

DOI: 10.3901/JME.2021.19.192

# 协同进化交叉熵算法求解浇次不固定的炼钢连铸调度问题\*

吕阳<sup>1,2</sup> 钱斌<sup>1,2</sup> 胡蓉<sup>1,2</sup> 张梓琪<sup>1</sup>

(1. 昆明理工大学信息工程与自动化学院 昆明 650500;  
2. 昆明理工大学云南省人工智能重点实验室 昆明 650500)

**摘要:** 浇次不固定的炼钢连铸调度问题(Cast uncertain steelmaking continuous casting scheduling problem, CU\_SCCSP)广泛存在于钢铁生产行业中。该问题对应炼铁、精炼和连铸三个连续生产阶段,其中炼铁和精炼阶段为带运输时间的混合流水线调度子问题,连铸阶段为带独立设置时间的复杂并行机调度子问题,且两个子问题相互耦合。针对该问题,建立优化目标为最小化最大完工时间和平均等待时间加权之和的排序模型,并提出一种协同进化交叉熵算法(Co-evolution cross-entropy optimization algorithm, CCOA)进行求解。设计前后子问题两段式编码和双向解码的策略,并采用启发式规则和随机方式初始化种群,以确保初始解的质量和分散性。在算法全局搜索阶段,采用分别对应前后子问题的双概率分布协同学习和积累优质解信息,并在采样概率分布生成新个体时引入考虑子问题耦合的模糊关系矩阵对概率分布取值进行适当调整,以增强算法较快到达优质解区域的能力,同时设计种群分裂机制来提高算法的引导性并扩大搜索范围。为提高算法的局部搜索能力,对分裂后的双种群中个体执行基于 interchange 和 insert 邻域操作的协同搜索,进而对当前历史最优解执行结合 SWAP 邻域快速评价的变邻域搜索,可增加算法在解空间中多个优质区域的搜索深度。仿真试验和算法比较验证了所提算法的有效性。

**关键词:** 炼钢连铸; 交叉熵方法; 协同进化; 快速评价方法; 模糊控制

**中图分类号:** TG156

## Co-evolution Cross-entropy Optimization Algorithm for Cast Uncertain Steelmaking-continuous Casting Scheduling

LÜ Yang<sup>1,2</sup> QIAN Bin<sup>1,2</sup> HU Rong<sup>1,2</sup> ZHANG Ziqi<sup>1</sup>

(1. Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500;

2. Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming 650500)

**Abstract:** The cast uncertain steelmaking continuous casting scheduling problem (CU\_SCCSP) widely exists in the steel industry. This problem corresponds to three continuous production stages, i.e., ironmaking stage, refining stage and continuous casting stage. The ironmaking and refining stages can be modeled as a hybrid flow shop scheduling subproblem with transportation times, and the continuous casting stage can be regarded as a complex parallel machine scheduling subproblem with independent setting times. These two sub-problems are coupled with each other. To solve CU\_SCCSP, a permutation-based model is established, and a co-evolution cross-entropy optimization algorithm (CCOA) is proposed to minimize the objective of weighted sum of the maximum completion time and the average waiting time. A two-stage encoding strategy and a bidirectional decoding strategy for the subproblems are designed, and a heuristic rule and a random method are adopted to initialize the population to ensure the quality and dispersion of the initial solutions. In the global search phase of CCOA, two probability matrices corresponding to the former and the latter subproblems

\* 国家自然科学基金资助项目(61963022, 51665025)。20201014 收到初稿,  
20210324 收到修改稿

are used to collaboratively learn and accumulate the information of high-quality solutions or individuals. Moreover, before generating new individuals by sampling the probability matrices, a fuzzy relation matrix considering the coupling between two subproblems is proposed to adjust the values in the probability matrices appropriately to enhance the ability of CCOA to reach the high-quality solution regions quickly. Meanwhile, a population splitting mechanism is designed to enhance CCOA's guiding ability and expand its search range. In order to improve the local search ability of CCOA, a cooperative search based on interchange neighborhood operation and insert neighborhood operation is executed on each individual in two splitting populations, and then a variable neighborhood search (VNS) method combined with the speedup evaluation of the swap neighborhood is performed on the current historical optimal solution, which can increase the search depth of the algorithm in multiple high-quality regions in solution space. Simulation experiments and algorithm comparisons verify the effectiveness of the proposed CCOA.

**Key words:** steelmaking-continuous casting; cross-entropy method; co-evolution; rapid evaluation method; fuzzy control

## 0 前言

钢铁行业是国民经济的支柱产业,对其生产过程的优化具有重要现实意义<sup>[1]</sup>。炼钢连铸调度问题(Steelmaking continuous casting scheduling problem, SCCSP)是钢铁行业中非常重要的一类问题,研究其有效调度算法以实现对生产资源(设备、原材料等)的合理分配,是提高相关企业生产效率和核心竞争力的重要途径,也是促进国民经济快速发展的关键手段<sup>[2]</sup>。

SCCSP 是一类将混合流水车间中的装料调度问题与并行机中的浇铸调度问题结合在一起的集成调度问题。该问题可以描述为炼铁、精炼和连铸三个连续生产阶段,每个阶段都存在多台相同并行机。其中,炼铁和精炼阶段为带运输时间的混合流水线调度(Hybrid flowshop scheduling, HFS)<sup>[2-4]</sup>子问题,“炉次”是该阶段的基本单元;连铸阶段为带独立设置时间的复杂并行机调度子问题,“浇次”该阶段的基本单元。为避免前一阶段“炉次”中钢水温度降低而需在后一阶段“浇次”中进行升温处理,需尽量确保前一子问题的炉次序列(该子问题的解)中靠前炉次的钢水在后一子问题的浇次序列(该子问题的解)中靠前加工,即这两个子问题间存在耦合。这也说明 SCCSP 的优质解应为其前后子问题的两个耦合度较高的解组成。总体来说,SCCSP 可分为两类:(1)浇次顺序和加工时间表确定的炼钢连铸调度问题(Cast certain steelmaking continuous casting scheduling problem, CC\_SCCSP);(2)浇次排序和加工时间表未确定的炼钢连铸调度问题(Cast uncertain steelmaking continuous casting scheduling problem, CU\_SCCSP)。相对于 CC\_SCCSP, CU\_SCCSP 更符合实际且更加复杂。由于 CU\_SCCSP 包含混合流水线调度问题和复杂并行机调度问题,而一般流水线调度问题已被证明

为 NP-hard<sup>[5]</sup>,故根据计算复杂性中的归约原理,可知 CU\_SCCSP 也属于 NP-hard 问题,其求解难度随问题规模的增大呈指数增长<sup>[6]</sup>。因此,本文针对 CU\_SCCSP 进行有效求解研究,具有重要的现实和理论意义。

SCCSP 的研究始于 1954 年 Johnson 对两台机器流水作业问题(Flow shop problem, FSP)优化算法的研究。起初主要针对小规模简单调度问题,之后随着问题规模的逐步扩大以及人们对算法研究的深入,调度理论及优化方法逐渐成熟,在实际问题中的应用逐步成为研究人员关注的焦点。LEE<sup>[3]</sup>和 TANG<sup>[4]</sup>是较早研究钢铁生产调度问题的学者。TANG 在其综述性文献中对钢铁企业生产及管理中的问题进行了分析和比较,并对生产计划与调度系统的开发及调度问题求解方法进行了总结。随着精确算法的发展,部分学者采用数学规划或动态规划等精确方法以获得问题的全局最优解。在 CC\_SCCSP 方面, XUAN 等<sup>[7]</sup>以最小化最大完工时间为优化目标,建立了整数规划模型,并提出了基于拉格朗日松弛算法求解 CC\_SCCSP 的批次解耦算法。MAO 等<sup>[8]</sup>以最小化提前/推迟完工时间和等待时间之和为优化目标,提出了拉格朗日松弛方法求解 CC\_SCCSP,同时放松了机器容量约束。LIANG 等<sup>[9]</sup>以最短的总装料等待时间和准时交货为优化目标,建立了混合整数非线性数学模型,通过增强的拉格朗日松弛算法解决 CU\_SCCSP。在 CU\_SCCSP 方面, CUI 等<sup>[10]</sup>以在合理的计算时间内获得近似最优解为优化目标,建立了混合整数非线性数学模型,通过松弛一些复杂的约束来解决该 CU\_SCCSP,并引入了一种改进的条件替代子梯度算法来解决拉格朗日对偶(LD)问题。CUI 等<sup>[11]</sup>针对 CU\_SCCSP 调度问题建立了数学模型,通过放宽机器容量约束,提出了一种 DC(Difference of convex functions)算法进行求解。

上述精确算法以线性代数和几何分析为基本工具,利用问题优化目标函数和约束式的结构信息构造搜索,可在几分钟至几十分钟内获取较小规模问题(炉次小于等于 30)的最优解。但是,SCCSP 具有非凸特性,其几何结构和最优解间的关系仍属开放问题,不存在多项式时间获取最优解的算法,故对应的运筹学算法均是靠遍历或部分遍历解空间来确保求解质量,这使其需要数小时至数天才可获得较大规模问题(炉次数在 100 至 300 之间)的最优解或满意解。因此一些学者采用启发式方法或智能优化算法以求在较短时间内获得问题的近似最优解。在 CC\_SCCSP 方面,ZHU 等<sup>[12]</sup>以提高生产调度的效率和性能为优化目标,提出了一种优化模型,该模型与并行反向推理算法和遗传算法相结合用于求解 CC\_SCCSP。PENG 等<sup>[13]</sup>以最大最小完工时间和平均等待时间为优化目标,提出了一种改进的人工蜂群算法求解 CC\_SCCSP,采用启发式方法对初始解进行优化,并采用两种变邻域搜索算法来生成新的高质量的解。在 CU\_SCCSP 方面,TANG 等<sup>[14]</sup>以优化生产过程中各个生产阶段的正在生产和将要生产的任务为目标,采用实数矩阵编码方式,提出了一种改进的差分演化算法对 CU\_SCCSP 进行求解。PAN 等<sup>[15]</sup>提出了一个人工蜂群算法求解多阶段的 HFS 型炼钢连铸生产调度问题,其算法融入了启发式规则、领域生成方法和改进策略,使得算法更为有效。ZENG 等<sup>[16]</sup>以最优解的质量和收敛速度为目标,提出了一种改进鲸鱼算法(WSA),并引入阈值参数来改进了 WSA 的迭代规则。上述智能算法均表现出良好性能。这表明采用智能算法是求解 SCCSP 是合理且必要的。

在上述文献中,对 CU\_SCCSP 的研究大多对问题进行整体编码,并采用智能算法或者启发式算法进行求解,但由于 CU\_SCCSP 比 CC\_SCCSP 具有更多的决策变量、约束条件,从而使得解空间扩大且编码解码较为复杂。这时仅靠智能算法本身难以有效引导算法至优质解区域搜索,容易导致算法实际搜索效率偏低。因此,也有部分学者将 CU\_SCCSP 分解为两个子问题,然后针对子问题分别进行编码和求解,从而将算法搜索限定于解空间中部分较优区域内,然而,已有的分解方法基本没有考虑子问题之间存在的耦合性,导致算法在搜索过程中难以避免对大量非耦合解进行无效搜索,从而导致算法的实际搜索效率不高。因此,针对具有子问题耦合特性的 CU\_SCCSP,设计有效智能求解算法具有重要意义。

