2024

文献调研汇报　汇报人：刘颖

文献简介

论文名：A Novel Evolutionary Algorithm for Energy-Efficient Scheduling in Flexible Job Shops

发表期刊和年份：IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 27, NO. 5, OCTOBER 2023

摘要：

- 现代制造业中，以牺牲高能耗为代价来提高生产力往往是不可能的。然而，通过高效的调度技术，能够在降低能源成本的同时保持高生产力
- 采用**分时电价**的柔性作业车间调度问题，在满足**预定义最大完工时间约束**的前提下，**最小化总能耗**
- 提出一种新颖的基于两个个体的进化（TIE）算法，该算法结合**禁忌搜索程序**、**基于拓扑顺序的重组运算符**、针对该特定问题的**新邻域结构**以及**近似邻域评估**方法
- 大量实验表明，所提出的 TIE 算法优于传统的基于轨迹和基于种群的方法

问题场景

在分时电价（TOU）背景下进行柔性作业车间调度

作业(Jobs)：作业集 J 有n个作业

操作：每个作业i有n_i个连续操作

机器(Machines)：机器集 M 有m台机器

每个角色的作用及交互方式

作业：具有一系列操作，每个操作可在多台兼容机器上加工，是调度的对象

操作：每个操作o可在兼容机器子集$M(o)$中的任一机器上加工

机器：执行作业的操作

场景设置

考虑操作在不同机器上的加工时间差异、不同时段的电价差异以及操作的先后顺序等因素，合理安排作业与机器的匹配以及操作的时间安排

所提方法 基于两个个体的进化算法

- 基于两个个体的进化算法
  - 算法结合禁忌搜索算法、基于拓扑顺序的重组运算符、针对该特定问题的新邻域结构以及近似邻域评估方法这四种方法
  - 柔性车间调度问题(FJSP)结合分时电价(TOU)涉及三个决策：1）机器分配；2）操作顺序；3）操作时间表，解空间庞大，传统单一个体轨迹搜索（如禁忌搜索）易陷入局部最优，而大规模种群进化算法计算成本高。
  - 因此，提出一种基于两个个体的进化算法，两个个体分别负责强化搜索（专注历史最优解的邻域优化）和多样化搜索（探索新解空间，避免早熟收敛）

所提方法　基于两个个体的进化算法

输入：问题实例(包含作业数n、机器数m处理时间$P(o,k)$、TOU电价方案、加工周期约束、$\overline{C}$等)
输出：最优解(总能耗最小的机器分配、工序排序)

Algorithm 3 TIE for FJSP With TOU Scheme

Input: Problem instance
Output: The best solution S^* found
$gen \leftarrow 0$; $S_1, S_2, S_c^*, S_p^*, S^* \leftarrow$ Init()
while stopping condition is not reached **do**
  $S_1' \leftarrow$ TOCX(S_1, S_2), $S_2' \leftarrow$ TOCX(S_2, S_1)
  $S_1 \leftarrow$ TS(S_1'), $S_2 \leftarrow$ TS(S_2')
  $S_c^* \leftarrow$ save_best(S_1, S_2, S_c^*)
  $S^* \leftarrow$ save_best(S_c^*, S^*)
  if gen is equal to an integer parameter p **then**
    $S_1 \leftarrow S_p^*$, $S_p^* \leftarrow S_c^*$, $S_c^* \leftarrow$ Init(), $gen \leftarrow 0$
  end if
  if $S_1 \approx S_2$ **then**
    $S_2 \leftarrow$ Init()
  end if
  $gen \leftarrow gen + 1$
end while
return S^*

所有解初始都随机生成：
S_1:进化个体1
S_2:进化个体2
S_c^*:当前周期最佳解
S_p^*:前一周期最佳解
S^*:全局最优

初始化：
- 每个作业的每个操作O等概率选择机器，确保操作顺序约束
- 初始解满足约束最大完工时间$makespan \leq \overline{C}$
- $\overline{C} = (1+\varepsilon)LB, \varepsilon = 0.1, LB$: 松弛下界

所提方法　基于两个个体的进化算法

Algorithm 4 Topological Order Recombination (TOCX)

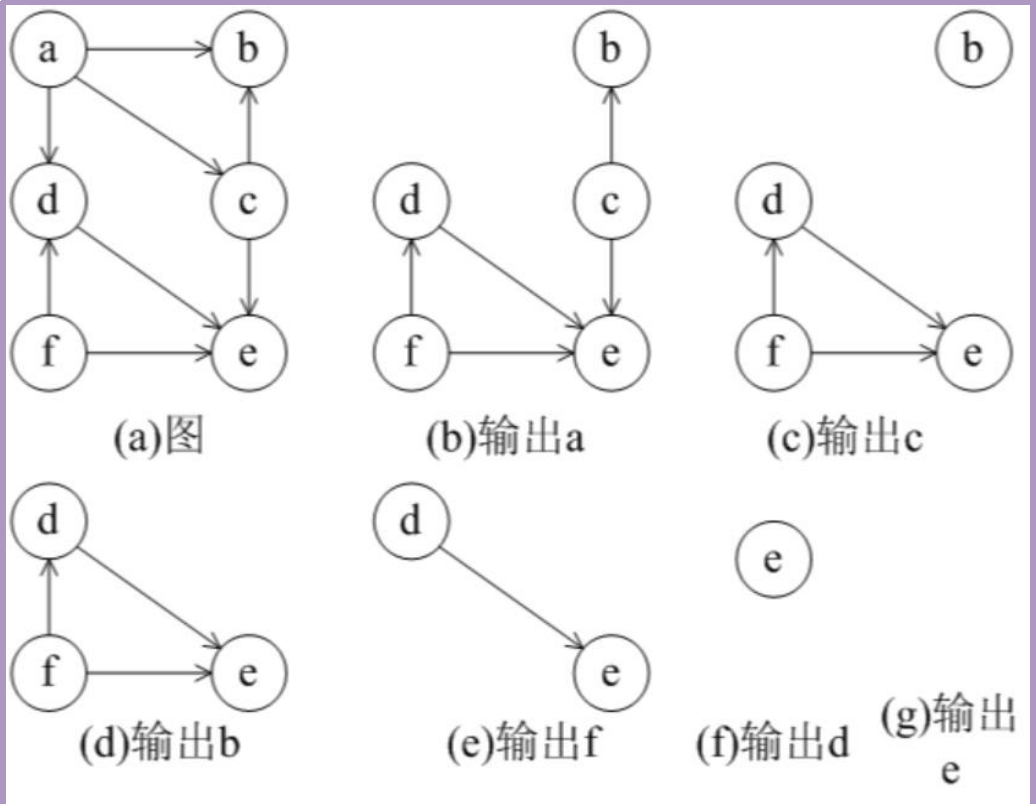
Input: parent solutions S_1 and S_2, γ
Output: An offspring solution S_o
calculate the topological order T_1 of S_1
calculate the topological order T_2 of S_2
an empty topological order list T, $\varphi \leftarrow 0$
while $\varphi <=$ total number of operations **do**
  choose the first operation o from $T_{\varphi/2+1}$
  $N \leftarrow \emptyset$
  for each position i in list T **do**
    insert o into position i of T, results in a sub li
    record the corresponding machine assignment
in list T_o^i
    mapping T_o^i into a sub solution S_o^i
    if S_o^i is feasible **then**
      $N \leftarrow N \cup \{S_o^i\}$
    end if
  end for
  if $rand(0, 1) < \gamma$ **then**
    $S_o^{i_{\min}} \leftarrow \arg\min\{makespan(S_o^i) | S_o^i \in N\}$
  else
    randomly select a solution $S_o^{i_{\min}}$ from N
  end if
  insert o into position $i_{\min}$ of list T
  record the corresponding machine assignment $\phi($
list T
  remove o from T_1 and T_2
  $\varphi \leftarrow \varphi + 1$
end while
mapping T into a complete solution S_o
return S_o

