文章编号: 1001-0920(2018)03-0503-06 DOI: 10.13195/j.kzyjc.2017.0124

混合灰狼优化算法求解柔性作业车间调度问题

姜 天 华†

(鲁东大学 交通学院，山东 烟台 264025)

摘 要: 将灰狼优化算法(GWO)用于柔性作业车间调度问题(FJSP),以优化最大完工时间为目标,提出一种混合灰狼优化算法(HGWO). 首先,采用两段式编码方式,建立GWO连续空间与FJSP离散空间的映射关系;其次,设计种群初始化方法,保证算法初始解的质量;然后,嵌入一种变邻域搜索策略,加强算法的局部搜索能力,引入遗传算子,提升算法的全局探索能力;最后,通过实验数据验证HGWO算法在求解FJSP问题方面的有效性.

关键词: 柔性作业车间调度；最大完工时间；灰狼优化算法；变邻域搜索策略；遗传算法

中图分类号: TH165 **文献标志码:** A

Flexible job shop scheduling problem with hybrid grey wolf optimization algorithm

JIANG Tian-hua†

(School of Transportation，Ludong University，Yantai 264025，China)

Abstract: Grey wolf optimization(GWO) algorithm is applied to the flexible job shop scheduling problem(FJSP), and a hybrid GWO(HGWO) is proposed with the objective of minimizing the makespan. Firstly, a two-phase encoding method is employed, and a mapping relationship is set up between the continuous space and the discrete space of FJSP. Then, a population initialization strategy is designed to ensure the quality of the initial solutions, and then a variable neighborhood search(VNS) strategy is embedded to enhance the local search ability. In addition, genetic operators are introduced to improve the global exploration capability of the algorithm. Finally, experimental data show that the proposed HGWO is effective for solving the FJSP.

Keywords: flexible job shop scheduling；makespan；grey wolf optimization algorithm；variable neighborhood search；genetic algorithm

0 引 言

柔性作业车间调度问题(FJSP)作为经典作业车间调度问题的一种延伸形式,具有更强的应用背景和更大的求解难度,已被证明是一种具有NP难特性的组合优化问题[1]. 与JSP问题相比,FJSP中工件柔性加工路径的特性在减小机器约束的同时,又增加了调度的灵活性,使其更能贴近于实际生产[2],但是这也大大提升了其复杂程度. 该问题的求解算法研究已成为车间调度领域的热点,目前元启发式算法已成为最流行的方法,它为FJSP问题提供了更多更新的求解思路和方法,引起了国内外学者们的广泛关注.

Zhang等[1]针对不同性能指标下柔性作业车间调度问题进行了研究,提出了一种具有双层子代产生模式的改进遗传算法;Zhang等[2]提出了一种改进遗传算法,以优化FJSP问题中的3个指标:最大完工时间、机器总负载和瓶颈机器负荷;Liu等[3]提出了一种改进双链量子遗传算法求解具有模糊交货期的柔性作业车间调度问题,以优化系统完工时间、总成本和提前/拖期惩罚;Karthikeyan等[4]提出了一种混合离散萤火虫算法求解有限资源约束下的多目标柔性作业车间调度问题;Yuan等[5]提出了一种混合差分进化算法,以优化柔性作业车间工件的最大完工时间;Li等[6]将遗传算法和禁忌搜索相结合,提出了一种混合算法以优化最大完工时间. 尽管各种元启发式算法在FJSP问题中已得到了广泛的研究,但目前仍没有任何一种算法能够获得所有问题的最优解,因此学者们仍在不断积极探索,以获得更丰富且更有效的方法.

收稿日期: 2017-02-12；**修回日期:** 2017-04-26.

基金项目: 山东省自然科学基金项目(ZR2016GP02)；鲁东大学引进人才科研项目(32660301).

责任编委: 唐加福.

作者简介: 姜天华(1983–),男,讲师,博士,从事生产调度和智能算法等研究.