交叉熵 (Cross-entropy, CE) 算法是 RUBINSTEIN 等<sup>[17]</sup>将信息熵中的交叉熵概念应用于小概率事件的仿真中,结合重要度采样方法用于评估复杂随机网络中小概率事件发生的概率,其主要思想是将优化问题与其相关的概率估计问题相关联,通过不断更新概率分布来寻找最优解。此进化方式可一定程度上避免传统进化算法中交叉、变异等操作所带来的模式破坏问题,从而使其搜索具有更好的引导性<sup>[18]</sup>。已有部分学者对 CE 算法进行研究并将其应用到组合优化问题的求解中。SHE 等<sup>[19]</sup>以同时最小化模糊最大完工时间和模糊总能耗为优化目标,建立模糊分布式装配流水线低碳调度问题模型,并提出了一种混合交叉熵算法进行求解,QIAN 等<sup>[20]</sup>以同时最小化模糊最大完工时间和模糊总能耗为优化目标,建立模糊分布式流水线绿色调度问题的模型,进而提出一种超启发式交叉熵算法进行求解。CASERTA 等<sup>[21]</sup>以最大载重为优化目标,通过交叉熵最小原理建立分布参数的更新规则,并结合元启发式算法进行了求解,经过大规模实例的对比试验其性能优于分支定界法<sup>[22]</sup>。CE 算法通过更新概率分布引导搜索方向,很大程度能避免盲目搜索,但同时也存在解的分布较差,搜索效率较低,且容易较早陷入局部最优。因此有研究学者在标准 CE 算法基础上,结合 SCCSP 特点,提出了改进的 CE 算法。WANG 等<sup>[23]</sup>以分时电价环境下电力成本最小化为优化目标,建立概率分布模型,提出了双层串级 CE 算法,用于求解带有分时电价的炼钢连铸模型,通过不断迭代更新来优化工件在每个阶段的加工过程参数,试验表明,串级 CE 算法在求解此类问题时,取得了较好的效果。YANG 等<sup>[24]</sup>以最小开浇费用和生产过程惩罚费用为优化目标,建立浇次计划的概率分布模型,并提出了一种结合启发式规则的改进 CE 算法求解 SCCSP 中的浇次调度计划。现有研究表明,对于 CU\_SCCSP,尚未有采用 CE 算法求解此类问题。

与大多数智能调度算法一样,已有交叉熵算法在求解 SCCSP 或部分具有耦合性的问题时,没有考虑问题之间存在的耦合性<sup>[25]</sup>(例如文献[15]、[23]),导致算法在求解此类问题时的有效性降低,全局搜索能力受到限制,此外,在进行局部搜索时也只是执行简单的重复邻域操作(例如文献[13]),从而导致局部搜索的实际效率不高<sup>[26]</sup>。为了解决这一问题,本文提出一种结合模糊关系矩阵<sup>[27]</sup>的协同进化交叉熵算法(Co-evolution cross-entropy optimization algorithm, CCOA),CCOA 在传统 CE 算法的基础

上通过对其中一个子问题采样生成的解进行处理,并结合隶属度函数构造模糊关系矩阵来对另一个子问题的采样概率分布进行调整,从而较充分考虑两个子问题解之间的耦合性。此外,在算法局部搜索阶段设计了两条可对邻域操作进行有效性进行快速判定的定理,用以提升算法的局部搜索效率。仿真试验和算法比较验证了所提 CCOA 是求解 CU\_SCCSP 的有效算法。

## 1 问题描述

### 1.1 CU\_SCCSP 介绍

在钢铁生产过程中,高炉产生的符合一定条件的铁水和废钢被送到指定转炉进行加工,按批次进入转炉(Linz-Donawitz-Verfahren converter, LD)的高温铁水在转炉中经过快速加温,把铁水和废钢加工成均匀的液态钢水,并经过快速脱碳、供氧转换、脱磷等工序,使其钢水的碳含量达到期望的比例。转炉产生出的钢水由台车运送到精炼炉(Ruhrstahl Heraeus, RH)进行精炼加工,精炼设备为钢水炉次提供脱碳、脱硫、去除杂质、排渣和添加必要成分等功能。在实际生产过程中,根据产品不同指标要求,可以对产品进行多次加工。因此,产品的精炼过程包含多个阶段。连铸又称作浇铸,精炼后的钢水被运载到连铸机(Continuous caster, CC)上固化,冷却,拉流,切割,使其转变成各种尺寸的板坯,炼钢连铸工艺图如图 1 所示。生产阶段需要把炉次或浇次按照算法求得的加工顺序,在遵循必要约束关系的条件下分配到指定的机器上。一个炉次的钢水正好是一台转炉中的钢水,在同一台连铸机上连续浇铸的炉次的有序集合成为一个浇次,每个浇次中所包含的炉次在连铸阶段的顺序是提前确定的,为了避免由于长时间等待导致炉次的温度过低,需要二次加热所带来的成本损耗,每个浇次里的炉次要连续加工。在整个生产过程中,钢水温度是重点控制的工艺参数,为了减少或避免高温钢水在各工序之间的温度下降,生产过程中应尽量减少钢水在工序之间的等待时间,从而防止因钢水温度损失而需要升温处理所带来的资源消耗、生产效率降低等问题。因此,寻找高效的调度方法,在保证尽早完工的前提下,减少钢水在工序之间的等待时间,降低钢铁生产所带来的能耗,提高设备利用率,从而提高企业生产效率,是 SCCSP 研究的重点。

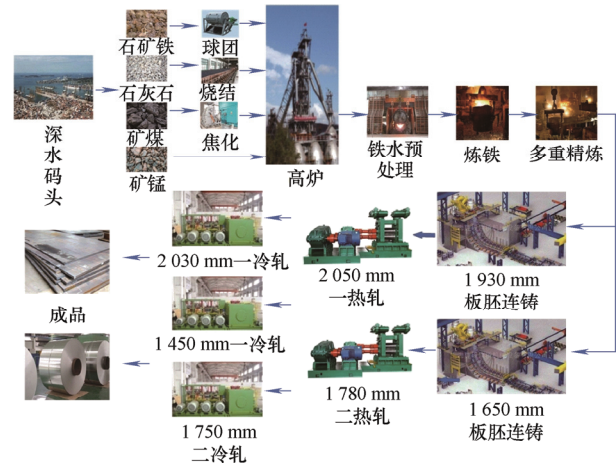


图 1 炼钢连铸工艺流程图

### 1.2 符号定义表

本文涉及符号含义如表 1 所示。

表 1 符号定义表

符号	说明
$s$	阶段下标, $s = 1, 2, \dots, S$ , $S$ 为阶段总数
$z$	浇次下标, $z = 1, 2, \dots, Z$ , $Z$ 为浇次总数
$k$	炉次下标, $k = 1, 2, \dots, N$ , $N$ 为炉次总数
$u$	炉次集合, $u = \{\pi_1, \pi_2, \dots, \pi_N\}$
$v$	浇次集合, $v = \{\pi_1^z, \pi_2^z, \dots, \pi_Z^z\}$
$\pi_i^s$	阶段 $s$ 炉次序列中的第 $i$ 个炉次
$\pi_i^z$	浇次序列中的第 $i$ 个浇次
$\Omega^z$	$\pi_i^z$ 中的炉次集合 $\Omega^z = \{l_{i-1} + 1, l_{i-1} + 2, \dots, l_i\}, l_0 = 0$
$\alpha_i$	浇次 $i$ 中所含炉次的数量, $\alpha_i = \ \pi_i^z\ $
$m_s$	阶段 $s$ 所包含的机器数
$M$	整个生产过程所包含的机器总数, $M = \sum_{s=1}^S m_s$
$t_s$	钢水从 $s-1$ 阶段运送到 $s$ 阶段所需要的时间
$ST$	每个浇次的独立设置时间
$T_s(i)$	第 $s$ 阶段的第 $i$ 台设备上加工炉次或者浇次总数
$\Omega^*$	第 $s$ 阶段的第 $i$ 台设备上工件的加工顺序集合, $\Omega^* = \{\pi_1^s, \pi_2^s, \dots, \pi_{T_s(i)}^s\}$
$D$	第 $s$ 阶段的第 $a$ 台设备上, 从浇次 $i$ 开始加工到浇次 $j$ 结束加工所持续的时间
$p$	工件加工所需时间
$L$	工件开始加工时间
$C$	工件的完工时间
$f_m$	最大最小完工时间
$f_w$	平均等待时间
$\psi_1$	最大最小完工时间的权重系数
$\psi_2$	平均等待时间权重系数

### 1.3 模型建立

CU\_SCCSP 描述如下: 炼铁和精炼阶段,  $N$  个

炉次需要完成  $S-1$  道工序, 每道工序存在  $m_s$  台独立并行机, 每个炉次在完成一道工序后需要经过  $t_s$  的时间运送到下一道工序加工。连铸阶段,  $N$  个炉次根据生产计划组成  $Z$  个浇次, 每个浇次只需要完成一道工序, 加工每个浇次之前需要经过设置时间, 所需的设置时间与浇次相关。且上述过程需要满足: 任何时刻每个浇次(炉次)都只能在一台机器上加工, 每台机器只能同时加工一个浇次(炉次), 连铸阶段要求每个浇次连续加工, 不能出现浇断。

$p(\pi_i^s)$  为  $\pi_i^s$  的加工时间 ( $p(\pi_0^s)=0$ ),  $p(\pi_i^z)=\sum_{j=1}^{\alpha_i} p(l_{i-1}+j)$  为  $\pi_i^z$  的加工时间, 其值等于它所包含的多个炉次在连铸阶段加工时间之和。 $L(\pi_i^{s(a)})$  为  $\pi_i^{s(a)}$  在第  $s$  阶段的第  $a$  台设备上的开始加工时间( $i=1,2,\dots,T_s(a)$ ,  $L(\pi_i^{0(a)})=0$ )。 $C(\pi_i^{s(a)})$  为工件  $\pi_i^{s(a)}$  在第  $s$  阶段的第  $a$  台设备上的完工时间。

CU\_SCCSP 的排序模型可表示如下

$$\min f = \psi_1 f_m + \psi_2 f_w \quad (1)$$

$$f_m = C \max(\pi) = \max_{a=1,2,\dots,m_s} (\sum_{i=1}^{T_s(a)} C(\pi_i^{s(a)})) \quad (2)$$

$$f_w = \sum_{i=1}^N (L(\pi_i^z) - L(\pi_i^1) - p(\pi_i^1)) / N \quad (3)$$

$$C(\pi_i^{l(a)}) = \sum_{j=1}^i p(\pi_j^{l(a)}), a=1,2,\dots,m_s, i=1,2,\dots,T_l(a) \quad (4)$$

$$C(\pi_i^{s(a)}) = \max(C(\pi_i^{s-1}) + t_s, C(\pi_{i-1}^{s(a)})) + p(\pi_i^{s(a)}) \quad (5)$$

$$s=2,\dots,S, a=1,2,\dots,m_s, i=1,2,\dots,T_s(a)$$

$$\Delta = C(\pi_{l_{i-1}+j+1}^{S(a)}) - p(\pi_{l_{i-1}+j+1}^{S(a)}) - C(\pi_{l_{i-1}+j}^{S(a)})$$

$$j=1,2,\dots,\alpha_i, a=1,2,\dots,m_s, i=1,2,\dots,Z \quad (6)$$

如果  $\Delta \leq 0$ , 则

$$C(\pi_{l_{i-1}+j+1}^S) = p(\pi_{l_{i-1}+j+1}^S) + C(\pi_{l_{i-1}+j}^S) \quad (7)$$

