李伟铭, 王德贤, 杨敬辉. 求解置换流水车间调度问题的混合乌鸦搜索算法[J]. 智能计算机与应用,2025,15(7):83-92.
DOI:10.20169/j.issn.2095-2163.250712

求解置换流水车间调度问题的混合乌鸦搜索算法

李伟铭[1], 王德贤[2], 杨敬辉[1,2,3]

(1 上海第二工业大学 计算机与信息工程学院, 上海 201209; 2 上海第二工业大学 智能制造与控制工程学院, 上海 201209;
3 武义智能制造产业技术研究院, 浙江 金华 321200)

摘 要: 关于置换流水车间调度问题(Permutation Flow-shop Scheduling Problem,PFSP),提出一种混合乌鸦搜索算法,用来实现最大完工时间的最小化,以期不断改善企业生产成本和效率,帮助创造更多收益。首先,对基于 NEH 的启发算法进行了改进,提出了一种新方法,用于改善初始种群的质量和多样性;其次,引入 SPV 规则进行编码,使算法能够有效处理离散调度问题;最后,为增强对解空间搜索能力,选取适应度值在前 20%的个体进行局部操作,同时设计出一种全新邻域结构,且规模能够自适应变化的动态邻域搜索算法,动态改变局部搜索能力,真正实现广域、局域两种搜索平衡,最终大幅增强混合算法性能,使问题得到更有效处理。结合实际情况,本文以 Rec 与 Taillard 两种测试集完成算法性能测试,并与当前处理 PFSP 问题效果显著的元启发式算法进行对比分析,从而做出准确判断。混合乌鸦搜索算法在最佳相对误差和平均相对误差方面的表现显著,其平均值相较其他算法至少降低了 88.3%和 87.5%,证明该算法在寻优效率和稳定性方面的显著优势,突出了其在复杂问题求解中的高效性和可靠性。

关键词: 乌鸦搜索算法; 置换流水车间; 种群初始化; 自适应动态邻域搜索
中图分类号: TP301 **文献标志码**: A **文章编号**: 2095-2163(2025)07-0083-10

Hybrid crow search algorithm for permutation flow-shop scheduling problem

LI Weiming[1], WANG Dexian[2], YANG Jinghui[1,2,3]

(1 School of Computer and Information Engineering, Shanghai Second Polytechnic University, Shanghai 201209, China;
2 School of Intelligent Manufacturing and Control Engineering, Shanghai Second Polytechnic University, Shanghai 201209, China;
3 Wuyi Intelligent Manufacturing Industry Technology Research Institute, Jinhua 321200, Zhejiang, China)

Abstract: Regarding the Permutation Flow-shop Scheduling Problem (PFSP), this paper proposes a hybrid crow search algorithm through research and comparison to minimize the maximum completion time, in order to continuously improve the company's production costs and efficiency, and help the company create more profits. Firstly, this article improves the heuristic algorithm based on NEH (Nawaz-Enscore-Ham) and proposes a new method to improve the quality and diversity of the initial population; second, the SPV (Smallest-Position-Value) rule is encoded to enable the algorithm to handle discrete scheduling problems; finally, to enhance the search ability for solution space, the individuals with the top 20% of the fitness value are selected for local operation, and a dynamic neighborhood search method in which the neighborhood size can change adaptively is designed in the local search to achieve dynamic adjustment of the local search ability, thus helping the algorithm to achieve a balance between wide area search and local area search and further improving the performance of the hybrid algorithm. By testing the performance of the algorithm using Rec and Taillard comparing it with the current metaheuristic that is going to work best in solving the PFSP problem, INHSA performs significantly in terms of optimal relative error and average relative error, with an average decrease of at least 88.3% and 87.5% compared to other algorithms. This data clearly demonstrates the significant advantages of the algorithm in terms of optimization efficiency and stability, highlighting its efficiency and reliability in solving complex problems.

Key words: crow search algorithm; permutation flow shop; population initialization; adaptive dynamic neighborhood search

0 引 言

根据设备环境(工件在加工设备上的流动方式)的不同,生产调度问题可分为多种类型,如流水车间调度、单机调度和并行调度等。而按照类型差异,需要选择对应方法来处理,从而达到最理想效

作者简介: 李伟铭(1999—),男,硕士,主要研究方向:优化算法,车间调度; 王德贤(1988—),男,硕士,工程师,主要研究方向:智能优化与调度,数字孪生。
通信作者: 杨敬辉(1968—)男,博士,教授,主要研究方向:精益生产,智能制造。Email:l.lippmann@foxmail.com。

收稿日期: 2023-12-04

果。其中,PFSP 属于典型流水车间调度类型,不管在流程工业或在装备制造业,均具备广阔应用场景。通过高效提供各种优质生产调度方案,为公司生产决策提供依据,将显著改善公司资源利用率、生产成本及效率,借此帮助公司抢占市场机遇,不断增强公司市场竞争力,占据市场主导地位。研究表明,PFSP 是 NP-hard 问题[1],由于一些因素影响,现阶段仍求解困难。因此,对其研究有着重大现实意义。

早在 1954 年,Johnson[2] 率先提出了 PFSP 模型,随之众多学者纷纷对其展开研究工作,并提出了大量的求解方法。关于求解方法,能够划分成两大类:最优求解法与近似求解法。在探讨最优求解法时,国际学界通过长期研究,推出了众多实用算法。其中包括分支定界[2]、整数或动态规划、Lagrange 松弛法等。由于这些方法计算时间较长,主要适用于处理小规模的 PFSP。在近似求解法领域,存在两大类算法。首先是启发式算法,涵盖了如 Johnson 规则[3]、NEH[4] 等方法,特别是 NEH[5-9] 在处理 PFSP 问题上表现最佳。然而,该类算法虽能快速提供解决方案,但其解的质量难以保证。二是元启发式算法。当前,结合多种策略以提高算法效率和质量的新颖元启发式算法成为研究热点。在求解 PFSP 问题时,常用的元启发式算法包括混合差分进化算法[10](LHDE)。混合共生生物搜索[11](HSOS)算法、混合蝙蝠算法[12](HBA)、新布谷鸟搜索[13](NCS)算法等等。

乌鸦搜索算法(Crow Search Algorithm,CSA)[14] 作为一种基于群智能的优化算法,其设计灵感源自于对乌鸦的记忆能力、交流技巧以及其储存食物、相互跟随窃取食物等社会行为的研究。相比传统的群智能优化算法,CSA 具有算法参数少、鲁棒性强、可扩展性好、简单易用等优点。虽然现阶段对其研究很少,但是 CSA 在图像分割、故障诊断、特征提取等领域的成功应用,揭示出其有较好的解决优化问题的潜力,值得期待。Huang 等[15] 提出了混沌乌鸦搜索算法(Hybrid Crow Search Algorithm,HCSA),主要针对以完工时间最小化为目标的 PFSP 问题,实用性较强,可有效缩短完工时间;闫红超[16] 等提出了一种新混合乌鸦搜索算法(NEW Hybrid Crow Search Algorithm,NHCSA),用于求解以完工时间最小化为目标的 PFSP,是目前仅能查阅到的采用 CSA 求解该问题的研究成果;但是该算法较复杂,实用效果尚有很大改进空间,必须进行更深入研究与探索,才能发挥其更大价值。

