一种基于协同进化的流水线向 *Seru* 系统转化方法

吴旭辉[1] 杜劭峰[2] 郝慧慧[2] 于 洋[3] 殷 勇[4] 李冬妮[1]

摘 要 *Seru* 生产系统是一种被广泛应用于电子制造产业的新型生产模式, 但由于流水线向 *Seru* 系统转化问题 (Line-*seru* conversion) 包含有 *Seru* 构建与 *Seru* 调度两个相互耦合的子问题, 现有算法难以在同时兼顾解的质量与计算效率的情况下对问题进行求解. 因此, 本文针对流水线向 *Seru* 系统转化问题的特点, 提出了一种协同进化算法, 即在进化算法中加入了协同机制, 将 *Seru* 构建与 *Seru* 调度子问题作为两个子种群利用该机制进行协同进化, 从而弥补了现有算法的不足. 并且, 本文还针对问题特点设计了个体基因编码方式, 从而使规划获得的 *Seru* 生产系统具有更优的生产性能及均衡性能. 实验表明, 采用加入了协同机制的进化算法比传统解决流水线向 *Seru* 系统转化问题的方法具有更好的性能, 本文所提的方法在最小化产品流通时间和劳动时间有较好的性能表现, 并且具有较高的计算效率.

关键词 协同进化, 流水线, *Seru* 系统, 转化

引用格式 吴旭辉, 杜劭峰, 郝慧慧, 于洋, 殷勇, 李冬妮. 一种基于协同进化的流水线向 *Seru* 系统转化方法. 自动化学报, 2018, **44**(6): 1015−1027

DOI 10.16383/j.aas.2018.c160642

A Line-*seru* Conversion Approach by Means of Cooperative Coevolution

WU Xu-Hui[1] DU Shao-Feng[2] HAO Hui-Hui[2] YU Yang[3] YIN Yong[4] LI Dong-Ni[1]

Abstract Line-*seru* conversion is an innovative assembly system applied widely in the electronics industry. However, extant algorithms can hardly come into play in solving the line-*seru* conversion problem. The reason lies in that the line-*seru* conversion problem consists of two interacting subproblems, i. e., *seru* formation and *seru* loading, so it is difficult to obtain high quality solutions with affordable computation efficiency. Thus, an evolutionary algorithm with a cooperation mechanism is proposed in this paper. With the cooperation mechanism, the two subproblems can cooperatively evolved as two subpopulations simultaneously so as to address the aforementioned problem. Moreover, the coding of chromosomes representing scheduling result is modified to satisfy the specific requirement of the conversion and acquire solutions with enhanced performance and balancing ability. Computational result shows a better performance of the proposed method in minimizing total throughput time, total labor hours and computational costs.

Key words Cooperative coevolution, assembly line, *seru* system, conversion

Citation Wu Xu-Hui, Du Shao-Feng, Hao Hui-Hui, Yu Yang, Yin Yong, Li Dong-Ni. A line-*seru* conversion approach by means of cooperative coevolution. *Acta Automatica Sinica*, 2018, **44**(6): 1015−1027

当前, 我国的制造业成本已达美国的 90 % 以上, 传统的劳动密集型制造业竞争力已逐渐消失[1], 产品多样性和定制化急剧增加, 产品生命周期显著缩短, 导致市场需求的波动性和不确定性随之加剧. 日趋多变的市场环境要求企业具有更为强大和迅速的应变能力, 这种能力已经很难通过传统生产模式得到满足.

以索尼 (Sony) 和佳能 (Canon) 为例, 由于它们的生产具有多品种、小批量、高附加值的特点, 且产品频繁进行设计升级与更新换代, 因此在生产技术和配置上要求生产系统能够根据生产需要进行快速调整, 而传统的基于精益思想构建的同步式、集成式的流水线已不能满足这样的需求. 因此索尼和佳能通过构建一种新型生产系统, 即 *Seru* (单元) 生产系统, 在多变的市场环境中脱颖而出[2−4]. 佳能与索尼对 *Seru* 单元的实践转化遵循以下步骤: 1) 当顾客需求产生波动, 组装流水线的不足凸现出来, 为了获得响应能力, 可以采取战略上的重新选择; 2) 拆除流水线, 通过资源集中放置和去除/替换、多能工培训,

收稿日期 2016-09-08 录用日期 2017-07-12
Manuscript received September 8, 2016; accepted July 12, 2017
国家自然科学基金 (71401014), 特种车辆及其传动系统智能制造国家重点实验室开放课题 (GZ2016KF003) 资助
Supported by National Natural Science Foundation of China (71401014) and State Key Laboratory of Smart Manufacturing for Special Vehicles and Transmission Systems (GZ2016KF003)
本文责任编委 宋士吉
Recommended by Associate Editor SONG Shi-Ji
1. 北京理工大学计算机学院智能信息技术北京市重点实验室 北京 100081 中国 2. 特种车辆及其传动系统智能制造国家重点实验室 包头 014000 中国 3. 东北大学流程工业综合自动化国家重点实验室 沈阳 110819 中国 4. 同志社大学大学院商学研究科 京都 日本
1. Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081, China 2. State Key Laboratory of Smart Manufacturing for Special Vehicles and Transmission System, Baotou 014000, China 3. Institute of Systems Engineering, State Key Laboratory of Synthetic Automation for Process Industries, Northeastern University, Shenyang 110819, China 4. Graduate School of Business, Doshisha University, Kyoto, Japan

获得独立性和自治性, 构建分割式 *Seru* 单元来取代流水线; 3) 随着 *Seru* 单元系统的成熟, 构建起 *Seru* 生产系统来达到供需平衡. 基于以上步骤, *Seru* 生产系统在实践中取得了令人瞩目的成效. 索尼在构建了 *Seru* 生产系统后, 累计拆除了 35 000 米的传送带流水线, 节约了 71 万平方米的场地空间, 节省了 36 846 个工作岗位 (相当于其之前劳动力总数的 1/4). 佳能的 54 家工厂在应用 *Seru* 生产系统后的 5 年间拆除了 20 000 米的组装流水线, 释放出 72 万平方米的厂房 (相当于 12 家大型工厂), 节省了 35 976 个岗位, 总计降低成本 2 亿 3 000 万日元, 平均生产效率甚至超过了丰田[5].

继索尼和佳能之后, *Seru* 生产系统在电子 (如三星、松下、富士通等)、汽车零部件 (如通用、丰田、本田等) 以及食品行业得到推广并获得成功[2−8]. 在我国, 随着外资企业大量建厂, 带动了 *Seru* 生产系统这一新型生产模式的传播和推广. 目前, 海尔、美的以及许多东部沿海的生活家电企业已经推行了 *Seru* 生产系统进行生产, 此外, 内蒙一机等企业在其小型零部件的装配中也开始实施 *Seru* 生产系统. 这些令人瞩目的现象引起了研究领域的浓厚兴趣 — 为什么 *Seru* 生产系统不仅节约了资源, 还能够获得更快的响应速度、更高的效率和更大的利润? 许多学者 (包括 INFORMS 会士 Stecke、Decision Sciences Institute (DSI) 主席 Swink、顶级期刊 *Production and Operations Management* (POM) 创始人 Singhal、顶级期刊 *Manufacturing & Service Operations Management* (M & SOM) 主编 Tang 等) 先后围绕 *Seru* 生产系统开展研究或将其作为一个新兴研究领域进行介绍[2, 9], 认为 *Seru* 生产系统代表了 "下一代生产系统" 的发展趋势[7].

目前的研究主要集中于两方面:

一方面是通过与传统生产方式的对比论证 *Seru* 生产系统的优势. Yin 等[5] 通过研究佳能企业的生产转型过程, 指出了将传统流水线改造为 *Seru* 生产系统的重要性. De Treville 等[7] 指出 *Seru* 生产系统在不确定市场环境下, 可使生产系统具备比现有的装配流水线更加强大的应变能力, Liu 等[10−12] 也提出了相同的观点. Manupati 等[4] 将 *Seru* 生产系统与传统流水线对比, 认为 *Seru* 生产系统更加节省人力资源, 并且更加规范和灵活. Liu 等[13−14] 认为, *Seru* 生产系统是一种更为先进的生产模式, 其潜力远大于传统流水线, 同时也论证了 *Seru* 生产系统具有缩短提前期、降低库存的效果. Stecke 等[2] 在探索可以适应市场需求多变的生产模式时指出, *Seru* 生产系统具有长期学习和进化的能力. Zhang 等[15] 指出, *Seru* 生产系统可以节约大量的环境资源, 减少费用支出, 具有可持续发展性.