†通讯作者. E-mail: jth1127@163.com

灰狼优化算法是根据狼群捕食原理提出的一种新兴元启发式算法[7]. 该算法虽然出现的时间比较晚,但目前已被成功运用于各类优化问题的求解中[8-11]. 然而,该算法在生产调度领域的研究尚不多见,且现有的几篇文献仅集中于简单流水车间调度问题的研究[12-14]. 因此,本文将灰狼优化算法扩展应用于柔性作业车间调度问题,并针对基本灰狼优化的算法进行一系列设计和改进,提出一种混合型灰狼优化算法(HGWO). 采用两段式个体编码方式实现离散调度解的连续编码,设计一种种群初始化方法,并利用变邻域搜索加强算法局部搜索能力,使决策层个体更好地引导其他个体进行搜索. 此外,引入遗传算法的交叉和变异操作,更好地避免算法出现早熟收敛. 通过对大量基准和随机算例的仿真实验测试算法的计算性能,并验证其有效性.

1 柔性作业车间调度问题

某车间内包含m台机器和n个待加工工件,每个工件的加工需经过$J_i(J_i \geqslant 1)$道工序,且工序间必须遵循一定的加工顺序. 与JSP问题相比,FJSP问题中工序具有柔性加工路径,每道工序可在多于一台的机器上处理,其加工时间取决于所分配的机器的能力. 本文FJSP问题的优化目标是为工序选择合适的机器,并确定各机器上工序加工顺序,使车间最大完工时间最小. 对于该类问题,存在以下假设: 1) 所有工件和机器在零时刻均处于就绪状态; 2) 同一时刻一台机器只可处理一个工件; 3) 工件加工过程具有不可中断性; 4) 仅考虑同工件工序间的先后约束关系,而不同工件工序间相互独立; 5) 各工件加工优先级相同; 6) 忽略工序加工前机器所需的准备时间.

令$C_{\max}$表示工件最大完工时间,C_{iJ_i}表示工件的完工时间,则目标函数可表示为

$$\min C_{\max} = \min\max(C_{iJ_i}),\ 1 \leqslant i \leqslant n. \tag{1}$$

2 基本灰狼优化算法

灰狼优化算法是一种基于群智能的元启发式优化算法,它是通过模拟狼群捕食的行为机制而提出的. 根据基本灰狼优化算法的特点,算法迭代过程中需按照适应度值从大到小的顺序将狼群个体划分为4类,用于模拟狼群的社会等级,狼群中适应度值排在前3位的个体分别依次定义为α、β和δ,而其余个体均定义为ω. 基本灰狼优化算法假设α、β和δ拥有获得潜在猎物所在位置的能力,在算法每次迭代时均先找出目前为止算法最优的3个个体α、β和δ,并保存好3者的个体位置,然后以此更新下一代个体的位置,通过不断迭代,最终达到捕食猎物的目的. 该过程可用以下公式表示:

$$\boldsymbol{D}_\alpha = |\boldsymbol{C}_1\boldsymbol{X}_\alpha(t) - \boldsymbol{X}(t)|,\ \boldsymbol{D}_\beta = |\boldsymbol{C}_2\boldsymbol{X}_\beta(t) - \boldsymbol{X}(t)|,$$
$$\boldsymbol{D}_\delta = |\boldsymbol{C}_3\boldsymbol{X}_\delta(t) - \boldsymbol{X}(t)|; \tag{2}$$

$$\boldsymbol{X}_1 = \boldsymbol{X}_\alpha(t) - \boldsymbol{A}_1\boldsymbol{D}_\alpha,\ \boldsymbol{X}_2 = \boldsymbol{X}_\beta(t) - \boldsymbol{A}_2\boldsymbol{D}_\beta,$$
$$\boldsymbol{X}_3 = \boldsymbol{X}_\delta(t) - \boldsymbol{A}_3\boldsymbol{D}_\delta, \tag{3}$$

$$\boldsymbol{X}(t+1) = \frac{\boldsymbol{X}_1 + \boldsymbol{X}_2 + \boldsymbol{X}_3}{3}; \tag{4}$$

$$\boldsymbol{A} = 2\boldsymbol{a}\boldsymbol{r}_1 - \boldsymbol{a}, \tag{5}$$

$$\boldsymbol{C} = 2\boldsymbol{r}_2. \tag{6}$$

其中: t表示当前迭代次数,$\boldsymbol{A}$和$\boldsymbol{C}$表示协同系数向量,$\boldsymbol{X}$表示灰狼的个体位置向量,$\boldsymbol{a}$中元素在迭代过程中由2线性递减到0,$\boldsymbol{r}_1$和$\boldsymbol{r}_2$均表示随机数向量. 由式(2)和(3)计算出个体与α、β和δ的距离,然后由式(4)确定个体向猎物移动的方向. 由于篇幅限制,基本GWO算法的步骤可参见文献[7].