如果  $\Delta > 0$ , 则

$$C(\pi_{l_{i-1}+j}^S) = C(\pi_{l_{i-1}+j+1}^S) - p(\pi_{l_{i-1}+j+1}^S) \quad (8)$$

$$C(\pi_i^z) = \max\{C(\pi_{l_{i-1}+1}^S) - p(\pi_{l_{i-1}+1}^S),$$

$$C(\pi_{l_{i-1}}^z) + ST(\pi_i^z)\} + p(\pi_i^z) \quad (9)$$

其中, 式(1)表示 CU\_SCCSP 的加权目标函数, 通过给指标赋予不同的权重, 从而把多目标问题转化为最小化单目标问题, 考虑到实际生产中要求尽早完成生产任务, 本文选择  $\psi_1=10$ ,  $\psi_2=1$  (权重指标可以根据实际生产需求适当调整)。式(2)表示最小化最大完工时间。式(3)表示平均等待时间, 式(4)和式

(5)表示正向调度过程每个工件完工时间的计算公式。式(6)至(9)表示逆向调度过程消除连铸阶段浇次内的存在的浇断公式。

#### 1.4 样例分析

本节给出一个三阶段样例, 其中包括两台转炉 1#LD 和 2#LD、两台精炼炉 1#RH 和 2#RH、两台连铸机 1#CC 和 2#CC。有三个浇次需要加工, 浇次 1 包含炉次 1、炉次 2, 浇次 2 包含炉次 3、炉次 4、炉次 5, 浇次 3 包含炉次 6、炉次 7。运输时间  $t_1=13$ ,  $t_2=10$ ,  $t_3=14$ , 设置时间  $ST_1=26$ ,  $ST_2=28$ ,  $ST_3=31$ 。每个炉次在各阶段的加工时间由下面的矩阵给出

$$[p_{s \times k}]_{3 \times 7} = \begin{pmatrix} 47 & 44 & 45 & 36 & 48 & 40 & 47 \\ 36 & 42 & 44 & 50 & 41 & 47 & 47 \\ 47 & 44 & 37 & 47 & 38 & 47 & 50 \end{pmatrix}$$

甘特图示例如图 2 所示, 此时连铸阶段浇次 2 中炉次 3 和炉次 4 之间存在浇断, 需要通过逆向调度消除浇断。消除浇断后的甘特图如图 3 所示, 此时最大完工时间为 326, 平均等待时间为 99, 总的目标值为 3 359。

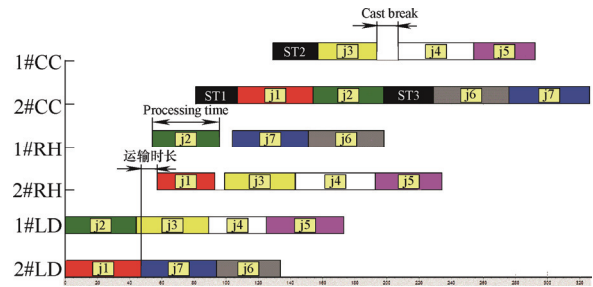


图 2 存在浇断的甘特图图例

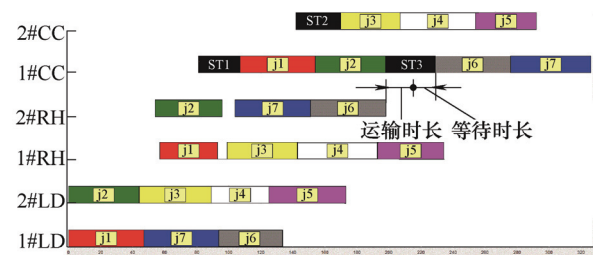


图 3 消除浇断的甘特图图例

## 2 协同进化交叉熵算法(CCOA)

本节提出了一种用于求解 CU\_SCCSP 的改进 CE 算法, 即 CCOA, 并设计对应炉次子问题和浇次子问题两段式编码和双向解码的策略。在 CCOA 的全局搜索阶段, 采用分别对应前后子问题的双概率



分布协同学习和积累优质解信息,并在采样概率分布生成新个体时引入考虑子问题耦合的模糊关系矩阵对概率分布取值进行适当调整,同时设计种群分裂机制来提高算法的引导性并扩大搜索范围。然后,在 CCOA 的局部搜索阶段,对分裂后的双种群中个体执行基于 interchange 和 insert 邻域操作的协同搜索,进而对当前历史最优解执行结合 swap 邻域快速评价的变邻域搜索。

## 2.1 编码与解码

### 2.1.1 两段式编码

本文采用两段式编码,前半部分为炉次序列(即前一子问题的解),后半部分为浇次序列(即后一子问题的解),种群中的第  $i$  个个体可表示为  $I_i = (u_i, v_i)$ 。

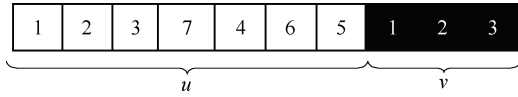


图 4 两段式编码

### 2.1.2 双向解码策略

为了在搜索过程中更好地对解进行更新,本文提出了一种双向解码策略。

在前向解码过程中,根据炼铁和精炼阶段炉次的加工顺序,按照排序模型,依次把每个炉次分配到第一台可以使用的机器上进行加工。连铸阶段需要把每个炉次根据约束关系分配到第一台可利用的连铸机上,且要保证同一个浇次里的炉次必须在同一台机器上加工。

在反向解码时,首先,从连铸阶段最后一个浇次的最后一个炉次开始,如果同一个浇次里后一个炉次与前一个炉次之间有空闲时间,则需要把前面一个炉次进行右移,消除浇断。然后,固定连铸阶段的炉次,在不违反约束的前提下,依次把每个阶段的炉次向右移,从而达到减少工件在加工期间的等待时间,优化目标值。

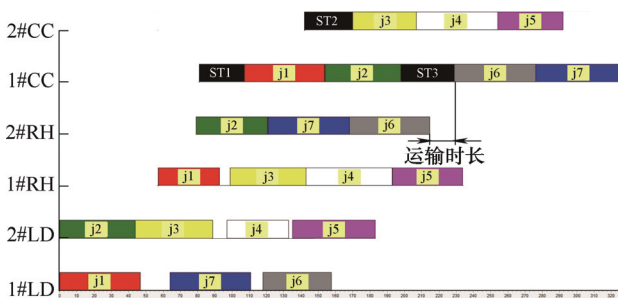


图 5 右移后的甘特图

例 1.4 经过反向解码策略平移后得到的甘特图

如图 5 所示,与图 3 相比减少了等待时间。此时最大完工时间为 326,平均等待时间为 90,总的目标值为 3 350。可以看出,平移后总目标值减少了 9。

## 2.2 种群初始化规则

针对问题特点,设计启发式规则如下。

(1) 根据 LPT 规则,将每个浇次的加工时间从小到大排列,从而得到浇次工序。

(2) 将得到的浇次序列进行逆向调度,根据连铸阶段每个炉次的开始加工时间按递增进行顺序,从而得到炉次加工序列。

(3) 随机生成剩余的个体的浇次序列,按照(2)中相同的方法生成炉次序列。

上述启发式规则生成解的过程中,既包含了利用 LPT 规则生成的高质量解,同时也包含了随机生成的分散解。

## 2.3 结合模糊关系矩阵的双向概率分布更新机制

### 2.3.1 概率分布的初始化规则

为了保证算法开始对解空间均匀搜索,概率分布参数采用均匀分布进行初始化,如下所示:

$$P_{\text{charge}}^0 = \begin{pmatrix} 1/N & 1/N & \cdots & 1/N \\ 1/N & 1/N & \cdots & 1/N \\ \vdots & \vdots & \ddots & \vdots \\ 1/N & 1/N & \cdots & 1/N \end{pmatrix}$$

$$P_{\text{cast}}^0 = \begin{pmatrix} 1/Z & 1/Z & \cdots & 1/Z \\ 1/Z & 1/Z & \cdots & 1/Z \\ \vdots & \vdots & \ddots & \vdots \\ 1/Z & 1/Z & \cdots & 1/Z \end{pmatrix}$$

### 2.3.2 CE 算法及概率分布采样机制

CE 是一种基于蒙特卡罗方法和重要性采样技术来解决复杂模拟和优化问题的有效算法,通过统计学习和更新,使最优解出现的概率逐渐增大,最终趋于稳定。本文采用文献[28]所设计的概率模型和更新方式,对于  $PS \times \rho$  个有效样本工序进行统计,其中  $PS$  为样本总数,概率模型  $P_{ij}$  可如下给出

$$P_{ij} = \frac{E_P I_{\{S(X) \leq \gamma\}} I_{\{x_i = j\}}}{E_P I_{\{S(X) \leq \gamma\}}} = \frac{\sum_{k=1}^m I_{\{S(X_k) \leq \gamma\}} I_{\{x_{ki} = j\}}}{\sum_{k=1}^m I_{\{S(X_k) \leq \gamma\}}} \quad (10)$$

其中,  $\sum_{k=1}^m I_{\{S(X_k) \geq \gamma\}}$  表示  $PS \times \rho$  个样本中目标值

优于  $\gamma$  的样本总数,示性函数  $I_{\{x_{ki} = j\}}$  表示当前样本  $k$  中工件编码  $i$  位是否选择工件  $j$ 。

算法通过采样概率分布来生成新个体。首先以轮盘赌方式采样概率模型的每一列,从前到后确定

工序排列矢量的每一位,其中工件号*i*出现在位置*j*的概率 $P_{ij}$ 。若*i*已经出现了,表明工件*i*的加工顺序已经确定,于是将概率模型的第*i*行置零,并将剩余列归一化。采用上述方式,采样浇次概率模型 $P_{\text{cast}}$ 和炉次概率模型 $P_{\text{charge}}$ 产生解序列,进而通过解码即可获得相应的调度解。通过上述采样产生 $PS$ 个新个体,便构成了基础种群。

以2个浇次,5个炉次问题为例,其中浇次1里包含炉次1和炉次2,浇次2里包含炉次3、炉次4、炉次5。选取四个样本

$$\begin{aligned} I_1 &= \{(1,3,2,4,5)(1,2)\} & I_2 &= \{(4,5,2,1,3)(2,1)\} \\ I_3 &= \{(4,5,3,1,2)(2,1)\} & I_4 &= \{(3,5,4,2,1)(2,1)\} \end{aligned}$$

根据式(10)可得

$$P_{\text{charge}}^{\text{gen}} = \begin{pmatrix} 1/4 & 0 & 0 & 2/4 & 1/4 \\ 0 & 0 & 2/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 0 & 1/4 \\ 2/4 & 0 & 1/4 & 1/4 & 0 \\ 0 & 3/4 & 0 & 0 & 1/4 \end{pmatrix}$$

$$P_{\text{cast}}^{\text{gen}} = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

以轮盘赌采样得到个体:  $I_i = \{(4,5,2,1,3)(2,1)\}$  为例,下面介绍结合模糊关系矩阵的双向概率分布更新机制。

### 2.3.3 模糊关系矩阵

设论域  $U$ ,  $V$ , 乘积空间上  $U \times V = \{(u,v) | u \in U, v \in V\}$  上的一个模糊子集  $R$  为从集合  $U$  到集合  $V$  的模糊关系。如果模糊关系  $R$  的隶属度函数为

$$\mu_R : U * V \rightarrow [0,1], (x,y) \mapsto \mu_R(x,y) \quad (11)$$

则称隶属度  $\mu_R(x,y)$  为  $(x,y)$  关于模糊关系  $R$  的相关程度。

根据本文涉及到的参量模型,建立因素集  $U = \{u_1, u_2, \dots, u_n\}$ ,  $u_i$  表示工序集合中的第*i*个工件,并根据工件在工序集里的位置建立评语集  $V = \{v_1, v_2, \dots, v_n\}$ ,  $v_j$  表示某个工件在工序位置*j*处加工。 $\mu_R(u_i, v_j) = r_{ij} \in [0,1]$  为工件*i*在位置*j*加工的隶属度函数,  $i=1,2,\dots,m, j=1,2,\dots,n$ , 记  $R = (r_{ij})_{m \times n}$ , 则  $R$  为模糊关系矩阵。