为此,本文经过深入研究,提出一种新混合乌鸦搜索算法(INHCSA)。首先,改进了一种基于 NEH 的启发式算法——NEHLJP1[6],有效提高了初始种群的质量与多样性。继而运用 SPV 规则[13] 进行编码,最终在自适应动态邻域搜索算法的加成下设计了一种更强寻优能力和稳定性的混合乌鸦算法。

1 置换流水车间调度问题

PFSP 是典型的生产规划问题。若有 n 个工件 $\{J_1,J_2,\cdots,J_n\}$ 遵循相同的顺序,依次在串行排列的 m 台机器 $\{M_1,M_2,\cdots,M_n\}$ 上加工。已知各个工件在各台机器上的加工时间,要求找到工件在各机器上的加工顺序(工件排序),使得调度目标最优。对于 PFSP 通常做出如下假设:

1)各工件最多可以被各设备加工 1 次;

2)在每一时刻,各设备最多可以加工 1 个工件、各工件最多可以被 1 台设备加工;

3)加工期间不允许中断;

4)不考虑准备时间;

5)各工件在各设备加工时间为定值,且不受加工次序影响;

6)各设备上工件加工次序完全一致。

按照实际情况可知,最具代表性 PFSP 当属以最大完工时间最小化为目标,通常被表示为:

$$F_m \mid prmu \mid C_{\max}$$

其中,$P_{i,k}$ 为工件 J_i 在机器 M_k 上的加工时间,$(i=1,2,\cdots,n,k=1,2,\cdots,m)$;$\pi=\{\pi(1),\pi(2),\cdots,\pi(n)\}$ 为任意一个调度方案;$C(\pi(i),k)$ 为工件 $\pi(i)$ 在机器 M_k 上的完工时间,$\pi(i)\in\{J_1,J_2,\cdots,J_n\}$。则有:

$$C(\pi(1),1)=P_{\pi(1),1} \tag{1}$$

$$C(\pi(i),1)=C(\pi(i-1),1)+P_{\pi(i),1} \tag{2}$$

$$C(\pi(1),k)=C(\pi(1),k-1)+P_{\pi(1),k} \tag{3}$$

$$C(\pi(i),k)=\max\{C(\pi(i-1),k),\ C(\pi(i),k-1)\}+P_{\pi(i),k} \tag{4}$$

上述公式中,$i=2,3,\cdots,n$;$k=2,3,\cdots,m$。

最大完工时间 $C_{\max}$ 为最后一个工件 $\pi(n)$ 在最后一台机器 M_m 上的完工时间,即:

$$C_{\max}(\pi)=C(\pi(n),m) \tag{5}$$

求解 $F_m \mid prmu \mid C_{\max}$,为找到一个最佳工序 π^*,使得

$$C_{\max}(\pi^*)\leqslant C_{\max}(\pi),\ \forall\pi\in\Pi \tag{6}$$

式中 Π 表示所有调度方案的集合。

2　混合乌鸦搜索算法求解 PSFP 问题

2.1　编码方式与 SPV 转换

在深入探索乌鸦搜索算法时，国际学界认为其适用性主要针对连续问题的优化，并不直接适用于 PFSP 的求解。针对此限制，采用 SPV 规则将连续值位置转换成工件排序，以此适配 PFSP 问题。此转换过程遵循以下步骤：从一系列位置值中，选取最小值对应的维度，作为接下来要加工的工件，持续此过程直至所有位置值均被遍历。

如图 1 所示，空间维度 D 取值为工件的数量 n。依据 SPV 规则，将 6 维空间中乌鸦 i 的位置 x_i = [1.35，-2.46，-1.52，2.31，0.52，-1.68] 转换为工件排序 π_i = [2，6，3，5，1，4]。反之，可根据工件排序调整代表位置连续值的顺序，使之符合 SPV 规则。

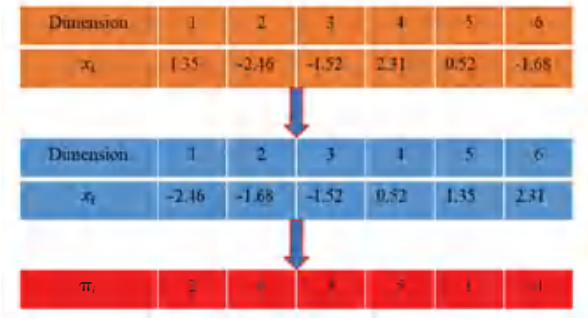

图 1　SPV 规则

Fig. 1　SPV Rules

将乌鸦 i 对应的工件排序 π_i 的完工时间 $C_{\max}(\pi_i)$ 作为乌鸦 i 的适应度值 $f(x_i)$，即：

$$f(x_i) = C_{\max} \tag{7}$$

显然，$f(x_i)$ 最小者就是最优个体。

2.2　种群初始化

一个良好的初始种群可以增强算法性能，而 NEH 启发式算法可以短时间生成优质解。结合 NEH 进行种群初始化，能够保证初始种群中有质量较高的个体，因而是最常用的方法。

鉴于 NEH 具有简单、有效的特点，一部分学者通过全面研究，发布了一系列基于 NEH 的改进算法，方便不同场景使用，如，NEHD[5]、NEHLJP1[6]、NEHFF[7]、NEHKK1[8] 等。其中，NEHLJP1 是寻优能力最强的一种算法。但在实践应用阶段，NEHLJP1 同样有着一些不足之处。NEHLJP1 在最后一个工件的调度过程中，往往会出现多个工件排序，其完工时间最小但顺序不同。然而，最终只能选择其中一个作为算法的结果，这样就会丢失掉高质量解的信息。

1）对 NEHLJP1 改进

为了弥补上述不足，对 NEHLJP1 算法进行了精细而有益的改良。具体改良措施为：在调度的最后阶段，如果出现多个具有相同最小完工时间但工件排序不同的情形，则将这些排序方案均予以保留。由于新增的处理机制无需对工件排序做出优劣判定，改进后的 NEHLJP1 算法的计算复杂度得以保持不变，仍为 $O(mn^2)$。