3 混合灰狼优化算法

3.1 编码机制

FJSP问题包含机器分配和工序排序两个子问题,因此每个个体可采用基于随机键的两段式编码,其中各段长度相等,且分别对应机器分配方案和工序排序方案. 假设个体位置向量的长度为$2l$,则可表示为$\boldsymbol{X} = \{x(1), x(2), \cdots, x(l), x(l+1), \cdots, x(2l)\}$,各元素均在$[-\varepsilon, \varepsilon]$内任意取值. 假设车间内包含3个工件,每个工件包含两道工序,则个体位置向量的总长度为12,各元素在$[-3, 3]$中任意取值,并按照一定的顺序存储,如图1所示,其中OP_{ij}表示工件i的第j道工序.

OP_{11}	OP_{12}	OP_{21}	OP_{22}	OP_{31}	OP_{32}	OP_{11}	OP_{12}	OP_{21}	OP_{22}	OP_{31}	OP_{32}
−1.0	1.1	−0.5	0.3	1.5	−0.9	−1.2	2.1	0.5	−1.3	2.0	1.6

图1 个体位置向量

3.2 转换机制

由图1可以看出,灰狼优化算法中个体位置向量元素均为连续值,而柔性作业车间调度解为离散值,因此还需解决连续个体位置向量与离散调度解间的转换问题. 为了实现算法解空间与问题空间之间的映射,分别在前后两部分采用不同的转换方法.

1) 个体位置向调度方案的转换.

i) 机器分配: 采用文献[15]中的转换方法,按照式(7)将个体位置向量元素值转换成工序可选机器集中机器的序号,由此可获得机器的编号

$$u(h) = \mathrm{round}\Big(\frac{x(h)+\varepsilon}{2\varepsilon}(z(h)+1)+1\Big),\ 1 \leqslant h \leqslant l. \tag{7}$$

其中: $z(h)$ 表示元素 h 对应的工序的可选机器数; $u(h) \in [1, z(h)]$ 表示所选机器在工序可选机器集中的序号,即工序可选机器集中第 $u(h)$ 个元素就是为工序分配的机器.

ii) 工序排序: 采用文献[16]中的升序排列ROV (Ranked order value)规则,首先按升序排列的顺序为每个位置元素赋予一个唯一的ROV值,然后根据ROV值即可构造工序排序方案,如图2所示,其中相同工序编号表示同工件不同工序.

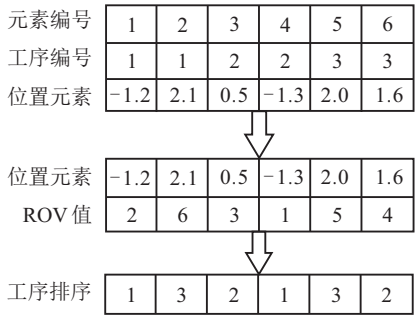

图 2　个体位置转换成工序排序过程

2) 调度方案向个体位置的转换.

i) 机器分配: 采用文献[15]中的转换方式,即一般情况下,可按照式(7)的逆操作执行,如下式所示:

$$x(h) = \frac{2\varepsilon}{z(h)-1}(u(h)-1) - \varepsilon,\ z(h) \neq 1. \qquad (8)$$

但存在一种特殊情况,即若工序仅有一台可选机器, $z(h)$=1,则 $x(h)$ 在 $[-\varepsilon, \varepsilon]$ 内任取数值.

ii) 工序排序: 首先在 $[-\varepsilon, \varepsilon]$ 间随机产生 l 个随机数,按升序顺序为每个位置元素赋予ROV值,并与工序排序方案对应,根据工序编号找到对应的ROV值,然后根据重排ROV值确定个体位置向量中各元素的值,转换过程如图3所示.

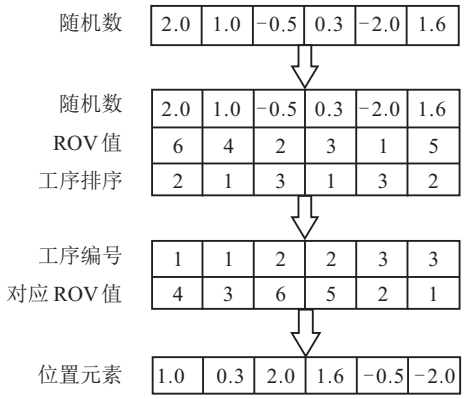

图 3　工序排序转换成个体位置过程

3.3　种群初始化

根据个体位置向量的表达形式,种群初始化将分别针对机器分配和工序排序进行. 在机器分配部分采用文献[2]中基于启发式算法与随机生成法相结合的方式生成初始机器分配方案;在工序排序部分采用文献[17]中基于搜索的方法,即对于一个已生成的机器分配方案,随机生成若干个工序排序方案,将每一个排序方案与该机器分配方案结合,选择最优的一组作为一个初始调度解,重复上述过程,直到生成初始种群为止.