另一方面集中于如何将传统流水线改造为 *Seru* 生产系统. Luo 等[16] 研究了将传统流水线转为 *Seru* 生产系统过程中的人员分配问题. Yu 等[17−19] 在研究传统流水线向 *Seru* 生产系统转化的过程中, 提出了一种流水线分割与重组的数学模型, 他们还研究了转化过程中考虑时间成本、劳动力等多目标优化问题, 以及如何减少劳动力的问题. Liu 等[10−12,14,18] 提出传统流水线转化为 *Seru* 生产系统的框架与原则, 并且研究了多能工的培训分配问题. Zhang 等[15] 也研究了多能工对传统流水线转化为 *Seru* 生产系统的作用. Yu 等[20] 研究了 *Seru* 生产系统的形成与分配问题.

在对流水线向 *Seru* 系统转化 (Line-*seru* conversion) 这个问题的研究中, 可将问题简化为两个决策过程: *Seru* 构建和 *Seru* 调度. 将 *Seru* 生产系统投入实际生产的首要步骤是如何将现有流水线拆分为若干 *Seru*, 以完成 *Seru* 生产系统的构建, 即 *Seru* 构建. *Seru* 调度决策批次如何分配到 *Seru* 中. 目前的研究都是对 *Seru* 调度阶段采用了例如 FCFS (First come first served) 等简单的启发式规则进行规划, 然后再设计算法来求解 *Seru* 构建的精确解或较优解. 例如 Yu 等[21] 通过修改原始 NSGA (Non-dominated sorting genetic algorithm)-II 交叉、变异操作符的具体实现方式, 以适应 *Seru* 构建的特点, 从而将其运用在解决 *Seru* 构建问题上. Yu 等[19] 为了提高算法的知识利用能力, 将局部搜索算法与 NSGA-II 相结合, 以提高 *Seru* 构建的质量. Liu 等[18] 提出了一种三阶段启发式算法, 解决了 *Seru* 生产系统构建阶段中工人分配的问题.

但是, 上述研究并没有联合决策过程进行全局寻优, 因此通过这类方法规划出的 *Seru* 生产系统无法充分发挥系统的灵活性与均衡能力较强的优势. 实际上, 求解流水线向 *Seru* 转化问题的最优解应该是联合考虑 *Seru* 构造和 *Seru* 调度两个问题, 但如果同时对 *Seru* 构建与 *Seru* 调度两个方面联合进行寻优, 那么问题的解空间就会变得过于巨大, 现有方法难以在可接受的时间内得到 *Seru* 系统的最优规划方案. 所以, 如何在合理的算法运行时间内, 同时对两个方面进行规划以获得较优的规划方案, 是解决该问题的关键.

本文参考 Kaku 等[22] 研究, 引入了两项评估指标对 *Seru* 生产系统的性能进行评估, 即产品流通时间 (Total throughput time, TTPT) 和总劳动时间 (Total labor hours, TLH) 作为优化目标. TTPT 即为系统中所有产品批次加工完成所花费的时间, 用来评估系统的生产率. TLH 即为系统中所有工人的累计劳动时间, 用来评估系统的人员劳动效率.

基于以上分析, 本文提出一种多目标协同进

化算法 (Multi-objective cooperative coevolution, MOCC), 用以解决流水线向 *Seru* 转化的联合优化问题. 本文方法在传统进化算法思想的基础上, 加入了子种群间的协同机制. 算法针对流水线向 *Seru* 转化问题的特点, 建立两个子种群分别对应 *Seru* 构建与 *Seru* 调度两个子问题. 通过协同机制, 两个子种群便可相互协同进化, 即在进化过程中同时考虑 *Seru* 构建与 *Seru* 调度且不增加算法运行时间, 以此解决现有方法存在的不足, 获得更优的转化结果. 另外, 本文在解的基因编码中引入了冗余码, 以期在规划过程中获得更优的均衡性能. 通过应用 MOCC, 转化后的 *Seru* 生产系统在灵活性与生产能力上都会获得较大的提升.

本文其余部分组织如下: 第 1 节介绍了在进化算法中引入协同机制的相关研究; 第 2 节给出了流水线向 *Seru* 生产系统转换问题抽象出的问题模型; 第 3 节提出了一种带有协同机制的进化算法; 第 4 节通过仿真实验验证该算法的性能; 第 5 节给出结论并展望下一步工作.

1　相关工作

本节针对调度问题, 介绍了在进化算法中引入协同机制的相关研究.

进化算法在解决复杂调度问题上有着良好的应用效果[23−24]. 但是一些调度问题, 例如流水线向 *Seru* 系统转化往往涉及到一个完整的解由若干个子问题的解组成的情况 (在流水线向 *Seru* 系统转化问题中, 子问题即为 *Seru* 构建与 *Seru* 调度), 这些子问题往往耦合度较高. 但传统的进化算法只能够针对一个独立的问题进行调度规划, 对于上述这类问题, 只能够将每个子问题单独进化, 再将解进行组合. 如贾凌云等[25] 针对运输能力受限条件下的跨单元问题, 提出了一种基于混合蛙跳与遗传规划的超启发式算法, 该算法首先对工序分派、工序排序、运输组批和路径决策等 4 个子问题分别求解, 最后再将 4 个子问题的解进行组合. 田云娜等[26] 提出一种基于动态决策块和蚁群优化的超启发式方法, 利用决策块分别解决跨单元生产调度和运输调度问题, 并最终生成调度解. 这样的方式虽然可以得到一个完整的解, 但子问题之间的依赖关系在进化过程中被忽略, 多个子问题的解组合后协作效果往往难以满足生产需要. 因此传统的进化算法并不适用于解决这类问题.

为了充分利用进化算法对于解决调度问题的优势, 一种多阶段协作的方法被应用到求解的过程中. Cheng[27] 提出了一种分阶段: 先使用作业调度 (Job dispatching) 再进行交货期安排决定 (Due-date assignment decisions) 的方法来解决作业车间调度问题. Baker[28] 使用了调度规则和交货期安排两种方式分别解决两个阶段的问题并分析了影响分派规则的各类参数. Miyazaki[29] 给出了一种在调度系统中加入顺序方式的分配方法用来降低作业延迟. 这样的方法在子问题之间耦合关系较弱时, 可以取得很好的应用效果, 但如果子问题间相互依赖, 由于前序子种群的解已被确定, 当前阶段的子种群的知识探索便会被严重的制约, 无法跳出已有的约束搜索到更优的解. 而实际生产环境中, 子问题间往往耦合度较高, 因此, 这样的方法依然不能很好地求解多个子问题.

为了充分考虑子种群之间依赖关系, 并将其在调度过程中加以体现, Cochran 等[30] 提出了一种基于代表机制的种群协同方式, 从而实现多个子种群同时进化. Goh 等[31] 使用的方法在进化初期允许分解优化问题, 并在各个种群中加入合作与竞争的机制. 通过这样的方式, 子种群间的依赖关系就可以被充分考虑并且算法的计算效率也可以得到大幅度的提高.

基于以上分析, 为了保证算法在实际生产环境中的成功应用, 本文结合以上思路, 针对流水线向 *Seru* 生产系统转换问题的特点, 考虑了子种群之间的相互作用与影响, 设计了一种带有协同机制的进化算法. 本文的主要贡献在于以下两点: 1) 提出了充分考虑子种群间依赖关系的协同机制, 并将其应用在流水线向 *Seru* 生产系统转化的问题上; 2) 对于协同机制两个子种群间个体相互依赖的特点, 设计了具有针对性的个体基因编码方式, 使协同机制能够被应用到进化算法中去.

2　问题模型

流水线向 *Seru* 系统转化问题已受到广泛的关注, 本文参考 Yu 等[21] 提出的问题模型, 以最小化 TTPT 和 TLH 作为优化调度目标, 同时考虑了 *Seru* 构建和 *Seru* 调度两个子问题的协同.

2.1　问题描述

带有装配流水线和装配 *Seru* 的装配系统主要有以下三种: 纯 *Seru* 系统、纯装配线系统、混合装配系统[22]. 为了保留 *Seru* 生产系统基本特点并简化问题模型, 本文讨论的流水线向 *Seru* 转化问题是将流水线转换为纯 *Seru* 系统.

纯 *Seru* 生产中, *Seru* 间生产的平衡极为重要. 实际生产中, 生产批次的大小差异很大, 这使得在 *Seru* 构建和 *Seru* 调度的过程中极易出现 *Seru* 间生产不平衡的情况. Yu 等[21] 在研究中, 将生产批次的大小差异设计为正态分布. 而本文的问题模型中, 生产批次的大小分布为均匀分布, 更贴合实际生

产状况.

本文中对流水线向 *Seru* 转换方法的评估是基于帕累托 (Pareto) 多目标最优, 对 TTPT 和 TLH 两个指标进行评价.