改进后的 LEHLJP1 步骤如下：

Step 1　计算 $c_i, i = 1,2,\cdots,n$，按照 c_i 非增的顺序对工件进行排序，得到优先序列 γ。令 $\pi = \{\gamma(1)\}, K = card(\pi) = 1$。$c_i$ 的计算方法：

$$c_i = AVG_i + STD_i + abs(SKE_i) \tag{8}$$

式中：$abs(SKE_i)$ 表示 SKE_i 的绝对值，AVG_i、STD_i 和 SKE_i 的计算方法如下：

$$AVG_i = \frac{1}{m}\sum_{j=1}^{m} P_{i,j} \tag{9}$$

$$STD_i = \sqrt{\frac{1}{m-1}\sum_{j-1}^{m}(P_{i,j} - AVG_i)^2} \tag{10}$$

$$SKE_i = \frac{\frac{1}{m}\sum_{j-1}^{m}(P_{i,j} - AVG_i)^3}{\left(\sqrt{\frac{1}{m}\sum_{j-1}^{m}(P_{i,j} - AVG_i)^2}\right)^3} \tag{11}$$

Step 2　令 $r = \gamma(K+1)$，将工件 r 插入 π，根据下式计算出使完工时间最大的插入位置集合 S：

$$S = \min_k C_{\max}(\pi(r,k)) \tag{12}$$

其中，$K = 1,2,\cdots,K+1$，$\pi(r,k)$ 表示将工件 r 插入 π 中位置 k 得到的工件排序。

Step 3　令 $K = K+1, h = card(S)$，判断：

若 $h = 1$，令 $\pi = \pi(r, S(1))$，并进一步判断：如果 $K < n$，继续执行 Step 2；否则，将 π 作为最终的调度序列输出，算法结束；

若 $h > 1$，则进一步判断：如果 $K < n$，则根据 TB 机制[6] 确定 r 在 π 中插入的位置 k^*，令 $\pi = (\pi(r, k^*))$，继续执行 Step 2；否则，将 $\pi_l = \pi(\pi, S(l))$，$l = 1,2,\cdots,h$，共 h 个工件排序均作为算法的结果输出，算法结束。

2）种群初始化

为保证初始种群质量与多样性，在改进的

NEHLJP1 基础上,本文给出了种群初始化新方法。首先,通过改良版的 NEHLJP1 算法生成一组工件序列;利用这些工件序列,初始化种群中大约 10% 的个体,确保每个工件序列产生的个体数量基本均匀;最后,通过随机方式产生种群中的剩余个体。具体步骤如下:

Step 1 基于改进的 NEHLJP1 算法产生一组数量为 $h(h > 1)$ 的工件排序 $\pi_l, l = 1,2,\cdots,h$。

令 IntNum = $\lceil N/10 \rceil$,若 $h >$ IntNum,则从中随机选择 IntNum 个工件排序用于初始化,并且令 $h =$ IntNum。

Step 2 根据下式,使用 π_l 初始化数量约为 $K = \lceil \text{IntNum}/10 \rceil$ 的个体,$l = 1,2,\cdots,h$:

$$x^0_{i,j} = x_{\max} - (x_{\max} - x_{\min}) \times (\pi_i(j) - 1)/n \quad (13)$$

式中:$x_{\max} = 5, x_{\min} = -5, j \in \{1,2,\cdots,n\}$,

$i \in \{(l-1) \times K + 1,\cdots,\min(l \times K, \text{IntNum})\}$

Step 3 随机生成剩余个体:

$$x^0_{(i,j)} = x_{\max} - (x_{\max} - x_{\min}) \times r_i \quad (14)$$

式中:$i \in \{\text{IntNum} + 1,\cdots,N\}$, $j = \{1,2,\cdots,n\}$, r_i 为(0,1)区间均匀分布的随机数。

显然,相比文献[10]、[11]和[13]采用的基于 NEH 对种群中 10%的个体进行初始化的方法,新的初始化方法使得种群中含有 10%质量更好并且存在差异的个体,因而初始种群的质量和多样性都得到了进一步的改善。

2.3 自适应随机动态邻域搜索算法

相较之下,面对一些复杂问题,CSA 有着易发生局部最优、易早熟收敛等问题,造成结果不够理想,需要合理优化与改进。为尽可能避免上述情况发生,选取适应度值在前 20%的个体进行局部操作,并且考虑到对于前 20%的个体完成局部操作后结果可能变差,因此通过模拟[17]退火按照相应概率得到次优解,防止发生局部最优问题。

针对 PFSP 等组合优化问题,局部搜索直接决定算法处理效果。传统的邻域结构主要是两点交换和插入,两者相对简单且方便实现,因此对于生产调度问题,许多算法都采用这两种邻域结构[18]。李坤等[19]针对 PSO 设计并运用了一种变邻域搜索算法(Variable Neighbourhood Search, VNS),促使最终结果更加合理。由于该算法流程相对复杂,本文提出采用全新邻域结构(基于插入移动)算法来实现。对于一个解 $\pi_0 = (\pi(1),\pi(2),\cdots,\pi(n))$,其局部搜索流程为:

Step 1 令迭代次数 $t = 1$,设定当前的最好解 $\pi_b = \pi_0$。

Step 2 从解 π_b 中随机选择一个位置 r,再从开始位置连续删除 d 个工件。如果 $n - r + 1 \geqslant d$,则连续删除工件 $\pi(r)$, $\pi(r+1)$, $\cdots$, $\pi(r+d-1)$;反之,删除工件 $\pi(r),\cdots,\pi(n)$,以及工件 $\pi(1),\cdots,\pi(d-(n-r+1))$(详见图 2)。为快捷简述,把被删除工件次序简记成 $\pi(r_1),\pi(r_2),\cdots,\pi(r_d)$。

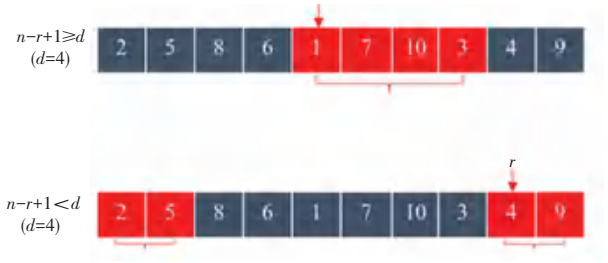

图 2 工件删除方法示意图

Fig. 2 Schematic diagram of workpiece deletion method

Step 3 从 $\pi(r_1)$ 开始,将其插入当前部分解内最优处;循环执行这一流程,直至这些被删除工件均插入部分解内,从而生成新的完整解 π'。

Step 4 如果目标函数值 $C_{\max}(\pi') < C_{\max}(\pi_b)$,则令 $\pi_b = \pi'$。

Step 5 令 $t = t + 1$,如果 $t > t_{\max}$,则停止;否则转到 Step 2 执行。

鉴于邻域规模直接决定算法性能,假定初始期间邻域较大,算法必定消耗较多计算时间,而在算法的后期,其更加偏向于局部搜索,容易导致局部最优解的出现。因此,对于邻域规模的设计需求,本文采取了自适应策略,即通过评估全部邻域规模的历史搜索效果,来判断各邻域规模的选择概率。