3.4　变邻域搜索

在灰狼优化算法的搜索过程中,种群个体根据决策层3个个体(α、β和δ)来更新自身的位置信息,并最终达到捕食猎物的目的. 因此,决策层个体的好坏对算法的计算性能具有很大的影响. 本文将变邻域搜索嵌入到灰狼优化算法中,并在每次迭代中均作用于α、β和δ,从而改善算法性能. 变邻域搜索算法中,针对个体位置向量采用以下3种邻域结构.

1) 邻域结构 N_1: 在工序排序部分中任选两个元素,所选元素需对应不同工件的工序,然后对所选元素进行交换操作.

2) 邻域结构 N_2: 在工序排序部分中任选两个元素,然后将后一元素插入到前一元素之前的位置.

3) 邻域结构 N_3: 在机器分配部分任选一个元素,该元素对应工序的可选机器应多于一台,然后将该工序分配至不同机器上,并根据式(8)生成新元素值.

变邻域搜索算法的具体步骤如下.

Step 1: 将决策层个体作为初始解 $\boldsymbol{X}$,设置 $q \leftarrow 1$, $\lambda_{\max} \leftarrow 3$ 及终止迭代次数 $q_{\max}$.

Step 2: 设置 $\lambda \leftarrow 1$.

Step 3: 振动. 若 $\lambda = 1$,则 $\boldsymbol{X}' \leftarrow N_1(\boldsymbol{X})$;若 $\lambda = 2$,则 $\boldsymbol{X}' \leftarrow N_2(\boldsymbol{X})$;若 $\lambda = 3$,则 $\boldsymbol{X}' \leftarrow N_3(\boldsymbol{X})$.

Step 4: 局部搜索. 将 $\boldsymbol{X}'$ 作为初始解,获得局部最优解 $\boldsymbol{X}''$.

Step 5: 若 $\boldsymbol{X}''$ 优于 $\boldsymbol{X}$,则 $\boldsymbol{X} \leftarrow \boldsymbol{X}''$,并设置 $\lambda \leftarrow 1$;否则设置 $\lambda \leftarrow \lambda + 1$.

Step 6: 判断 $\lambda > \lambda_{\max}$ 是否满足. 若满足,则设置 $q \leftarrow q + 1$,转到Step 7;否则,转到Step 3.

Step 7: 判断 $q > q_{\max}$ 是否满足. 若满足,则转到Step 8;否则,转到Step 2.

Step 8: 算法结束.

变邻域搜索算法中局部搜索采用文献[18]中的阈值接受法,具体步骤如下.

Step 1: 获取初始解 $\boldsymbol{X}'$,并设置阈值 $\psi > 0$, $\rho \leftarrow 1$, $\vartheta \leftarrow 1$ 及终止迭代次数 $\rho_{\max}$.

Step 2: 若 $\vartheta = 1$,则 $\boldsymbol{X}'' \leftarrow N_1(\boldsymbol{X}') \bigcup N_3(\boldsymbol{X}')$;若 $\vartheta = 0$,则 $\boldsymbol{X}'' \leftarrow N_2(\boldsymbol{X}') \bigcup N_3(\boldsymbol{X}')$.

Step 3: 判断 $C_{\max}(\boldsymbol{X}'') - C_{\max}(\boldsymbol{X}') \leqslant \psi$ 是否满足. 若满足,则 $\boldsymbol{X}' \leftarrow \boldsymbol{X}''$;否则,设置 $\vartheta \leftarrow |\vartheta - 1|$.

Step 4: 令$\rho \leftarrow \rho + 1$,并判断$\rho > \rho_{\max}$是否满足. 若满足,则$\boldsymbol{X}'' \leftarrow \boldsymbol{X}'$,转到Step 5;否则,转到Step 2.

Step 5: 算法结束.