2.2 问题假设

本文的基本假设条件如下:

1) 有 N 个产品类型, M 个产品批次 (每个批次只有一种产品类型), 且批次大小已知;

2) *Seru* 中的大多数工序是用简单低价的设备完成的, 复制它们通常成本较小[6], 故复制设备的成本可忽略不计;

3) 一个产品批次只能在一个 *Seru* 中完成, 不考虑拆分;

4) 每一个工序都在其给定的设备中完成, 若某类型产品无需某种工序, 则跳过该工序的设备;

5) 在 *Seru* 中的工序与流水线中的工序一致, 共有 W 种不同的工序, 与工人人数相同;

6) 在 *Seru* 中, 每个工人能执行 *Seru* 的所有工序, 且忽略相邻工序之间的延时;

7) 在流水线生产方式中, 每个工人只执行一个工序, 工人数与工序数相等;

8) 不同 *Seru* 中的工人数可以不同, 但不超过工人总人数;

9) 两种不同类型产品相邻加工时, 考虑设备的准备时间, 否则忽略准备时间.

2.3 符号列表

与本文问题相关的符号变量定义如下所示.

索引:

i 为工人的索引集 ($i = 1, 2, \cdots, W$), W 为工人总数;

j 为 *Seru* 序号的索引集 ($j = 1, 2, \cdots, J$);

n 为产品型号的索引集 ($n = 1, 2, \cdots, N$);

m 为产品批次的索引集 ($m = 1, 2, \cdots, M$);

k 为产品批次在一个 *Seru* 中加工顺序的索引集 ($k = 1, 2, \cdots, M$).

参数:

$$V_{mn} = \begin{cases} 1, & \text{表示批次的产品类型是 } n \\ 0, & \text{其他} \end{cases}$$

B_m: 批次 m 的大小;

T_n: 为产品类型 n 在流水线的平衡时间;

SL_n: 为在流水线中加工产品类型 n 的准备时间;

SCP_n: 为在 *Seru* 中加工产品类型 n 的准备时间;

η_i: 为工人 i 在 *Seru* 中有效操作工序个数的上界, 当工人在 *Seru* 中操作工序的个数超过这个上界时, 其工序加工时间将变长;

CW_i: 工人 i 在 *Seru* 中操作多个工序的能力系数;

ϵ_i: 为多能工系数;

β_{ni}: 为工人 i 加工产品类型 n 的技术系数, 值越小表示能力越高.

决策变量:

$$X_{ij} = \begin{cases} 1, & \text{工人 } i \text{ 配到了 } Seru\ j \\ 0, & \text{其他} \end{cases}$$

$$Z_{mjk} = \begin{cases} 1, & \text{批次 } m \text{ 被分配到了 } Seru\ j \text{ 中} \\ & \text{并被第 } k \text{ 个生产} \\ 0, & \text{其他} \end{cases}$$

变量:

$$CW_i = \begin{cases} 1+\varepsilon_i(W-\eta_i), & W > \eta_i \\ 1, & W \leq \eta_i \end{cases}, \forall i \tag{1}$$

TC_m: 批次 m 的一个产品在 *Seru* 中一个工序的加工时间, 等于 *Seru* 中所有工人对批次 m 加工时间的平均值, 如式 (2) 所示:

$$TC_m = \frac{\sum_{n=1}^{N}\sum_{i=1}^{W}\sum_{j=1}^{J}\sum_{k=1}^{M} V_{mn}T_n\beta_{ni}CW_iX_{ij}Z_{mjk}}{\sum_{i=1}^{W}\sum_{j=1}^{J}\sum_{k=1}^{M} X_{ij}Z_{mjk}} \tag{2}$$

SC_m: 批次 m 在 *Seru* 中的准备时间, 如果批次 m 与 *Seru* 中上一批次的产品类型相同, 则其准备时间为 0; 否则为 SCP_nV_{mn}, 如式 (3) 所示:

$$SC_m = \begin{cases} SCP_nV_{mn}, & V_{mn} = 1, V_{m'n} = 0 \\ 0, & V_{mn} = V_{m'n} = 1 \end{cases} \tag{3}$$

$(m'jZ_{mjk} = 1,\ Z_{m'j(k-1)} = 1,\ \forall j, k)$

FC_m: 批次 m 在 *Seru* 中的流通时间, 与批次大小、TC_m、工序个数和 *Seru* 中的工人个数相关, 如式 (4) 所示:

$$FC_m = \frac{B_mTC_mW}{\sum_{i=1}^{W}\sum_{j=1}^{J}\sum_{k=1}^{M} X_{ij}Z_{mjk}} \tag{4}$$

FCB_m: 批次 m 在 *Seru* 中的开始时间, 等于该 *Seru* 中前序批次流通时间和准备时间的总和, 如式 (5) 所示:

$$FCB_m = \sum_{s=1}^{m-1}\sum_{j=1}^{J}\sum_{k=1}^{m}(FS_s + SC_s)Z_{mjk}Z_{sj(k-1)} \tag{5}$$

2.4　问题形式化描述

Seru 系统的产品流通时间 TTPT 和总劳动时间 TLH 表示如下:

$$\mathrm{TTPT}=\min\left\{\max_{m}(FCB_m+FC_m+SC_m)\right\} \tag{6}$$

$$\mathrm{TLH}=\min\sum_{m=1}^{M}\sum_{i=1}^{W}\left(\sum_{j=1}^{J}\sum_{k=1}^{M}FC_mX_{ij}Z_{mjk}\right) \tag{7}$$

根据实际生产中的问题特性和约束, 本文的约束条件描述如下:

$$1\leq\sum_{i=1}^{W}X_{ij}\leq \mathrm{W},\ \ \forall j \tag{8}$$

$$\sum_{j=1}^{J}X_{ij}=1,\ \ \forall i \tag{9}$$

$$\sum_{j=1}^{J}\sum_{k=1}^{M}Z_{mjk}=1,\ \ \forall m \tag{10}$$

$$\sum_{m=1}^{M}\sum_{k=1}^{M}Z_{mjk}=0,\forall j\mid\sum_{i=1}^{W}X_{ij}=0 \tag{11}$$

$$\sum_{j=1}^{J}\sum_{k=1}^{M}Z_{mjk}\leq\sum_{j'=1}^{J}\sum_{k'=1}^{M}Z_{(m-1)j'k'},m=2,3,\cdots,M \tag{12}$$

其中, 式 (8) 表示每个 *Seru* 中的工人数不可超过工人总数; 式 (9) 表示每个工人必须且仅可被分配至一个 *Seru*; 式 (10) 表示每个批次被且仅被一个 *Seru* 加工; 式 (11) 表示一个产品批次不可以被分配至没有工人的 *Seru*; 式 (12) 表示产品批次必须按顺序被分配.

2.5　问题性质分析

流水线向 *Seru* 系统转化包含 *Seru* 构建和 *Seru* 调度两个决策过程. Yu 等[20] 已证明将现有流水线拆分为若干 *Seru* 的 *Seru* 构建问题是一个 NP 难问题. 对于决策批次如何分配到 *Seru* 中的 *Seru* 调度问题的复杂度, 下面将给出简要说明.

已知多处理机调度问题是一个 NP 难问题, 描述如下: 设有 n 个独立的作业 $\{1,2,\cdots,\ n\}$, 由 m 台相同的机器 $\{1,2,\cdots,\ m\}$, 进行加工处理. 作业 i 所需时间为 t_i. 任何作业可以在任何一台机器上加工处理, 但未完工前不允许中断处理, 任何作业不能拆分成更小的子作业. 在满足以上约束下, 要求给出一种作业调度方案, 使所给的 n 个作业在尽可能短的时间内由 m 台机器加工处理完成.

在 *Seru* 调度问题中, 可将每一个已经被构建的 *Seru* j, 对应于多机器调度问题中的一个机器, 每个加工批次 m, 对应于作业. 此处, 不同于多处理机问题中的机器, 各个 *Seru* 之间存在加工能力差异. *Seru* 调度问题所求解的帕累托最优结果, 相较于多处理机调度问题也更为复杂. 由此可见, 对于更为复杂的 *Seru* 调度问题, 其在多项式时间内也无法得到最优解.

3　MOCC 算法

为解决流水线向 *Seru* 转化的问题, 本文提出了一种基于个体编码内部交换的进化算法.

由于所考虑的问题可分为 *Seru* 构建和 *Seru* 调度两个子问题, 并且子问题之间存在着交互作用, 本方法采用了协同进化的策略, 即形成两个子种群分别对应两个子问题. 两个子种群互相协作共同进化, 最后进化出协同效果较好的完整解.

针对两个子种群的特点, 本方法设计了一种增加了冗余编码的编码方案, 以更好地进行 *Seru* 之间的平衡.

算法流程:

步骤 1. 初始种群, 随机产生个体编码, 形成两个子种群;

步骤 2. 使用个体编码内部交换操作对其中个体进行进化;

步骤 3. 每个子种群在其他子种群代表的协助下评估种群内部个体适应度并更新代表;

步骤 4. 每代种群进化后采用精英策略生成下一代种群;

步骤 5. 若未达到迭代次数, 返回步骤 2; 若达到最终迭代次数, 停止.

