一种常见干扰条件下的开放式车间重调度研究①

刘 乐，周 泓
（北京航空航天大学经济管理学院，北京 100191）

摘要： 在开放式车间中，针对由机器持续不可用干扰（MUAD）及其引发的工时缩减现象，致力于研究如何有效、及时地实施重调度活动. 其中，重调度性能由总完工期来度量，而重调度稳定性则体现在序位偏差与结束时间偏差上. 基于右移、受影响工序、全局三种典型的重调度策略分别提出并实现了三种特定的重调度方法（即 sRSR，sAOR 与 sTR_GOS）以响应所关注的干扰条件. 在仿真实验中通过模拟大量的重调度情景，考察了原调度生成机制的选取问题并对比了三种方法在相同情景下的各自指标绩效. 实验结果显示：当 MUAD 干扰持续较短时间且以“中断—可续”模式恢复被中断工序时，推荐选用 sAOR 实施重调度；当 MUAD 干扰发生在晚期、工时低幅缩减、算例较小且中断模式为“中断—不可续”时，适宜采取 sTR_GOS 实施重调度.

关键词： 重调度；开放式车间；干扰；稳定性；受影响工序重调度
中图分类号： F406.2 **文献标识码：** A **文章编号：** 1007－9807（2014）06－0028－21

0 引 言

在实际生产环境中会随时遭遇到各种干扰的影响，如紧急订单到达、机器故障、因质检不达标而返工、订单撤销、释放期变更等. 突发干扰事件往往会导致原先的调度方案执行起来不顺畅、效率低下，甚至不再可行；此时亟需对原调度进行修复或再生成，于是引出了生产运作管理中的重调度问题. 重调度是预测—反应式调度（predictable-reactive scheduling）中的重要环节[1]，其宗旨在于尽量小牺牲调度性能（efficiency）的同时，追求调度稳定性（stability）的优越[2-3]. 调度稳定性体现在干扰负面影响的度量上；扰后针对原调度做出的任何变动都会对先期的人、物、财力投入造成废弃，因而有效控制或缩小新、原调度之间的偏差程度对降低运作成本、提高运作效率具有重要的现实意义[4-5].

机器持续不可用（machine unavailability for a duration，MUAD）干扰是生产运作系统中最常见的干扰类型之一. 机器故障与维护、物料或能源供应不利、员工缺勤等都可归结为 MUAD 干扰. 然而伴随着 MUAD 干扰的发生，在制造实践中生产管理者往往会通过加班（taking overtime）或转包（outsourcing）等方式不失时机地缩减个别受影响工序的加工时间，以主动地弥补 MUAD 干扰所导致的时间损失. 从运作管理的角度看，以一定的制造成本人为地缩减工序工时同样会影响后续的生产调度决策[6]. 由此，MUAD 干扰及其伴随出现的工时缩减（compressions of processing times，CPT）现象联合形成了制造实践中常见的一种干扰情景，即：MUAD_{CPT} 干扰. MUAD_{CPT} 干扰比单一 MUAD 干扰更具普适性、现实性. Li 等[7] 首先将 MUAD_{CPT} 干扰引入生产反应式调度研究中；不过自此以后，该干扰并未引起足够的学术关注.

除了生产干扰之外，机器配置环境也是区分重调度研究对象的准则之一[8]. 制造车间是生产重调度研究中主要关注的配置环境. 在车间环境下加工各工件需要经历多道工序，每道工序要在

① 收稿日期：2012－03－14；修订日期：2013－05－21.
基金项目：国家自然科学基金资助项目（71071008；70821061）.
作者简介：刘 乐（1984—），男，山东济南人，博士生. Email：liu_le@sem.buaa.edu.cn

不同机器上执行,因而每个工件将按某一次序(即加工路线)到达各机器以完成加工. 如果所有工件的加工路线事先未限定,则该情形下的调度问题为开放式车间调度问题(open shop scheduling problem, OSSP). 由于工件的加工路线也成为优化对象,开放式车间调度问题在解空间上远大于同等规模的流水、作业车间调度问题. 尽管求解起来高度困难,它在制造实践、日常生活中却普遍存在,且现实意义较强.

右移重调度(right shift rescheduling, RSR)、部分重调度(partial rescheduling)以及全局重调度(total rescheduling, TR)是三类重要的车间重调度实施策略[2-3,9]. 在车间制造实践中,这三种策略执行起来效果不一,往往要根据所面临的干扰条件酌情选取. 纵观重调度文献,动态车间环境中有关重调度机制、方法的探究主要集中于流水[10-14]、作业[2,8-9,15-19]、柔性制造车间[7,20-23]三种配置类型,而在相对复杂的开放式车间环境下的重调度方法研究成果则迄今罕见[24].

为此,在开放式车间环境中,本文针对 $MUAD_{CPT}$ 干扰发生后如何高效率、低成本地实施重调度进行了系统的方法与仿真研究. 预期通过该研究,为调度实践者在开放式车间中合理地选取重调度实现机制以妥善应对机器干扰及其并发影响,提供有价值的启示与思路.

1　相关研究

重调度问题根植于生产运作管理的实践中,一直以来备受国内外调度学者的关注. 动态生产环境下各种引发重调度的干扰事件自由地存在;不仅形式多样,而且影响效果各异. 为贴近于本文的研究,在此不妨回顾一下车间环境中面向单一 MUAD 干扰的重调度研究成果;其中假定 MUAD 干扰在原调度生成后至多出现一次.

首先关注流水车间环境. Akturk 和 Gorgulu[11] 针对机器故障干扰致力于重调度策略研究,根据控制反馈机制提出了有效确定匹配时刻(match-up point)的方法,以求通过部分调整来实现尽可能跟原调度相匹配的效果. Kurihara 等[12] 基于分支定界法提出了一种新型重调度算法,对比分析结果验证了该新算法的有效性. 李铁克等[14] 研究了机器故障下的混合流水车间重调度问题,文中不仅构建了动态约束满足模型,还提出了基于局部性修复的有效重调度算法.

在作业车间环境下,Abumaizar 和 Svestka[2] 提出了面向受影响工序的启发式重调度方法(affected operations rescheduling, AOR),该方法实施起来兼顾调度性能与稳定性的优化,并在与右移、全局重调度方法的实验对比分析中显示出优越性. Mason 等[9] 以半导体制造为背景,针对作业车间中的机器故障干扰,利用专门的实验分析手段考察了三种所提方法的重调度表现. 同样面对单一的 MUAD 干扰,Dong 和 Jang[8] 开发出两种新颖的启发式重调度算法以优化工件的平均延误时间: 即 AWI-J 和 AWI-O; 仿真实验结果表明,AWI-J 在调度性能上优于 AOR,而 AWI-O 在调度性能和稳定性上则均优于 AOR.

对于更复杂车间环境,有关研究则相对少见. 在柔性车间环境中,Li 等[7] 基于“调度二叉树”的构建思路及物资需求计划中的净变(net change)方法,提出了面向受影响工序的部分重调度算法. Louis 和 Xu[24] 针对开放式车间环境,将遗传算法与援例推理相结合,研究了机器故障与更新干扰影响下的重调度问题.

2　干扰因素与重调度任务

为表述方便起见,表 1 给出了文中用到的符号表示及其说明.

2.1　开放式车间完工期调度问题

在 OSSP 研究文献中,最小化完工期(makespan)[25-31] 是迄今研究最多、最深入的问题形式之一; 根据通用的三域调度问题表示法可将其记为: $O_m \mid\mid C_{\max}$. 本文拟以 $O_m \mid\mid C_{\max}$ 问题为原始问题展开重调度研究.

问题 $O_m \mid\mid C_{\max}$ 可具体描述如下. 工件集 J 欲在机器集 M 上进行加工,每个工件 J_j 需经历 m 道工序,每道工序 $(O_{i,j}, \forall i, \forall j)$ 须在不同的机器上执行. 工件 $J_j(j = 1, 2, \cdots, n)$ 在机器 $M_i(i = 1, 2, \cdots, m)$ 上的加工时间 $p_{i,j} > 0$ 固定且已知. 两个关键的假设条件分别为: 1) 一台机器上不能同

时加工多个工件，且一个工件不能同时在多台机器上进行加工；2) 同一机器上工件的加工次序任意，每个工件的加工路线亦无限制. 严格地，其他假设还包括：所有工件相互独立并在 0 时刻均处于可得状态(available)；工序间不允许抢先占有；工序的准备时间计入其加工时间. 调度的目标在于：在满足各种假设与约束条件的前提下，确定出面向所有 $m \times n$ 道工序的调度 S^*，以使得对任一面向全体工序集的调度 S，满足 $C_{\max}(S^*) = \max_{M_i \in M, J_j \in J}\{t_{i,j}(S^*) + p_{i,j}\} \leqslant C_{\max}(S)$.