- 拓扑重组生成子代

输入：两个个体S_1、S_2，重组率$\gamma = 0.3$

输出：子代S_0

1. 计算两个父代拓扑序



拓扑排序序列：acbfde

入度：指向该节点边的个数
找入度为0的节点
生成一个可行的拓扑排序序列

所提方法　基于两个个体的进化算法

Algorithm 4 Topological Order Recombination (TOCX)

Input: parent solutions S_1 and S_2, γ
Output: An offspring solution S_o
calculate the topological order T_1 of S_1
calculate the topological order T_2 of S_2
an empty topological order list T, $\varphi \leftarrow 0$
while φ <= total number of operations **do**
    choose the first operation o from $T_{\varphi/2+1}$
    $N \leftarrow \emptyset$
    for each position i in list T **do**
        insert o into position i of T, results in a sub list T_o^i
        record the corresponding machine assignment $\phi(o, S_{\varphi/2+}$
in list T_o^i
        mapping T_o^i into a sub
        if S_o^i is feasible **then**
            $N \leftarrow N \cup \{S_o^i\}$
        end if
    end for
    if $rand(0,1) < \gamma$ **then**
        $S_o^{i\min} \leftarrow \arg\min\{makes$
    else
        randomly select a soluti
    end if
    insert o into position $i_{\min}$
    record the corresponding machine assignment $\phi(o, S_{\varphi/2+1})$
list T
    remove o from T_1 and T_2
    $\varphi \leftarrow \varphi + 1$
end while
mapping T into a complete solution S_o
return S_o

尝试将o插入当前序列T中的所有位置

贪婪选择（局部最优）+随机选择（多样性）

- 拓扑重组生成子代

输入：两个个体S_1、S_2，参数$\gamma = 0.3$(控制是否使用贪婪选择)

输出：子代S_0

1. 计算两个父代拓扑序T_1、T_2

T_0^i：操作o插入到位置i之前的工序排序

$while\ \varphi \le$ 总操作数:
　交替选择父代拓扑序T_1、T_2操作
　$N = \emptyset$
　$for\ o\ in\ T$中的所有插入位置i:
　　插入o到位置i,生成子列表T_o^i
　　记录插入后的机器分配$\phi(o, s_{\varphi/2+1})$ →子解S_o^i
　　可行性检查：S_o^i满足满足工序顺序约束,则加入候选集N，$N = N \cup \{S_0^i\}$
　$end\ for$
　$if\ rand(0,1) < \gamma$:
　　从N中选择完工时间最小的候选解$S_0^i \rightarrow S_o^{i_{min}}$
　$else$:
　　随机从N中选择一个解→ $S_o^{i_{min}}$
　$end\ if$
　将o插入到选定位置i_{min}
　从T_1、T_2中删除操作o
　$\varphi = \varphi + 1$
$end\ while$
拓扑顺序T和记录的机器分配→完整调度解S_o

所提方法　基于两个个体的进化算法

Algorithm 3 TIE for FJSP With TOU Scheme

Input: Problem instance
Output: The best solution S^* found
$gen \leftarrow 0$; $S_1, S_2, S_c^*, S_p^*, S^* \leftarrow \text{Init}()$
while stopping condition is not reached **do**
　$S_1' \leftarrow \text{TOCX}(S_1, S_2)$, $S_2' \leftarrow \text{TOCX}(S_2, S_1)$
　$S_1 \leftarrow \text{TS}(S_1')$, $S_2 \leftarrow \text{TS}(S_2')$
　$S_c^* \leftarrow \text{save\_best}(S_1, S_2, S_c^*)$
　$S^* \leftarrow \text{save\_best}(S_c^*, S^*)$
　if gen is equal to an integer parameter p **then**
　　$S_1 \leftarrow S_p^*$, $S_p^* \leftarrow S_c^*$, $S_c^* \leftarrow \text{Init}()$, $gen \leftarrow 0$
　end if
　if $S_1 \approx S_2$ **then**
　　$S_2 \leftarrow \text{Init}()$
　end if
　$gen \leftarrow gen + 1$
end while
return S^*