3.5 交叉和变异算子

灰狼优化算法进化时,个体仅根据适应度最优的3个个体进行个体位置更新,这使得该算法易陷入局部最优解,对此本文引入变邻域搜索算法,用于增强算法的局部开发能力. 良好的算法能够在搜索过程中平衡全局探索与局部开发能力. 考虑到遗传算法具有很强的全局搜索能力,在灰狼优化算法中引入遗传算法的交叉和变异算子,提高该算法的全局探索能力,从而进一步增强该算法的计算性能. 对于交叉操作,在机器分配部分采用双点交叉法,在工序排序部分采用部分匹配交叉法;对于变异操作,在机器分配部分采用单点变异法,在工序排序部分采用逆序变异法,即在个体位置向量中任选两个位置,然后对所选位置间的元素进行逆序排序.

3.6 HGWO算法步骤

HGWO算法的具体步骤如下.

Step 1: 设置算法参数并创建初始种群.

Step 2: 评估种群中个体的适应度值,并确定α、β和δ.

Step 3: 对最优个体均执行变邻域搜索,然后根据适应度值从大到小更新α、β和δ.

Step 4: 根据式(2) ~ (6)更新个体位置信息.

Step 5: 判断算法是否满足终止条件. 若满足,则转到Step 6;否则,按一定概率对个体进行交叉和变异操作,然后转到Step 2.

Step 6: 算法结束并输出结果.

4 实验分析

本节针对文献[19-20]中FJSP问题的15个基准算例(Kacem01 ~ Kacem05, MK01 ~ MK10)以及本文随机生成的15个算例(RM01 ~ RM15)进行仿真. 对于RM01 ~ RM15,机器数和工件数如表1所示,其他数据取值服从一定范围内的均匀分布,其中工件的工序数在[1, 5]内取值,工序加工时间在[1, 20]内取值. 此外,采用随机的方式确定各工序的可加工机器集,即对于一台机器,若产生的随机数大于0.5,则该机器为对应工序的一台可加工机器.