本文选择正态分布来计算隶属度。正态分布在确定某参数的评价区间后,当取较高的隶属度时,说明采用正态分布能够收集更多隶属度高的信息,当取较低隶属度时,采用正态分布能够屏蔽更多的

隶属度低的信息。高斯函数计算各参量的隶属度为

$$r(\eta_{ij}) = e^{-\frac{(\eta_{ij}-\mu)^2}{2\sigma^2}} \quad (12)$$

式中,  $\mu$  为分布的均值;  $\sigma$  为分布的方差。

### 2.3.4 反向概率分布更新机制

根据给定浇次序  $v = \{\pi_1^z, \pi_2^z, \dots, \pi_Z^z\}$ , 确定炉次序  $u = \{l_0+1, \dots, l_1, \dots, l_{e-1}+1, \dots, l_{e-1}+i, \dots, l_e, \dots, l_Z\}$ ,

且满足  $\sum_{e=0}^{Z-1} \sum_{i=1}^{l_e} (l_{e-1}+i) = N$ , 此工序表明了浇次的优先级会影响其内部炉次的优先级,令  $i (i=1,2,\dots,N)$  为炉次  $l_{e-1}+i$  在工序中的位置,  $r_{(l_{e-1}+i)j}$  表示炉次  $l_{e-1}+i$  在位置*j*处的隶属度,则模糊关系矩阵  $R_{\text{charge}}$  可由下面公式求出

$$r_{(l_{e-1}+i)j} = e^{-\frac{(i-j)^2}{2\sigma^2}} \quad (13)$$

式中,  $\sigma$  为尺度参数(或方差),用于调节工件在某个位置隶属度的大小,通过选取不同级别的参数  $\sigma$ ,可以更好地对模糊关系矩阵进行调节。将式(13)代入式(10),可得

$$P_{ij}^{\text{gen}} = P_{ij}^{\text{gen}} r_{(l_{e-1}+i)j} \quad (14)$$

对  $P_{ij}^{\text{gen}}$  进行归一化处理可得

$$P_{ij}^{\text{gen}} = P_{ij}^{\text{gen}} / \sum_{j=1}^N P_{ij}^{\text{gen}} \quad (15)$$

通过采样式(15)可以得到炉次工序。

对于 2.2.2 中采样得到的解  $I_i = \{(4,5,2,1,3)(2,1)\}$ , 选取浇次序列  $v = (2,1)$  根据上述方法构造模糊关系矩阵,此处选取参数  $\sigma=1$ 。模糊关系矩阵构造如下

$$R_{\text{charge}} = \begin{pmatrix} 0.01 & 0.14 & 0.61 & 1 & 0.61 \\ 0 & 0.01 & 0.14 & 0.61 & 1 \\ 1 & 0.61 & 0.14 & 0.01 & 0 \\ 0.61 & 1 & 0.61 & 0.14 & 0.01 \\ 0.14 & 0.61 & 1 & 0.61 & 0.14 \end{pmatrix}$$

利用浇次序列构造模糊关系矩阵对炉次概率分布进行更新

$$P_{\text{charge}}^{\text{gen}} = P_{\text{charge}}^{\text{gen}} \cdot R_{\text{charge}} \quad (16)$$

归一化处理后可得

$$P_{\text{charge}}^{\text{gen}} = \begin{pmatrix} 0 & 0 & 0 & 0.73 & 0.35 \\ 0 & 0 & 0.27 & 0.22 & 0.57 \\ 0.45 & 0.25 & 0.14 & 0 & 0 \\ 0.55 & 0 & 0.59 & 0.05 & 0 \\ 0 & 0.75 & 0 & 0 & 0.08 \end{pmatrix}$$

以轮盘赌方式采样矩阵  $\mathbf{P}_{\text{charge}}^{\text{gen}}$ , 从前到后确定工序排列矢量的每一位, 把得到的新的炉次序列与  $\mathbf{I}_i$  的浇次序列组合成一个新的个体  $\mathbf{I}_i^{\text{charge}}$ 。

### 2.3.5 正向概率分布更新规则

根据给定的炉次序  $\mathbf{u} = \{\pi_1, \dots, \pi_i, \dots, \pi_N\}$ , 对于某个工件  $\pi_i$  在位置  $j$  处的隶属度为  $r_{\pi_i j} = e^{\frac{-(i-j)^2}{2\sigma^2}}$ , 可得到一个关于炉次的模糊关系矩阵  $\mathbf{R}_{\text{temp}}$ , 设浇次  $\pi_i^z$  包含炉次集合  $\Omega^z = \{l_{i-1} + 1, \dots, l_{i-1} + e \dots, l_i\}$ ,  $r_{\pi_i^z j}$  为浇次  $\pi_i^z$  在位置  $j$  处的隶属度, 则模糊关系矩阵  $\mathbf{R}_{\text{cast}}$  可由下面公式求出

$$r_{\pi_i^z j} = \frac{1}{\alpha_i} \sum_{e=1}^{\alpha_i} \sum_{a=1}^{\alpha_i} r_{(l_{i-1}+e)(l_{j-1}+a)} \quad (17)$$

将式(17)代入式(10), 可得

$$P_{ij}^{\text{gen}} = P_{ij}^{\text{gen}} r_{\pi_i^z j} \quad (18)$$

对  $P_{ij}^{\text{gen}+1}$  进行归一化处理可得

$$P_{ij}^{\text{gen}} = P_{ij}^{\text{gen}} / \sum_{j=1}^Z P_{ij}^{\text{gen}} \quad (19)$$

通过采样式(19)可以得到浇次工序。

针对第 2.3.2 节中采样得到的解, 选取炉次加工序列  $\mathbf{u} = (4, 5, 2, 1, 3)$ , 根据上述方法构造模糊关系矩阵, 此处选取参数  $\sigma = 1$ 。模糊关系矩阵构造如下

$$\mathbf{R}_{\text{cast}} = \begin{pmatrix} 0.225 & 0.59 \\ 0.457 & 0.284 \end{pmatrix}$$

利用炉次序列构造模糊关系矩阵对浇次概率分布进行更新

$$\mathbf{P}_{\text{cast}}^{\text{gen}} = \mathbf{P}_{\text{cast}}^{\text{gen}} \cdot \mathbf{R}_{\text{cast}} \quad (20)$$

$$\mathbf{P}_{\text{cast}}^{\text{gen}} = \begin{pmatrix} 0.276 & 0.724 \\ 0.617 & 0.383 \end{pmatrix}$$

以轮盘赌方式采样矩阵  $\mathbf{P}_{\text{cast}}^{\text{gen}}$ , 把得到的新的浇次序列与  $\mathbf{I}_i$  的炉次序列组合成一个新的个体。

## 2.4 种群分裂机制

当算法采样生成新种群, 则认为满足种群分裂条件, 将当前历史最优解的前段(后段)和采样生成种群中每个个体的后段(前段)逐一组合为新个体, 进而把种群分裂为双种群, 具体步骤如下。

(1) 选取当前种群中的最优个体  $\bar{\mathbf{I}} = (\bar{\mathbf{u}}, \bar{\mathbf{v}})$ 。

(2) 将当前种群中的个体的浇次序替换成最优个体的浇次序, 得到  $\mathbf{I}_i^1 = (\mathbf{u}_i, \bar{\mathbf{v}})$ , 由个体  $\mathbf{I}_i^1$  组成新

种群  $\text{pop1}$ 。

(3) 将当前种群中的个体的炉次序替换成最优个体的炉次序, 得到  $\mathbf{I}_i^2 = (\bar{\mathbf{u}}, \mathbf{v}_i)$ , 由个体  $\mathbf{I}_i^2$  组成新种群  $\text{pop2}$ 。

## 2.5 两阶段局部搜索算法

局部搜索算法的设计和邻域结构的分析是组合优化领域的重要研究问题, 有效的局部搜索可明显的加快算法的搜索效率, 从而提高整个算法的搜索性能。本节首先对双种群每个个体中非当前历史最优解部分执行基于 interchange 和 insert 邻域操作的迭代搜索。然后, 分析 swap 邻域操作的性质并提出该邻域的快速评价方法, 并对当前历史最优解执行结合所提快速评价的变邻域搜索。

### 2.5.1 第一阶段局部搜索算法

(1) Interchange 和 insert 邻域操作介绍。

① Interchange( $\mathbf{u}, i$ ): 交换序列  $\mathbf{u}$  中  $i$  前后两个位置的编码; ② Insert( $\mathbf{u}, i, j$ ) ( $i < j$ ): 产生一个 0 到 1 之间的随机数  $r$ , 若  $r < 0.5$ , 则将序列  $\mathbf{u}$  中的第  $i$  位编码插入到第  $j$  位后面, 若  $r \geq 0.5$ , 则将序列中的第  $j$  位编码插入到第  $i$  位后面。

(2) 第一阶段局部搜索。

首先针对  $\text{pop1}$ , 为增强算法的搜索能力, 需对  $\text{pop1}$  中的每个个体的炉次序的近邻区域进行较细致的邻域搜索操作, 操作如下: 选取  $\text{pop1}$  中的一个个体  $\mathbf{I}_i^1$ , 对其炉次序列执行基于 interchange 和 insert 的变邻域搜索操作, 将得到的新炉次序列和  $\mathbf{I}_i^1$  中的浇次序列组成新个体, 若新个体的目标值优于老个体, 则用新个体取代老个体, 并继续沿用当前的局部搜索操作, 如果产生的新个体的目标值差于老个体, 则采用另外一种局部操作对炉次序列进行搜索, 并不断更新  $\text{pop1}$  中的个体, 直到终止条件满足。

针对  $\text{pop2}$ , 采用同样的方法对每个个体的浇次序列进行变邻域搜索, 并更新种群中的个体。

### 2.5.2 第二阶段局部搜索算法

(1) 结合 swap 邻域的快速评价搜索操作。

Speedup\_swap( $\mathbf{v}, i, j$ ): 遍历交换序列  $\mathbf{v}$  中不同两个位置  $i$  和  $j$  的工件所得的所有序列或邻域, 利用定理 1 和定理 2 进行操作有效性判定, 并找到遍历过程中最好的解。

假设连铸阶段有  $m_S$  台连铸机, 每台连铸机之间相互独立,  $R_k$  表示第  $k$  台机器的释放时间。若  $R_{k^*} = \max\{R_k | k = 1, \dots, m_S\}$ , ( $R_{k^*} = C_{\max}$ ) 则称  $M_{k^*}$  为关键机器。调度解的目标值有关键机器决定, 因此当且仅当关键机器的释放时间都被缩短时, 邻域