对基于拓扑顺序的交叉解S_1'、S_2'进行禁忌搜索
目的：寻找局部最优S_1、S_2

所提方法 基于两个个体的进化算法

- 禁忌搜索算法

输入：初始解S_0、迭代次数λ、最大完工时间$\bar{C}$

输出：最优解S^*

初始化：当前解$S_c \leftarrow S_0$、最优

$Iter \leftarrow 0$、局部搜索改进标志i

时间关键操作o:$s^{max}(o) - s^{min}(o) = \bar{C} - C_{max}$)
即开始时间之差=最大完工时间之间的松弛量
找“最靠近调度边界，几乎不能被延迟” 的操作

$while\ Iter < \lambda$:
　　for 时间关键操作$o\ in\ S_c$:
　　　　$\boxed{N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)}$
　　　　在邻域中找出加工时间最小且不在禁忌中的解S'(若全是禁忌解随机选取)
　　　　将$Move(S_c, S')$添加进禁忌表，防止回跳
　　$S_c \leftarrow S'$
　　$end\ for$
　　$if\ makespan(S^*) > makespan(S_c)$:
　　　　$S^* \leftarrow S_c; Iter \leftarrow 0$
　　$end\ if$
　　……

N^k邻域结构(机器重分配)

机器重分配禁忌：$\theta_1 = m + rand(\quad)\%(2m)$
若操作o从机器m_o中移除则在接下来的θ_1次的迭代中禁止将其分配给m_o

每台新机器×若干插入位置→ 可行解（前驱操作冲突+资源冲突）

Algorithm 1 Tabu Search Procedure

Input: Initial solution S_0, λ, $\bar{C}$
Output: The best found solution S^*
$S_c \leftarrow S_0$, $S^* \leftarrow S_0$, $N \leftarrow \emptyset$, $Iter \leftarrow 0$, $is\_imp \leftarrow true$
while $Iter < \lambda$ **do**
　for each time-critical operation o in S_c **do**
　　$N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$
　　$S' \leftarrow \arg\min\{makespan(S) | S \in N, S$ is in not tabu status$\}$
　　if S' do not exists **then**/*all neighborhood solutions in N are in tabu status*/
　　　Randomly select one solution S' from N
　　end if
　　Insert the move $Move(S_c, S')$ into tabu list
　　$S_c \leftarrow S'$; $N \leftarrow \emptyset$
　end for
　if m…
　　$S^* \leftarrow S'$; $S_c \leftarrow S'$; $is\_imp \leftarrow true$ **break**
　　end if
　　$N \leftarrow N \setminus \{S'\}$
　end while
…**l while**
…; $Iter \leftarrow Iter + 1$
…**le**
…**n** S^*

所提方法 基于两个个体的进化算法

- 禁忌搜索算法

输入：初始解S_0、迭代次数λ、最

输出：最优解S^*

初始化：当前解$S_c \leftarrow S_0$、最优解

$Iter \leftarrow 0$、局部搜索改进标志$is_$

N^π邻域结构(顺序重分配基于N^7邻域结构)

顺序重分配禁忌:$\theta_2 = n + rand(\quad)\%n$,禁止最近的关键块结构在$\theta_2$步内重复出现

例：$abcdef \rightarrow dabcef \Longrightarrow$部分块$abcd$在$\theta_2$次迭代中禁止出现

$while\ Iter < \lambda$:

  for 时间关键操作$o\ in\ S_c$:

    $N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$

    在邻域中找出加工时间最小且不在禁忌中的解S'(若全是禁忌则随机选取)

    将$Move(S_c, S')$添加进禁忌表，防止回跳

  $S_c \leftarrow S'$

  $end\ for$

  $if\ makespan(S^*) > makespan(S_c)$:

    $S^* \leftarrow S_c; Iter \leftarrow 0$

  $end\ if$

  ……

邻域搜索与选择 → 解更新 → 优化TEC

Algori

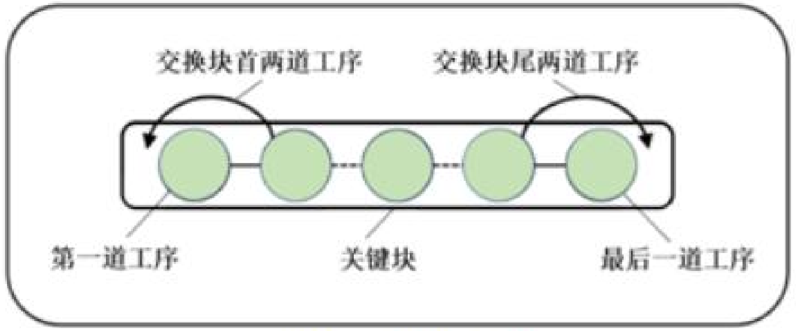


图1 N5邻域结构

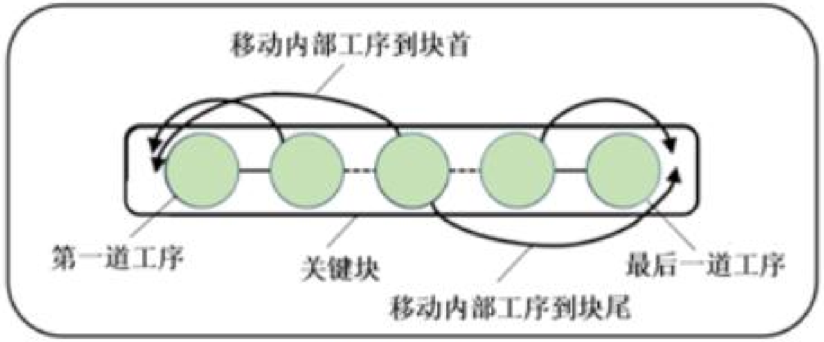


图2 N6邻域结构

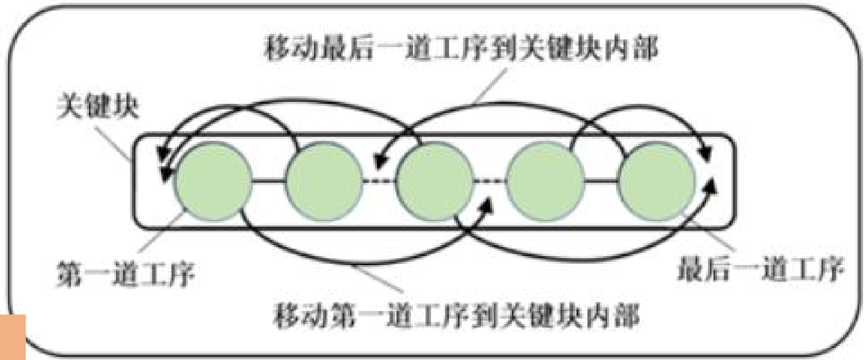


图3 N7邻域结构

所提方法　基于两个个体的进化算法

优化TEC

- 禁忌搜索算法

输入

$while\ Iter < \lambda$:

　… …

　$if\ makespan(S_c) \le \bar{C}$:

　　$while\ is\_tmp\ do$:

　　　$is\_tmp \leftarrow false$

　　　for 操作$o\ in\ S_c$:

　　　　$N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$

　　　$end\ for$

　　　$while\ N$不为空:

　　　　从N中随机选取解S'

　　　　$if\ TEC(S^*) > TEC(S')\ and\ makespan(S') \le \bar{C}$

　　　　　$S^* \leftarrow S'; S_c \leftarrow S'; is_{tmp} \leftarrow true; break$

　　　　$end\ if$

　　　　从N中减去解S'

　　　$end\ while$

　　$end\ while$

　$end\ if$

　$N \leftarrow \emptyset; Iter \leftarrow Iter + 1$

$end\ while$

$first - improvement$策略:找到能耗更优就立刻接受

Algorithm 1 Tabu Search Procedure

1: **Input**: Initial solution S_0, λ, $\bar{C}$
2: **Output**: The best found solution S^*
3: $S_c \leftarrow S_0$, $S^* \leftarrow S_0$, $N \leftarrow \emptyset$, $Iter \leftarrow 0$, $is\_imp \leftarrow true$
4: **while** $Iter < \lambda$ **do**
each time-critical operation o in S_c **do**
$\leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$
arg min$\{makespan(S)|S \in N, S$ is in not tabu status$\}$
do not exists **then**/*all neighborhood solutions in N are in
andomly select one solution S' from N
f
the move $Move(S_c, S')$ into tabu list
S'; $N \leftarrow \emptyset$
$pan(S^*) > makespan(S_c)$ **then**
S_c; $Iter \leftarrow 0$
$pan(S_c) <= \bar{C}$ **then**
is_imp is true **do**
$\_imp \leftarrow false$
r each operation o in S_c **do**
om N
$makespan(S') <= \bar{C}$ **then**
$S^* \leftarrow S'$; $S_c \leftarrow S'$; $is\_imp \leftarrow true$ **break**
end if
$N \leftarrow N \setminus \{S'\}$
nd while
while
; $Iter \leftarrow Iter + 1$
ile
urn S^*

所提方法　基于两个个体的进化算法

创新优化

• 算法

输入：初始解S_0、迭代次数λ、最大完工时间$\bar{C}$

输出：最优解S^*

初始化：当前解$S_c \leftarrow S_0$、最优解$S^* \leftarrow S_0$、候选解N

It

$while\ Iter < \lambda$:

　for 时间关键操作$o\ in\ S_c$:

　$\boxed{N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)}$

　在邻域中找出加工时间最小且不在禁忌

取)

　将$Move(S_c, S')$添加进禁忌表，防止回路

　$S_c \leftarrow S'$

　$end\ for$

　$if\ makespan(S^*) > makespan(S_c)$:

　　$S^* \leftarrow S_c; Iter \leftarrow 0$

　$end\ if$

　$if\ makespan(S_c) \leq \bar{C}$:

Algorithm 1 Tabu Search Procedure

提出邻域剪裁：提前排除无效邻域变换,优化$N^\pi(S_c, o)$

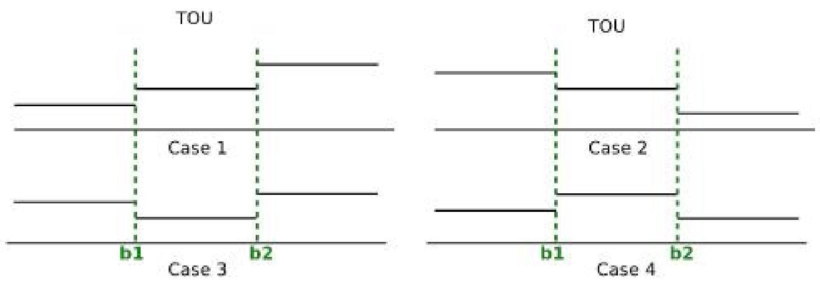


Fig. 3.　Combinations of TOU with three adjacent i

Lemma 1: The energy cost of operation u cannot be reduced on the same machine:

1) if $r(MP[u]) \geq s(u)$ holds for case 1;
2) if $r(MS[u]) \leq s(u)$ holds for case 2;
3) if $b_1 \leq s(u)$ and $b_2 \geq s(u) + P(u)$ hold for case 3;
4) if the following conditions hold for case 4:

$$b_1 \leq \min\{r(JP[u]) + P(JP[u]), r(MP[u]) + P(MP[u])\} \quad (5)$$

$$b_2 \geq \bar{C} - \min(P(JS[u]) + q(JS[u]), P(MS[u]) + q(MS[u]).$$

in

引理1给出无法被剪枝的四种情况：

（1）$MP(u)$最早开始时间≥操作u开始时间 无法左移降低成本，可剪枝

（2）$r(MS(u)) \leq s(u)$ 无法右移降低成本，可剪枝

（3）操作u开始时间和结束时间在$[b_1, b_2]$区间 不可节能，可剪枝

（4）操作u的最早开始时间和最晚完成时间在$[b_1, b_2]$区间 整段在高峰，可剪枝

$N \leftarrow N \setminus \{S'\}$

end while

while

; $Iter \leftarrow Iter + 1$

le

rn S^*

所提方法　基于两个个体的进化算法

创新优化

• 算法

Algorithm 1 Tabu Search Procedure

1: **Input**: Initial solution S_0, λ, $\bar{C}$
2: **Output**: The best found solution S^*