3.1　编码方案

Seru 生产系统的出现, 能够实现各个生产 *Seru* 间的平衡, 非常好地适应了当今加工周期短、多品种、小批量的生产模式. 为了更好地实现 *Seru* 生产系统的平衡特性, 针对前文中提出的流水线向 *Seru* 转化问题所包含的 *Seru* 构建和 *Seru* 调度两个子问题, 我们分析了其各自的特点, 设计出了一种加入冗余码的编码方案. 这种加入冗余码的编码方式可以更好地引导进化方向, 通过使 *Seru* 间的生产状态达到平衡, 更好地进行 *Seru* 构建和 *Seru* 调度.

针对 *Seru* 构建问题中可能出现的 *Seru* 个数及 *Seru* 内工人数量与加工货物批次大小间不能互相平衡协调的问题, 在 *Seru* 构建的编码中加入的冗余码用于对工人编码的分隔, 使各个 *Seru* 内包含的工人数量有所区别, 构成规模不同的 *Seru*.

针对 *Seru* 调度问题中可能出现的货物批次大小与 *Seru* 规模不匹配造成的不平衡问题, 在 *Seru* 调度的编码中加入用于占位的冗余码, 使规模不同的 *Seru* 有机会被分配到不同个数的货物.

3.1.1 *Seru* 构建

Seru 构建是流水线向 *Seru* 转化问题中首要决策步骤, 它决定了 *Seru* 生产系统中 *Seru* 的个数以及每个 *Seru* 内的工人. *Seru* 构建的结果将影响 *Seru* 生产系统的性能, 为了使构建好的 *Seru* 生产系统具有良好的平衡特性, 我们采用文献 [21] 中编码方式解决 *Seru* 构建子问题, 即一种通过加入冗余码对工人编码进行分隔的编码方式.

工人编码表示了构成当前 *Seru* 生产系统中所有的工人, 工人的不同组合可以构成不同的 *Seru*, 所以 *Seru* 构建的过程就是将工人进行组合的过程. 因此用一条编码表示一种全部工人的组合方式, 不同的编码组成一个种群在进化算法中进行进化. 编码中不仅有全部工人的信息, 还加入了用于分隔的冗余码, 可将表示工人的编码进行划分. 被冗余码分隔开的 (在两个不相邻的冗余码之间的) 工人编码同属于一个 *Seru*, 这样不仅确定了 *Seru* 的个数, 也同时确定了 *Seru* 内工人的分配.

在进化过程中, 用于分隔的冗余码可能进化出不同的位置, 这样会使得每个 *Seru* 内包含的工人有所变化, 使 *Seru* 的规模出现变化, 更能适应变批量的生产情况, 有更强的平衡能力.

下面举例说明该编码方式:

考虑有 5 个工人编号为 1～5, 冗余码为 6～9 的情况.

若编码为:

1	7	6	2	5	9	3	4	8

图 1 *Seru* 构建编码示例 1

Fig. 1 Example 1 of coding for *seru* formation

则 *Seru* 构建情况为: 工人 1 为一个 *Seru*, 工人 2、5 为一个 *Seru*, 工人 3、4 为一个 *Seru*.

若编码为:

5	3	2	7	8	1	4	6	9

图 2 *Seru* 构建编码示例 2

Fig. 2 Example 2 of coding for *seru* formation

则 *Seru* 构建情况为: 工人 5、3、2 为一个 *Seru*, 工人 1、4 为一个 *Seru*.

3.1.2 *Seru* 调度

Seru 调度是指给各个 *Seru* 安排需要加工的批次. 在解决流水线向 *Seru* 转化问题时, *Seru* 调度往往在 *Seru* 构建好之后由简单的启发式规则来完成. 而本文提出的方法将 *Seru* 调度也作为一个子问题通过进化算法给出调度方案. 在解决 *Seru* 调度子问题时, 为了使所有的批次都能够被分配至构建的 *Seru* 中, 我们设计了一种基于求模运算的 "批次－*Seru*" 映射机制; 为了能使各个 *Seru* 更好地平衡对变批量批次的加工, 我们在批次的编码内设计了带有用于占位的冗余码的编码方式.

通过对批次编码的操作可以实现 *Seru* 的调度. 在批次的编码中加入冗余码, 在每次进化后, 根据 *Seru* 构建的结果对每个编码的位置进行求模运算, 将每个编码安排入对应的 *Seru*. 若为冗余码, 则该位置空闲, 若为批次编码则将该批次放入该 *Seru* 进行加工.

在进行 *Seru* 调度的过程中, 不同的调度结果有可能导致 *Seru* 间的不平衡, 为使各个 *Seru* 保持平衡, 批次编码中增加了冗余编码. 在添加冗余编码后, 批次编码分配入不同 *Seru* 的个数会产生变化, *Seru* 的加工能力将通过这一方式再次平衡.

下面举例说明该编码方式: 考虑有 3 个 *Seru*, 5 个批次编号为 1～5, 冗余码为 6～15 的情况.

若编码为:

1	7	11	6	2	15	13	5	14	9	3	4	11	10	8

图 3 *Seru* 调度编码示例 1

Fig. 3 Example 1 of coding for *seru* loading

则 *Seru* 调度情况为: 第一个 *Seru* 中有位置模 3 为 0 的编码: 11, 15, 14, 4, 8. 去除冗余码后, 第一个 *Seru* 加工的批次为: 4.

第二个 *Seru* 中有位置模 3 为 1 的编码: 1, 6, 13, 9, 11. 去除冗余码后, 第二个 *Seru* 加工的批次为: 1.

第三个 *Seru* 中有位置模 3 为 2 的编码: 7, 2, 5, 3, 10. 去除冗余码后, 第三个 *Seru* 加工的批次为: 2, 5, 3.

若编码为:

12	14	5	13	3	15	2	7	8	11	1	4	6	10	9

图 4 *Seru* 调度编码示例 2

Fig. 4 Example 2 of coding for *seru* loading

则 *Seru* 调度情况为: 第一个 *Seru* 中有位置模 3 为 0 的编码: 5, 15, 8, 4, 9. 去除冗余码后, 第一个 *Seru* 加工的批次为: 5, 4.

第二个 *Seru* 中有位置模 3 为 1 的编码: 12, 13, 2, 11, 6. 去除冗余码后, 第一个 *Seru* 加工的批次为: 2.

第三个 *Seru* 中有位置模 3 为 2 的编码: 14, 3, 7, 1, 10. 去除冗余码后第一个 *Seru* 加工的批次为:

3, 1.

3.2　种群进化

种群的进化是根据问题模型设计基于个体编码的一种简单的进化算法. 该方法针对一个种群, 通过对个体编码的内部操作, 经过种群内的进化淘汰迭代, 不断地优化个体结构, 产生更优解.

3.2.1　交换操作

内部交换是种群进化的最基本的方法, 它可以使种群内部产生新的个体, 使种群保持进化. 交换个体内部编码可使新个体依据亲代的编码产生变化, 在一个种群内部, 可保持种群多样性. 具体的做法是, 对于一个父代个体, 随机选取其自身编码两位进行交换产生新个体.

交换前:

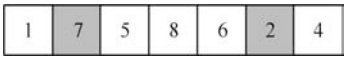

图 5　交换前编码示例

Fig. 5　Example of coding before exchange

如图, 深色的两位即是随机选取的两位编码, 交换后的个体编码如下:

交换后:

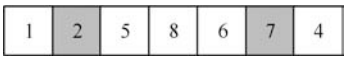

图 6　交换后编码示例

Fig. 6　Example of coding after exchange

3.2.2　精英策略

精英策略指的是种群在进化过程中不断保存下部分性能最优个体的做法. 在种群内部每一代进化过程中, 筛选出部分性能最优的个体, 这部分个体会进行保留, 不对其进行内部交换. 用这些最优个体替换掉经过内部交换后较差的部分个体, 形成新的种群, 再进行下一次进化.

下面给出精英策略的形式化描述:

Parent = $\{p \mid$ 亲代全部个体编码$\}$

Elitist = $\{elite \mid elite \in$ Parent $\cap$ nondominated rank_{elite} =1$\}$

Offspring′= $\{p' \mid p'$ 进化后个体编码$\}$

Eliminated = $\{elim \mid elim \in$ Offspring′ $\cap$ nondominated rank_{elim} = Max(nondominated rank)$\}$

Offspring = Offspring′– Eliminated $\cup$ Elitist

其中, Parent 为亲代种群, Elitist 为精英个体集合, Eliminated 为被淘汰个体种群, Offspring′ 为过渡子代种群, Offspring 为最终子代种群.