表 1 文中用到的符号表示及其说明

Table 1 Notations used throughout this paper and their definitions

符号	说明
n; m	工件数、机器数
$\boldsymbol{J} = \{J_j \mid j = 1, 2, \cdots, n\}$	待加工的工件集合
$\boldsymbol{M} = \{M_i \mid i = 1, 2, \cdots, m\}$	机器集合
$O_{i,j}$	工件 J_j 在机器 M_i 上的加工工序
$\boldsymbol{O} = \{O_{i,j} \mid M_i \in \boldsymbol{M};\ J_j \in \boldsymbol{J}\}$	0 时刻由全部 $m \times n$ 个工序组成的工序集
$p_{i,j}$	工序 $O_{i,j}$ 在 0 时刻的加工时间
$\boldsymbol{S}$	调度方案(包括全部工序及其开始、结束时间)
$t_{i,j}(\boldsymbol{S})$; $e_{i,j}(\boldsymbol{S})$	工序 $O_{i,j}$ 在 $\boldsymbol{S}$ 中的开工时间、结束时间
$C_{\max}(\boldsymbol{S}) = \max\{t_{i,j}(\boldsymbol{S}) + p_{i,j}, M_i \in \boldsymbol{M};\ J_j \in \boldsymbol{J}\}$	调度 $\boldsymbol{S}$ 的完工期
$DM \in \boldsymbol{M}$	出现 MUAD 干扰的机器
t_D; DT	MUAD 干扰的发生时间、持续时长
$\boldsymbol{S_O}$, $\boldsymbol{S_N}$	原调度和实施重调度后的新调度
$\boldsymbol{O}' = \{O_{i,j} \mid t_{i,j}(\boldsymbol{S_O}) > t_D$ or $t_{i,j}(\boldsymbol{S_O}) < t_D < e_{i,j}(\boldsymbol{S_O});\ M_i \in \boldsymbol{M};\ J_j \in \boldsymbol{J}\}$	在 t_D 时刻 $\boldsymbol{S_O}$ 中由所有未开工或被中断的工序组成的集合($\mid \boldsymbol{O}' \mid$ 等于集合 $\boldsymbol{O}'$ 中的工序数目)
$O_{DM,j^*} \in \boldsymbol{O}'(J_{j^*} \in \boldsymbol{J})$	原方案 $\boldsymbol{S_O}$ 中的直接受扰工序
$Y_{\min}$	MUAD 干扰对工序 O_{DM,j^*} 造成的最小结束时间推迟量
$p'_{i,j}$	MUAD_{CPT} 干扰发生后工序 $O_{i,j} \in \boldsymbol{O}'$ 的工时
$t'_{i,j} = t_{i,j}(\boldsymbol{S_N})$; $e'_{i,j} = e_{i,j}(\boldsymbol{S_N})$	工序 $O_{i,j} \in \boldsymbol{O}'$ 在 $\boldsymbol{S_N}$ 中的开工时间、结束时间
$SD(\boldsymbol{S_O}, \boldsymbol{S_N})$, $TD(\boldsymbol{S_O}, \boldsymbol{S_N})$	新、原调度之间的序位、结束时间偏差量

注：在不会引起歧义的情况下，符号 $t_{i,j}(\boldsymbol{S})$、$e_{i,j}(\boldsymbol{S})$ 和 $C_{\max}(\boldsymbol{S})$ 可分别简记为 $t_{i,j}$、$e_{i,j}$ 和 $C_{\max}$.

2.2 干扰因素的引入与界定

MUAD 干扰通常由发生时刻、所处机器和持续时间三个属性来界定、描述[2]. 本文假定 MUAD 干扰发生后这三个属性已确知，发生时刻不晚于原调度的完工期，且持续时长有限. MUAD 干扰突然发生后，倘若需要实施重调度，那么在原调度中必出现了一批受扰工序(affected operations)；这些工序受机器不可用或加工能力的限制而无法按照预定的起止时间完成加工. 此时，为了将原调度重新调整为可执行的，不得不延迟某些工序的结束时间或推迟某些工件的完工时间，继而导致整个工期延长、机器负载加重、在制品库存成本增加、遭到拖期惩罚等一系列负面影响. MUAD 干扰发生后，运作管理者在主观上不情愿依旧按原定工时重新安排调度方案，而会在实际条件允许的前提下，想方设法地通过加班、转包等人为手段缩减原调度中个别受扰工序的剩余工时，以求缓解 MUAD 干扰所引起的诸多负面影响.

在甘特图表示中，如果某工序的加工时段跟 MUAD 干扰的持续时段出现部分重叠(overlapping)，则该工序被称为直接受扰工序(directly affected operation)[2]. 由生产运作实践和文献[7]可发现，工时缩减的对象工序往往为率先遭遇到 MUAD 干扰的直接受扰工序或其后

继工序. 直接受扰工序的加工过程有可能被 MUAD 干扰所中断,而中断后的剩余加工任务要待干扰恢复后再执行. 根据加工特征的不同,工序的中断模式通常有两种选择[2]: 中断—可续(interrupt-resume);中断—非可续(interrupt-repeat). 两中断模式的区别在于对被中断工序剩余工时的计量: 对于中断—可续模式,剩余工时等于工序总工时减去干扰发生之前已耗用的加工时间; 对于中断—非可续模式,剩余工时为该被中断工序的原定工时.

受 MUAD 干扰影响的工序将面临工时缩减; 可能的缩减量即使源自人为确定,也须综合考虑工序(剩余)工时、干扰持续时长、中断模式、工时缩减成本等客观因素后再做决定. 由于工时缩减的对象集中于直接受扰工序及其后继工序上,在此不妨将总工时缩减的最大量限定为 MUAD 干扰对直接受扰工序所造成的最小结束时间推迟量,记为 $Y_{\min}$. 事实上,根据直接受扰工序的受影响情形和中断模式的不同,$Y_{\min}$ 的确定存在以下三种可能:

1) 如果 MUAD 干扰的发生时刻 t_D 不晚于直接受扰工序 O_{DM,j^*} 的开始时间 $t_{DM,j^*}(S_O)$,如图 1(a) 所示,则 O_{DM,j^*} 的最小结束时间推迟量为: $Y_{\min} = t_D + DT - t_{DM,j^*}(S_O)$.

2) 如果 MUAD 干扰中断了工序 O_{DM,j^*} 的加工,即 $t_{DM,j^*}(S_O) < t_D < e_{DM,j^*}(S_O)$,且中断模式为 interrupt-resume,如图 1(b) 所示,则 O_{DM,j^*} 的最小结束时间推迟量为 $Y_{\min} = DT$.

3) 如果 MUAD 干扰中断了工序 O_{DM,j^*} 的加工且中断模式为 interrupt-repeat,如图 1(c) 所示,则 O_{DM,j^*} 的最小结束时间推迟量为 $Y_{\min} = DT + t_D - t_{DM,j^*}(S_O)$.

综上,$Y_{\min}$ 的统一计算方式可表示为

$$Y_{\min} = (t_D + DT) - \max\{t_D, t_{DM,j^*}(S_o)\} + (1 - I) \cdot \max\{t_D - t_{DM,j^*}(S_o), 0\} \quad (1)$$

其中 0 - 1 变量 I 指明直接受扰工序所采取的中断模式,即

$$I = \begin{cases} 0, \text{若中断模式为 interrupt-repeat} \\ 1, \text{若中断模式为 interrupt-resume} \end{cases} \quad (2)$$

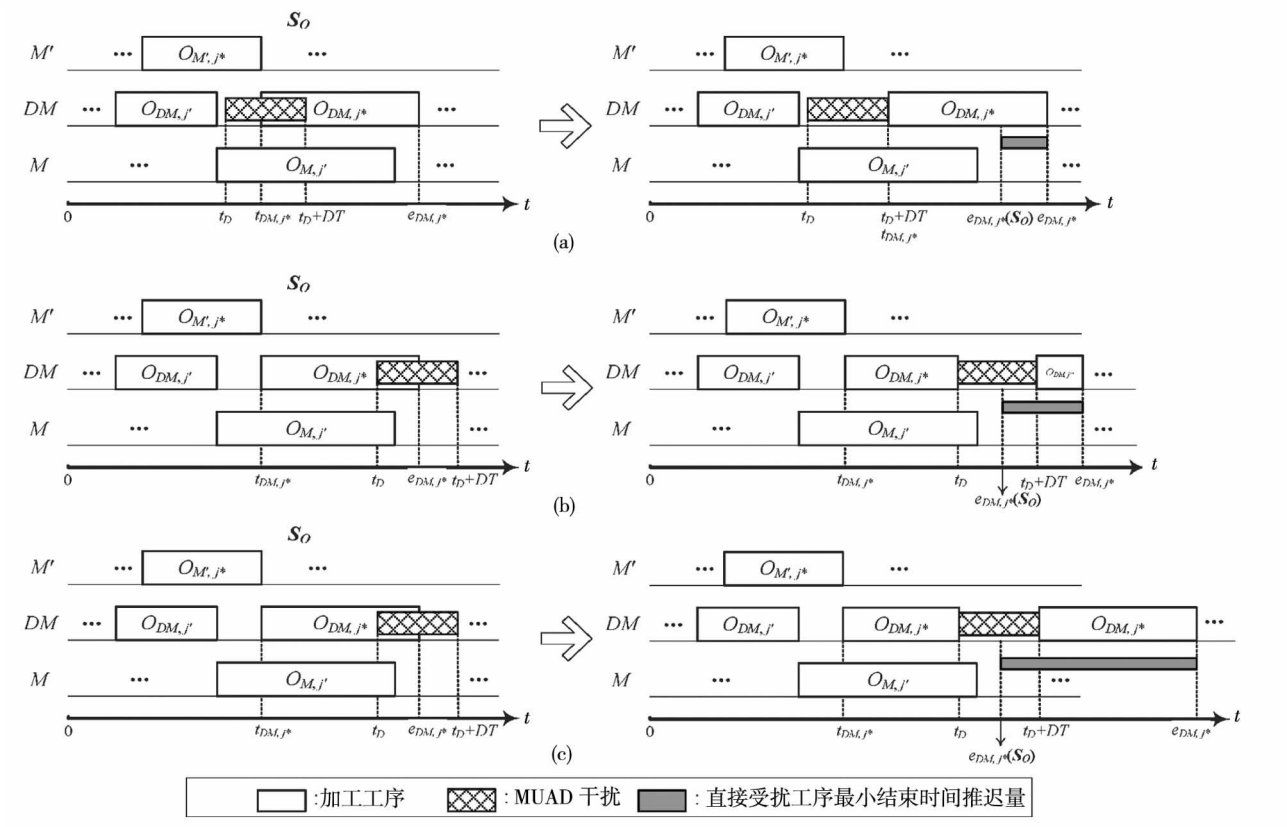

图 1　MUAD 干扰对直接受扰工序影响情况示意图

Fig. 1 Illustration of directly affected operation by MUAD disruption

2.3 重调度绩效评价与度量

重调度的绩效优劣主要体现在性能、稳定性及计算代价上. 重调度的计算代价可由整个重调度过程的耗用时间来衡量. 重调度性能水平通常以某个调度目标函数为评价依据,如完工期、总完工时间、最大延迟时间、总加权延误时间等. 一种常见做法是,以生成原调度时所优化的目标函数作为重调度性能指标. 本研究中的原调度通过求解问题 $O_m \mid\mid C_{\max}$ 而得到,则重调度性能水平由 $C_{\max}$ 值来反映. 另外,本研究通过统计新、原调度在工序序位和时间表(time table)上偏差量来度量重调度的稳定性,详见以下两量化指标.