$while\ Iter < \lambda$:
　… …
　$if\ makespan(S_c) \le \bar{C}$:
　　$while\ is\_tmp\ do$:
　　　$is\_tmp \leftarrow false$
　　　for 操作$o\ in\ S_c$:
　　　　$N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$
　　　$end\ for$
　　　$while\ N$不为空:
　　　　从N中随机选取解S'
　　　　$if\ TEC(S^*) > TEC(S')\ and\ makespan(S')$
　　　　　$S^* \leftarrow S'; S_c \leftarrow S'; is_{tmp} \leftarrow true; break$
　　　　$end\ if$
　　　　从N中减去解S'
　　　$end\ while$
　　$end\ while$
　$end\ if$
　$N \leftarrow \emptyset; Iter \leftarrow Iter + 1$
$end\ while$

for 关键操作$o\ in\ S_c$:
　$N \leftarrow N \cup N^t(S_c, o) \cup N^k(S_c, o) \cup N^\pi(S_c, o)$

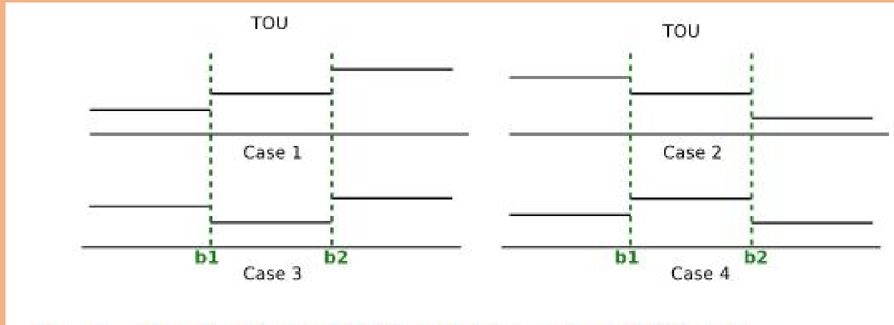


Fig. 3. Combinations of TOU with three adjacent intervals.

（1）$case$ 1:若操作u移动，将能量关键操作后继操作全部左移
（2）$case$2:同上右移
（3）$case$3:在操作u移动后，所有能量关键前驱和后继操作都移动
（4）$case$4: 不移动

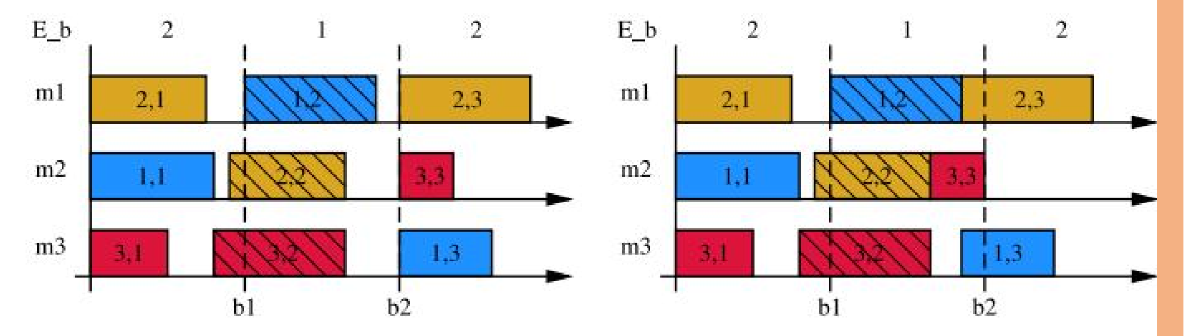


Fig. 1. Illustration of partial compactness.

所提方法 基于两个个体的进化算法

创新优化

• 算法

$while\ Iter < \lambda:$

… …

$if\ makespan(S_c) \le \bar{C}:$

$while\ is\_tmp\ do:$

$is\_tmp \leftarrow false$

for 操作$o\ in\ S_c:$

$N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$

$end\ for$

$while\ N$不为空:

从N中随机选取解S'

$if\ \boxed{TEC(S^*) > TEC(S')}\ an$ … $espan(S') \le$

$S^* \leftarrow S'; S_c \leftarrow S'; is_{tmp} \leftarrow true; break$

$end\ if$

从N中减去解S'

$end\ while$

$end\ while$

$end\ if$

$\boxed{N \leftarrow \emptyset; Iter \leftarrow Iter + 1}$

$end\ while$

TEC估算：

输入：当前解S、邻域解S'

输出：邻域解S'近似能耗

初始化：能耗变化值$\Delta \leftarrow 0$

Algorithm 1 Tabu Search Procedure

1: **Input**: Initial solution S_0, λ, $\bar{C}$

2: **Output**: The best found solution S^*

3: $S_c \leftarrow S_0$, $S^* \leftarrow S_0$, $N \leftarrow \emptyset$, *Iter* $\leftarrow 0$, *is_imp* $\leftarrow$ *true*