3.3　种群协同

本文所考虑的问题可分为 *Seru* 的构建和 *Seru* 的调度两个子问题, 并且子问题之间存在着交互作用. 由于以上所述的编码和进化过程中, 单个种群只针对两种子问题中的一个进行进化, 但是单个子种群中的个体无法构成一个完整的解, 所以在对一个子种群中个体的进化进行评价时, 需要加上其他子种群的个体形成一个完整的解.

本方法采用了协同进化的策略, 在本问题的两个子问题间建立协同机制, 使得对应的两个子种群互相协作共同进化, 最后进化出协同效果较好的完整解. 在进化过程中, 在对每一个个体进行评估时, 都先将该个体与来自另一种群的代表个体组成对于问题的完整解, 再对该完整解进行评估, 即评估将由该个体和另一种群的代表个体协同完成. 在本方法中, 子种群代表个体由二进制锦标赛算法, 根据非支配等级 (Nondominated rank) 从子种群中进行选择.

如图 7 所示, 两个子种群 A 和 B 分别对应 *Seru* 构建和 *Seru* 调度两个子问题, 子种群之间为协同进化关系. a 和 b 分别为子种群 A 和 B 中评价出的当前最优个体, a 和 b 共同组成一个当前最优解. 黑色实线箭头表示更新, 当子种群中有评价值更好的个体则对当前最优解的该部分进行替换. 黑色虚线箭头代表子种群之间的协作, 表示在评估时使用另一个子种群的最优个体. 例如, 当前最优解为 ab, 当子种群 A 在评估种群内的某个个体 a' 时, 则使用 b 与 a' 组成一个完整解进行调度, 进而得到适应度赋给 a'. 如果此时 a' 的适应度比 a 高, 说明出现了一组协同更优的解, 我们把 a 替换成 a', 当前的最优解为 $a'b$. 根据此种方式, 子种群在进化中互相影响, 最终得到一个可使种群间协作较好的解.

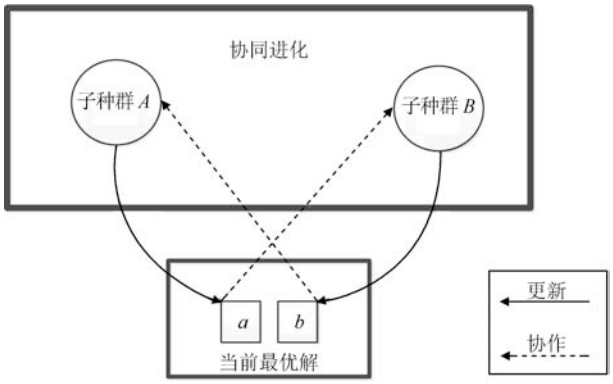

图 7　协同进化图

Fig. 7　Diagram of cooperative coevolution

4　对比实验

为了验证本方法的优化性能和计算效率, 本文进行了多组对比实验, 仿真实验采用 C# 语言实现, 运行在 3.10 GHz Core i5-2400 CPU, 4 GB RAM 的 PC 机上.

4.1 实验设计

本文实验设计的工人数为 20, 多能工系数 ε_i 按 N (0.2, 0.05) 分布, 具体如表 1 所示. 工人对不同类型产品熟练度数据的分布、工人对不同产品的熟练度以及 30 批产品的信息数据, 如表 A1, A2, A3, A4 所示. 进化算法交叉率 P_c 为 0.5, 精英策略中精英数量 N_{elite} 为 10.

表 1 算例产生的参数表

Table 1 Parameters of test problems

算例产生参数	取值
产品类型	5
批次大小	∼U[10, 110]
ε_i	∼N[0.2, 0.05]
SL_n	2.2
SCP_n	1.0
T_n	1.8
η_i	10

4.2 协同策略的影响

本文在进化算法中加入了协同机制, 在考虑两个子问题间交互关系的前提下, 同时对两个子问题进行规划以求获得性能更优的规划结果.

为了检验协同机制的有效性, 实验与没有加入协同机制的方法 (Multi-objective evolution, MOE) 进行了对比.

在这里, 我们引入了参考集 (由于问题规模较为庞大, 无法求得精确解, 故将算法迭代 100 次后获得的非支配集作为参考集), 并使用了 RNI (Ratio of non-dominated individuals)[32] 与 D[34−35] 两个指标对算法性能进行评估.

其中 RNI 定义了非支配集中解的个数与参考集中解个数的比值, RNI 越高即代表所得出非支配集的有效解越多; D 即为非支配集中的解与参考集中解的距离, D 越小即代表非支配集越接近参考集.

在没有加入协同机制的方法中, 先用本文方法解决 *Seru* 构建子问题; 再根据其结果用经典规则 FCFS 解决 *Seru* 调度子问题; 最后将对两个子问题的结果合成一个完整的解, 与本文中使用协同进化的算法结果进行比较.

在 5 个工人的情况下, 其两个评价指标 TLH 和 TTPT 的帕累托前沿结果对比如图 8 所示.

其余对比结果如表 2 所示, MOCC 与未加入协同机制的算法之间的 Gap 值计算方法均参考公式 (13)、(14), Av 为 Average 的缩写:

$$\mathrm{Gap}_{\mathrm{RNI}\_AV} = \frac{Av\ \mathrm{RNI}_{\mathrm{MOCC}} - Av\ \mathrm{RNI}_{\ \mathrm{MOE}}}{Av\ \mathrm{RNI}_{\mathrm{MOCC}}} \quad (13)$$

$$\mathrm{Gap}_{D\_AV} = \frac{Av\ D_{\mathrm{MOCC}} - Av\ D_{\mathrm{MOE}}}{Av\ D_{\mathrm{MOCC}}} \quad (14)$$

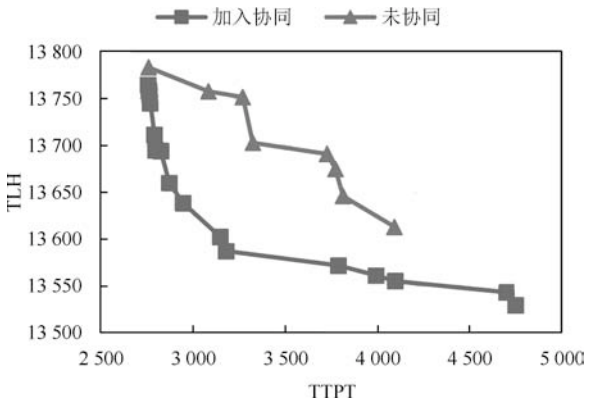

图 8 工人数量为 5 时, MOCC 与未加入协同算法的非支配集间的比较

Fig. 8 The non-dominated solutions of MOCC and the one without cooperation strategy with 5 workers

由实验结果可知, 在不同规模的测试问题下, 加入协同机制的方法均优于未加入协同机制方法的优化性能, $\mathrm{Gap}_{\mathrm{RNI}\_AV}$ 平均值为 36.68 %, Gap_{D_AV}

表 2 与未使用协同策略的性能对比

Table 2 Comparison proposed approach and the one without cooperation strategy

W	R	Proposed algorithm				MOE				$\mathrm{Gap}_{\mathrm{RNI}\_AV}$	Gap_{D_AV}
		Av RNI	Min RNI	$Av\ D_{av}$	$Av\ D_{\max}$	Av RNI	Min RNI	$Av\ D_{av}$	$Av\ D_{\max}$	(%)	(%)
5	20	0.66	0.50	0.04	0.09	0.42	0.35	0.13	0.37	57.14	256.83
10	43	0.50	0.33	0.12	0.68	0.45	0.35	0.16	0.81	10.00	36.63
15	62	0.51	0.32	0.03	0.22	0.40	0.45	0.06	0.48	27.50	117.42
20	57	0.65	0.47	0.02	0.13	0.41	0.35	0.05	0.36	56.89	169.32
25	31	0.64	0.54	0.11	0.26	0.52	0.44	0.16	0.21	22.96	49.06
30	39	0.69	0.56	0.09	0.28	0.41	0.46	0.18	0.32	45.57	87.23
Average										36.68	119.42

注: W 表示工人数量, R 表示参考集中解的数量

平均值为 119.42 %. 这组实验表明, 由加入协同机制的算法产生的解不仅有效解更多, 而且也更靠近参考集, 优化性能较未加入协同机制的算法有明显的提升.

产生这样结果的原因在于, 加入协同机制之后, 问题的两个子问题 (即 *Seru* 构建与 *Seru* 调度) 协同进化, 算法生成的解在解决对应子问题时, 考虑了子问题间的耦合关系, 因此与单独对子问题进化所产生的解相比, MOCC 所产生解的协同性能较为理想的. 而 *Seru* 生产系统能够发挥其优势的关键正是在于两阶段能够同时进行调整所产生的柔性与灵活性, 因此协同性能更好的解往往会为 *Seru* 系统带来更佳的应用效果. 由此可见, 未加入协同机制的方法虽然能够在单个子问题上获得较好的应用效果, 但当两个子问题的解组合为完整的解时, 由于其进化时模拟的场景不尽相同, 综合求解能力也就与加入协同机制的方法相比较差.