操作才是有效的。针对浇次序基于快速评价的邻域操作如下所述。

定理 1 若交换的工件序号为  $\pi_i^z$  和  $\pi_j^z$  ( $j > i$ ) 在同一台机器  $M_k$  上加工时,  $R_k$  为交换前机器  $M_k$  的释放时间,  $R'_k$  为交换后机器  $M_k$  的释放时间, 令  $\Delta_k = R'_k - R_k$ , 当且仅当  $M_k$  为关键机器且  $\Delta_k < 0$  时, 即  $L'(\pi_i^{z(k)}) + L'(\pi_j^{z(k)}) - L(\pi_i^{z(k)}) - L(\pi_j^{z(k)}) < 0$ ; 对浇次序进行 Speedup\_swap 操作是有效的, 且新调度解最大完工时间为

$$C'_{\max} = \max \{ \max \{ R_k \mid k = 1, \dots, m_S, k \neq M_k \}, C_{\max} + \Delta_k \}$$

证明

执行操作前

$$R_k = D(\pi_1^{z(k)}, \pi_{i-1}^{z(k)}) + L(\pi_i^{z(k)}) - C(\pi_{i-1}^{z(k)}) + p(\pi_i^{z(k)}) + \\ D(\pi_{i+1}^{z(k)}, \pi_{j-1}^{z(k)}) + L(\pi_j^{z(k)}) - C(\pi_{j-1}^{z(k)}) + p(\pi_j^{z(k)}) + \\ D(\pi_{j+1}^{z(k)}, \pi_{T_S(k)}^{z(k)})$$

执行操作后

$$R'_k = D(\pi_1^{z(k)}, \pi_{i-1}^{z(k)}) + L'(\pi_j^{z(k)}) - C(\pi_{i-1}^{z(k)}) + p(\pi_i^{z(k)}) + \\ D(\pi_{i+1}^{z(k)}, \pi_{j-1}^{z(k)}) + L'(\pi_i^{z(k)}) - C(\pi_{j-1}^{z(k)}) + p(\pi_j^{z(k)}) + \\ D(\pi_{j+1}^{z(k)}, \pi_{T_S(k)}^{z(k)})$$

定理 2 若交换的工件序号为  $\pi_i^z$  和  $\pi_j^z$  ( $j > i$ ) 在不同两台机器  $M_{k_1}$  和  $M_{k_2}$  上加工时,  $R_k$  为交换前机器  $M_k$  的释放时间,  $R'_k$  为交换后机器  $M_k$  的释放时间, 令  $\Delta_{k_1} = R'_{k_1} - R_{k_1}$ ,  $\Delta_{k_2} = R'_{k_2} - R_{k_2}$ , 当且仅当  $M_{k_1}$  或者  $M_{k_2}$  中有一台关键机器且  $\Delta_{k_1} < 0$ ,  $\Delta_{k_2} < 0$  时, 即  $L'(\pi_j^{z(k_1)}) + p(\pi_i^{z(k_1)}) - L(\pi_i^{z(k_1)}) - p(\pi_i^{z(k_1)}) < 0$ ;  $L'(\pi_i^{z(k_2)}) + p(\pi_i^{z(k_2)}) - L(\pi_j^{z(k_2)}) - p(\pi_j^{z(k_2)}) < 0$ ; 时, 操作 Speedup\_swap(u) 是有效的, 且新调度解的最大完工时间为

$$C'_{\max} = \max \{ \max \{ R_k \mid k = 1, \dots, m_S, k \neq M_{k_1}, M_{k_2} \}, \\ R_{k_1} + \Delta_{k_1}, R_{k_2} + \Delta_{k_2} \};$$

证明:

设  $M_{k_1}$  为关键机器,  $R_{k_1} = C_{\max}$

执行操作前

$$R_{k_1} = D(\pi_1^{z(k_1)}, \pi_{i-1}^{z(k_1)}) + L(\pi_i^{z(k_1)}) - C(\pi_{i-1}^{z(k_1)}) + \\ p(\pi_i^{z(k_1)}) + D(\pi_{i+1}^{z(k_1)}, \pi_{T_S(k_1)}^{z(k_1)}) \\ R_{k_2} = D(\pi_1^{z(k_2)}, \pi_{j-1}^{z(k_2)}) + L(\pi_j^{z(k_2)}) - C(\pi_{j-1}^{z(k_2)}) + \\ p(\pi_j^{z(k_2)}) + D(\pi_{j+1}^{z(k_2)}, \pi_{T_S(k_2)}^{z(k_2)})$$

执行操作后

$$R'_{k_1} = D(\pi_1^{z(k_1)}, \pi_{i-1}^{z(k_1)}) + L'(\pi_j^{z(k_1)}) - C(\pi_{i-1}^{z(k_1)}) +$$

$$p(\pi_j^{z(k_1)}) + D(\pi_{i+1}^{z(k_1)}, \pi_{T_S(k_1)}^{z(k_1)})$$

$$R'_{k_2} = D(\pi_1^{z(k_2)}, \pi_{j-1}^{z(k_2)}) + L'(\pi_i^{z(k_2)}) - C(\pi_{j-1}^{z(k_2)}) +$$

$$p(\pi_i^{z(k_2)}) + D(\pi_{j+1}^{z(k_2)}, \pi_{T_S(k_2)}^{z(k_2)})$$

$$\Delta_{k_1} = R'_{k_1} - R_{k_1} =$$

$$L'(\pi_j^{z(k_1)}) + p(\pi_j^{z(k_1)}) - L(\pi_i^{z(k_1)}) - p(\pi_i^{z(k_1)})$$

$$\Delta_{k_2} = R'_{k_2} - R_{k_2} =$$

(2) 第二阶段局部搜索具体操作如下所述。

此阶段选取 pop1 和 pop2 中目标值最小的个体

$I_{best} = \min(I_{best}^{cast}, I_{best}^{charge})$  (即当前最优解) 进行局部搜索操作, 其中  $I_{best}^{charge}$  为 pop1 中的最优解,  $I_{best}^{cast}$  为 pop2 中的最优解。

首先, 采用第 2.5.1 节中的变领域搜索算法对当前最优解的炉次序列进行搜索, 若生成的新个体优于当前最优解, 则用生成的新个体取代最优解, 并继续沿用当前算法进行搜索, 若生成的新个体差于当前最优解, 则对当前最优解的浇次序序列执行 Speedup\_swap 操作, 根据上面所提定理来避免在遍历交换过程中的无效操作, 并对执行有效操作后得到的新解进行快速评价, 从而找到与当前炉次序列组合在一起目标值最小的个体, 来对当前最优解进行更新。重复上述操作, 直到停止条件满足。

通过 Speedup\_swap 操作可以快速遍历搜索整个浇次序解空间, 再配合针对炉次的变领域操作, 可以更有效的找到解空间中存在的优质解。

## 2.6 种群合并机制

传统的种群合并机制是把分裂后的种群进行简单组合, 然后对其进行后续操作, 这样的合并机制随机性较大, 上一代搜索过程中的优质解信息容易丢失。本文所提的种群合并操作与传统方法不同, 根据 PAN<sup>[15]</sup> 分析最优个体携带的优质解信息较多, 围绕其进行搜索有可能发现更多优质解, 于是本文设计了一种新的种群合并机制(种群生成机制), 通过对当前最优解进行扰动操作从而生成新种群, 这样极大程度的避免了随机搜索带来的算法性能下降, 且能最大化保留优质解的结构信息, 进而有助于算法更快的在解空间找到优质解。具体步骤如下。当种群局部搜索操作都停止时, 则认为满足种群的合并条件, 并恢复全局搜索。针对每次迭代所得到的最优个体的浇次序进行扰动操作, 生成新的浇次序, 并采用第 2.1.1 节中的规则生成对应的炉次序, 从而产生一个新的个体, 重复上述操作, 直到新种群生成并

恢复对种群的全局搜索。在进行扰动操作时, 若扰动次数过少, 则算法不容易跳出局部最优, 若扰动次数过多, 则会导致算法后续搜索的随机性较大, 使得算法搜索效率下降。经过试验表明, 当扰动次数为 3 次时, 算法性能最好。

(1) 扰动算子。

① Inter\_reverse( $v$ ): 随机选择序  $v$  中的两个位置  $i, j$ , 并把  $i, j$  中间的序列逆序。

② Intra\_reverse( $v$ ): 随机选择序  $v$  中的两个位置  $i, j$ , 并把  $i, j$  两端的序列逆序。

③ Insert\_f( $v$ ): 随机选择序  $v$  中的一个位置  $i$ , 并把位置  $i$  后面的序插到序列最前面。

④ Insert\_b( $v$ ): 随机选择序  $v$  中的一个位置  $i$ , 并把位置  $i$  前面的序插到序列最后面。

(2) 扰动操作。

首先选取当前种群中的最优解  $I_{\text{best}}$ , 并将其浇次序列赋值给一个临时变量  $I_{\text{temp}}$ , 然后对  $I_{\text{temp}}$  浇次序列执行重复 3 次的扰动操作, 扰动操作通过随机产生的 1 到 4 四个随机数进行选择, 3 次扰动操作执行完毕后, 根据第 2.1.1 节中的规则生成对应的炉次序, 从而组成新个体  $I_{\text{disturb}}$ 。

## 2.7 CCOA 流程

令  $pop(gen)$  为算法第  $gen$  代种群,  $gen\_max$  为最大运行次数,  $PS$  为种群大小, 本文取  $PS=40$ 。种群更新步骤如下。

步骤 1: 令  $gen=1$ , 由 2.1 节启发式规则初始化种群, 并计算种群中每个个体的目标值。根据式(10)生成第 1 次迭代的炉次和浇次的概率分布参

数  $P_{\text{cast}}^1, P_{\text{charge}}^1$ , 并根据式(21), (22)对概率分布进行平滑处理,  $\alpha$  为平滑系数。

$$P_{\text{charge}}^1 = \alpha P_{\text{charge}}^1 + (1-\alpha) P_{\text{charge}}^0 \quad (21)$$

$$P_{\text{cast}}^1 = \alpha P_{\text{cast}}^1 + (1-\alpha) P_{\text{cast}}^0 \quad (22)$$

步骤 2: 通过轮盘赌的方式采样生成基础种群  $pop(gen)$ ,  $I_1, I_2, \dots, I_n$  为种群中的个体, 对于基础种群个体  $I_i$  采用 2.3 中的双向更新规则生成  $I_i^{\text{cast}}$  和  $I_i^{\text{charge}}$ , 并选取  $\hat{I}_i = \text{best}\{I_i, I_i^{\text{cast}}, I_i^{\text{charge}}\}$  更新当种群,  $\hat{I}_1, \hat{I}_2, \dots, \hat{I}_n$  为种群中的个体。

步骤 3: 根据第 2.4 节种群分裂机制把种群分裂成两个子种群  $pop1$  和  $pop2$ 。并对  $pop1$  和  $pop2$  中的每个个体进行变领域搜索, 并更新当前种群。找到当前种群中的最优解, 对其执行第二级局部搜索操作, 并更新当前最优解。

步骤 4: 根据第 2.6 节的种群合并机制对种群进行合并, 按式(20)确定下次迭代的概率分布  $P_{\text{cast}}^{\text{gen}+1}, P_{\text{charge}}^{\text{gen}+1}$ , 并根据式(23)、(24)对概率分布进行平滑处理。

$$P_{\text{charge}}^{\text{gen}+1} = \alpha P_{\text{charge}}^{\text{gen}+1} + (1-\alpha) P_{\text{charge}}^{\text{gen}} \quad (23)$$

$$P_{\text{cast}}^{\text{gen}+1} = \alpha P_{\text{cast}}^{\text{gen}+1} + (1-\alpha) P_{\text{cast}}^{\text{gen}} \quad (24)$$

步骤 5: 令  $gen = gen + 1$ , 如果  $gen \leq gen\_max$ , 转至步骤 2, 否则输出当前最优解  $I_{\text{best}}$ 。