表1 基准算例和随机算例对比结果

算例	$n \times m$	BGWO				GWOin				GWOin+vns				HGWO			
		Time	Best	Avg	RPD	Time	Best	Avg	RPD	Time	Best	Avg	RPD	Time	Best	Avg	RPD
Kacem01	4 × 5	0.6	11	11.5	0	0.7	11	11.0	0	5.1	11	11.0	0	5.6	11	11.0	0
Kacem02	8 × 8	1.6	23	24.6	64.2	1.6	15	16.2	7.1	13.5	14	14.6	0	14.8	14	14.3	0
Kacem03	10 × 7	1.8	19	21.6	72.7	1.9	14	14.8	27.3	15.3	11	12.0	0	16.3	11	11.6	0
Kacem04	10 × 10	1.9	13	16.4	85.7	2.0	8	8.3	14.3	16.2	7	7.8	0	17.5	7	7.5	0
Kacem05	15 × 10	4.3	27	31.5	107.7	4.4	17	18.4	30.8	37.3	14	14.7	7.7	40.7	13	14.1	0
MK01	10 × 6	3.8	42	45.2	5.0	4.0	42	42.0	5.0	33.8	40	41.8	0	36.3	40	41.6	0
MK02	10 × 6	4.0	37	39.1	27.6	4.2	33	36.1	13.8	35.5	29	30.7	0	38.7	29	30.3	0
MK03	15 × 8	15.8	221	229.7	8.3	16.2	204	208.3	0	155.7	204	204.3	0	165.8	204	204.1	0
MK04	15 × 8	7.7	73	76.5	12.3	7.8	68	71.8	4.6	70.3	65	67.5	0	75.9	65	67.4	0
MK05	15 × 4	9.4	179	188.6	2.3	9.7	181	186.7	3.4	89.9	176	179.1	0.6	96.7	175	178.2	0
MK06	10 × 15	16.1	123	129.6	66.2	16.6	92	100.5	24.3	157.4	74	76.6	0	168.6	79	79.9	6.8
MK07	20 × 5	9.2	184	193.4	23.5	9.4	159	166.0	6.7	85.4	150	155.1	0.7	92.1	149	156.4	0
MK08	20 × 10	30.8	523	549.4	0	31.4	523	529.9	0	316.5	523	523.0	0	340.8	523	523.0	0
MK09	20 × 10	33.9	382	400.4	17.5	34.8	364	396.8	12.0	353.3	339	348.5	4.3	378.9	325	342.3	0
MK10	20 × 15	34.8	327	345.6	29.2	36.1	300	309.1	18.6	359.9	255	265.6	7.9	388.5	253	262.7	0
RM01	4 × 5	0.8	39	39.9	0	0.8	39	39.0	0	5.6	39	39.0	0	7.0	39	39.0	0
RM02	8 × 8	1.6	36	40.3	12.5	1.6	32	34.2	0	13.1	32	32.0	0	14.8	32	32.0	0
RM03	10 × 7	2.1	43	52.6	10.3	2.2	43	47.0	10.3	17.4	39	40.1	0	17.9	39	39.0	0
RM04	10 × 10	2.1	36	40.9	28.6	2.1	32	32.9	14.3	17.8	28	29.0	0	20.9	28	28.5	0
RM05	15 × 10	4.1	52	59.0	44.4	4.1	45	50.0	25.0	36.2	37	39.0	2.8	39.9	36	37.9	0
RM06	10 × 6	2.1	51	57.7	6.3	2.1	51	53.6	6.3	17.3	48	50.1	0	18.8	48	49.9	0
RM07	15 × 8	4.0	63	68.2	37.0	4.2	51	57.7	10.9	35.2	48	49.8	4.3	38.8	46	49.0	0
RM08	15 × 4	4.0	104	110.8	13.0	4.2	96	100.7	4.3	38.6	92	94.4	0	44.4	92	95.6	0
RM09	10 × 15	2.2	39	43.3	5.4	2.1	37	37.4	0	18.9	37	37.0	0	20.6	37	37.0	0
RM10	20 × 5	6.0	121	128.4	16.3	6.0	109	112.6	4.8	63.4	106	107.8	1.9	71.1	104	106.6	0
RM11	20 × 10	5.9	72	79.6	53.2	6.0	63	66.9	34.0	54.9	47	49.7	0	60.7	50	52.4	6.4
RM12	20 × 15	6.3	59	65.6	84.4	6.4	43	44.8	34.3	58.8	34	36.2	6.3	62.3	32	35.0	0
RM13	30 × 10	11.9	107	118.4	67.2	13.1	77	87.5	20.3	125.8	64	68.5	0	141.3	65	69.9	1.5
RM14	30 × 15	11.9	88	92.7	104.7	12.7	58	58.4	34.9	136.5	45	45.5	4.7	145.3	43	44.4	0
RM15	30 × 20	13.3	72	78.0	176.9	13.2	32	35.1	23.1	139.1	29	29.7	11.5	152.3	26	28.2	0
Mean					39.4				13.0				1.8				0.5

采用Fortran程序设计语言,在Win XP系统下内存2 G的Pentium CPU G2030 @3.00 GHz, 2.99 GHz计算机上运行. 算法参数设置为:种群大小为100,最大迭代次数为800,变邻域搜索和局部搜索最大迭代次数$q_{\max}$和$\rho_{\max}$均为10,交叉率为0.8,变异率为0.2.

首先对算法中几种改善机制的有效性进行验证,其中包括种群初始化方法、变邻域搜索算法和交叉变异操作. 各算法针对不同算例分别独立运行10次后进行比较,对比结果如表1所示. 其中:BGWO表示不包含任何改善机制的基本灰狼优化算法;GWOin表示在BGWO中增加种群初始化方法而得到的算法;GWOin+vns表示在GWOin中嵌入变邻域搜索算法;HGWO表示本文所提出的算法,即在GWOin+vns中又增加了交叉和变异算子;Time表示算法运行时间;Best表示算法运行10次获得的最佳结果;Avg表示算法10次运行结果的平均值;RPD表示相对百分比偏差,计算公式为$\mathrm{RPD} = 100 \times (\mathrm{Best} - \mathrm{Min})/\mathrm{Min}$,Best表示各算法运行10次获得的最佳结果,Min表示各算法最佳结果中的最小值;Mean表示30个算例下各算法RPD的平均值. 由表1中的数据可知,本文算法可以在较短时间内获得较满意的解,但由于GWOin+vns和HGWO中包含变邻域搜索算法,使其计算时间相对较长. 从计算结果看,HGWO算法的计算结果最好,能够在27个算例中获得最佳结果,明显多于其他算法. 此外,与GWOin+vns算法相比,HGWO算法计算时间稍长,但其Mean值更小,且相同Best值的情况下,HGWO算法在10个算例中Avg值较好,因此遗传算子的引入能够进一步改善算法的性能.