在此,定义 $N(d)$ 为移除连续 d 个工件后,逐个重新安置这些工件至剩余序列中最合适位置形成的邻域,$d_{\max}$ 表示 d 的最大取值,而 P_k 表示第 k 种邻域的选择概率,t 表示迭代次数,$t_{\max}$ 表示最大允许的迭代次数,则自适应随机动态邻域搜索算法流程如图 3 所示。

由于 P_k 按其使用后能否优化当前解进行更新,所以算法可自适应选取最适合邻域类型,最终防止类型由大至小单一变化,保证邻域搜索深度自适应变化,增强其搜索能力,提高所得结果的科学性与合理性。

由此可见,最佳工件排序即为种群最优个体所对应的工件排序(如图 4 所示)。

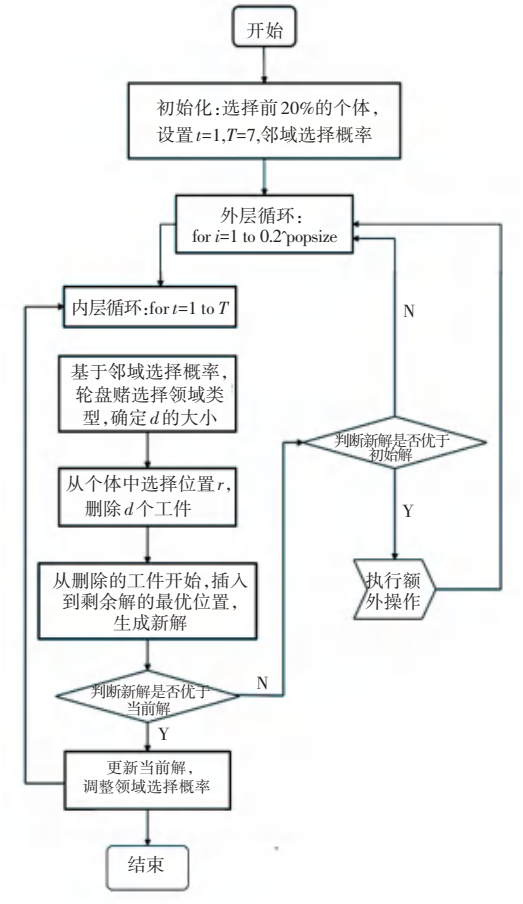

图 3　自适应随机动态邻域搜索算法流程图

Fig. 3　Algorithm flowchart of dynamic neighborhood search method

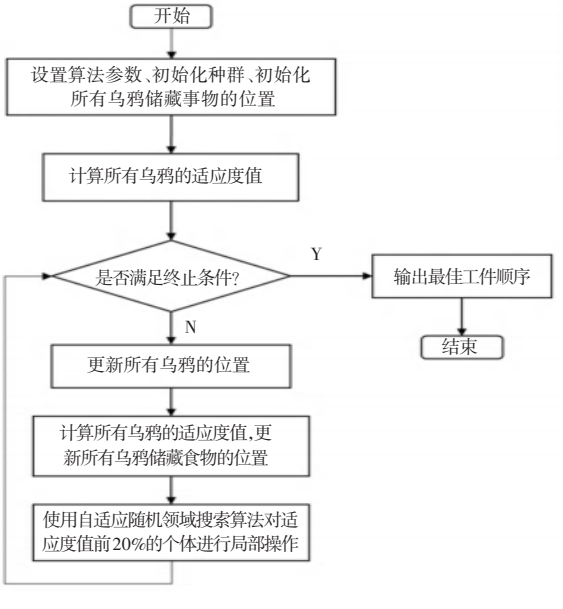

图 4　INHCSA 的算法流程图

Fig. 4　Algorithm flowchart of INHCSA

3　实验与分析

3.1　实验环境及参数设置

实验环境为：Windows10 64 位 OS，主频 3.6 GHz 的 Intel Core i7-4790 CPU、8.0 GB 内存，编程语言及版本为 MATLAB R2019b。

基于 Rec 与 Taillard 两种测试集完成各种算法对比分析。其中，Rec 测试集由 21 个算例组成，Taillard 由 120 个算例（各种规模 10 个，共 12 种规模）组成。针对 Taillard ，为提高验证结果代表性，常用的做法是从每个规模的算例中选取第一个或最后一个组成测试集进行验证。Rec 与 Taillard 算例由 OR-LIBRARY 获得。

实验中，各算例均独立运行 20 次。利用最佳相对误差（BRE）、平均相对误差（ARE）进行对比，计算公式如下：

$$BRE = \frac{C_{\text{best}} - C^*}{C^*} \tag{15}$$

$$ARE = \frac{C_{\text{avg}} - C^*}{C^*} \tag{16}$$

式中：C_{best} 为规定运行次数内完工时间的最小值，C_{avg} 为规定运行次数内完工时间的平均值 C^*，对于 Rec 测试集为算例的已知最优解、对于 Taillard 测试集为算例的已知上界值。使用 $\overline{BRE}$、$\overline{ARE}$ 表示不同规模算例的 BRE、ARE 的平均值。

INHCSA 包含 N、G、AP、fl、T_p 和 d 共 6 个参数，为了确定这些参数的取值，以最大迭代次数作为算法终止的条件，将各参数分别定义成一个因素、各因素设定 3 个水平，选择 $L18(3^6)$ 正交表基于算例 Rec037 进行正交实验。使用每种参数组合独立进行 20 次仿真，将完工时间的平均值作为响应变量（Response Variable，RV）。表 1 给出了因素水平的设置，表 2 给出了正交实验的结果，图 5 给出了不同参数设置对算法性能的影响趋势。

表 1　因素水平

Table 1　Factor level

水平	N	G	AP	fl	T_p	d
1	30	1 000	0.1	1	0.4	3
2	40	2 000	0.2	3	0.6	5
3	50	3 000	0.3	5	0.8	7