1) 序位偏差

沿用文献[2]中的做法统计序位偏差量: 设函数 $F_1(O_{i,j}, S)$ 返回调度 S 中位于机器 M_i 上所有排在 $O_{i,j}$ 之前的工序集合,函数 $F_2(O_{i,j}, S)$ 返回调度 S 中位于机器 M_i 上所有排在 $O_{i,j}$ 之后的工序集合; 对于任一工序 $O_{i,j} \in \boldsymbol{O}'$,取 $Q_{i,j}(S_O, S_N) = |F_1(O_{i,j}, S_O) \cap F_2(O_{i,j}, S_N)|$. 此时,新、原方案间的序位偏差量 $SD(S_O, S_N)$ 等价于工序集 $\boldsymbol{O}'$ 中各工序对应 $Q_{i,j}(S_O, S_N)$ 值的累加和,即

$$SD(S_O, S_N) = \sum_{O_{i,j} \in \boldsymbol{O}'} Q_{i,j}(S_O, S_N) \tag{3}$$

2) 结束时间偏差

在车间环境下,可通过累加各工序在新、原调度中的开始时间或结束时间之差的绝对值来衡量干扰在时间表上造成的负面效应. 本文拟采取基于结束时间的偏差统计方式,即

$$TD(S_O, S_N) = \sum_{O_{i,j} \in \boldsymbol{O}'} |e'_{i,j} - e_{i,j}(S_o)| \tag{4}$$

2.4 重调度情景与任务

本研究所针对的重调度情景可描述如下. 在开放式车间环境中,通过求解问题 $O_m \mid\mid C_{\max}$ 得到原调度 S_O. t_D 时刻之前,加工过程将按照 S_O 有条不紊地执行. 在 t_D 时刻,MUAD 干扰在机器 $DM \in M$ 上发生,继而机器 DM 在之后的 DT 个时间单位内处于不可用状态. 一旦所发生的 MUAD 干扰直接影响到 S_O 中的某道工序(即 O_{DM,j^*}),则 O_{DM,j^*} 及其后续工序的(剩余)工时可能会被人为缩减(如 $p'_{DM,j^*} \leqslant p_{DM,j^*} - I \cdot \max\{t_D - t_{DM,j^*}(S_O), 0\}$),并满足总工时缩减量不超过 MUAD 干扰对工序 O_{DM,j^*} 所造成的最小结束时间推迟量. 由此,突发的 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰条件形成,此时原调度 S_O 不再有效,亟需对所有未完成的工序实施重新调度.

$\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰发生后所面临的重调度任务包括如下: 1) 将 t_D 时刻 S_O 中尚未开工或被中断的工序收集起来,组成扰后重调度工序集 O'; 2) 在满足开放式车间中加工能力约束的前提下,利用特定的重调度策略与方法得到面向工序集 O' 的新调度 S_N; 3) 为了评价新调度的重调度绩效,计算出 S_N 在完工期、序位和时间表偏差三指标上的结果值,并记录整个重调度过程的耗用时间.

3 不同策略下的重调度实施

右移重调度(RSR)与受影响工序重调度(AOR)是车间重调度理论与实践中两种重要的反应式调度修复手段,具备易实现、响应速度快的优势. 特别是面向受影响工序的部分重调度机制 AOR,凭借其在调度性能、稳定性及实施效率上的均衡表现,已引起了诸多调度学者的广泛关注[2,7-8,15,32]. 全局重调度(TR)或调度再生成是一种优化机理主导下的重调度策略[2,9],通常需在由所有未完成工序构成的解空间中执行专门的优化搜索或遍历过程. 它所消耗的计算代价往往要大于反应式调度修复机制,但由于在解空间内寻优也增加了求得高性能重调度方案的可能性. 本节拟针对开放式车间中的 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰,基于右移、受影响工序、全局重调度策略分别提出了三种重调度方法,即 sRSR、sAOR 和 sTR_GOS. 下面三子节将对三种方法的实现细节进行逐一探讨.

3.1 右移重调度方法

右移重调度是一种简单易行的调度修复手段. 它的普遍做法是: 将所有未完成的工序统一推迟特定长度个时间单位(在甘特图上表现为向右移动),以适应突发的干扰因素并确保新生成的调度可行. 由右移重调度得到的新调度保持了原调度中的工序间相对次序,因而不会导致序位偏差; 不过由于未完工工序的结束时间都被延后,则

右移修复后的新调度在完工期、总完工时间等指标上鲜有好的表现. 针对开放式车间中的 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰,本小节提出一种特效的右移重调度方法(specific RSR, sRSR) ,其详细步骤如下:

步骤 1　根据原调度 $\boldsymbol{S}_O$ 以及 MUAD 干扰信息 DM、t_D 和 DT,确定出干扰发生时全部未开工或被中断的工序,继而得到工序集合 $\boldsymbol{O}'$.

步骤 2　由式(1) 计算出 MUAD 干扰对工序 O_{DM,j^*} 所造成的最小结束时间推迟量 $Y_{\min}$,进而由式(5) 求出整体右移量(amount of right-shift, ARS) ,即

$$ARS = Y_{\min} - (p_{DM,j^*} - I \cdot \max\{t_D - t_{DM,j^*}(\boldsymbol{S}_O), 0\} - p'_{DM,j^*}) \quad (5)$$

步骤 3　确定直接受扰工序 O_{DM,j^*} 的开始时间和结束时间如下

$$t'_{DM,j^*} = t_D + DT, e'_{DM,j^*} = e_{DM,j^*}(\boldsymbol{S}_O) + ARS$$

步骤 4　将工序集 $\boldsymbol{O}' \backslash \{O_{DM,j^*}\}$ 中各工序的开始、结束时间统一延后 ARS 个时间单位,即

$$t'_{i,j} = t_{i,j}(\boldsymbol{S}_O) + ARS, e'_{i,j} = e_{i,j}(\boldsymbol{S}_O) + ARS \quad \forall O'_{i,j} \in \boldsymbol{O}' \backslash \{O_{DM,j^*}\}$$

步骤 5　输出得到的新调度 $\boldsymbol{S}_N$,结束.

sRSR 方法的计算代价主要消耗在了步骤 4 上. 考虑到工序集 $\boldsymbol{O}' \subseteq \boldsymbol{O}$,则至多 $m \times n$ 个工序的结束时间会被右移推迟. 由此,sRSR 方法的时间复杂度为 $O(m \times n)$.

3.2　面向受影响工序的重调度方法

受影响工序重调度(AOR) 是部分重调度的典型实现形式,仅对直接或间接受到机器干扰影响的工序实施反应式修复. 事实上,并非所有未开工的工序都会受到干扰的影响,某些工序的原定加工时间表在扰后仍然有效. AOR 方法实施起来须贯彻以下三项原则[2,7]: 1) 基于“二元分支(binary branching) ”机理从直接受影响工序开始逐一识别出所有受影响工序; 2) 满足相应机器环境下的加工约束条件,并保持原调度中工序间的相对次序; 3) 在妥善处理干扰的前提下尽量小地推迟受扰工序的加工时间表.

对于发生在开放式车间中的 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰,同样可通过构建二叉树结构的方式识别出原调度中所有受影响工序. 具体地,$\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰对原调度所引发的时间冲突效应从直接受扰工序开始,并以二叉树(binary tree) 式结构逐渐向后续工序传递,其中根节点代表惟一的直接受扰工序,其余节点表示它的后续工序; 每个节点最多引出左、右两个分支节点(假如存在的话) : 左分支引出与当前工序同在一条加工路线上的下一道工序; 右分支引出与当前工序同在一台机器上的下一道工序. 并非所有被引出的工序都会受到干扰的影响,至于是否受到干扰影响,还要视其原定的开工时间而定.

根据以上分析及表 2 中的符号表示,本小节设计出一种既有效应对开放式车间中的 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰又仅修复受影响工序的重调度方法——sAOR; 其详细步骤可描述如下:

输入原调度 $\boldsymbol{S}_O$,MUAD 干扰的属性值(即 t_D、DT、DM) ,各受扰工序的扰后工时②.

步骤 0　确定出重调度工序集 $\boldsymbol{O}'$,并对以下集合、变量进行初始化: $\boldsymbol{PAO} \leftarrow \emptyset$; $\boldsymbol{AO} \leftarrow \emptyset$; $\boldsymbol{ST}_{AO} \leftarrow \emptyset$; $\boldsymbol{ET}_{AO} \leftarrow \emptyset$; $a \leftarrow 0$; $JST(O_{i,j}) \leftarrow 0$, $MST(O_{i,j}) \leftarrow 0$, $\forall O_{i,j} \in \boldsymbol{O}'$; $\boldsymbol{S}_N \leftarrow \emptyset$.

步骤 1　置 $MST(O_{DM,j^*}) \leftarrow t_D + DT$; 将工序 O_{DM,j^*} 作为当前工序 CO (即 $CO \leftarrow O_{DM,j^*}$) , $\boldsymbol{PAO} \leftarrow \boldsymbol{PAO} \cup \{O_{DM,j^*}\}$; $PT \leftarrow p'_{DM,j^*}$.

步骤 2　更新当前工序 CO 的起止时间: $ST_N = \max\{MST(CO), JST(CO)\}$; $ET_N = ST_N + PT$.

步骤 3　如果 $CO \notin \boldsymbol{AO}$,则 $\boldsymbol{AO} \leftarrow \boldsymbol{AO} \cup \{CO\}$, $\boldsymbol{ST}_{AO} \leftarrow \boldsymbol{ST}_{AO} \cup \{ST_N\}$, $\boldsymbol{ET}_{AO} \leftarrow \boldsymbol{ET}_{AO} \cup \{ET_N\}$, $a \leftarrow a + 1$; 否则,确定出当前工序 CO 在集合 $\boldsymbol{AO}$ 中的编号 a' (即 $\boldsymbol{AO}[a'] = CO$) ,并对其开始、结束时间进行如下更新: $\boldsymbol{ST}_{AO}[a] \leftarrow \max\{ST_N, \boldsymbol{ST}_{AO}[a']\}$; $\boldsymbol{ET}_{AO}[a] \leftarrow \boldsymbol{ST}_{AO}[a] + PT$.