文献[19-26]中给出了不同算法求解15个FJSP问题基准算例下的工件最大完工时间. 为了验证本文HGWO算法的有效性,将其与上述文献中的各种算法进行比较,算法对比结果如表2和表3所示.

在表2和表3中:粗体表示相同算例下各算法比较后的最佳结果,"–"表示文献中未给出相应结果. 表2针对Kacem算例比较本文算法与其他算法的计算结果,数据显示HGWO算法只在算例Kacem05下稍逊于其他3种算法,但在算例Kacem01 ~ Kacem04下均能获得最优解. 表3通过对Brandimarte算例的计算进行进一步对比,数据显示HGWO算法能够获得最优解的个数(5个)仅次于文献[26]中的算法(6个),但若同时综合考虑表2和表3中13个基准算例,本文HGWO算法能够获得最优解的个数(8个)多于文献[26]中算法(7个).

为了进一步测试本文算法的优越性,将HGWO算法与文献[25-26]中算法的RPD值进行比较,如表4所示.

表2　Kacem算例对比

算例	文献[19]	文献[21]	文献[22]	文献[25]	文献[26]	HGWO
Kacem01	–	**11**	–	**11**	**11**	**11**
Kacem02	15	–	15	15	17	**14**
Kacem03	–	12	–	13	–	**11**
Kacem04	**7**	**7**	**7**	**7**	8	**7**
Kacem05	23	**12**	**12**	**12**	–	13

表3　Brandimarte算例对比

算例	文献[20]	文献[23]	文献[24]	文献[25]	文献[26]	HGWO
MK01	42	**40**	41	42	41	**40**
MK02	32	32	30	28	**26**	29
MK03	211	207	–	**204**	207	**204**
MK04	81	67	68	75	**65**	**65**
MK05	186	188	181	179	**171**	175
MK06	86	85	75	69	**61**	79
MK07	157	154	159	**149**	173	**149**
MK08	**523**	**523**	**523**	555	**523**	**523**
MK09	369	437	369	342	**307**	325
MK10	296	380	272	**242**	312	253

表4　算法RPD值对比

算例	文献[25]	文献[26]	HGWO
Kacem01	0	0	0
Kacem02	7.14	21.43	0
Kacem03	0	14.29	0
MK01	5.0	2.5	0
MK02	7.69	0	11.54
MK03	0	1.47	0
MK04	15.38	0	0
MK05	4.68	0	2.34
MK06	13.11	0	29.51
MK07	0	16.11	0
MK08	6.12	0	0
MK09	11.40	0	5.86
MK10	0	28.93	4.55
Mean	5.42	6.52	4.14

由表4中的数据可以看出,HGWO算法获得的Mean值在3种算法中最小. 因此,综合表2 ~ 表4中的数据对比结果,本文所提出的HGWO算法在求解FJSP问题中工件的最大完工时间方面具有一定的有效性.

5　结　论

本文针对柔性作业车间调度问题特点,对新型灰狼优化算法进行一系列改进,提出了一种混合灰狼优化算法: 1) 采用两段式编码,实现了连续个体位置与离散调度解间的转换;设计了种群初始化方法,确保种群的质量和多样性;提出了一种变邻域搜索策略,增强算法局部搜索能力;引入了交叉和变异算子,平衡算法全局探索与局部开发能力. 2) 针对30个FJSP问题的算例进行了仿真,验证了算法改善机制的有效性,并进一步测试了算法解决柔性作业车间调度问题方面的有效性. 3) 下一步将把灰狼优化算法扩展至

更复杂的车间调度问题中,并结合车间特点,设计出更有效的算法.

参考文献(References)

[1] Zhang C Y, Rao Y Q, Li P G, et al. Bilevel genetic algorithm for the flexible job-shop scheduling problem[J]. Chinese J of Mechanical Engineering, 2007, 43(4): 119-124.

[2] Zhang G H, Gao L, Li P G, et al. Improved genetic algorithm for the flexible job-shop scheduling problem[J]. Chinese J of Mechanical Engineering, 2009, 45(7): 145-151.

[3] Liu X B, Jiao X, Ning T, et al. Flexible job shop scheduling based on double chains quantum genetic algorithm[J]. Computer Integrated Manufacturing Systems, 2015, 21(2): 495-502.