表 2 正交实验结果

Table 2 Orthogonal experimental results

参数组合编号	因素水平						RV
	N	G	AP	d	T_P	fl	
1	30(1)	1 000(1)	0.1(1)	1(1)	0.4(1)	3(1)	5 038.5
2	30(2)	2 000(2)	0.2(2)	3(2)	0.6(2)	5(2)	5 025.1
3	30(3)	3 000(3)	0.3(3)	5(3)	0.8(3)	7(3)	5 018.8
4	40(1)	1 000(1)	0.1(1)	5(3)	0.6(2)	5(2)	5 030.1
5	40(2)	2 000(2)	0.2(2)	1(1)	0.8(3)	7(3)	5 017.3
6	40(3)	3 000(3)	0.3(3)	3(2)	0.4(1)	3(1)	5 028.7
7	50(1)	1 000(1)	0.2(2)	3(2)	0.8(3)	3(1)	5 055.4
8	50(2)	2 000(2)	0.3(3)	5(3)	0.4(1)	5(2)	5 047.3
9	50(3)	3 000(3)	0.1(1)	1(1)	0.6(2)	7(3)	5 014.7
10	30(1)	1 000(1)	0.3(3)	3(2)	0.6(2)	7(3)	5 009.4
11	30(2)	2 000(2)	0.1(1)	5(3)	0.8(3)	3(1)	5 029.4
12	30(3)	3 000(3)	0.2(2)	1(1)	0.4(1)	5(2)	5 034.5
13	40(1)	1 000(1)	0.2(2)	5(3)	0.4(1)	7(3)	5 028.4
14	40(2)	2 000(2)	0.3(3)	1(1)	0.6(2)	3(1)	5 039.3
15	40(3)	3 000(3)	0.1(1)	3(2)	0.8(3)	5(2)	5 019.9
16	50(1)	1 000(1)	0.3(3)	1(1)	0.8(3)	5(2)	5 028.8
17	50(2)	2 000(2)	0.1(1)	3(2)	0.4(1)	7(3)	5 003.6
18	50(3)	3 000(3)	0.2(2)	5(3)	0.6(2)	3(1)	5 034.1
k1	5 025.9	5 031.7	5 022.7	5 037.5	5 030.2	5 028.8	
k2	5 027.2	5 027.0	5 032.4	5 030.9	5 025.4	5 023.6	
k3	5 030.6	5 025.1	5 028.7	5 015.3	5 028.2	5 031.3	
极差	4.7	6.6	9.7	22.2	4.8	7.7	
等级	6	4	2	1	5	3	

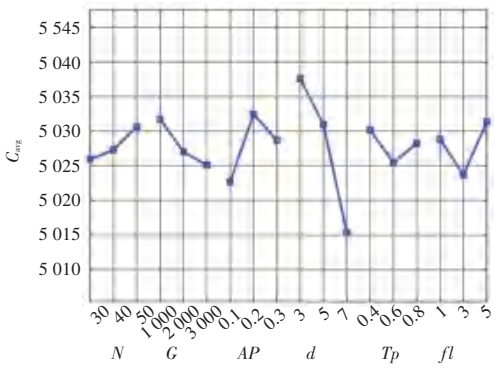

图 5 不同参数设置对算法性能的影响趋势

Fig. 5 Impact trend of different parameter settings on algorithm performance

从表 2 可知,参数会对算法性能造成一定影响。按照由大至小顺序进行排列,分别为 d、AP、fl、G、Tp 和 N。从图 5 中可以看出,当 d 取值较大时,既可以增强其摆脱局部最优的能力,又可以增加收敛精度;认知概率 AP 决定了乌鸦在搜索过程中的集中或分散,取 0.1 时效果最好;fl 在乌鸦搜索算法中对平衡算法的全局搜索和局部搜索有着重要的意义,不能太大也不能太小;G 较大的迭代次数 有助于算法进行更深入的搜索,但超过 2 000 时,G 的增大对算法寻优能力的提升变缓;N 和 Tp 对算法的影响较小。

基于上述分析,参数设置为:d = 7、N = 40、G = 2 000、fl = 3、AP = 0.1、Tp = 0.6 进行算法比较。其余算法的数据和参数若无专门说明均来自相应文献。

3.2 算法分析与比较

为了验证算法性能,将 INHCSA 与 NHCSA、HSOS、NCS、MCTLBO[20]、LHDE[10] 以及 HGA[21] 算法进行比较。

3.2.1 与 NHCSA、CSA 和 HCSA 算法比较

为验证本文算法有效性,将 INHCSA 算法与基本 NHCSA、HCSA 算法相比较,比较结果见表 3。

表 3　INHCSA 与 NHCSA、HCSA 的比较结果

Table 3　Comparison results of INHCSA, NHCSA, and HCSA

算例	n	m	C^*	HCSA		NHCSA		INHCSA	
				BRE	*ARE*	*BRE*	*ARE*	*BRE*	*ARE*
Ta001	20	5	1 278	**0**	**0**	**0**	**0**	**0**	**0**
Ta011	20	10	1 582	**0**	**0**	**0**	0. 006	**0**	**0**
Ta021	20	20	2 297	0	0. 18	**0**	0. 015	**0**	**0**
Ta031	50	5	2 724	0. 18	0. 18	**0**	**0**	**0**	**0**
Ta041	50	10	2 991	1. 24	1. 70	1. 137	1. 184	**1. 036**	**1. 159**
Ta051	50	20	3 850	5. 79	6. 25	0. 805	1. 127	**0. 051**	**0. 328**
Ta061	100	5	5 493	0	0. 02	0	0	**0**	**0**
Ta071	100	10	5 770	1. 31	1. 50	0	0. 004	**0**	**0**
Ta081	100	20	6 202	1. 59	2. 10	**1. 016**	**1. 528**	1. 402	1. 699
Ta091	200	10	10 862	1. 24	2. 56	0. 064	0. 089	**0. 018**	**0. 024**
Ta101	200	20	11 195	1. 26	1. 39	0. 777	1. 232	**0. 757**	**0. 932**
Ta111	500	20	26 059	0. 80	1. 06	0. 368	0. 544	**0. 257**	**0. 376**
AVG				1. 11	1. 41	0. 347	0. 477	**0. 293**	**0. 376**

由表 3 可以看出，在针对 *BRE* 的比较中，除了 Ta081 以外，INHCSA 无论在 *ARE* 的比较中还是在 *BRE* 的比较中，对其它 11 个算例的计算结果都优于 NHCSA 和 HCSA，从而证明 NHCSA 具有更强的寻优能力和更好的稳定性。图 6 给出了 INHCSA 与 NHCSA、HCSA、CSA 对 TA031、TA051、TA071、TA091 的收敛曲线。通过图例不难发现，针对以上各种算例，INHCSA 初始值、收敛速度与精度均优于其他算法。根本原因在于新种群初始化法可以增强其初始种群质量与多样性，提升了算法迭代的效率；基于改进贪婪迭代的局部搜索，可以优化收敛速度与精度。