步骤 4　确定工序 NO_J. 若 NO_J 存在且 $ST(NO_J, \boldsymbol{S}_O) < ET_N$,则 $\boldsymbol{PAO} \leftarrow \boldsymbol{PAO} \cup \{NO_J\}$; $JST(NO_J) \leftarrow ET_N$.

步骤 5　确定工序 NO_M. 若 NO_M 存在且 $ST(NO_M, \boldsymbol{S}_O) < ET_N$,则 $\boldsymbol{PAO} \leftarrow \boldsymbol{PAO} \cup \{NO_M\}$; $MST(NO_M) \leftarrow ET_N$.

---

② 假设直接受扰工序 O_{DM,j^*} 的工时缩减量为 Δp,则 O_{DM,j^*} 的扰后工时 $p'_{DM,j^*} = p_{DM,j^*} - I \cdot \max\{t_D - t_{DM,j^*}(\boldsymbol{S}_O), 0\} - \Delta p$.

表 2 sAOR 方法中用到的符号及其说明

Table 2 Notations used in the sAOR and their definitions

符号	说明
CO	当前正在处理的工序
PT	当前工序的加工时间
NO_J	原调度中跟工序 CO 同在一条加工路线的下一道工序
NO_M	原调度中跟工序 CO 同在一台机器的下一道工序
$JST(O_{i,j})$	由所在加工路线上的直接前驱工序所决定的工序 $O_{i,j}$ 的最早开始时间
$MST(O_{i,j})$	由所在机器上的直接前驱工序所决定的工序 $O_{i,j}$ 的最早开始时间
$\boldsymbol{PAO}$	潜在的受影响工序集
$\boldsymbol{AO}$, $\boldsymbol{ST}_{AO}$, $\boldsymbol{ET}_{AO}$	保存受影响工序及其更新后开始、结束时间的集合
a	对于集合 $\boldsymbol{AO}$、$\boldsymbol{ST}_{AO}$、$\boldsymbol{ET}_{AO}$ 中元素的编号
$\Phi[a]$	集合 Φ 中编号为 a 的元素，Φ 可为 $\boldsymbol{AO}$、$\boldsymbol{ST}_{AO}$、$\boldsymbol{ET}_{AO}$
$ST(O_{i,j}, \boldsymbol{S}_O)$	工序 $O_{i,j}$ 在原调度 $\boldsymbol{S}_O$ 中的开始时间
ST_N, ET_N	当前工序 CO 更新后的开始时间、结束时间

步骤 6 将工序 CO 从工序集 $\boldsymbol{PAO}$ 中删除. 如果 $\boldsymbol{PAO} \neq \emptyset$，则从集合 $\boldsymbol{PAO}$ 中任意选取一道工序作为当前工序 CO，更新当前工序的加工时间 PT，转回步骤 2.

步骤 7 对于集合 $\boldsymbol{O}'$ 中每道工序，如果存在于集合 $\boldsymbol{AO}$ 中，则将该工序及其更新后的时间表计入 $\boldsymbol{S}_N$ 中；否则，将该未受扰工序及其原起止时间计入 $\boldsymbol{S}_N$.

步骤 8 输出最终得到的完整调度 $\boldsymbol{S}_N$，结束.

由 sAOR 方法流程不难发现，集合 $\boldsymbol{O}'$ 中的某些工序可能先后两次被计入集合 $\boldsymbol{PAO}$ 中，那么步骤 2 – 6 至多重复执行 $2R$ 次. 在每次迭代中，由于需要扫描当前集合 $\boldsymbol{AO}$（$\boldsymbol{AO} \subseteq \boldsymbol{O}'$），步骤 3 至多耗费 $O(R)$ 时间；步骤 4、5 分别耗费 $O(m)$、$O(n)$ 时间；步骤 2、6 均耗费常数时间. 步骤 7 至多耗费 $O(R^2)$ 时间. 由于 $R \leqslant m \times n$，则 sAOR 方法的时间复杂度至多为 $O(m^2 \times n^2)$.

3.3 基于原调度生成的全局重调度方法

全局重调度策略的实现往往需要开发专门的重调度优化算法，所涉及的重调度目标依赖于实际生产工况及调度者的偏好. 沿用原调度生成方法来重新生成新调度方案是全局重调度策略的代表性实施方式之一，并且在生产重调度实践中也屡见不鲜. 本质上，基于原调度生成的全局重调度方法（total rescheduling based on generation of original schedule，TR_ GOS）是一种仅追求调度性能优越的单目标重调度算法，它比相应的多目标优化算法更节省计算开销，且实时响应性更强. 不过，由于在优化过程中没有顾及重调度的稳定性，由 TR_GOS 所得的新调度无论在工序间相对次序还是在加工时间表上，相比于原调度方案都会出现明显变化. 尽管 TR_GOS 中不顾稳定性而一味优化调度性能的做法缺乏科学性，但其实用性与学术参照价值不容低估；若干生产重调度研究文献已对 TR_GOS 方法进行了专门的考察与分析[2,9,22].

Gueret 和 Prins[30] 曾基于优先级向量比较规则，提出了两种不同版本的串列调度启发式（listing scheduling）：dynamic version of H1（D_H1）；static repetitive version of H1（SR_H1）. 其中，SR_H1 在优先级不断更新过程中将每次生成的串列调度解统一解译为主动式调度（active schedules），且充分的计算结果表明它在解质量上有着明显优于 D_H1 算法的表现. 然而，Naderi 等[31] 通过对比实验发现：将串列调度方案解译为无延迟调度（nondelay schedule）在问题 $O_m \mid\mid C_{\max}$ 的求解上有着比解译为主动式调度显著更优的优化效果. 事实上，由串列调度解解译出的无延迟调度集虽无法保证含有最优解，但其规模通常远小于主动式调度集；进而在一定程度上缩小了待搜索的解空间，有利于尽快搜索到更高质量的调度解. 由

此,可在算法 SR_H1 中引入两项改进措施,从而设计出基于 SR_H1 的变体启发式 V_SR_H1 (variant SR_H1) ,即: 1) 每当评价当前的串列调度解时,都将其解译为无延迟调度; 2) 对于解更新次数 $main\_step$,由原先的 50 次增加为依赖于待排工序数 N 的 $\lceil 50 \times \sqrt{N} \rceil$ 次.

继而,以 V_SR_H1 算法为基础可形成有效应对开放式车间中 $\mathrm{MUAD_{CPT}}$ 干扰的 TR_GOS 方法,记其为 sTR_GOS. 算法 sTR_GOS 的详细流程如下所述:

输入各工序的 0 时刻工时$\{p_{i,j}, O_{i,j} \in \boldsymbol{O}\}$; 原调度方案 $\boldsymbol{S}_O$; 有关 MUAD 干扰的信息(即 t_D、DT、DM) ; 工序集 $\boldsymbol{O}'$; 各受扰工序的扰后工时$\{p'_{i,j}, O_{i,j} \in \boldsymbol{O}'\}$.

步骤 0　根据原调度 $\boldsymbol{S}_O$ 和 MUAD 干扰发生时刻 t_D 确定出各机器的初始可用时间; 由下式计算出扰后调度方案的完工期下界 LB',其中 $LB = \max\{\max\limits_{M_i \in \boldsymbol{M}}\{\sum\limits_{j=1}^{n} p_{i,j}\}, \max\limits_{J_j \in \boldsymbol{J}}\{\sum\limits_{i=1}^{m} p_{i,j}\}\}$.

$$LB' = \max\left\{\begin{array}{l} LB, \\ \sum\limits_{O_{DM_{\boldsymbol{jj}}} \in \boldsymbol{O}' \setminus \{O_{DM_{\boldsymbol{jj}^*}}\}} p'_{DM_{\boldsymbol{jj}}} + \sum\limits_{O_{DM_{\boldsymbol{jj}}} \in \boldsymbol{O} \setminus \boldsymbol{O}'} p_{DM_{\boldsymbol{jj}}} + Y_{\min} + p'_{DM_{\boldsymbol{jj}^*}} + I \cdot \max\{t_D - t_{DM_{\boldsymbol{jj}^*}}(\boldsymbol{S}_O), 0\} \\ \sum\limits_{O_{i_{\boldsymbol{jj}^*}} \in \boldsymbol{O}' \setminus \{O_{DM_{\boldsymbol{jj}^*}}\}} p'_{i_{\boldsymbol{jj}^*}} + \sum\limits_{O_{i_{\boldsymbol{jj}^*}} \in \boldsymbol{O} \setminus \boldsymbol{O}'} p_{i_{\boldsymbol{jj}^*}} + Y_{\min} + p'_{DM_{\boldsymbol{jj}^*}} + I \cdot \max\{t_D - t_{DM_{\boldsymbol{jj}^*}}(\boldsymbol{S}_O), 0\} \end{array}\right\} \tag{6}$$

步骤 1　对于每个工序 $O_{i,j} \in \boldsymbol{O}'$,计算相应的优先级向量 $\boldsymbol{V}_{i,j}$. $\boldsymbol{V}_{i,j}$ 中所含的两个分量分别是机器 M_i 上的残余加工时间和工件 J_j 的残余加工时间; 两个分量按非增次序排列.

步骤 2　对集合 $\boldsymbol{O}'$ 中的所有工序按照对应优先级向量的非增次序③进行排列,进而得到初始串列 $\boldsymbol{L}_0$. 结合扰后各机器的初始可用时间,将 $\boldsymbol{L}_0$ 解译为对应的无延迟调度,记作 $\boldsymbol{V}_0$.

步骤 3　置 $\boldsymbol{S}_N \leftarrow \boldsymbol{V}_0$,并初始化变量 $t \leftarrow 0$, $main\_step \leftarrow \lceil 50 \times \sqrt{R} \rceil$.

步骤 4　在 $\boldsymbol{V}_t$ 中找出结束时间晚于 LB' 的所有工序,或者在被加工过程中至少一台机器处于空闲状态的工序,并将它们保存在工序集 $\boldsymbol{R}_t$ 中.