4.3　与其他方法对比

本文选择了 2 种解决流水线向 *Seru* 生产系统转化问题的方法对比我们的方法进行实验, 分别为: NSGA-II 和加入局部搜索的 NSGA-II. 除了引入 RNI 及 D 等指标进行性能对比, 为了更好地验证本方法对 *Seru* 生产系统进行均衡的能力, 我们还引入了 STDEV (Standard deviation)$_{\mathrm{TTPT}}$ 作为衡量算法均衡性能的指标. STDEV$_{\mathrm{TTPT}}$ 定义了方案中不同 *Seru* 间 TTPT 的标准差, STDEV$_{\mathrm{TTPT}}$ 越小, 即代表不同 *Seru* 间加工流通时间差距越小、*Seru* 系统加工能力的浪费越少, 算法的均衡性能也就越好.

4.3.1　与 NSGA-II 方法的对比

NSGA-II[36] 是一种基于 GA (Genetic algorithm) 的进化方法. Yu 等[21] 采用 NSGA-II, 以解决流水线向 *Seru* 系统转化问题. 本节将进行 MOCC 与 NSGA-II 的对比实验. 为了保证实验的公平性, 对比实验采用了与文献 [21] 相同的问题模型以及实验参数.

与本文中使用协同进化的算法结果进行比较, 其两个评价指标 TLH 和 TTPT 的帕累托前沿结果如图 9 所示. 其余对比结果如表 3 所示, MOCC 与 NSGA-II 之间的 Gap 值计算方法均参考公式 (15) ∼ (17):

$$\mathrm{Gap}_{\mathrm{RNI\_}AV} = \frac{Av\ \mathrm{RNI}_{\mathrm{MOCC}} - Av\ \mathrm{RNI}_{\mathrm{NSGA-II}}}{Av\ \mathrm{RNI}_{\mathrm{MOCC}}} \tag{15}$$

$$\mathrm{Gap}_{D\_AV} = \frac{Av\ D_{\mathrm{MOCC}} - Av\ D_{\mathrm{NSGA-II}}}{Av\ D_{\mathrm{MOCC}}} \tag{16}$$

$$\mathrm{Gap}_{\mathrm{STDEV}} = \frac{Av\ \mathrm{STDEV}_{\mathrm{MOCC}} - Av\ \mathrm{STDEV}_{\mathrm{NSGA-II}}}{Av\ \mathrm{STDEV}_{\mathrm{MOCC}}} \tag{17}$$

由实验结果可知, MOCC 与 NSGA-II 相比, RNI 平均提升了 71.17 %, *D* 平均提升了 146.59 %, STDEV TTPT 平均提升了 295.53 %. 这组实验表明, 本方法产生的规划方案不仅在生产性能指标上较 NSGA-II 有所提升, 均衡性能也有较为明显的改善.

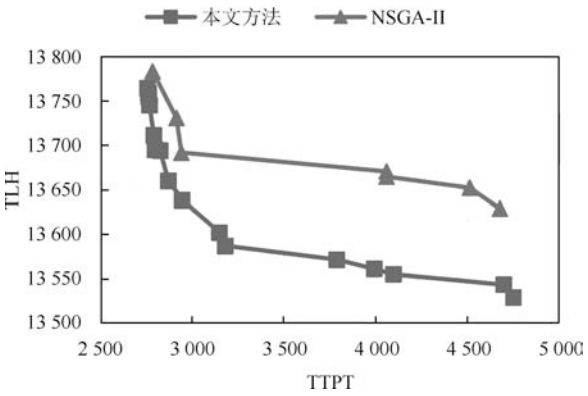

图 9　工人数量为 5 时, MOCC 与 NSGA-II 算法的非支配集间的比较

Fig. 9　The non-dominated solutions of MOCC and NSGA-II with 5 workers

分析实验结果, MOCC 较 NSGA-II 的优势主要有两个方面. 一方面, 协同机制的加入, 使得 MOCC 更能适应流水线向 *Seru* 生产系统转化问题的特点, 即两阶段子问题相互依赖、耦合度高, 避免了 NSGA-II 子问题单独进化时, 过度缩小搜索空间所导致解的多样性受到限制的问题. 另一方面, 由于引入了冗余码, MOCC 所产生的解不仅在均衡性上具有优势, 而且能够充分体现 *Seru* 生产系统的柔性, 避免了 *Seru* 在进行构建及调度时受到不必要约束, 从而提高了解的质量. 因此, 与传统方法相比, MOCC 在解决流水线向 *Seru* 生产系统转化的问题上具有一定优势.

4.3.2　与加入局部搜索的 NSGA-II 方法对比

加入局部搜索 (Local search) 的 NSGA-II 可以增强 NSGA-II 的寻优能力. Yu 等[19] 将局部搜索加入 NSGA-II, 以获得更佳的流水线向 *Seru* 系统转化方案. 本节将进行 MOCC 与加入局部搜索的 NSGA-II 的对比实验. 为了保证实验的公平性, 对比实验采用了与文献 [19] 相同的问题模型以及实验参数.

与本文中使用协同进化的算法结果进行比较, 其两个评价指标 TLH 和 TTPT 的帕累托前沿结果如图 10 所示. 其余对比结果如表 4 所示, MOCC 与加入局部搜索的 NSGA-II 之间的 Gap 值计算方法均参考公式 (18) ∼ (20):

$$\mathrm{Gap}_{\mathrm{RNI\_}AV} = \frac{Av\ \mathrm{RNI}_{\mathrm{MOCC}} - Av\ \mathrm{RNI}_{\mathrm{NSGA\text{-}II\_local\ search}}}{Av\ \mathrm{RNI}_{\mathrm{MOCC}}} \tag{18}$$

$$\mathrm{Gap}_{D\_AV} = \frac{Av\ D_{\mathrm{MOCC}} - Av\ D_{\mathrm{NSGA\text{-}II\_local\ search}}}{Av\ D_{\mathrm{MOCC}}} \tag{19}$$

$$\mathrm{Gap}_{\mathrm{STDEV}} = \frac{Av\ \mathrm{STDEV}_{\mathrm{MOCC}} - Av\ \mathrm{STDEV}_{\mathrm{NSGA\text{-}II\_\ local\ search}}}{Av\ \mathrm{STDEV}_{\mathrm{MOCC}}} \tag{20}$$

表 3 与 NSGA-II 方法的性能对比

Table 3 Comparison of proposed approach and NSGA-II

W	R	Proposed algorithm				NSGA-II				$\mathrm{Gap}_{\mathrm{RNI\_}AV}$ (%)	Gap_{D_AV} (%)	$\mathrm{Gap}_{\mathrm{STDEV}}$
		Av RNI	$Av\ D_{av}$	$Av\ D_{\max}$	STDEV TTPT	Av RNI	$Av\ D_{av}$	$Av\ D_{\max}$	STDEV TTPT			
5	20	0.66	0.04	0.09	134.91	0.37	0.14	0.40	390.95	78.38	281.42	189.79
10	43	0.45	0.12	0.68	330.33	0.35	0.14	0.69	1 101.28	25.71	20.61	233.39
15	62	0.40	0.03	0.22	552.93	0.27	0.06	0.45	2 518.01	48.71	142.42	355.39
20	57	0.65	0.02	0.13	1 672.87	0.23	0.06	0.38	5 443.23	176.02	263.64	225.38
25	31	0.64	0.11	0.26	1 551.91	0.49	0.18	0.31	7 613.63	31.52	73.58	390.60
30	39	0.69	0.09	0.28	1 286.32	0.41	0.19	0.27	6 156.87	66.68	97.87	378.64
Average										71.17	146.59	295.53

表 4 与加入 local search 的 NSGA-II 方法的性能对比

Table 4 Comparison of proposed approach and NSGA-II combining local search

W	R	Proposed algorithm				NSGA-II combining local search				$\mathrm{Gap}_{\mathrm{RNI\_}AV}$ (%)	Gap_{D_AV} (%)	$\mathrm{Gap}_{\mathrm{STDEV}}$
		Av RNI	$Av\ D_{av}$	$Av\ D_{\max}$	STDEV TTPT	Av RNI	$Av\ D_{av}$	$Av\ D_{\max}$	STDEV TTPT			
5	20	0.66	0.04	0.09	134.91	0.40	0.13	0.42	377.08	65.00	262.84	179.51
10	43	0.45	0.12	0.68	330.33	0.39	0.15	0.68	1 010.48	15.68	26.75	205.90
15	62	0.40	0.03	0.22	552.93	0.26	0.07	0.52	3 122.81	53.92	159.85	464.78
20	57	0.65	0.02	0.13	1 672.87	0.29	0.06	0.39	5 275.38	122.79	263.64	215.35
25	31	0.64	0.11	0.26	1 551.91	0.51	0.17	0.26	6 962.83	26.35	64.15	348.66
30	39	0.69	0.09	0.28	1 286.32	0.47	0.16	0.34	6 261.26	47.44	74.47	386.76
Average										55.20	141.95	300.16