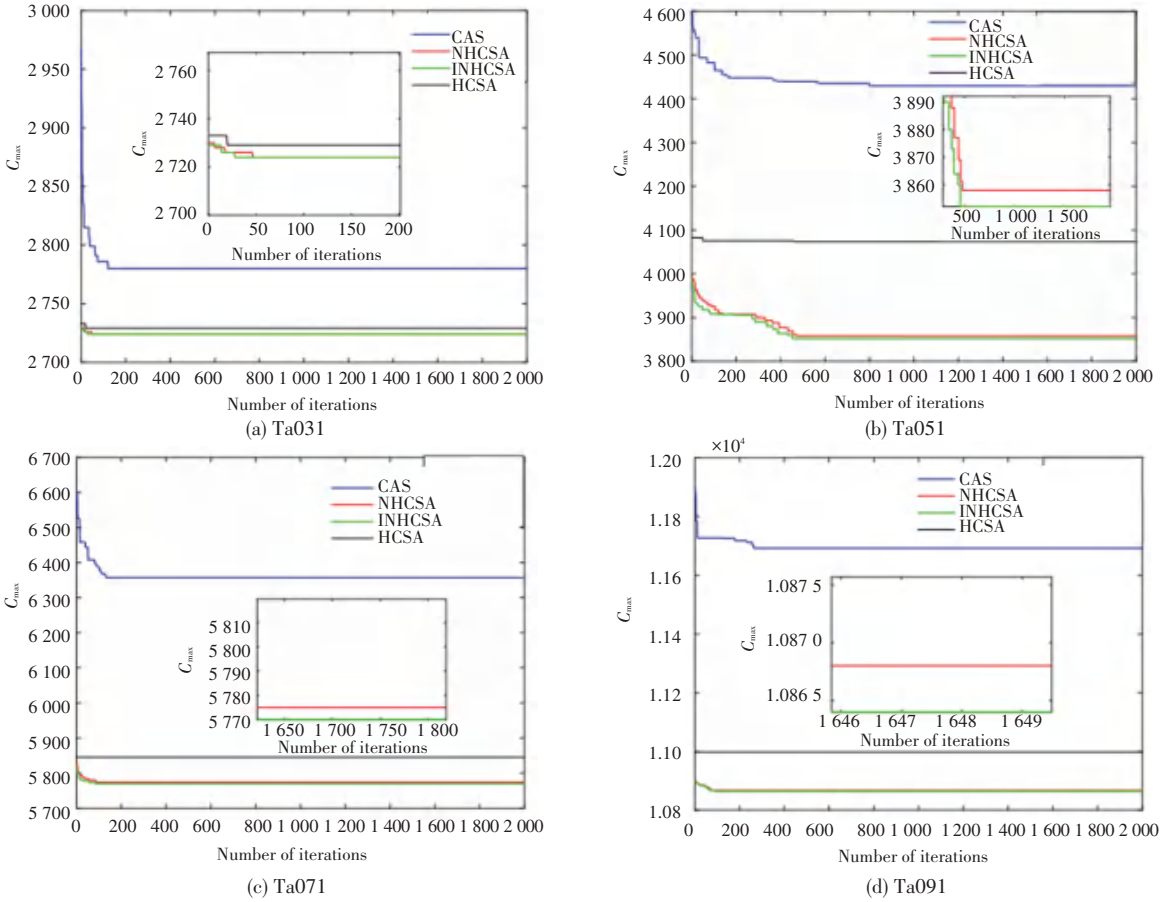

图 6　收敛曲线

Fig. 6　Convergence curves

3.2.2 与 NHCSA 等算法针对 Taillard 测试集比较

将 INHCSA 与 NHCSA、HSOS、NCS、HGA 及 LHDE 相比较,比较结果见表 4。

表 4 Taillard 测试集比较结果

Table 4 Comparison results of taillard test set

算法	算例	Ta010	Ta020	Ta030	Ta040	Ta050	Ta060	Ta070	Ta080	Ta090	Ta100	Ta110	Ta120	AVG
	n	20	20	20	50	50	50	100	100	100	200	200	500	
	m	5	10	20	5	10	20	5	10	20	10	20	20	
	C^*	1 108	1 591	2 178	2 782	3 065	3 756	5 322	5 845	6 434	10 675	11 288	26 457	
HSOS	C_{avg}	1 108	1 601	2 205	2 782	3 140	3 887	5 326	5 898	6 650	10 798	11 698	26 780	
	ARE	0	0.629	1.240	0	2.447	3.488	0.075	0.907	3.357	1.152	3.632	1.224	1.513
NCS	C_{avg}	1 108	1 606	2 184	2 782	3 131.2	3 860.6	5 326	5 891.4	6 602.8	10 734	11 633.6	26 897.2	
	ARE	0	0.943	0.275	0	2 160	2.785	0.075	0.794	2.624	0.553	3.062	1.664	1.244
HGA	C_{avg}	1 345.5	2 019.3	2 956.3	3 341.5	4 279	5 963.5	6 542.5	8 255.3	10 928.8	15 869.1	20 587.6	49 545.6	
	ARE	21.435	26.920	35.735	20.111	39.608	58.773	22.933	41.237	69.860	48.657	82.385	87.268	46.244
LHDE	C_{avg}	1 108	1 598.4	2 185		3 116.5	3 821.1	5 322	5 881					
	ARE	0	0.465	0.321		1.680	1.733	0	0.616					
NHCSA	C_{avg}	1 108	1 591	2 178.6	2 782	3 085.9	3 787.1	5 322	5 849.1	6 497.2	10 680.4	11 410.4	26 573.7	
	ARE	0	0	0.025	0	0.680	0.828	0	0.070	0.982	0.050	1.084	0.441	0.347
INHCSA	C_{avg}	1 108	1 591	2 178	2 782	3 085	3 770.9	5 322	5 845	6 483.4	10 678.1	11 385.6	26 547.4	
	ARE	0	0	0	0	0.652	0.328	0	0	0.767	0.024	0.932	0.376	0.256

根据文献[10],LHDE 算法针对 Taillard 测试集的计算结果仅涵盖了 Ta010 至 Ta030 及 Ta050 至 Ta080 这 7 个算例。将 INHCSA 与 NHCSA、LHDE、HGA、NCS 和 HSOS 算法相比较。可以看出,INHCSA 取得了 11 个最优值,NHCSA 取得了 4 个最优值,依次是 LHDE、NCS、HSOS 和 HGA。可见 INHCSA 在最优值的数量上均优于其它算法,从而证明了 INHCSA 具有更强的寻优能力。

3.2.3 与 NHCSA 等算法针对 Rec 测试集比较

由于文献[13]中的 NCS 算法未提供 Rec 测试集的计算结果,表 5 仅列出了 HSOS、MCTLBO、HGA、LHDE、NHCSA 和 INHCSA 算法的比较结果。为了更直观地展示比较结果,图 7、图 8 分别给出了各算法求得的最佳相对误差和平均相对误差,图 9 给出了各算法针对不同规模算例求得的最佳相对误差和平均相对误差平均值的比较结果。