步骤 5　将集合 $\boldsymbol{R}_t$ 中的各工序在串列 $\boldsymbol{L}_t$ 中的位次均前移一位,从而得到新串列 $\boldsymbol{L}_{t+1}$.

步骤 6　根据扰后各台机器的初始可用时间将串列 $\boldsymbol{L}_{t+1}$ 解译为无延迟调度 $\boldsymbol{V}_{t+1}$. 如果 $\boldsymbol{C}_{\max}(\boldsymbol{V}_{t+1}) < \boldsymbol{C}_{\max}(\boldsymbol{S}_N)$,则置 $\boldsymbol{S}_N \leftarrow \boldsymbol{V}_{t+1}$.

步骤 7　置 $t \leftarrow t + 1$. 如果 $t < main\_step$,则转回步骤 4; 否则,输出最终的调度 $\boldsymbol{S}_N$,结束.

4　仿真实验与分析

以 Taillard 系列 Benchmarks [33] 为测试算例集(含 4 × 4、5 × 5、7 × 7、10 × 10、15 × 15 和 20 × 20 六组问题规模,每组规模各含 10 个算例) ,对开放式车间中的 $\mathrm{MUAD_{CPT}}$ 干扰条件进行了仿真,并编程实现了三种原调度生成启发式(即 LPT 规则、SR_H1、V_SR_H1) 及三种重调度方法(即 sRSR、sAOR、sTR_GOS) ,继而在大量仿真实验的基础上,全面统计、对比了各重调度方法的绩效,探究了有关原调度生成机制的选取问题,并分析了干扰情景、加工特征等因素及其交互作用对重调度方法的取舍所带来的影响.

4.1　有关原调度生成机制的预实验

本研究中的原调度方案 $\boldsymbol{S}_O$ 需通过求解问题 $\boldsymbol{O}_m \mid\mid C_{\max}$ 而得到. 然而,当机器台数达到或超过 3 时,问题 $O_m \mid\mid C_{\max}$ 已被证实具备强 NP 困难性 [25] . 于是,可在合理计算时间内得到高质量近似解的启发式算法已成为问题 $O_m \mid\mid C_{\max}$ ($m \geqslant 3$) 的主要求解手段. 求解困难组合优化问题的启发式算法区分为两大类,元启发式(metaheuristics) ; 构造型启发式 (constructive heuristics) . 在问题 $O_m \mid\mid C_{\max}$ 的算法研究中,元启发式算法充分显示出全局搜索能力强、收敛速度快的优势 [26-28] ; 不过,它们在设计简单性、运行时间及计算鲁棒性上往往逊色于相关的构造型启发式 [29-31] . 构造型启发式利用待解问题的内在性质或特征从无到有地构造一个近似

③　给定两个优先级向量 $\boldsymbol{V}_1 = (a_1, b_1)$ 和 $\boldsymbol{V}_2 = (a_2, b_2)$,其中 $a_1 \geqslant b_1$, $a_2 \geqslant b_2$,规定 $\boldsymbol{V}_1 \geqslant \boldsymbol{V}_2$,当且仅当 $a_1 > a_2$ 或者($a_1 = a_2$ 且 $b_1 \geqslant b_2$) . 当 $a_1 = a_2$ 且 $b_1 = b_2$ 时,加工时间较长的工序拥有更高的优先级.

解;所得解质量或许不如元启发式算法,而简单易行、耗时短、可重复则是它的优势.

本小节旨在考察 LPT 规则、SR_H1[30] 以及 V_SR_H1(见 3.3 节)三种构造型启发式所生成的不同原调度,在面对 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰时,各自在不同响应机制下的重调度表现;通过对比三者的原调度结果及基于不同响应机制的重调度结果,确定出在正式重调度实验研究中所用的原调度生成方法. 对于原调度生成启发式(LPT、SR_H1 和 V_SR_H1)以及重调度方法(sRSR、sAOR 和 sTR_GOS)均采用 Java 语言编程实现;并在处理器为 Intel (R) Core(TM) 2.80GHz,内存为 2.0GB 的 PC 机上进行计算与仿真实验.

在针对原调度结果的对比实验中,LPT、SR_H1 和 V_SR_H1 三种启发式分别对 Taillard 系列的 OSSP Benchmarks 中的 60 个算例逐一进行求解;对于每个算例,各启发式的完工期结果与运行时间分别被记录下来. 表 3 给出了启发式 LPT、SR_H1 及 V_SR_H1 在解质量指标 Gap_LB 和运行时间 T 上的统计结果. 其中,有关指标 Gap_LB 的计算表达式为

$$Gap\_LB = \frac{Makespan_{CH} - LB}{LB} \times 100 \qquad (7)$$

式中 $Makespan_{CH}$ 表示启发式 $CH \in \{\mathrm{LPT}, \mathrm{SR\_H1}, \mathrm{V\_SR\_H1}\}$ 求解问题 $O_m \| C_{\max}$ 所得的完工期值,LB 则表示相应算例的 $C_{\max}$ 下界值,详见文献[33];指标 T 的统计单位为(s);表中各单元格为相应启发式求解对应规模中 10 个算例的指标均值. 从表 3 可直观看出:启发式 V_SR_H1 在总体解质量上的比较优势明显,在较大规模 5 组算例上的解质量优于 LPT 与 SR_H1;尽管 V_SR_H1 在计算时间上明显逊色于另两种启发式,但对于 $m \times n \leqslant 100$ 的算例,其耗时均值保持在 0.3 s 以内;即便对于 20×20 规模的算例,V_SR_H1 的平均耗时也在 10 s 以内.

表 3 三种构造型启发式在指标 *Gap_LB* 和 *T* 上的统计结果

Table 3 Statistical results of three constructive heuristics on *Gap_LB* and *T*

算例	LPT		SR_H1		V_SR_H1	
	Gap_LB	*T*	*Gap_LB*	*T*	*Gap_LB*	*T*
Tail_4 × 4	13.20	**0.000**	**2.85**	0.002	4.66	0.005
Tail_5 × 5	16.85	**0.000**	5.28	0.003	**4.27**	0.009
Tail_7 × 7	10.98	**0.000**	3.95	0.009	**2.09**	0.042
Tail_10 × 10	7.06	**0.008**	2.26	0.027	**1.19**	0.245
Tail_15 × 15	4.03	**0.053**	0.84	0.139	**0.34**	2.050
Tail_20 × 20	1.99	**0.217**	0.56	0.486	**0.15**	9.675
均值	9.02	**0.046**	2.62	0.111	**2.12**	2.004

注: 每行中在指标 Gap_LB、T 上表现最优者均以粗体标明;指标 T 的单位:s.

在基于不同策略重调度结果的对比实验中,对于每个 Taillard 系列的基准算例,三种启发式 LPT、SR_H1 与 V_SR_H1 分别生成各自的原调度方案 S_O^{CH},取 $CH \in \{\mathrm{LPT}, \mathrm{SR\_H1}, \mathrm{V\_SR\_H1}\}$;三种原调度各自针对 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰仿真生成器所随机产生的同一干扰,先后采取 sRSR、sAOR 与 sTR_GOS 三种方法予以应对,进而分别得到新调度 S_N^{RI},其中 $RI \in \{\mathrm{sRSR}, \mathrm{sAOR}, \mathrm{sTR\_GOS}\}$;对于方法 sRSR 或 sAOR 生成的新调度,统计它的完工期值($C_{\max}$)与结束时间偏差量(TD),而对于方法 sTR_GOS 生成的新调度,除了统计 $C_{\max}$ 与 TD 之外,还记录下新调度的序位偏差量(SD). 为保证实验统计结果的可靠性,对于每个算例随机产生 30 个独立的 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰事件. 对于 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰的模拟生成,各干扰属性按如下方式设定:t_D 服从 $[0.1 \times LB\rfloor, LB]$ 离散均匀分布;DM 服从 $[1, m]$ 离散均匀分布;DT 服从 $[0.01 \times LB\rfloor, \lceil 0.1 \times LB\rceil]$ 离散均匀分布;总工时缩减量服从 $[0.2 \times Y_{\min}\rfloor, \lceil 0.7 \times Y_{\min}\rceil]$ 离散均匀分布;$I \leftarrow rand(0,1)$.

为便于统计,按照由问题规模和干扰编号($k = 1, 2, \cdots, 30$)所构成的二元组将整个实验

划分为 $6 \times 30 = 180$ 个单元,对单元内计算出的各重调度指标值 $IND \in \{C_{\max}, TD, SD\}$ 根据原调度生成启发式(CH) 与重调度方法(RI) 的不同进行分类累加,进而得到相应的指标累加值 SUM $[IND(S_O^{CH}, S_N^{RI})]$. 在每个实验单元内针对特定的指标 IND 与重调度方法 RI,利用下式计算各指标累加值的相对比率(relative ratio).

$$RR_{IND}^{RI} = \frac{\text{SUM}\ [IND(S_O^{CH_h}, S_N^{RI})]}{\min\{\text{SUM}\ [IND(S_O^{CH_h}, S_N^{RI})], h = 1,2,3\}} \quad (8)$$

表 4 给出了三种原调度方法在各比率 RR_{IND}^{RI} 上对所有实验单元的总体均值结果.

表 4 表明,方法 V_SR_H1 在全部 7 项比率指标 RR_{IND}^{RI} 的均值比较中都优于另两种启发式,尤以指标 $RR_{TD}^{\text{sTR\_GOS}}$ 和 $RR_{SD}^{\text{sTR\_GOS}}$ 上的均值对比优势显著. 综合表 3 和 4 中的结果发现: V_SR_H1 方法通过牺牲可接受的计算代价(时间) 换来了更高质量的原调度方案; 并在利用不同策略实施重调度后,其产生的原调度对应于相对更优的扰后完工期值及时间表、序位偏差量. 更佳的基于不同策略的重调度结果有利于降低生产成本、提高生产效率,进而增加赢利. 为此,本文选取启发式 V_SR_H1 作为正式重调度方法比较研究中的原调度方案生成算法.