[4] Karthikeyan S, Asokan P, Nickolas S. A hybrid discrete firefly algorithm for multi-objective flexible job shop scheduling problem with limited resource constraints[J]. The Int J of Advanced Manufacturing Technology, 2014, 72(9/12): 1567-1579.

[5] Yuan Y, Xu H. Flexible job shop scheduling using hybrid differential evolution algorithms[J]. Computers & Industrial Engineering, 2013, 65(2): 246-260.

[6] Li X, Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem[J]. Int J of Production Economics, 2016, 174: 93-110.

[7] Mirjalili S, Mirjalili S M, Lewis A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69: 46-61.

[8] 姚鹏, 王宏伦. 基于改进流体扰动算法与灰狼优化的无人机三维航路规划[J]. 控制与决策, 2016, 31(4): 701-708.
(Yao P, Wang H L. Three-dimensional path planning for UAV based on improved interfered fluid dynamical system and grey wolf optimizer[J]. Control and Decision, 2016, 31(4): 701-708.)

[9] Song X, Tang L, Zhao S, et al. Grey wolf optimizer for parameter estimation in surface waves[J]. Soil Dynamics and Earthquake Engineering, 2015, 75: 147-157.

[10] Song H M, Sulaiman M H, Mohamed M R. An application of grey wolf optimizer for solving combined economic emission dispatch problems[J]. Int Review on Modelling and Simulations, 2014, 7(5): 838-844.

[11] Sulaiman M H, Mustaffa Z, Mohamed M R, et al. Using the gray wolf optimizer for solving optimal reactive power dispatch problem[J]. Applied Soft Computing, 2015, 32: 286-292.

[12] 吕新桥, 廖天龙. 基于灰狼优化算法的置换流水线车间调度[J]. 武汉理工大学学报, 2015, 37(5): 111-116.
(Lv X Q, Liap T L. Permutation flow-shop scheduling based on the grey wolf optimizer[J]. J of Wuhan University of Technology, 2015, 37(5): 111-116.)

[13] Komaki G M, Kayvanfar V. Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time[J]. J of Computational Science, 2015, 8: 109-120.

[14] Lu C, Xiao S, Li X, et al. An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production[J]. Advances in Engineering Software, 2016, 99: 161-176.

[15] Yuan Y, Xu H, Yang J. A hybrid harmony search algorithm for the flexible job shop scheduling problem[J]. Applied Soft Computing, 2013, 13(7): 3259-3272.

[16] 王凌. 微粒群优化与调度算法[M]. 北京: 清华大学出版社, 2008: 117-118.
(Wang L. Particle Swarm optimization and scheduling algorithms[M]. Beijing: Tsinghua University Press, 2008: 117-118.)

[17] Demir Y, İsleyen S K. An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations[J]. Int J of Production Research, 2014, 52(13): 3905-3921.

[18] Bouffard V, Ferland J A. Improving simulated annealing with variable neighborhood search to solve the resource-constrained scheduling problem[J]. J of Scheduling, 2007, 10(6): 375-386.

[19] Kacem I, Hammadi S, Borne P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems[J]. IEEE Trans on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2002, 32(1): 1-13.

[20] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993, 41(3): 157-183.

[21] Ho N B, Tay J C. GENACE: An efficient cultural algorithm for solving the flexible job-shop problem[C]. Proc of the IEEE Congress on Evolutionary Computation. Portland: IEEE, 2004, 2: 1759-1766.

[22] Xia W, Wu Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems[J]. Computers & Industrial Engineering, 2005, 48(2): 409-425.

[23] Henchiri A, Ennigrou M. Particle swarm optimization combined with tabu search in a multi-agent model for flexible job shop problem[C]. Int Conf in Swarm Intelligence. Harbin: Springer, 2013: 385-394.

[24] Teekeng W, Thammano A. A combination of shuffled frog leaping and fuzzy logic for flexible job-shop scheduling problems[J]. Procedia Computer Science, 2011, 6: 69-75.

[25] Ziaee M. A heuristic algorithm for solving flexible job shop scheduling problem[J]. The Int J of Advanced Manufacturing Technology, 2014, 71(1/2/3/4): 519-528.

[26] Nouiri M, Bekrar A, Jemai A, et al. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem[J]. J of Intelligent Manufacturing, 2015, 11(2): 1-13.

(责任编辑：闫 妍)