表 5 Rec 测试结果比较

Table 5 Comparison of Rec test results

算法	算例	Rec01	Rec03	Rec05	Rec07	Rec09	Rec11	Rec13	Rec15	Rec17	Rec19	Rec21	Rec23	Rec25	Rec27	Rec29	Rec31	Rec41	AVG
	n	20	20	20	20	20	20	20	20	20	30	30	30	30	30	30	50	75	
	m	5	5	5	10	10	10	15	15	15	10	10	10	15	15	15	10	20	
	C^*	1 247	1 109	1 242	1 566	1 537	1 431	1 930	1 950	1 902	2 093	2 017	2 011	2 513	2 373	2 287	3 045	4 960	
HSOS	BRE	0	0	0	0	0	0	0	0	0	0.620	1.437	0.348	0.835	0.969	0.831	0.427	2.388	0.583
	ARE	0	0	0	0	0	0	0.273	0.523	1.388	1.274	1.537	1.280	2.067	1.432	2.488	0.644	3.350	1.050
MCTLBO	BRE	0	0	0	0	0	0	0	0	0	0.287	1.468	0.149	0.199	0.253	0	0.427	1.956	0.367
	ARE	0.120	0.068	0.217	0.632	0	0	0.192	0.356	0.037	0.430	1.557	0.686	0.809	1.016	0.822	1.307	2.061	0.747
HGA	BRE	0	0	0	0	0	0	0.360	0.560	0.950	0.620	1.440	0.400	1.270	1.100	1.400	0.430	3.640	0.860
	ARE	0.140	0.090	0.290	0.690	0.640	1.100	1.680	1.120	2.320	1.320	1.570	0.870	2.540	1.830	2.700	1.340	4.920	1.600
LHDE	BRE	0	0	0.242	0	0	0	0	0	0	0.287	0.645	0.348	0.557	0.253	0.831	0.427	2.661	0.502
	ARE	0	0	0.242	0	0.026	0	0.275	0.523	0.363	0.702	1.279	0.428	1.082	0.951	1.049	0.644	3.351	0.757
NHCSA	BRE	0	0	0	0	0	0	0	0	0	0	0.149	0.099	0	0.126	0	0.099	0.948	0.140
	ARE	0	0	0.242	0	0	0	0	0	0	0.287	0.645	0.348	0.557	0.253	0.831	0.427	2.661	0.502
INHCSA	BRE	0	0	0.242	0	0.026	0	0.275	0.523	0.363	0.702	1.279	0.428	1.082	0.951	1.049	0.644	3.351	0.757
	ARE	0	0	0	0	0	0	0	0	0	0	0.149	0.099	0	0.126	0	0.099	0.948	0.140

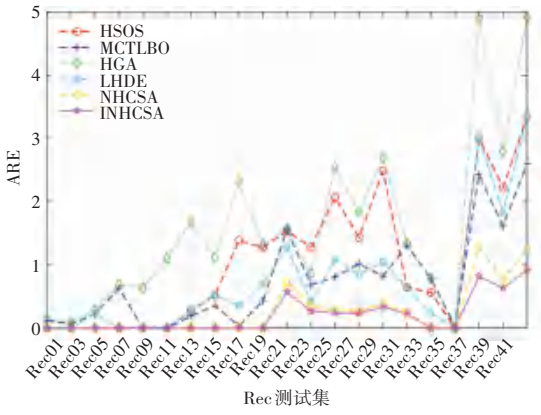

图 7　与 NHCSA 等算法针对测试集 ARE 的比较

Fig. 7　ARE comparison between NHCSA and other algorithms for the test set

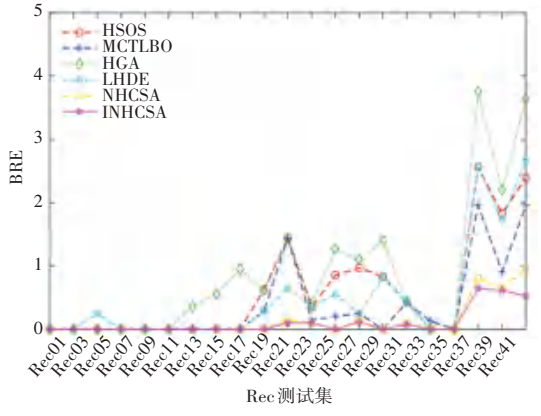

图 8　与 NHCSA 等算法针对测试集 BRE 的比较

Fig. 8　BRE comparison between NHCSA and other algorithms for the test set

从以上结果可以看出，在最佳相对误差的比较中，INHCSA 取得 21 个最优值，数量上排名第一，其次是 NHCSA 取得 14 个最优值，HSOS 和 MCTLBO 均取得了10个最优值，LHDE和HGA，分别取得9个和 7 个最优值；针对最佳相对误差的平均值 $\overline{BRE}$，INHCSA 的计算结果为 0. 10 排名第一，NHCSA 和 MCTLBO 平均值为 0. 14 和 0. 367，LHDE、HSOS 和 HGA 平均值分别为 0. 50、0. 58 和 0. 86。由此可见，无论是最优值的数量还是平均值，INHCSA 都优于其它算法，从而证明了 INHCSA 具有更强的寻优能力。

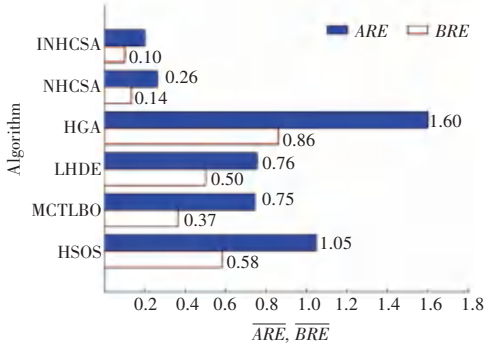

图 9　与 NHCSA 等算法针对 Rec 测试集 $\overline{ARE}$、$\overline{BRE}$ 的比较

Fig. 9　Comparison of NHCSA and other algorithms on ARE and BRE for Rec test sets

同样，在平均相对误差 $\overline{ARE}$ 的比较中，INHCSA 取得 21 个最优值，平均值为 0. 20，排名第一；NHCSA 取得 10 个最优值，平均值为 0. 26；HSOS 取得 6 个最优值，平均值为 1. 050；LHDE 取得 4 个最优值，平均值为 0. 757；MCTLBO 取得 2 个最优值，平均值为 0. 747；HGA 取得 1 个最优值，平均值为 1. 60；从而证明 INHCSA 具有更好的稳定性。

图 10 为算例 Rec21 的甘特图，对应的调度序列为[12,9,3,22,16,14,25,24,17,26,23,20,18,21,28,29,8,13,6,5,19,15,4,2,1,27,17,10,30,11]。