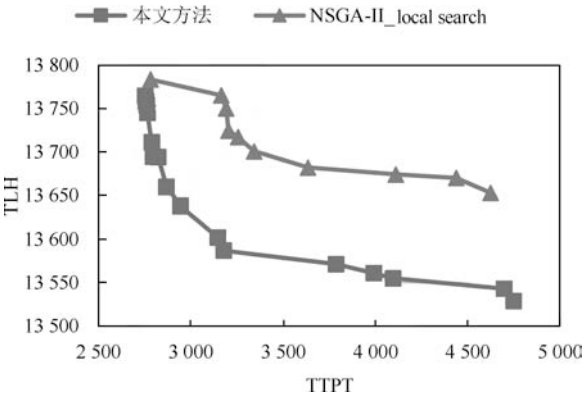

图 10 工人数量为 5 时, MOCC 与加入局部搜索的 NSGA-II 算法的非支配集间的比较

Fig. 10 The non-dominated solutions of MOCC and NSGA-II combining local search with 5 workers

由实验结果可知, MOCC 与加入局部搜索的 NSGA-II 方法相比, RNI 平均提升了 55.20 %, D 平均提升了 141.95 %, STDEV TTPT 平均提升了 300.16 %. 实验结果表明, 虽然加入了局部搜索使得 NSGA-II 的寻优性能有所提升, 产生的有效解数量较 NSGA-II 有一定程度上的增加, 但由于 *Seru* 系统转化问题的特点, 两个子问题解的协同性能也是影响完整解性能的一个重要因素, 因此本文加入了协同机制的方法依然体现出了能够同时考虑子问题间相互作用的优势. 在均衡性能上, 由于引入了冗余码, MOCC 的优势也较为明显.

4.4 在实际生产中的应用

为了进一步评估 MOCC 的性能, 本文构建了一个基于内蒙古第一机械集团 (内蒙一机) 实际生产环境的 *Seru* 生产系统, 并将 MOCC 应用到该系统的

生产调度中. 部分在实际生产中常用的经典的生产调度规则也在本节中与 MOCC 进行对比.

内蒙一机在传动装置的生产中, 操纵电控部件、液力变矩器、行星变速装置等小型零部件装配中已经初步实施了 *Seru* 生产系统. 本文将内蒙一机构建的一处 *Seru* 生产系统用于对比实验, 该系统包含有 10 个工人. 在本文构建的 *Seru* 生产系统在内蒙一机应用前, 内蒙一机的 *Seru* 调度是由调度人员根据经典的生产调度规则进行的, 例如, 先到先服务 (FCFS)、最短加工时间 (Shortest processing time, SPT)、最短劳动时间 (Shortest total labor hour, STLH)、最短等待时间 (Shortest waiting time, SWT). 内蒙一机的 *Seru* 生产系统中有两到三名调度员专门负责生产过程中的 *Seru* 调度. 虽然有上述调度规则的协助, 但由于 *Seru* 生产系统的复杂性, 人工调度的结果往往难以满足实际生产的需求, 使得 *Seru* 系统成为整个企业生产系统中的瓶颈.

本文采用了采集自内蒙一机 *Seru* 生产系统的实际生产数据, 将 MOCC 与上述 4 条经典的生产调度规则进行对比. 实验结果表明利用 MOCC 构建的 *Seru* 生产系统性能优于经典调度规则, RNI 平均提升了 79.65 %, D 平均提升了 274.61 %, STDEV TTPT 平均提升了 226.36 %.

由实验结果可以看出, MOCC 相较于经典调度规则, 规划结果更具有针对性, 因此具有更强的优化能力, 规划出的 *Seru* 生产系统也具有更优的均衡性能, 能够更好地发挥出 *Seru* 生产系统在实际生产中的潜力与优势.

5　结论

本文针对流水线向 *Seru* 生产系统转化问题的特点, 从实际应用的要求出发, 提出一种基于个体编码内部交换的 MOCC 算法, 并针对问题特点, 加入了种群间协同机制, 解决流水线向 *Seru* 生产系统转化问题. 由于所考虑的问题可分为 *Seru* 的构建和 *Seru* 的调度两个子问题, 并且子问题之间存在着交互作用, 所以该方法使用了两个子种群互相协作共同进化, 最后进化出协同效果较好的完整解. 通过这样的方式, 该方法可以在兼顾计算效率的前提下, 获得良好的优化能力. 实验结果表明, 与子问题单独进化相比, 协同进化能够有效扩大算法搜索范围, 充分发挥 *Seru* 生产系统的柔性与灵活性, 最终生成的规划方案也能够在实际生产中取得更好的应用效果. 与应用 NSGA-II 等传统算法求解该问题的方法相比, MOCC 不仅在生产指标 (如 TTPT 等) 上优于传统算法, 所产生规划方案的均衡性能也有较为突出的表现. 这种差异的原因在于, 通过子种群间的协同机制, 本方法同时考虑了 *Seru* 构建与 *Seru* 调度两个方面, 而传统方法往往将关注点置于 *Seru* 构建的规划上, 仅使用例如 FCFS 的启发式规则对 *Seru* 调度阶段进行规划, 虽然降低了问题的复杂性, 但也忽视了 *Seru* 调度阶段的重要性. 并且, 我们在编码方案中引入冗余码, 增加了解的多样性, 从而提高了算法的优化性能. 由此可见, MOCC 同时兼备计算效率和优化性能的优势, 可以在实际应用中高效地解决复杂性高, 产品批次大小波动较大的流水线向 *Seru* 生产系统转化问题.

附录 A

表 A1　工人 i 的多能工系数

Table A1　Worker i's coefficient of influencing level of doing multiple assembly task

工人	1	2	3	4	5
ε_i	0.18	0.19	0.2	0.21	0.2
工人	6	7	8	9	10
ε_i	0.2	0.2	0.22	0.19	0.19
工人	11	12	13	14	15
ε_i	0.18	0.23	0.24	0.22	0.16
工人	16	17	18	19	20
ε_i	0.24	0.18	0.18	0.21	0.18

表 A2　工人对不同类型产品熟练度数据的分布

Table A2　The data distribution of worker's level of skill for each product type

产品类型 1	2	3	4	5
N(1,0.05)	N(1.05, 0.05)	N(1.1, 0.05)	N(1.15, 0.05)	N(1.2, 0.05)

表 A3　工人对不同产品的熟练度

Table A3　The data of worker's level of skill

工人/产品	1	2	3	4	5
1	0.92	0.96	1.04	1.09	1.20
2	0.95	0.97	1.09	1.12	1.18
3	0.99	1.01	1.05	1.09	1.21
4	1.03	1.07	1.09	1.12	1.25
5	0.96	1.02	1.05	1.10	1.18
6	1.01	1.10	1.10	1.15	1.23
7	1.04	1.07	1.09	1.17	1.24
8	0.98	1.02	1.10	1.11	1.20
9	0.97	1.03	1.12	1.19	1.26
10	0.98	1.06	1.13	1.18	1.28
11	0.95	1.04	1.03	1.14	1.19
12	0.98	1.07	1.07	1.15	1.15
13	0.99	0.95	1.11	1.17	1.10
14	1.01	1.10	1.05	1.13	1.18
15	1.04	1.10	1.05	1.15	1.11
16	0.99	0.97	1.08	1.11	1.22
17	1.04	1.01	1.11	1.15	1.24
18	0.93	1.06	1.07	1.13	1.14
19	0.96	0.98	1.12	1.14	1.21
20	1.08	1.04	1.09	1.11	1.13

表 A4 30 批产品的信息数据

Table A4 The data of 30 batches

批次编号	产品类型	批次大小
1	3	46
2	5	68
3	3	45
4	4	19
5	1	36
6	4	45
7	1	62
8	2	30
9	2	60
10	3	67
11	2	9
12	4	24
13	3	38
14	4	32
15	5	52
16	5	48
17	1	68
18	4	71
19	2	46
20	5	25
21	1	26
22	3	52
23	4	46
24	5	44
25	2	32
26	3	75
27	1	33
28	4	103
29	2	74
30	3	53

References

1 盛朝迅, 黄汉权. 中美制造业成本比较及对策建议. 宏观经济管理, 2016, (9): 85−88

2 Stecke K E, Yin Y, Kaku I. Seru production: an extension of just-in-time approach for volatile business environments. *Analytical Approaches to Strategic Decision-Making: Interdisciplinary Considerations*, IGI Global, 2014. 45−58