4: **while** *Iter* $< \lambda$ **do**

… each time-critical operation o in S_c **do**

… $\leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$

… arg min{*makespan*(S)|$S \in N$, S is in not tabu status}

… do not exists **then**/*all neighborhood solutions in N are in

… andomly select one solution S' from N

… f

… the move *Move*(S_c, S') into tabu list

… S'; $N \leftarrow \emptyset$

… pan(S^*) > *makespan*(S_c) **then**

end if

$N \leftarrow N \setminus \{S'\}$

… nd while

… while

…; *Iter* $\leftarrow$ *Iter* + 1

… ile

… rn S^*

所提方法　基于两个个体的进化算法

- 能耗近似评估

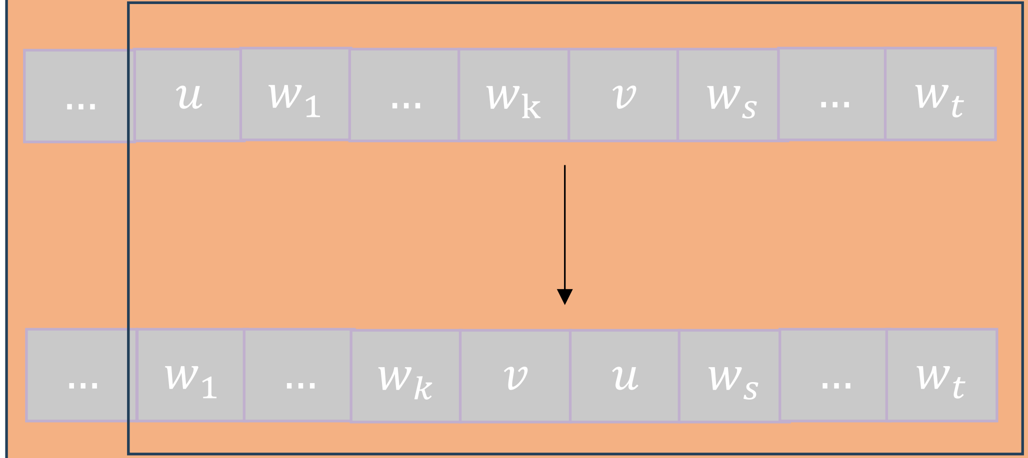
仅统计直接受影响的机器和操作

Algorithm 2 Approximate Estimation of TEC Method

Input: A current solution S, and a neighboring solution $S' \in N(S)$
Output: The approximate TEC value of S'
$\Delta \leftarrow 0$ /* reset the change of TEC to 0 */
if $S' \in N^{\pi}(S, o)$ **then**
　$p \leftarrow \min\{index(o, M_o(S))|S \in \{S, S'\}\}$
　for each operation $o' \in \{M_o(S)|index(o', M_o(S)) >= p\}$ **do**
　　estimate the approximate value $\hat{r}(o')$
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(o', S)$
　end for
else if $S' \in N^{k}(S, o)$ **then**
　$p \leftarrow \min\{index(o, M_o(S)|S \in \{S, S'\}\}$
　for each operation $o' \in \{M_o(S)|index(o', M_o(S)) >= p\}$ **do**
　　estimate the approximate value $\hat{r}(o')$
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(op, S)$
　end for
　for each operation
$o' \in \{M_o(S')|index(o', M_o(S')) >= index(o, M_o(S'))\}$ **do**
　　estimate the approximate value $\hat{r}(o')$
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(o', S)$
　end for
else
　$\Delta \leftarrow EC(o, S') - EC(o, S)$ /*for N^t neighborhoods, only one operation is changed*/
end if
$TEC(S') \leftarrow TEC(S) + \Delta$

顺序重分配能耗估算

$if\ S' \in N^{\pi}(S, o)$:
　$p \leftarrow o$在S和S'中的位置索引的最小值
　for p之后的每个操作o'的索引:
　　估计$\hat{r}(o')$即o'的最早开始时间
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(o', S)$
　$end\ for$
$else\ if\ S' \in N^{k}(S, o)$:
　$p \leftarrow o$在S和S'中的位置索引的最小值



$\hat{r}(o)$只统计直接受影响的机器和操作

$TEC(S') \leftarrow TEC(S) + \Delta$

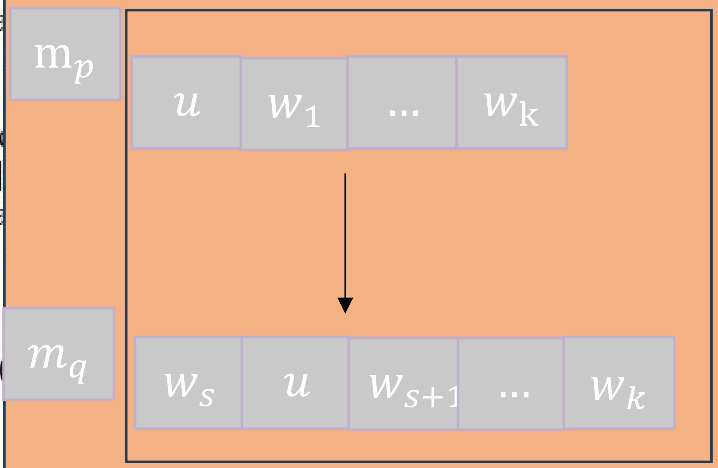
所提方法　基于两个个体的进化算法

- 能耗近似评估

仅统计直接受影响的机器和操作

Algorithm 2 Approximate Estimation of TEC Method

Input: A current solution S, and a neighboring solution $S' \in N(S)$
Output: The approximate TEC value of S'
$\Delta \leftarrow 0$ /* reset the change of TEC to 0 */
if $S' \in N^{\pi}(S, o)$ **then**
    $p \leftarrow \min\{index(o, M_o(S)) | S \in \{S, S'\}\}$
    for each operation $o' \in \{M_o(S) | index(o', M_o(S)) >= p\}$ **do**
        estimate the approximate value $\hat{r}(o')$
        $\Delta \leftarrow \Delta + EC(o', S') - EC(o', S)$
    end for
else if $S' \in N^{k}(S, o)$ **then**
    $p \leftarrow \min\{index(o, M_o(S)) | S \in \{S, S'\}\}$
    for each operation $o' \in \{M_o(S) | index(o', M_o(S)) >= p\}$ **do**
        estima
        $\Delta \leftarrow$
    end for
    for each
$o' \in \{M_o(S')|$
        estima
        $\Delta \leftarrow$
    end for
else
    $\Delta \leftarrow EC$
is changed*/
end if
$TEC(S') \leftarrow T$



$\hat{r}(o)$只统计直接受影响的机器和操作

$if\ S' \in N^{\pi}(S,o)$:
    $p \leftarrow o$在S和S'中的位置索引的最小值
    for p之后的每个操作o'的索引:
        估计$\hat{r}(o')$即o'的最早开始时间
        $\Delta \leftarrow \Delta + EC(o',S') - EC(o',S)$
    $end\ for$

机器重分配能耗估算

$else\ if S' \in N^{k}(S,o)$:
    $p \leftarrow o$在S和S'中的位置索引的最小值
    for p之后的每个操作o'的索引:
        估计$\hat{r}(o')$
        $\Delta \leftarrow \Delta + EC(o',S') - EC(o',S)$
    $end\ for$
    for 邻域解中索引大于o的:
        估计$\hat{r}(o')$
        $\Delta \leftarrow \Delta + EC(o',S') - EC(o',S)$
    $end\ for$
$else$:
    $\Delta \leftarrow \Delta + EC(o,S') - EC(o,S)$
$end\ if$
$TEC(S') \leftarrow TEC(S) + \Delta$