表 4　三种构造型启发式在比率 RR_{IND}^{RI} 上的总体均值对比结果

Table 4 The comparison of means for three constructive heuristics on RR_{IND}^{RI}

原调度方法	sRSR		sAOR		sTR_GOS		
	$RR_{C_{\max}}^{\text{sRSR}}$	RR_{TD}^{sRSR}	$RR_{C_{\max}}^{\text{sAOR}}$	RR_{TD}^{sAOR}	$RR_{C_{\max}}^{\text{sTR\_GOS}}$	$RR_{TD}^{\text{sTR\_GOS}}$	$RR_{SD}^{\text{sTR\_GOS}}$
LPT	1.111	1.518	1.098	1.442	1.108	1.605	2.717
SR_H1	1.052	1.115	1.051	1.190	1.030	1.810	3.429
V_SR_H1	**1.000**	**1.056**	**1.000**	**1.189**	**1.003**	**1.106**	**1.600**

注: 每列中的表现最优单元格均以粗体标明.

4.2　仿真实验设计与描述

在正式的重调度仿真实验中,仍选用 Taillard 系列 OSSP Benchmarks[33] 作为实验算例集,共计 60 个计算实例; 以本文提出的 V_SR_H1 启发式作为原调度方案生成算法; 采用 Java 语言编程实现了重调度方法 sRSR、sAOR、sTR_GOS 以及原调度生成启发式 V_SR_H1; 并在处理器主频为 2.80GHz、内存为 2GB 的 PC 机上进行仿真实验. 此外,实验中还考虑了以下六项相互独立、潜在影响重调度表现的因素:

1) 重调度实现方法(rescheduling implementation, **RI**): 它的三个层次对应于文中提出的三种重调度方法,即 sRSR、sAOR 和 sTR_GOS.

2) MUAD 干扰发生时间(occurrence time, **OT**): 分为早期(E) 、中期(M) 和晚期(L) 三层次,其划分依据为干扰发生时间 t_D 跟原调度完工期 $L_O = C_{\max}(S_O)$ 的相对大小关系.

3) MUAD 干扰持续时长(lasting duration, **LD**): 分为短期(S) 、中期(M) 和长期(L) ,由时长 DT 跟原调度完工期 L_O 的相对大小关系而确定.

4) 缩减率(**RATE**): 工时缩减幅度会受多方面的限制,在此区分为高(H) 、低(L) 两层次.

5) 算例规模(**SIZE**): 包含小规模(S) 和大规模(B) 两层次,根据原调度中的工序数来划分.

6) 中断模式(interrupt mode, **IM**): 分为中断 — 不可续、中断 — 可续两层次.

参考文献 [2] 中的做法,表 5 给出了 6 项影响因素的各自层次设置与划分情况. 从可控性上看,仅因素 **RI** 为可控因素(controllable factor) ,而其他五项因素均为客观的非可控因素. 其中,**OT**、**LD**、**RATE** 联合刻画出 MUAD_{CPT} 干扰; **OT**、**LD**、**RATE**、**SIZE** 及 **IM** 共同决定所面临的重调度情景,故称为重调度情景因素. 为便于表述,下文以五元组(**OT**, **LD**, **RATE**, **SIZE**, **IM**) 表示所遇到的重调度情景(族) ,如情景族(*, S, *, *, 1) 表示 MUAD 干扰持续较短时间,且被中断的工序将以“中断 — 可续”的方式恢复中断.

表 5 各重调度影响因素的层次设置与划分

Table 5 The levels setting for each of the six factors

层次	重调度方法 RI	干扰发生时间 OT	干扰持续时长 LD	缩减率* RATE	算例规模 SIZE	中断模式 IM
层次 1	sRSR	E ($\lfloor 0.05L_O \rfloor \leqslant t_D \leqslant \lceil 0.3L_O \rceil$)	S ($\lfloor 0.005L_O \rfloor \leqslant DT \leqslant \lceil 0.035L_O \rceil$)	H ($\alpha = 0.5$)	S ($m \times n \leqslant 50$)	$I = 0$ (interrupt-repeat)
层次 2	sAOR	M ($\lfloor 0.35L_O \rfloor \leqslant t_D \leqslant \lceil 0.6L_O \rceil$)	M ($\lfloor 0.04L_O \rfloor \leqslant DT \leqslant \lceil 0.07L_O \rceil$)	L ($\alpha = 0.2$)	B ($m \times n \geqslant 100$)	$I = 1$ (interrupt-resume)
层次 3	sTR_GOS	L ($\lfloor 0.65L_O \rfloor \leqslant t_D \leqslant \lceil 0.9L_O \rceil$)	L ($\lfloor 0.075L_O \rfloor \leqslant DT \leqslant \lceil 0.105L_O \rceil$)	—	—	—

注: * 为贴近生产实际、统一 CPT 干扰的模拟实现,实验中对两类工序实施工时缩减: 工序 O_{DM,j^*} 及其所在机器或加工路线上的直接后继工序(假如存在)中距 $e_{DM,j^*}(S_O)$ 时刻较近者,记其为 $O_{(DM),(j^*)}$. 由缩减率 $\alpha \in [0, 0.5]$,则 O_{DM,j^*} 和 $O_{(DM),(j^*)}$ 的扰后工时各为 $p'_{DM,j^*} = p_{DM,j^*} - \min\{\alpha \cdot Y_{\min}, p_{DM,j^*} - I \cdot \max\{t_D - t_{DM,j^*}(S_O), 0\}\}$; $p'_{(DM),(j^*)} = p_{(DM),(j^*)} - \min\{\alpha \cdot [Y_{\min} - (t_{(DM),(j^*)}(S_O) - e_{DM,j^*}(S_O))], p_{(DM),(j^*)}\}$.

实验中还提出了四项响应指标(response variables),分别用于观测重调度结果在完工期 $C_{\max}$、序位偏差 SD、时间表偏差 TD 及重调度耗时 T 四方面上的表现. 其中,除了以秒(s)为单位对重调度耗时(T)进行直接测量之外,为了尽量降低由于算例特征所致的绩效差异性,其余三项响应指标均在原始观测值的基础上进行必要的统计处理.

1) 完工期比率(ratio between makespans, RM)

$$RM(S_O, S_N) = \frac{C_{\max}(S_N)}{C_{\max}(S_O)} \tag{9}$$

2) 标准化时间表偏差的平方根(square root of normalized timetable deviation, SRNTD)

$$SRNTD(S_O, S_N) = \sqrt{\frac{\sum_{O_{i,j} \in O'} | e'_{i,j} - e_{i,j}(S_O) |}{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{i,j}}} \tag{10}$$

3) 标准化序位偏差(normalized sequence deviation, NSD)

$$NSD(S_O, S_N) = \frac{\sum_{O_{i,j} \in O'} Q_{i,j}(S_O, S_N)}{m \times n} \tag{11}$$

综上,本实验针对随机模拟的大量重调度情景,在由 V_SR_H1 启发式生成的原调度基础上,全面考察 sRSR、sAOR 和 sTR_GOS 三种重调度方法在相同情景下、四种响应指标上的各自绩效. 整个实验按照完全析因实验法(full-factorial experiments)进行设计与实施. 五项情景因素中,RATE、SIZE 和 IM 各包含两个层次,OT 和 LD 均区分为三个层次,合计共 $3^2 \times 2^3 = 72$ 种情景组合; 每种情景组合中,SIZE 的每个层次对应于 Taillard 算例集中的 3 种问题规模且每一规模包含 10 个算例,即每个情景组合将在 30 个算例上进行重复实验. 由此,整个实验由 $3^2 \times 2^3 \times 30 = 2\,160$ 个实验单元构成; 每个实验单元中,对三种方法实施同情景独立观测. 实验完成后,将来自 2 160 个实验单元的样本数据汇总起来,用于进一步的统计与分析.

4.3 实验结果与分析

4.3.1 对于指标 RM 的统计结果与分析

首先关注 sRSR、sAOR 和 sTR_GOS 三种方法在响应指标 RM 上的表现. 图 2(a - b)给出了它们在 72 种重调度情景下对于指标 RM 的均值与标准差统计结果. 对于指标 RM 的均值结果,方法 sTR_GOS 表现最优、方法 sAOR 次之、方法 sRSR 相对较差; 特别当处于情景(E, L, *, B, *)和(M, L, *, B, *)时,方法 sTR_GOS 的比较优势尤为显著. 通过比较两图中各情景下 RM 的标准差统计结果可发现,三种方法在小规模算例上

的 *RM* 指标值显示出了更高的数据离散程度，数　　据波动性更强.

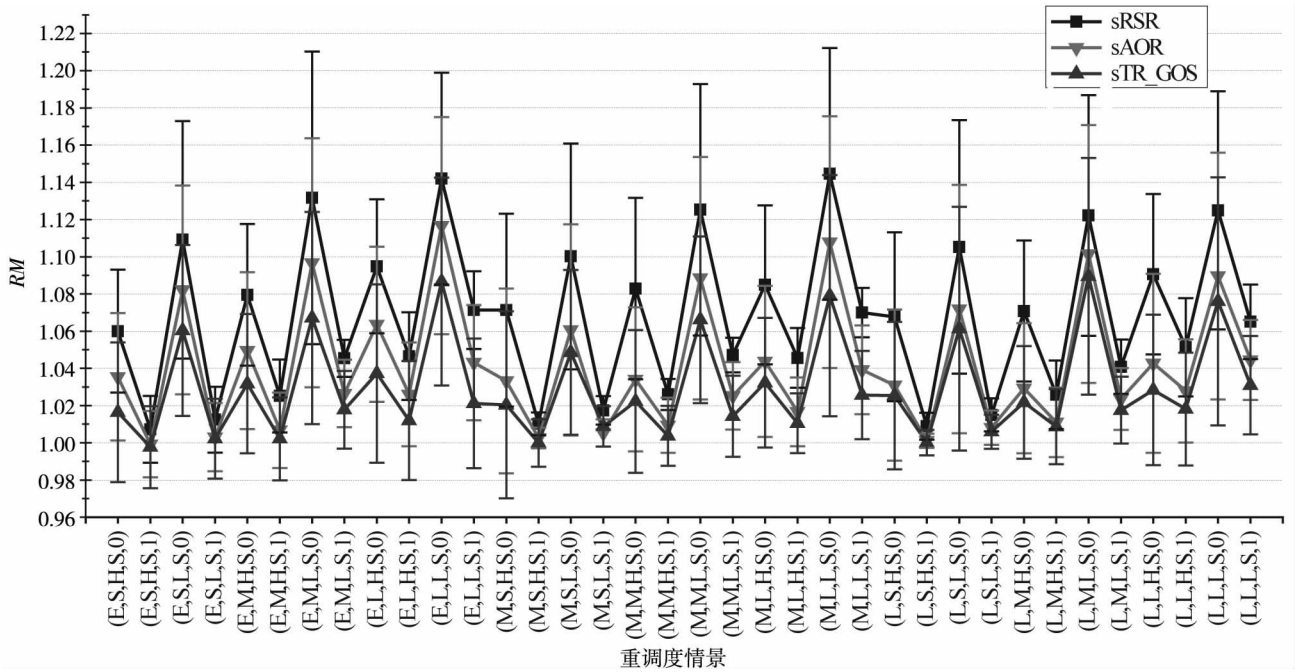

图 2(a)　情景(*，*，*，S，*)下三种重调度方法在指标 *RM* 上的均值及其标准差

Fig. 2(a) Means and standard deviations of three rescheduling methods on *RM* under the (*, *, *, S, *) scenarios

