnet. A MBSE method for system, Software and Hardware Architecture Design—ARCADIA and Capella [Z]. 2015.