算法流程图如图 6 所示。

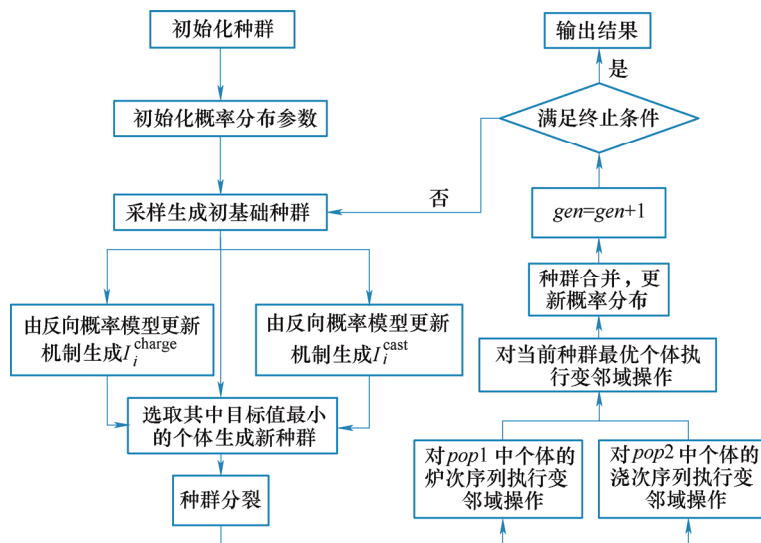


图 6 算法流程图

### 3 试验设计与分析

为确定算法的参数和性能, 对其开展试验设计<sup>[29]</sup>(Design of experiment, DOE), 选取  $S \in \{3, 4, 5, 6\}$ 、 $Z \in \{10, 15, 20, 25, 30\}$ , 测试的  $S \times Z$  组合包括:  $3 \times 10$ ,  $3 \times 15$ ,  $3 \times 20$ ,  $3 \times 25$ ,  $3 \times 30$ ,  $4 \times 10$ ,  $4 \times 15$ ,  $4 \times 20$ ,  $4 \times 25$ ,  $4 \times 30$ ,  $5 \times 10$ ,  $5 \times 15$ ,  $5 \times 20$ ,  $5 \times 25$ ,  $5 \times 30$ ,  $6 \times 10$ ,  $6 \times 15$ ,  $6 \times 20$ ,  $6 \times 25$ ,  $6 \times 30$ 。根据我国炼钢连铸实际生产情况, 生产过程中的参数由以下均匀分布随机产生:  $p_{s \times k} \in [36, 50]$ ,  $m_s \in [3, 5]$ ,  $\alpha_k \in [8, 12]$ ,  $t_s \in [10, 15]$ ,  $q_z \in [80, 100]$ , 测试算例均已上传 GitHub\*。所有算法和测试程序均由 Delphi 7 编码实现, 操作系统为 Win10, CPU 主频为 4.5 GHz, 内存为 16 GB。每种算法独立运行 20 次, 算法的运行时间为  $T^*$ , 算法终止条件为  $T > Z \times S \times \lambda$ , 每次运行记录目标值  $f$ , 并采用一下三个指标对算法性能进行衡量

$$\begin{cases} BST = \min\{f_i | i = 1, \dots, 20\} \\ AVG = \sum_{i=1}^{20} f_i / 20 (i = 1, \dots, 20) \\ WST = \max\{f_i | i = 1, \dots, 20\} \end{cases} \quad (25)$$

式中,  $BST$  表示算法独立运行 20 次的最好值,  $AVG$  表示算法独立运行 20 次的均值,  $WST$  表示算法独立运行 20 次的最差值。特别说明: 本文仿真试验所用测试算例可在 <https://github.com/ly726564418/testdate.git> 下载。

#### 3.1 算法参数设置

本文所涉及到的四个参数尺度参数  $\sigma$ 、分位数  $\rho$ 、平滑系数  $\alpha$  和迭代次数  $MI$  均取四个水平(表 2)选择规模为  $L16(4^4)$  的正交试验表, 选取一组中等规模问题( $5 \times 15$ )采用 DOE 进行试验分析, 每组参数组合下的测试问题均进行 20 次独立试验, 以式(25)中的  $AVG$  为评价指标进行测试试验参数相应值如

表 3 所示, 根据表 3 所示的正交表和  $AVG$  统计, 可得到各参数基于  $AVG$  的相应值(表 4), 根据各参数的响应值, 可以得到各参数相应值得变化趋势如图 7 所示。由表 4 和图 7 可以看出  $\sigma$ ,  $\rho$ ,  $\alpha$ ,  $MI$  4 个主要参数的平均响应值和影响力等级, 其影响力等级一栏值越小代表该参数的影响力排名越高, 不难看出迭代次数  $MI$  对算法影响程度最大。根据表 4 和图 7 中 4 个参数的变化对算法的影响分析如下。

表 2 参数水平表

参数	水平			
	1	2	3	4
$\sigma$	0.6	0.8	1.0	1.2
$\rho$	5	10	15	20
$\alpha$	0.1	0.3	0.5	0.7
$MI$	20	25	30	35

尺度参数  $\sigma$  影响采样生成的炉次或者浇次序列之间的耦合程度大小, 增大  $\sigma$ , 会加强生成序列之间的耦合性, 但如果  $\sigma$  过大, 会出现过耦合现象, 导致生成的解不符合问题的约束关系, 使解的质量变差。分位数  $\rho$  影响算法对于优质解信息的统计, 选择过小的分位数  $\rho$  会导致算法在统计优质解信息时不充分, 所统计的信息不足以全面反映优质解的特征, 使得算法在后续的搜索引导能力下降。选择过大的分位数  $\rho$  会导致算法统计时间加长, 使得算法搜索效率降低。

平滑参数  $\alpha$  影响概率参数的收敛速度, 由于 CE 算法的性能很大程度上取决于概率参数的收敛程度,  $\alpha$  取值过大容易导致算法早熟收敛, 取值过小容易导致算法收敛过缓。

迭代次数  $MI$  影响算法局部搜索性能,  $MI$  过小会导致算法搜索深度不足,  $MI$  过大会导致算法进行大量无效搜索, 使得算法局部搜索性能下降。

表 3 各参数组合的相应值

组合	参数水平				RVG	组合	参数水平				RVG
	$\sigma$	$\rho$	$\alpha$	$MI$			$\sigma$	$\rho$	$\alpha$	$MI$	
1	1	1	1	1	23 390.55	9	3	1	3	4	23 414.35
2	1	2	2	2	23 386.95	10	3	2	4	3	23 383.65
3	1	3	3	3	23 394.75	11	3	3	1	2	23 389.85
4	1	4	4	4	23 408.7	12	3	4	2	1	23 395.15
5	2	1	2	3	23 380.1	13	4	1	4	2	23 391.55
6	2	2	1	4	23 412.9	14	4	2	3	1	23 380.95
7	2	3	4	1	23 393.1	15	4	3	2	4	23 409.85
8	2	4	3	2	23 390.85	16	4	4	1	3	23 391.45

表 4 各参数组合性能响应值

水平	$\sigma$	$\rho$	$\alpha$	$MI$
1	23 395.2	23 394.1	23 396.2	23 389.9
2	23 394.3	<b>23 391.1</b>	<b>23 388.4</b>	23 389.8
3	23 395.8	23 396.9	23 395.2	<b>23 387.5</b>
4	<b>23 393.5</b>	23 396.5	23 394.3	23 411.5
极差	2.3	5.8	7.8	23.9
优方案	4	2	2	3
等级	4	3	2	1

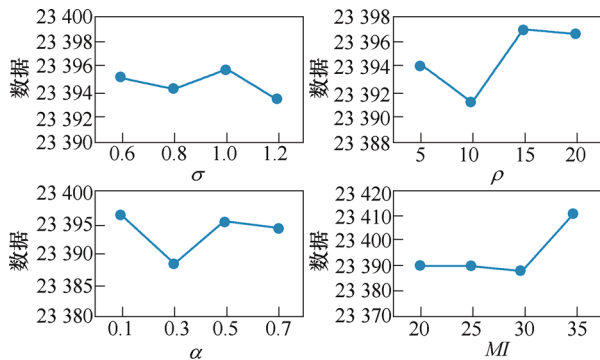


图 7 响应趋势图

通过对各参数组合的影响等级和影响趋势的分析, 不难看出参数的选择对算法性能有着显著的影响。若选择的参数过大, 会导致算法偏离优质解区域, 若参数过小, 又会使得算法的搜索能力不足。综合考虑, 选取表 4 中的优方案作为 CCOA 的参数设置, 即尺度参数  $\sigma$  选择水平 4, 其值为 1.2, 分位

数  $\rho$  选择水平 2, 其值为 10, 平滑参数  $\alpha$  选择水平 2, 其值为 0.3, 迭代次数  $MI$  选择水平 3, 其值为 30, 此时, 算法可表现出良好性能。

### 3.2 CCOA 与变形算法对比

CCOA 的关键操作主要包括: ① 种群初始化规则; ② 基于模糊关系矩阵的双向概率分布更新机制; ③ 双种群协同的变邻域搜索操作; ④ 基于邻域操作有效性分析的局部搜索。因此, 考虑如下变形算法, 参数设置与 CCOA 一致。

(1) 标准的 CE 算法。

(2) CE\_V1: 在标准的 CE 算法基础上增加种群初始化规则。

(3) CE\_V2: 在 CE\_V1 的基础上使用基于模糊关系矩阵的双向概率分布更新机制。

(4) CE\_V3: 在 CE\_V2 的基础上增加对当前种群中最优解的浇次和炉次序列执行表 2 和表 3 中的变邻域局部搜索算法。

(5) CE\_V4: 在 CE\_V2 的基础上增加两阶段局部搜索操作, 第一阶段跟 CCOA 算法一致, 第二阶段采用 CE\_V3 中的算法针对种群最优解执行局部搜索操作。

在相同时间内对 CCOA 和五种变形算法进行试验对比, 从而验证 CCOA 算法中每个关键环节的有效性。设定每种算法的运行时间均为  $Z \times S \times 200$ , 每个测试问题均进行 20 次独立试验, 对应的最优结果用粗体表示。试验结果如表 5 所示。