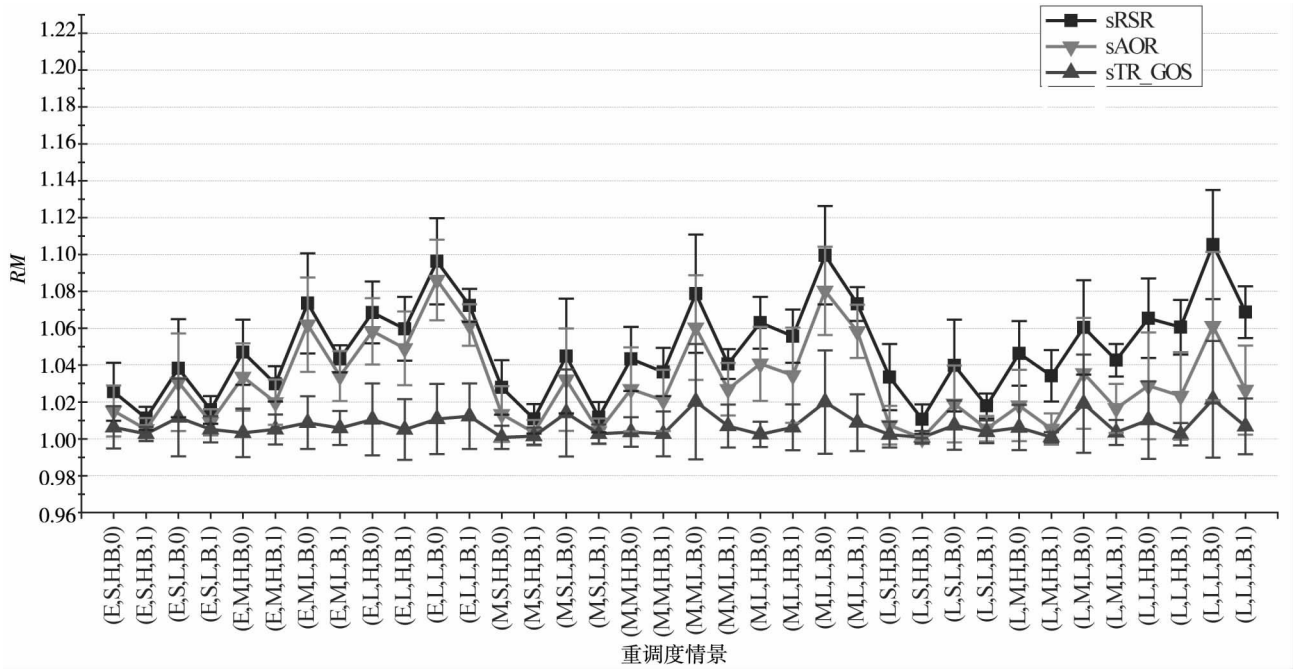

图 2(b)　情景(*，*，*，B，*)下三种重调度方法在指标 *RM* 上的均值及其标准差

Fig. 2(b) Means and standard deviations of three rescheduling methods on *RM* under the (*, *, *, B, *) scenarios

为了进一步考察三种方法在指标 *RM* 上的表现是否存在显著差异，须在各重调度情景下分别实施不同的方法，进而对收集起来的 *RM* 样本数据实施显著性统计推断. 在此选用 Friedman 检验法对收集的 3 - 相关样本所来自的总体分布进行非参数检验，详细的检验结果如表 6 所示. 表中结果表明，三种方法在指标 *RM* 上的表现差异具有统计学意义，即在 0.1% 的显著性水平下显著($p = 0.000 < 0.001$)；另鉴于三者在总体均值、中位数上的大小关系及在各重调度情景下的表现

情况(见图 2),可以得出结论: 方法 sTR_GOS 的 *RM* 绩效显著优于方法 sAOR 与 sRSR; 而方法 sAOR 在指标 *RM* 上跟方法 sRSR 相比则明显更好. 究其原因,归结于三者对调度性能指标的不同偏好程度: 方法 sTR_GOS 以性能优化为惟一目标,无定序约束,在优化搜索机理的指导下更可能得到调度性能优越的重调度方案; 而 sRSR 以定序、简单易行为重,在调度性能上难有好的表现.

仔细观察图 2 不难发现,在某些特定重调度情景下,方法 sAOR 与 sTR_GOS 在指标 *RM* 上的均值表现十分接近(相应均值点彼此临近或重合),进而有必要在相应情景下对两者的 *RM* 样本所来自的总体分布进行显著性检验. 为此,专门针对 sAOR 和 sTR_GOS 表现相接近的重调度情景执行了 *RM* 绩效的 Wilcoxon 符号秩检验,检验结果如表 7 所示,其中显著性水平为 0.05; h 值取 0 表示方法 sAOR 跟 sTR_GOS 相比表现无差异,取 1 代表 sTR_GOS 表现显著优于 sAOR. 表 7 中共计有 13 个 $h = 0$ 情景; 除了(L, M, H, S, 1) 之外,其余情景恰好组成情景族(*, S, *, *, 1). 可见,当 MUAD 干扰持续较短时间且采取"中断—可续"方式恢复被中断工序时,无统计结果显示方法 sAOR 的 *RM* 表现显著逊色于 sTR_GOS.

表 6 指标 *RM* 上 3 - 相关样本的 Friedman 检验统计结果

Table 6 Statistical results of Friedman test for 3-related samples on *RM*

3 - 相关样本					样本容量	χ^2 统计量	自由度	p 值
重调度方法	均值	中位数	标准差	秩均值				
sRSR	1.058	1.049	0.047	2.89	2 160	3 352.443	2	0.000***
sAOR	1.036	1.025	0.042	1.80				
sTR_GOS	1.018	1.003	0.036	1.31				

注: *** 表示在 0.1% 的显著性水平下显著.

表 7 特定情景下方法 sAOR、sTR_GOS 在指标 *RM* 上的 Wilcoxon 符号秩检验结果

Table 7 Results of Wilcoxon signed-ranks test for sAOR, sTR_GOS on *RM* under specific scenarios

重调度情景	sAOR - sTR_GOS			重调度情景	sAOR - sTR_GOS		
	Z 统计量	p 值	h 值		Z 统计量	p 值	h 值
(E, S, H, S, 1)	**-0.459[b]**	**0.646**	**0**	(L, M, H, S, 0)	-2.366[a]	0.018	1
(E, S, L, S, 1)	**-0.432[a]**	**0.666**	**0**	**(L, M, H, S, 1)**	**-1.820[a]**	**0.069**	**0**
(E, M, H, S, 1)	-2.411[a]	0.016	1	(L, M, L, S, 1)	-2.521[a]	0.012	1
(E, M, L, S, 1)	-2.556[a]	0.011	1	**(E, S, H, B, 1)**	**-1.741[a]**	**0.082**	**0**
(M, S, H, S, 1)	-1.079[a]	0.281	0	**(E, S, L, B, 1)**	-1.826[a]	0.068	0
(M, S, L, S, 1)	-1.852[b]	0.064	0	**(M, S, H, B, 1)**	-0.852[a]	0.394	0
(M, M, H, S, 1)	-2.401[a]	0.016	1	(M, S, L, B, **1**)	**-0.057**[a]	**0.954**	**0**
(M, L, H, S, 1)	-2.343[a]	0.019	1	(L, S, H, B, 0)	-3.040[a]	0.002	1
(L, S, H, S, 0)	-2.666[a]	0.008	1	**(L, S, H, B, 1)**	-0.561[b]	0.575	0
(L, S, H, S, 1)	-0.943[a]	0.345	0	**(L, S, L, B, 1)**	-1.500[a]	0.134	0
(L, S, L, S, 1)	-1.753[a]	0.080	0	(L, M, H, B, 1)	-2.490[a]	0.013	1

注: [a] 基于负秩; [b] 基于正秩.

4.3.2 对于指标 *SRNTD* 的统计结果与分析

类似地,图 3(a - b) 给出了三种方法对于指标 *SRNTD* 在 72 种重调度情景下的均值与标准差统计结果. 通过比较图中结果可发现: 总体上,方法 sAOR 显示出优于方法 sRSR 和 sTR_GOS 的显著优势,尤其对于干扰发生在早期或中期的情形; 当面对情景族(E, *, *, B, *) 和(M, *, *, B, *) 时,方法 sTR_GOS 在指标 *SRNTD* 上的比较劣势明显; 从标准差上看,三种方法在情景(L, *, *, *, *) 下的 *SRNTD* 样本数据有着比情景(E, *, *, *, *) 和(M, *, *, *, *) 更高的数据集中程度.