3 Isa K, Tsuru T. Cell production and workplace innovation in Japan: toward a new model for Japanese manufacturing? *Industrial Relations: A Journal of Economy and Society*, 2002, **41**(4): 548−578

4 Manupati V K, Vemkata Deepthi T, Ramakotaiah K, Rao S S. Reconfiguration of networked seru production systems in an Indian perspective. In: Proceedings of the 2015 IEEE International Conference on Industrial Engineering and Operations Management. Dubai, United Arab Emirates: IEEE, 2015. 1−7

5 Yin Y, Kaku I, Stecke K E. The evolution of seru production systems throughout Canon. *Operations Management Education Review*, Scotland, UK: Neilson Journals Publishing, 2008. 27−40

6 Stecke K E, Yin Y, Kaku I, Murase Y. Seru: the organizational extension of JIT for a super-talent factory. *International Journal of Strategic Decision Sciences*, 2012, **3**(1): 106−119

7 De Treville S, Ketokivi M, Singhal V R. Competitive manufacturing in a high-cost environment. *Journal of Operations Management*, 2017, **49−51**: 1−88 DOI: 10.1016/j.jom.2017.02.001

8 刘晨光, 廉洁, 李文娟, 殷勇. 日本式单元化生产−生产方式在日本的最新发展形态. 管理评论, 2010, **22**(5): 93−103

9 Yin Y, Stecke K E, Swink M, Kaku I. Lessons from seru production on manufacturing competitively in a high cost environment. *Journal of Operations Management*, 2017, **49−51**: 67−76

10 Liu C G, Li W J, Lian J, Yin Y. Reconfiguration of assembly systems: from conveyor assembly line to serus. *Journal of Manufacturing Systems*, 2012, **31**(3): 312−325

11 Liu C G, Stecke K E, Lian J, Yin Y. An implementation framework for seru production. *International Transactions in Operational Research*, 2014, **21**(1): 1−19

12 Liu C G, Yang J, Lian J, Li W J, Evans S, Yin Y. Sustainable performance oriented operational decision-making of single machine systems with deterministic product arrival time. *Journal of Cleaner Production*, 2014, **85**: 318−330

13 Liu C G, Lian J, Yin Y, Li W J. Seru Seisa — an innovation of the production management mode in Japan. *Asian Journal of Technology Innovation*, 2010, **18**(2): 89−113

14 Liu C G, Dang F, Li W J, Evans S, Yin Y. Production planning of multi-stage multi-option seru production systems with sustainable measures. *Journal of Cleaner Production*, 2015, **105**: 285−299

15 Zhang X L, Liu C G, Li W J, Evans S, Yin Y. Effects of key enabling technologies for seru production on sustainable performance. *Omega*, 2017, **66**: 290−307

16 Luo L, Zhang Z, Yin Y. Seru loading with worker-operation assignment in single period. In: Proceedings of the 2016 IEEE International Conference on Industrial Engineering and Engineering Management. Bali, Indonesia: IEEE, 2016. 1055−1058

17 Yu Y, Tang J F, Sun W, Yin Y, Kaku I. Reducing worker (s) by converting assembly line into a pure cell system. *International Journal of Production Economics*, 2013, **145**(2): 799−806

18 Liu C G, Yang N, Li W J, Lian J, Evans S, Yin Y. Training and assignment of multi-skilled workers for implementing seru production systems. *The International Journal of Advanced Manufacturing Technology*, 2013, **69**(5−8): 937−959

19 Yu Y, Tang J, Sun W, Yin Y, Kaku I. Combining local search into non-dominated sorting for multi-objective line-cell conversion problem. *International Journal of Computer Integrated Manufacturing*, 2013, **26**(4): 316−326

20 Yu Y, Gong J, Tang J F, Yin Y, Kaku I. How to carry out assembly line-cell conversion? A discussion based on factor analysis of system performance improvements. *International Journal of Production Research*, 2012, **50**(18): 5259−5280

21 Yu Y, Tang J, Gong J, Yin Y, Kaku I. Mathematical analysis and solutions for multi-objective line-cell conversion problem. *European Journal of Operational Research*, 2014, **236**(2): 774−786

22 Kaku I, Gong J, Tang J F, Yin Y. A mathematical model for converting conveyor assembly line to cellular manufacturing. *Industrial Engineering & Management Systems*, 2008, **7**(2):160−170

23 Dimopoulos C, Zalzala A M S. Investigating the use of genetic programming for a classic one-machine scheduling problem. *Advances in Engineering Software*, 2001, **32**(6): 489−498

24 Nguyen S, Zhang M, Johnston M, Tan K C. Automatic programming via iterated local search for dynamic job shop scheduling. *IEEE Transactions on Cybernetics*, 2015, **45**(1): 1−14

25 Jia Ling-Yun, Li Dong-Ni, Tian Yun-Na. An intercell scheduling approach using shuffled frog leaping algorithm and genetic programming. *Acta Automatica Sinica*, 2014, **40**(5): 936−948
(贾凌云, 李冬妮, 田云娜. 基于混合蛙跳和遗传规划的跨单元调度方法. 自动化学报, 2014, **40**(5): 936−948)

26 Tian Yun-Na, Li Dong-Ni, Liu Zhao-He, Zheng Dan. A Hyper-heuristic Approach with Dynamic Decision Blocks for Inter-cell Scheduling. *Acta Automatica Sinica*, 2016, **42**(4): 524−534
(田云娜, 李冬妮, 刘兆赫, 郑丹. 一种基于动态决策块的超启发式跨单元调度方法. 自动化学报, 2016, **42**(4): 524−534)

27 Cheng T C E. Integration of priority dispatching and due-date assignment in a job shop. *International Journal of Systems Science*, 2007, **19**(9): 1813−1825

28 Baker Kenneth R. Sequencing rules and due-date assignments in a job shop. *Management Science*, 1984, **30**(9): 1093−1104

29 Miyazaki S. Combined scheduling system for reducing job tardiness in a job shop. *International Journal of Production Research*, 1981, **19**(2): 201−211

30 Cochran J K, Horng S M, Fowler J W. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers & Operations Research*, 2003, **30**(7): 1087−1102

31 Goh C K, Tan K C. A Competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 2009, **13**(1): 103−127

32 Tan K C, Lee T H, Khor E F. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 2001, **5**(6): 565−588

33 Ulungu E L, Teghem J, Ost C. Efficiency of interactive multi-objective simulated annealing through a case study. *Journal of the Operational Research Society*, 1998, **49**(10): 1044−1050

34 Czyzżk P, Jaszkiewicz A. Pareto simulated annealing — a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 1998, **7**(1): 34−47

35 Wang X D, Hirsch C, Kang S, Lacor C. Multi-objective optimization of turbomachinery using improved NSGA-II and approximation model. *Computer Methods in Applied Mechanics and Engineering*, 2011, **200**(9−12): 883−895

吴旭辉 北京理工大学计算机学院硕士研究生. 主要研究方向为演化计算和生产调度. E-mail: harry_wxh@163.com
(**WU Xu-Hui** Master student at the School of Computer Science, Beijing Institute of Technology. His research interest covers evolutionary computation and production scheduling.)

杜劭峰 特种车辆及其传动系统智能制造国家重点实验室主任. 主要研究方向为数字化与智能制造.
(**DU Shao-Feng** Directory of State Key Laboratory of Smart Manufacturing for Special Vehicles and Transmission System. His research interest covers digitization and smart manufacturing.)

郝慧慧 特种车辆及其传动系统智能制造国家重点实验室成员. 主要研究方向为数字化仿真技术.
(**HAO Hui-Hui** Member of State Key Laboratory of Smart Manufacturing for Special Vehicles and Transmission System. Her main research interest is digital simulation.)

于 洋 东北大学信息科学与工程学院副教授. 主要研究方向为工业工程, 绿色物流. E-mail: yuyang@ise.neu.edu.cn
(**YU Yang** Associate professor at the School of Information Science and Engineering, Northeastern University. His research interest covers green logistics and industrial engineering.)

殷 勇 同志社大学大学院商学研究科教授. 主要研究方向为 *Seru* 制造与工业 4.0.
E-mail: yyin@mail.doshisha.ac.jp
(**YIN Yong** Professor at the Graduate School of Business, Doshisha University. His research interest covers *Seru* production and Industry 4.0.)

李冬妮 北京理工大学计算机学院副教授. 主要研究方向为智能优化, 企业计算, 物流管理. 本文通信作者.
E-mail: ldn@bit.edu.cn
(**LI Dong-Ni** Associate professor at the School of Computer Science, Beijing Institute of Technology. Her research interest covers intelligent optimization, enterprise computation, and logistics management. Corresponding author of this paper.)