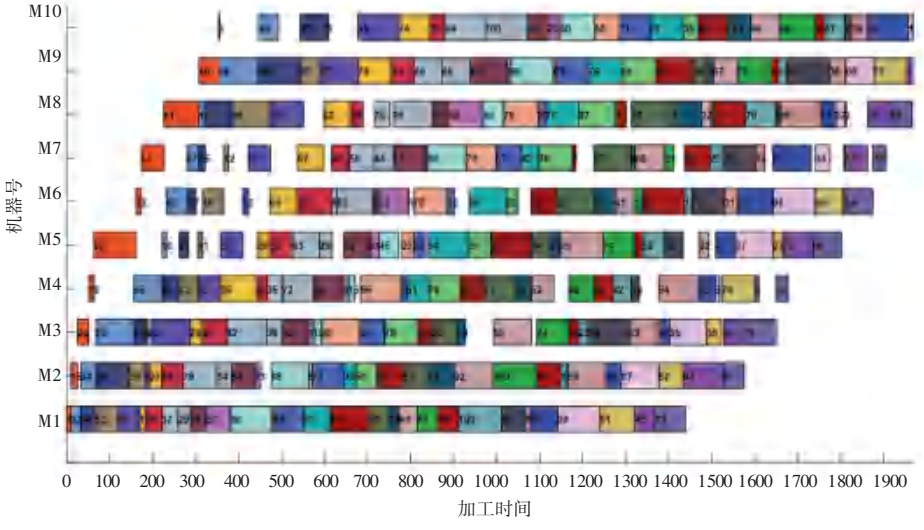

图 10　算例 Rec21 甘特图

Fig. 10　Gantt chart of example Rec21

为在统计学意义上阐述 6 种算法性能有无显著差异,可通过方差分析法深入分析表 4 中数据。6 种算法具有 95%置信区间的最小显著误差区间均值图,如图 11 所示。结合图例,若两区间不存在重叠处,则两者所代表个体在统计学意义上有显著差异。

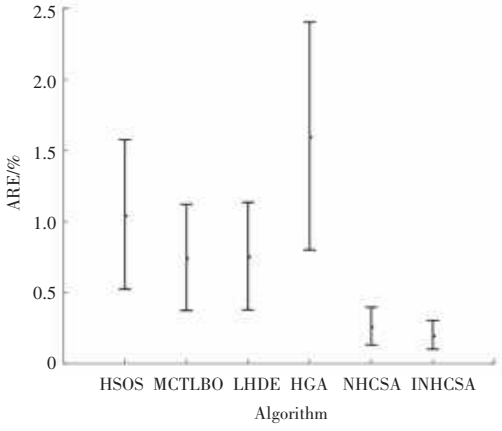

图 11 各种算法的 ARE 计算结果 95%置信区间均值图

Fig. 11 95% confidence interval mean plot of ARE calculation results for algorithms

在图 11 中,虽然 INHCSA 与 NHCSA 有大部分的重叠,但均值点低于 NHCSA。但较其他 4 种算法都不重叠,说明了算法在统计意义上明显好于 HSOS、MCTLBO、LHDE 和 HGA 这 4 种算法,说明连续算法与新提出自适应随机动态邻域策略相结合,可以取得更优异和稳定的结果。

4 结束语

针对处理最小化最大完工时间的置换流水车间调度问题,本文提出了一种创新的 INHCSA 算法。通过改良 NEH 算法,提升初始种群质量与多样性,优先优化前 20%个体,引入动态邻域结构,实现全局与局部搜索平衡。这些创新显著提高了混合算法的整体性能,使得所面临问题得到更有效的处理。针对算法性能的评估,本研究采用 Rec 与 Taillard 测试集,通过与当前先进的元启发式算法进行对比分析,从而进行准确评估。测试结果表明,这一改进方案显著提升了 CSA 算法的收敛速度和精度,进一步增强了其整体性能。与 NHCSA、HCSA、HSOS、MCTLBO、LHDE、NCS 和 HGA 等算法相比较,INHCSA 在寻优能力和稳定性方面展现出更为理想的表现。

参考文献

[1] GAREY M R, JOHNSON D S, SETHI R. The complexity of flowshop and jobshop Scheduling[J]. Mathematics of Operations Research, 1976, 1(2): 117-129.

[2] JOHNSON S M. Optimal two - and three - stage production schedules with setup times included[J]. Naval Research Logistics Quarterly, 1954, 1(1): 61-68.

[3] NERON E, BAPTISTE P, GUPTAJN D. Solving hybrid flow shop problem using energetic reasoning and global operations [J]. Omega, 2001, 29(6): 501-511.

[4] NAWAZ M, ENSCORE E, HAM I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem[J]. Omega, 1983, 11(1): 91-95.

[5] DONG X, HUANG H, CHEN P. An improved NEH - based heuristic for the permutation flowshop problem[J]. Computers & Operations Research, 2008, 35(12): 3962-3968.

[6] LIU W, JIN Y, PRICE M. A new improved NEH heuristic for permutation flow shop scheduling problems [J]. International Journal of Production Economics, 2017, 193(11): 21-30.

[7] FERNANDEZ-VIAGAS V, FRAMINAN J M. On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem [J]. Computers & Operations Research, 2014, 45(3): 60-67.

[8] PAWEL J K, JERZY K. An improved NEH heuristic to minimizemakespan in permutation flow shops [J]. Computers & Operations Research, 2008, 35(9): 3001-3008.

[9] VICTOR F, RUBÉN R, JOSE M F. A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation [J]. European Journal of Operational Research, 2017, 257 (3): 707-721.

[10] LIU Y, YIN M H, GU W X. An effective differential evolution algorithm for permutation flow shop scheduling problem[J]. Applied Mathematics and Computation, 2014, 248(1): 143-159.

[11] 秦旋,房子涵,张赵鑫. 混合共生生物搜索算法求解置换流水车间调度问题[J]. 浙江大学学报(工学版), 2020, 54(4): 712-721.

[12] TOSUN O, MARICHELV M K. Hybrid bat algorithm for flow shop scheduling problems [J]. International Journal of Mathematics in Operational Research, 2016, 9(1): 125-138.

[13] WANG H, WANG W, SUN H, et al. A new Cuckoo search algorithm with hybrid strategies for flow shop scheduling problems [J]. Soft Computing, 2017, 21(15): 4297-4307.

[14] ASKARZADEH A. A novel metaheuristic method for solving constrained engineering optimization problems-crow search algorithm [J]. Computers and Structures, 2016, 169(6): 1-12.

[15] HUANG K, GIRSANG A, WU Z, et al. A hybrid crow search algorithm for solving permutation flow shop scheduling problems [J]. Applied Sicences, 2019, 9(7): 1353-1368.

[16] 闫红超,汤伟,姚斌. 基于新混合乌鸦搜索算法的置换流水车间调度[J]. 计算机集成制造系统,2024,30(5):1834-1846.

[17] SHIBUCHI H, YOSHIDA T, MURATA T. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling [J]. IEEE Transactions on Evolutionary Computation, 2003, 7(2):204-223.

[18] FRAMINAN J M, LEISTEN R, RUIZ-USANO R. Comparsion of heuristic for flowtime minimisation in permutation flowshops[J]. Computers & Operations Research, 2005, 32(5): 1237-1254.

[19] 李坤,徐铮,田慧欣. 基于自适应变邻域搜索算法的一类混合流水车间调度问题[J]. 系统工程, 2015, 33(11):9.

[20] 张其文,张斌. 基于教学优化算法求解置换流水车间调度问题[J]. 系统仿真学报,2022,34(5):1054-1063.

[21] 冯世扣,鲍敏,张伟. 基于混合遗传算法的车间调度研究[J]. 机电工程,2015,32(10):1315-1319.