表 5 CCOA 与多种 CE 变形算法对比结果

问题规模	CE		CE_V1		CE_V2		CE_V3		CE_V4		CCOA	
	AVG	BST	AVG	BST	AVG	BST	AVG	BST	AVG	BST	AVG	BST
3×10	21 044	20 100	14 519	14 462	14 530	14 478	14 443	14 384	14 446	14 396	<b>14 428</b>	<b>14 386</b>
3×15	32 807	31 575	21 367	21 327	21 369	21 343	21 349	21 314	21 341	21 303	<b>21 315</b>	<b>21 295</b>
3×20	38 856	37 231	22 324	22 136	22 088	21 903	21 988	21 833	21 905	21 775	<b>21 813</b>	<b>21 774</b>
3×25	68 088	66 385	45 205	44 350	44 337	43 953	44 183	43 754	44 081	43 550	<b>43 934</b>	<b>43 477</b>
3×30	94 168	91 517	55 323	55 151	54 324	53 923	54 025	53 676	<b>54 014</b>	<b>53 630</b>	54 047	53 705
4×10	26 162	25 149	16 346	16 229	16 048	15 969	15 951	15 856	15 914	15 867	<b>15 894</b>	<b>15 843</b>
4×15	38 174	36 761	22 576	22 503	22 586	22 534	22 542	22 466	22 500	22 461	<b>22 458</b>	<b>22 429</b>
4×20	52 580	51 269	30 368	30 309	30 041	29 971	29 992	29 936	29 957	29 904	<b>29 925</b>	<b>29 887</b>
4×25	65 701	64 473	43 931	43 296	43 195	42 856	42 997	42 559	<b>42 845</b>	<b>42 494</b>	42 911	42 568
4×30	93 223	90 185	54 427	53 987	54 034	53 779	53 782	53 511	53 732	53 402	<b>53 670</b>	<b>53 371</b>
5×10	26 872	25 869	16 935	16 833	16 945	16 873	16 838	16 727	16 804	16 739	<b>16 761</b>	<b>16 725</b>
5×15	39 505	37 902	23 980	23 905	23 558	23 439	23 475	23 386	23 448	23 390	<b>23 385</b>	<b>23 339</b>
5×20	65 006	63 704	39 612	38 686	38 907	38 285	38 761	38 112	38 440	38 090	<b>37 983</b>	<b>37 708</b>
5×25	64 899	62 544	37 729	37 677	37 043	36 960	37 012	36 940	36 969	36 918	<b>36 930</b>	<b>36 901</b>
5×30	82 820	81 913	46 884	46 802	46 906	46 839	46 834	46 782	46 823	46 774	<b>46 790</b>	<b>46 728</b>
6×10	28 654	27 731	18 458	18 388	18 150	18 087	18 048	17 987	18 028	17 989	<b>18 017</b>	<b>17 974</b>
6×15	41 326	40 378	25 373	25 263	24 934	24 820	24 867	24 756	24 828	24 743	<b>24 764</b>	<b>24 726</b>
6×20	56 827	54 957	33 826	33 729	33 842	33 744	33 714	33 606	33 687	33 595	<b>33 614</b>	<b>33 542</b>
6×25	78 806	76 738	47 343	46 830	46 946	46 457	46 941	46 375	46 549	46 323	<b>46 253</b>	<b>46 101</b>
6×30	81 557	80 335	46 985	46 883	46 445	46 344	46 315	46 231	46 325	46 224	<b>46 261</b>	<b>46 191</b>
均值	54 854	53 336	33 175	32 937	32 811	32 628	32 703	32 510	32 632	32 478	<b>32 559</b>	<b>32 434</b>

为了进一步验证 CCOA 中各关键环节对算法的影响,根据每种变形算法独立运行 20 次的算法结果的均值进行方差分析(ANOVA)。由于未采用启发式规则的 CE 算法与其他 5 种变形算法差距较大,因此在进行方差分析时未将其加入其中。图 8 显示了 CCOA 中关键环节对算法运行结果的影响,以及均值变化线和 95%置信度下的 Tukey's HSD 检验的置信区间。结合表 5 可以看出,CE\_V1 解的质量明显优于 CE,表明在求解 CU\_SCCSP 时,启发式规则的使用对提高初始解质量有显著效果。CE\_V2 解的质量优于 CE\_V1,表明在 CE\_V1 基础上采用结合模糊关系矩阵的概率分布更新机制有利于引导算法朝优质解空间的方向进行搜索,提高算法搜索效率,也进一步验证了本文所提的结合模糊关系矩阵的 CE 在求解相关子问题时,有显著的效果。CE\_V3 与 CE\_V4 对比表明双种群协同的局部搜索的性能明显优于单纯对最优解进行局部搜索。CCOA 解的质量优于 CE\_V4,表明基于邻域操作有效性分析的 Speeduo\_swap 搜索操作在保证较短运行时间的前提下,可以对解空间进行较为细致充分的搜索,从而大大增加发现优质解或近似最优解的概率。

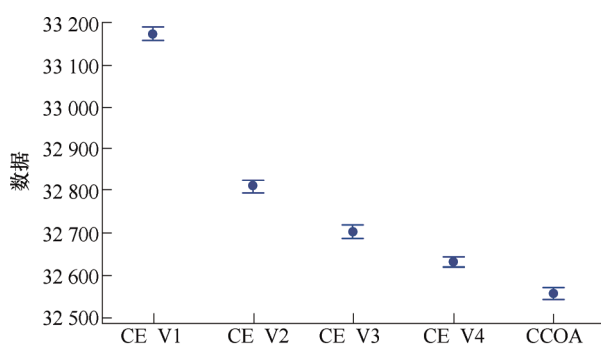


图 8 CCOA 关键环节对比区间图

综上所述,CCOA 中的双向概率分布更新机制有助于提高算法的全局搜索能力,基于变邻域操作和 Speeduo\_swap 操作的两阶段局部搜索算法可以加强算法的局部搜索能力,从而使算法在全局搜索和局部搜索之间达到较好的平衡。

### 3.3 CCOA 与其他算法对比

为了进一步验证 CCOA 的性能,本节将 CCOA 和重要国内外期刊中的有效算法 CCABC<sup>[15]</sup>、OIABC<sup>[13]</sup>、GA<sup>[30]</sup>进行比较。其中 CCABC 是求解 CU\_SCCSP 公认的有效算法, IABC 是解决 CC\_SCCSP 的有效算法,本文在 IABC 基础上进行改进,设计出了 OIABC,使其在保留 IABC 搜索性能的同时,也适用于求解 CU\_SCCSP 问题。设定每

种算法的运行时间均为  $Z \times S \times \lambda$ , 每个测试问题均进行 20 次独立试验,最优结果用粗体表示。

在所有对比算法中,针对 GA,本文设计了部分映射交叉(PMX)、顺序交叉(OX)和单点交叉(OP)三种交叉算子用于执行交叉操作,并以 50%的概率随机选取插入或者交换算子来对个体执行变异操作。试验表明,当设置种群大小为 10、交叉概率为 0.5、变异概率为 0.1 时,算法性能表现良好。OIABC 是通过对文献[13]中的 IABC 算法进行改进的到的。IABC 算法的优势在于针对雇佣蜂和围观蜂阶段分别设计两种变领域搜索操作,来实现对不同邻域的解空间进行搜索,于是本文在保留其变领域局部搜索算法的基础上,针对 CC\_SCCSP 原本固定不变的浇次序列也执行了变领域搜索操作,并通过更改原有的编码方式和目标函数,让其适用于 CU\_CCABC。

CCABC、OIABC、GA 均采用本文所提的种群初始化规则来初始化种群,以确保算法具有相同的初始解质量。从表 6 中( $\lambda = 300$ )可知,CCOA 在绝大部分问题上的测试结果都明显优于其他比较算法,这验证了 CCOA 具有良好的性能。为了进一步验证各算法组内差异,根据  $\lambda$  在 200、300、400 三种取值下的算法结果的 AVG 进行方差分析(ANOVA)。图 9 显示了四种算法在不同终止运行时间下的均值变化线及 95%置信度下的 Tukey's HSD 检验的置信区间,可以看出,CCOA 在均值水平上与其他算法有显著差异。

GA 通过模仿自然界的选择与遗传的机理来寻找最优解,其具有良好的全局搜索能力,可以快速地对解空间中的全体解进行搜索,而不会陷入局部最优解的快速下降陷阱,但是遗传算法的局部搜索能力较差,采用传统的交叉,变异等邻域操作生成新种群以实现搜索,这使其实际搜索深度较为有限,难以到达复杂解空间中真正存在优质解的区域,导致算法后期搜索效率较低。CCABC 通过在常规的群智能算法基础增加协同进化机制,并在蜂群探索阶段加入了探索策略和增强策略来引导算法向优质解区域进行搜索。OIABC 在传统蜂群多阶段搜索的基础上增加了变领域搜索操作,以增加算法在不同阶段的寻优能力。上述两种基于 ABC 的改进算法在求解 SCCSP 都取得了较好的效果,但由于 ABC 产生新解只是基于其父解(旧解),导致好的信息无法在种群中快速传播,同时每次变异只修改父解的一个维度,并且改变幅度较小,所以这导致 ABC 局部寻优能力较弱,收敛速度较慢,尤其在解决约束问



题、复合函数、不可分函数上性能欠佳。而本文涉及到的 CU\_SCCSP 问题是一个典型的多约束组合优化问题, 因此, ABC 与大多数群智能算法类似, 在实际搜索过程中算法存在早熟收敛、易陷入局部最优, 进化后期收敛较慢等问题。针对上述对比算法所存在的问题, 本文所提的 CCOA, 在全局搜索阶段通过模糊关系矩阵改进 CE 算法来缩小问题的解空间, 从而提升算法的全局搜索效率, 并在局部

阶段提出了一种双阶段局部搜索算法来引导算法在多个不同区域进行较深的搜索, 从而提升算法搜索的深度。另外, 为了防止算法过早收敛, 陷入局部最优, 本文还提出了一种种群合并机制, 通过对当前最优解进行扰动操作来生成新个体, 从而使算法能够跳出局部最优, 来保证算法能够发现复杂解空间中的优质解。因此 CCOA 可在上述测试问题中取得较好效果。

表 6 CCOA 与 CCABC、OIABC 和 GA 的比较( $\lambda = 300$ )

问题规模	CCOA			CCABC			OIABC			GA		
	AVG	BST	WST	AVG	BST	WST	AVG	BST	WST	AVG	BST	WST
3×10	<b>14 417</b>	14 376	<b>14 455</b>	14 561	14 505	14 677	14 560	14 499	14 619	14 640	14 606	14 672
3×15	<b>21 315</b>	21 294	<b>21 337</b>	21 569	21 542	21 602	21 568	21 537	21 607	21 601	21 556	21 634
3×20	<b>21 801</b>	21 755	<b>21 869</b>	21 872	21 830	21 914	21 868	21 825	21 946	21 896	21 871	21 920
3×25	43 972	43 542	44 424	<b>43 609</b>	<b>43 461</b>	43 851	43 617	43 512	<b>43 792</b>	43 777	43 591	44 042
3×30	53 895	53 556	54 141	<b>53 649</b>	53 537	53 702	53 650	53 623	<b>53 657</b>	53 696	<b>53 515</b>	53 706
4×10	<b>15 905</b>	<b>15 848</b>	<b>15 963</b>	15 968	15 929	16 014	15 958	15 898	15 999	16 005	15 967	16 090
4×15	<b>22 457</b>	<b>22 431</b>	<b>22 503</b>	22 562	22 526	22 591	22 559	22 519	22 594	22 589	22 558	22 625
4×20	<b>29 930</b>	<b>29 899</b>	<b>29 970</b>	30 082	30 037	30 122	30 092	30 057	30 129	30 124	30 093	30 150
4×25	42 815	42 452	43 252	<b>42 524</b>	42 442	<b>42 685</b>	42 544	<b>42 419</b>	42 745	42 616	42 433	42 811
4×30	53 666	53 436	53 825	<b>53 497</b>	<b>53 358</b>	53 571	53 532	53 367	53 590	53 566	53 450	<b>53 610</b>
5×10	<b>16 767</b>	<b>16 708</b>	<b>16 824</b>	16 851	16 810	16 895	16 855	16 806	16 925	16 925	16 897	16 951
5×15	<b>23 381</b>	<b>23 350</b>	<b>23 431</b>	23 546	23 483	23 610	23 547	23 508	23 612	23 606	23 566	23 657
5×20	<b>37 937</b>	<b>37 733</b>	<b>38 288</b>	38 115	37 844	38 479	38 155	37 869	38 418	38 284	37 817	38 560
5×25	<b>36 928</b>	<b>36 883</b>	<b>36 960</b>	37 124	37 086	37 176	37 130	37 092	37 173	37 174	37 131	37 214
5×30	<b>46 777</b>	<b>46 685</b>	<b>46 883</b>	47 001	46 913	47 057	47 010	46 922	47 086	47 056	47 013	47 100
6×10	<b>18 010</b>	<b>17 967</b>	<b>18 043</b>	18 120	18 069	18 157	18 109	18 063	18 133	18 152	18 118	18 180
6×15	<b>24 766</b>	<b>24 733</b>	<b>24 809</b>	24 914	24 864	24 970	24 919	24 848	25 006	24 984	24 898	25 031
6×20	<b>33 592</b>	<b>33 534</b>	<b>33 675</b>	33 815	33 725	33 890	33 809	33 747	33 882	33 923	33 865	33 973
6×25	<b>46 251</b>	<b>46 057</b>	<b>46 468</b>	46 275	46 106	46 488	46 315	46 176	46 473	46 333	46 184	46 498
6×30	<b>46 245</b>	<b>46 168</b>	<b>46 289</b>	46 551	46 455	46 624	46 524	46 464	46 625	46 621	46 565	46 680
均值	<b>32 541</b>	<b>32 420</b>	<b>32 670</b>	32 610	32 526	32 704	32 616	32 538	32 701	32 678	32 585	32 755