所提方法　基于两个个体的进化算法

- 能耗近似评估

仅统计直接受影响的机器和操作

Algorithm 2 Approximate Estimation of TEC Method

Input: A current solution S, and a neighboring solution $S' \in N(S)$
Output: The approximate TEC value of S'
$\Delta \leftarrow 0$ /* reset the change of TEC to 0 */
if $S' \in N^{\pi}(S, o)$ **then**
　$p \leftarrow \min\{index(o, M_o(S))|S \in \{S, S'\}\}$
　for each operation $o' \in \{M_o(S)|index(o', M_o(S)) >= p\}$ **do**
　　estimate the approximate value $\hat{r}(o')$
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(o', S)$
　end for
else if $S' \in N^{k}(S, o)$ **then**
　$p \leftarrow \min\{index(o, M_o(S)|S \in \{S, S'\}\}$
　for each operation $o' \in \{M_o(S)|index(o', M_o(S)) >= p\}$ **do**
　　estimate the approximate value $\hat{r}(o')$
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(op, S)$
　end for
　for each operation
$o' \in \{M_o(S')|index(o', M_o(S')) >= index(o, M_o(S'))\}$ **do**
　　estimate the approximate value $\hat{r}(o')$
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(o', S)$
　end for
else
　$\Delta \leftarrow EC(o, S') - EC(o, S)$ /*for N^t neighborhoods, only one operation is changed*/
end if
$TEC(S') \leftarrow TEC(S) + \Delta$

$if\ S' \in N^{\pi}(S, o)$:
　$p \leftarrow o$在S和S'中的位置索引的最小值
　for p之后的每个操作o'的索引:
　　估计$\hat{r}(o')$即o'的最早开始时间
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(o', S)$
　$end\ for$
$else\ if S' \in N^{k}(S, o)$:
　$p \leftarrow o$在S和S'中的位置索引的最小值
　for p之后的每个操作o'的索引:
　　估计$\hat{r}(o')$
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(o', S)$
　$end\ for$
　for 邻域解中索引大于o的:
　　估计$\hat{r}(o')$
　　$\Delta \leftarrow \Delta + EC(o', S') - EC(o', S)$
　$end\ for$
$else$:
　$\boxed{\Delta \leftarrow \Delta + EC(o, S') - EC(o, S)}$
$end\ if$
$TEC(S') \leftarrow TEC(S) + \Delta$

N^t能耗估算

所提方法　基于两个个体的进化算法

创新优化

• 算法

$while\ Iter < \lambda$:

… …

$if\ makespan(S_c) \le \bar{C}$:

$while\ is\_tmp\ do$:

$is\_tmp \leftarrow false$

for 操作$o\ in\ S_c$:

$N \leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$

$end\ for$

$while\ N$不为空:

从N中随机选取解S'

$if\ TEC(S^*) > TEC(S')\ and\ makes$

$S^* \leftarrow S'; S_c \leftarrow S'; is_{tmp} \leftarrow true$

$end\ if$

从N中减去解S'

$end\ while$

$end\ while$

$end\ if$

$\boxed{N \leftarrow \emptyset; Iter \leftarrow Iter + 1}$

$end\ while$

添加随机扰动机制

目标：通过扰动和局部搜索跳出局部最优

若500次迭代内，最优解无改进，则对$0.2 \times |N_c|$个操作随机执行扰动$N^k \cup N^\pi$（N_c为时间关键操作）

Algorithm 1 Tabu Search Procedure

1: **Input**: Initial solution S_0, λ, $\bar{C}$

2: **Output**: The best found solution S^*

3: $S_c \leftarrow S_0$, $S^* \leftarrow S_0$, $N \leftarrow \emptyset$, $Iter \leftarrow 0$, $is\_imp \leftarrow true$

4: **while** $Iter < \lambda$ **do**

each time-critical operation o in S_c **do**

$\leftarrow N \cup N^k(S_c, o) \cup N^\pi(S_c, o)$

arg min$\{makespan(S)|S \in N, S$ is in not tabu status$\}$

do not exists **then**/*all neighborhood solutions in N are in

andomly select one solution S' from N

f

the move $Move(S_c, S')$ into tabu list

S'; $N \leftarrow \emptyset$

$pan(S^*) > makespan(S_c)$ **then**

S_c; $Iter \leftarrow 0$

$pan(S_c) <= \bar{C}$ **then**

is_imp is true **do**

$\_imp \leftarrow false$

)

$'$ from N

$kespan(S') <= \bar{C}$ **then**

$\leftarrow true$ **break**

$N \leftarrow N \setminus \{S'\}$

nd while

while

; $Iter \leftarrow Iter + 1$

ile

rn S^*

所提方法　基于两个个体的进化算法

Algorithm 3 TIE for FJSP With TOU Scheme

Input: Problem instance
Output: The best solution S^* found
$gen \leftarrow 0$; $S_1, S_2, S_c^*, S_p^*, S^* \leftarrow \text{Init}()$
while stopping condition is not reached **do**
　$S_1' \leftarrow \text{TOCX}(S_1, S_2)$, $S_2' \leftarrow \text{TOCX}(S_2, S_1)$
　$S_1 \leftarrow \text{TS}(S_1')$, $S_2 \leftarrow \text{TS}(S_2')$
　$S_c^* \leftarrow \text{save\_best}(S_1, S_2, S_c^*)$
　$S^* \leftarrow \text{save\_best}(S_c^*, S^*)$
　if gen is equal to an integer parameter p **then**
　　$S_1 \leftarrow S_p^*$, $S_p^* \leftarrow S_c^*$, $S_c^* \leftarrow \text{Init}()$, $gen \leftarrow 0$
　end if
　if $S_1 \approx S_2$ **then**
　　$S_2 \leftarrow \text{Init}()$
　end if
　$gen \leftarrow gen + 1$
end while
return S^*

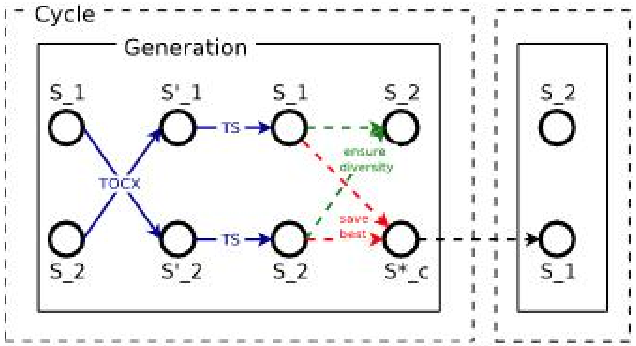


Fig. 5.　General framework of TIE.