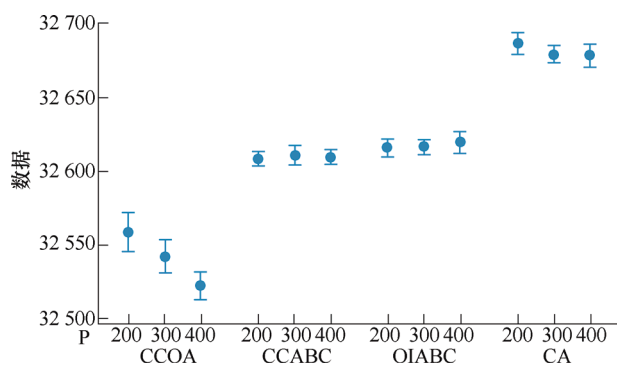


图 9 方差分析区间图

## 4 结论

(1) 建立 CU\_SCCSP 的排序模型, 并设计前后子问题两段式编码和双向解码的策略。

(2) 在算法的全局搜索部分, 结合传统 CE 算法, 设计了一种针对耦合性问题求解的双向概率分布更新机制, 通过基于隶属度函数构造的模糊关系矩阵来对采样概率分布进行优化, 使得算法能在解空间中更快地找到优质解, 从而提高算法的全局搜索效率。

(3) 在算法的局部搜索部分, 构造了基于 interchange 和 insert 邻域的变邻域局部搜索操作, 并结合问题特性, 针对 swap 邻域操作进行有效性分析, 提出了 speedup\_swap 操作, 结合两阶段的局部搜索, 对全局得到的优质解执行较为细致的搜索。

通过不同测试问题上的仿真试验和算法比较, 验证了 CCOA 是求解 CU\_SCCSP 的有效算法。本文所提的 CCOA 引入了基于隶属度函数建立的模糊关系矩阵, 这对其他智能调度算法在求解具有耦合性问题具有一定的借鉴意义。后续工作是针对不确定多目标的 SCCSP, 设计结合机器学习的有效多目标 CCOA 进行求解。

### 参 考 文 献

- [1] 殷瑞钰. 关于智能化钢厂的讨论——从物理系统一侧出发讨论钢厂智能化[J]. 钢铁, 2017, 52(6): 1.  
YIN R Y. A discussion on “smart” steel plant-view from physical system side[J]. Iron Steel, 2017, 52(6): 1.
- [2] CUI H, LUO X. An improved lagrangian relaxation approach to scheduling steelmaking-continuous casting process[J]. Computers & Chemical Engineering, 2017, 106(2): 33-146.
- [3] LEE H S, MURTHY S S, HAIDER S, et al. Primary production scheduling at steelmaking industries[J]. IBM Journal of Research & Development, 1996, 40(2): 231-252.
- [4] TANG L X, LIU J Y, RONG A Y, et al. A mathematical programming model for scheduling steelmaking continuous casting production[J]. European Journal of Operational Research, 2000, 120(2): 423-435.
- [5] GAREY M R, JOHNSON D S, SETHI R. The complexity of flow shop and job shop Scheduling[J]. Mathematics of Operations Research, 1976, 1(2): 117-129.
- [6] ZHAO X Q, RONG G. Survey of production scheduling in the process industry[J]. Control and Instruments in Chemical Industry, 2004, 31(6): 8-13.
- [7] XUAN, H, TANG, L. Scheduling a hybrid flow shop with batch production at the last stage[J]. Computer & Operations Research, 2007, 34(9): 2718-2733.
- [8] MAO K, PAN Q K. A novel lagrangian relaxation approach for a hybrid flowshop scheduling problem in the steelmaking-continuous casting process[J]. European Journal of Operational Research, 2014, 236: 51-60.
- [9] LIANG L, SUN H, JIN Y Q, et al. Research on Steelmaking continuous casting production scheduling problem based on augmented lagrangian relaxation algorithm under multi-coupling constraints[J]. IFAC Papers on Line, 2019, 52(1): 820-825.
- [10] CUI L, LI G, ZHU Z, et al. A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization[J]. Information Sciences, 2017, 414(1): 53-67.
- [11] CUI H, LUO X, WANG Y & et al. Scheduling of steelmaking continuous casting process using deflected surrogate Lagrangian relaxation approach and DC algorithm[J]. Computers & Industrial Engineering, 2020, 140: 106271.
- [12] ZHU D F, ZHENG Z, GAO X Q. Intelligent optimization based production planning and simulation analysis for steelmaking and continuous casting process. International Journal of Iron and Steel Research, 2010, 17(9): 19-24.
- [13] PENG K, PAN Q, ZHANG B. An improved artificial bee colony algorithm for steelmaking-refining-continuous casting scheduling problem[J]. Chinese Journal of Chemical Engineering, 2018, 26(08): 135-143.
- [14] TANG L, ZHAO Y, LIU J. An improved differential evolutionary algorithm for practical dynamic scheduling in steelmaking-continuous casting production[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(2), 209-225.
- [15] PAN Q K. An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling[J]. European Journal of Operational Research, 2016, 250(3): 702-714.
- [16] 曾冰, 王梦雨, 高亮, 等. 改进鲸鱼群算法及其在炼钢连铸调度中的应用[J]. 郑州大学学报, 2018, 39(6): 14-22, 35.  
ZENG Bing, WANG Mengyu, GAO Liang, et al. Improved whale swarm algorithm and its application in steelmaking and continuous casting scheduling[J]. Journal of Zhengzhou University, 2018, 39(6): 14-22, 35.
- [17] RUBINSTEIN R. The cross-entropy method for combinatorial and continuous optimization[J]. Methodology and Computing in Applied Probability, 1999, 1(2): 127-190.
- [18] 杨海军, 李建武, 李敏强. 进化算法的模式、涌现与困难性研究[M]. 北京: 科学出版社, 2012.  
YANG Haijun, LI Jianwu, LI Minqiang. Research on pattern, emergence and difficulty of evolutionary algorithm[M]. Beijing: Science Press, 2012.
- [19] 余明哲, 钱斌, 胡蓉等. 混合交叉熵算法求解模糊分布式装配流水线低碳调度问题[J]. 控制理论与应用, 2020, 37(10): 12-23.  
SHE Mingzhe, QIAN Bin, HU Rong, et al. Hybrid cross-entropy algorithm to solve fuzzy distributed assembly line low-carbon scheduling problem[J]. Control Theory and Application, 2020, 37(10): 12-23.

- [20] 钱斌, 余明哲, 胡蓉, 等. 超启发式交叉熵算法求解模糊分布式流水线绿色调度问题[J]. 控制与决策, 2021, 36(6): 1387-1396.  
QIAN Bin, SHE Mingzhe, HU Rong, et al. Super heuristic cross-entropy algorithm to solve the fuzzy distributed pipeline green scheduling problem[J]. Control and Decision, 2021, 36(6): 1387-1396.
- [21] CASERTA M, RICO E Q. A cross entropy algorithm for the Knapsack problem with setups[J]. Computers & Operations Research, 2008, 35(1): 241-252.
- [22] URBAN T L. Optimal balancing of U-shaped assembly lines[J]. Management Science, 1998, 44(5): 738-741.
- [23] 王桂荣, 李歧强, 丁然等. 加工时间不确定的炼钢连铸生产调度串级交叉熵算法[J]. 控制与决策, 2016, 31(7): 1153-1160.  
WANG Guirong, LI Qiqiang, DING Ran, et al. Cascade cross entropy algorithm for steelmaking and continuous casting production scheduling with uncertain processing time [J]. Control and Decision, 2016, 31 (7): 1153-1160.
- [24] 杨凡, 李歧强, 刘珊, 等. 浇次计划编制的混合启发式—交叉熵算法[J]. 计算机集成制造系统, 2014, 20(9): 2241-2247.  
YANG Fan, LI Qiqiang, LIU Shan, et al. Hybrid gHeuristic-Cross Entropy Algorithm for Pouring Plannin[J]. Computer Integrated Manufacturing System, 2014, 20(9): 2241-2247.
- [25] 梁吉业, 冯晨娇, 宋鹏, 等. 大数据相关分析综述[J]. 计算机学报, 2016, 39(1): 1-18.  
LIANG Jiye, FENG Chenjiao, SONG Peng, et al. Overview of big data correlation analysis [J]. Chinese Journal of Computers, 2016, 39(1): 1-18.
- [26] 杨凡, 李歧强, 王桂荣. 柔性宽度浇次计划编制的一种混合改进算法[J]. 控制与决策, 2015, 30(2): 348-352.  
YANG Fan, LI Qiqiang, WANG Guirong. Hybrid improved algorithm for cast planning problem with flexible width[J]. Control and Decision, 2015, 30(2): 348-352.
- [27] SUN Feng, QU Xiaobing, WANG Xueping, et al. On pre-solution matrices of fuzzy relation equations over complete Brouwerian lattices[J]. Fuzzy Sets & Systems, 2020, 384: 34-53.
- [28] 郑逸凡, 钱斌, 胡蓉, 等. CE-GA 协同进化算法求解人机共同作业的 U 形装配线平衡问题[J]. 机械工程学报, 2020, 56(9): 213-228.  
ZHENG Yifan, QIAN Bin, HU Rong, et al. CE-GA co-evolutionary algorithm to solve the U-shaped assembly line balance problem of human-machine cooperation[J]. Journal of Mechanical Engineering, 2020, 56(9): 213-228.
- [29] MONTGOMERY D C. Design and Analysis of Experiments[M]. Hoboken: John Wiley and Sons, 2005.
- [30] RUBÉN R, CONCEPCIÓN M. A genetic algorithm for hybrid flow shops with sequence dependent setup times and machine eligibility[J]. European Journal of Operational Research, 2007, 169(3): 781-800.

作者简介: 吕阳, 男, 1996 年出生。主要研究方向为智能算法与优化调度。

E-mail: 726564418@qq.com

钱斌(通信作者), 男, 1976 年出生, 教授, 博士研究生导师。主要研究方向为优化调度理论与方法。

E-mail: bin.qian@vip.163.com

胡蓉, 女, 1973 年出生, 副教授, 硕士研究生导师。主要研究方向为优化方法和决策支持系统。

E-mail: ronghu@vip.163.com

张梓琪, 男, 1989 年出生, 博士研究生。主要研究方向为智能算法与优化调度。

E-mail: 768894018@qq.com