更新当前周期最优解→ S_c^*
更新全局最优化→ S^*
检查周期：
　上一周期最优解→ S_1
　当前周期最优解→ S_p^*
　下一周期初始化→ S_c^*

停止条件：算法运行时间≥ T_{max}

试验比较

COMPARISON (RPD %) WITH SP, DP, AND THEIR HYBRIDS

Ins.	SP	DP	TS(DP)	TS(SP)		TIE	
				1 m.	10 m.	1 m.	10 m.
BCdata	3.07	3.61	3.35	2.20	2.10	0.93	**0.04**
BRdata	15.44	15.25	3.02	2.55	**1.12**	3.31	1.65
DPdata	0.67	0.36	0.69	0.61	**0.51**	2.24	0.92
HUdata/sdata	3.21	2.95	2.27	1.57	1.39	1.21	**0.37**
HUdata/edata	2.79	2.31	2.22	1.63	1.51	1.00	**0.29**
HUdata/rdata	2.09	1.63	1.40	1.00	0.82	[illegible]	[illegible]
HUdata/vdata	2.77	2.32	1.03	0.77	**0.37**	[illegible]	[illegible]
Mean	3.03	2.69	1.82	1.31	1.07	[illegible]	[illegible]

TOU	SP	DP	TS(DP)	TS(SP)		TIE	
				1 m.	10 m.	1 m.	10 m.
TOU0	6.27	3.82	2.60	1.94	1.66	1.81	**0.67**
TOU1	2.38	2.34	1.40	0.97	0.77	1.02	**0.40**
TOU2	2.49	1.45	1.51	1.20	0.93	1.64	**0.86**
TOU3	4.05	3.31	1.72	1.15	0.90	1.17	**0.44**
TOU4	2.53	2.63	1.30	1.93	1.60	2.46	**1.23**

SUMMARY OF TEST RESULTS OF TIE, HEA, AND ILS

TOU	ILS				HEA				TIE			
	TEC_{avg}	TEC_{sd}	$time_{avg}$	RPD	TEC_{avg}	TEC_{sd}	$time_{avg}$	RPD	TEC_{avg}	TEC_{sd}	$time_{avg}$	RPD
TOU0	724.69	4.03	298.35	0.99	724.29	3.69	311.67	0.93	717.59	3.20	318.78	0.00
TOU1	771.54	2.53	300.25	0.47	771.10	2.15	307.81	0.41	767.96	1.92	325.77	0.00
TOU2	940.30	3.40	301.46	0.64	940.47	3.76	321.27	0.66	934.29	3.24	306.77	0.00
TOU3	944.99	3.89	311.27	0.67	943.84	3.25	311.50	0.55	938.66	2.72	325.94	0.00
TOU4	776.73	5.70	268.88	1.04	775.81	4.52	316.39	0.92	768.71	3.63	310.43	0.00
TOU5	800.74	2.93	292.43	0.60	799.79	2.52	312.64	0.48	795.96	2.14	323.48	0.00
TOU6	988.81	4.18	301.59	0.66	988.34	3.74	311.83	0.61	982.30	3.33	324.47	0.00
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	995.55	4.21	303.13	0.64	989.23	3.51	311.57	0.00
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	867.40	3.48	312.03	0.65	**861.84**	**2.96**	318.40	**0.00**

, and ILS on *BCdata*, *BRdata*, and *DPdata*.
cates that the three kinds of optimization frameworks distinguish each other statistically.

COMPARISON OF DIFFERENT CROSSOVER OPERATORS IN TIE WITH TOU5 ON *BRdata*

Ins.	POX			PRX			TOCX		
	TEC_{min}	TEC_{avg}	TEC_{sd}	TEC_{min}	TEC_{avg}	TEC_{sd}	TEC_{min}	TEC_{avg}	TEC_{sd}
Mk01	31.4	31.5	0.1	31.2	31.32	0.1	30.8	31.04	0.15
Mk02	29	29.12	0.13	29	29.14	0.13	28.8	28.96	0.08
Mk03	157.2	159.86	1.16	157.4	159.21	0.76	157.1	158.93	1.36
Mk04	72.2	72.95	0.38	72.3	72.8	0.22	71.7	72.07	0.24
Mk05	149	149.49	0.31	148.8	149.21	0.25	148.8	149.04	0.23
Mk06	82.3	83.08	0.47	81.6	82.76	0.54	80.9	82.11	0.6
Mk07	156	157.08	0.76	155.1	157.02	0.88	156.1	156.95	0.51
Mk08	459.3	461.93	1.84	458.6	461.84	1.55	452.6	458.03	2.68
Mk09	457.2	460.56	2.13	452.8	458.52	2.77	456.7	460.8	3.08
Mk10	425.1	427.01	1.43	423.8	425.97	1.33	422.8	425.82	1.61
Avg.	201.87	203.26	0.87	201.06	202.78	0.85	**200.63**	**202.38**	1.05

Friedman's test is conducted on TEC_{avg} obtained by TIE, TIE with POX, and TIE with PRX on *BCdata*.
The resulting small value of $p < 0.001$ indicates the three crossover operators distinguish each other statistically.

p-VALUES OF WILCOXON'S SIGNED RANK TESTS ON THE CUTOFF TIMES OF TIE

	1 min.	5 min.	10 min.	20 min.	30 min.
1 min.	-	0.0002	0.0002	0.0002	0.0002
5 min.	-	-	0.0004	0.0002	0.0003
10 min.	-	-	-	0.0512	0.0576
20 min.	-	-	-	-	0.0581
30 min.	-	-	-	-	-

THANK YOU