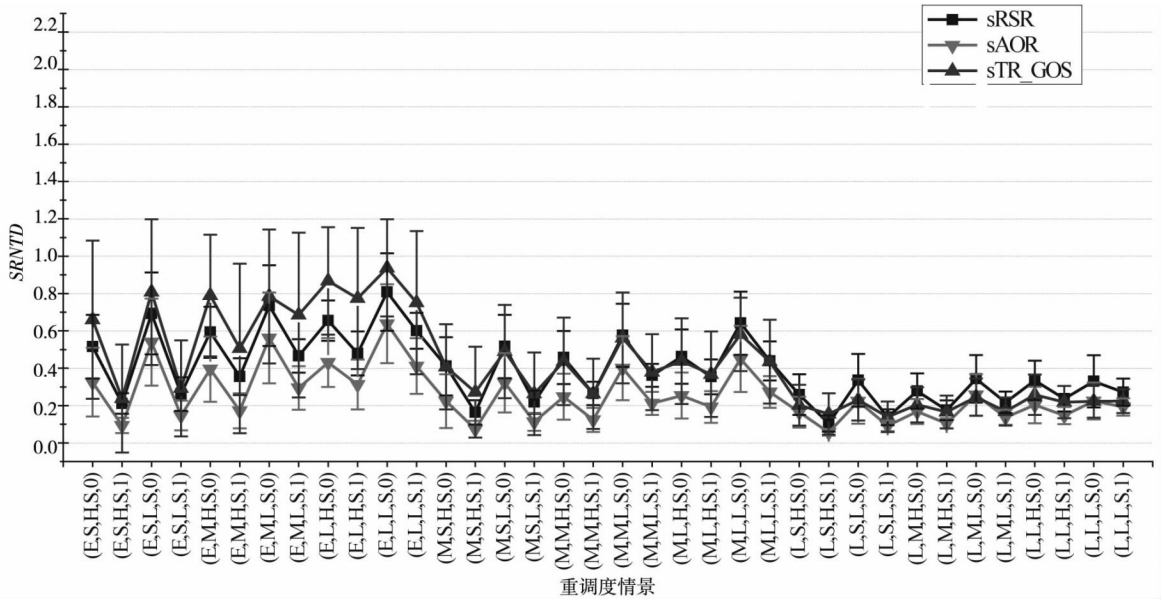

图 3(a)　情景(*,*,*,S,*)下三种重调度方法在指标 *SRNTD* 上的均值及其标准差

Fig. 3(a) Means and standard deviations of three rescheduling methods on *SRNTD* under the (*,*,*,S,*) scenarios

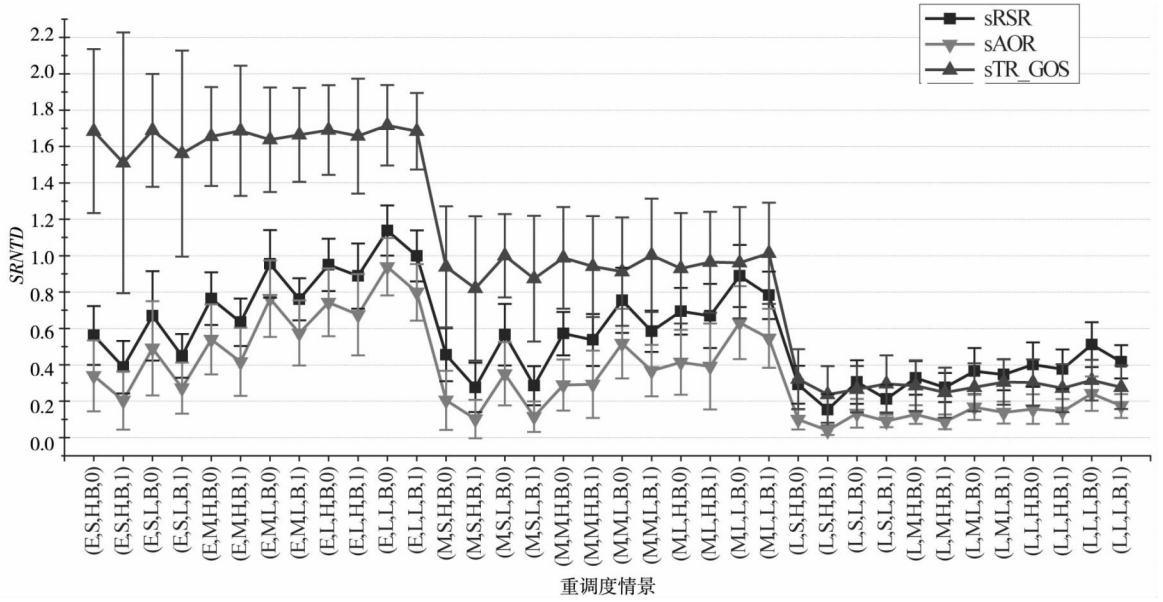

图 3(b)　情景(*,*,*,B,*)下三种重调度方法在指标 *SRNTD* 上的均值及其标准差

Fig. 3(b) Means and standard deviations of three rescheduling methods on *SRNTD* under the (*,*,*,B,*) scenarios

三种重调度方法在指标 *SRNTD* 上的表现存在显著差异,这一论断在 Friedman 检验结果中得以证实,如表 8 所示. 其中,$p = 0.000 < 0.001$,表明三者间的差异在 0.1% 的显著性水平下显著;具体地,方法 sAOR 的指标 *SRNTD* 表现显著优于 sRSR 和 sTR_GOS,而在后两者之间的比较中,sTR_GOS 明显逊色于 sRSR. 即便如此,进一步观察图 3(a) 可发现,在情景(L,*,*,S,*)下方法 sAOR、sTR_GOS 在指标 *SRNTD* 上的表现较为接近(误差棒上的均值点彼此重叠、难以辨清).

为此,在情景(L,*,*,S,*)下,专门对 sAOR 和 sTR_GOS 两种方法执行了 *SRNTD* 绩效的 Wilcoxon 符号秩检验,检验结果如表 9 所示. 其中,显著性水平为 0.05;$h = 0$ 表示方法 sAOR 跟 sTR_GOS 相比表现无差异,而 $h = 1$ 代表 sAOR 显著优于 sTR_GOS. 在全部 12 种重调度情景中,仅在(L,*,L,S,0)所代表的情景下无证据说明两种方法在 *SRNTD* 指标上的表现存在显著差异,而在其余 9 种重调度情景中相应 p 值都不超过 0.003,说明 sAOR 的 *SRNTD* 绩效显著优于 sTR_GOS.

表 8 指标 *SRNTD* 上 3 – 相关样本的 Friedman 检验统计结果

Table 8 Statistical results of Friedman test for 3-related samples on *SRNTD*

3 – 相关样本					样本容量	χ^2 统计量	自由度	p 值
重调度方法	均值	中位数	标准差	秩均值				
sRSR	0.479	0.424	0.264	2.41	2 160	2 578.294	2	0.000***
sAOR	0.302	0.220	0.242	1.14				
sTR_GOS	0.694	0.468	0.573	2.45				

注: *** 表示在 0.1% 的显著性水平下显著.

表 9 情景(L,*,*,S,*)下方法 sAOR、sTR_GOS 在指标 *SRNTD* 上的 Wilcoxon 符号秩检验结果

Table 9 Results of Wilcoxon signed-ranks test for sAOR, sTR_GOS on *SRNTD* under scenarios (L,*,*,S,*)

重调度情景	sAOR – sTR_GOS			重调度情景	sAOR – sTR_GOS		
	Z 统计量	p 值	h 值		Z 统计量	p 值	h 值
(L, S, H, S, 0)	– 3.107[b]	0.002	1	**(L, M, L, S, 0)**	**– 1.712[a]**	**0.087**	**0**
(L, S, H, S, 1)	– 3.621[b]	0.000	1	(L, M, L, S, 1)	– 2.981[b]	0.003	1
(L, S, L, S, 0)	**– 0.866[b]**	0.386	0	(L, L, H, S, 0)	– 3.101[b]	0.002	1
(L, S, L, S, 1)	– 2.934[b]	0.003	1	(L, L, H, S, 1)	– 3.980[b]	0.000	1
(L, M, H, S, 0)	– 3.045[b]	0.002	1	**(L, L, L, S, 0)**	**– 0.465[a]**	**0.642**	**0**
(L, M, H, S, 1)	– 3.296[b]	0.001	1	(L, L, L, S, 1)	– 3.076[b]	0.002	1

注: [a] 基于负秩; [b] 基于正秩.

4.3.3 对于指标 *NSD* 的统计结果与分析

由于方法 sRSR 和 sAOR 所生成的新调度没有造成任何序位变化,则这两种定序修复方法在指标 *NSD* 上均对应最佳值 0. 然而,方法 sTR_GOS 是一种调度再生成手段,所得的新调度难免会出现序位变化. 图 4 以柱状图的形式描绘了方法 sTR_GOS 在各重调度情景下对于指标 *NSD* 的均值统计结果. 从图中可明显看出,MUAD 干扰越早发生,方法 sTR_GOS 所造成的 *NSD* 指标值越大;算例规模变大,所对应的序位偏差量也随之增加. 当 MUAD 干扰发生在早期且算例规模大时,方法 sTR_GOS 在序位偏差稳定性上的劣势充分显现出来. 然而,当干扰发生在晚期,方法 sTR_GOS 在指标 *NSD* 上的表现有所好转,相应的 *NSD* 均值仅为 0.03,远小于干扰出现在早期时的对应均值 0.92.

4.3.4 对于指标 *T* 的统计结果与分析

实验中还记录下了各实验单元的耗时数据,并根据各情景因素的层次变化对耗时数据进行分类统计,统计结果如表 10 所示. 从表 10 可看出,方法 sRSR 具备最快的干扰响应速度,几乎不耗用计算时间;且在指标 *T* 上表现极为稳定,基本不受重调度情景变化的影响. 方法 sAOR 的干扰响应同样迅速,虽不及方法 sRSR,但无论遭遇何种重调度情景都会在 2 s 内完成重调度过程,并在情景族(L,*,*,*,*) 和(*,*,*,S,*) 下跟方法 sRSR 的 *T* 指标表现几近相同,均可在瞬间内完成调度修复. 另外,表 10 中的结果显示,方法 sAOR 对所遭遇的重调度情景表现得相当“敏感”,统计出来的标准差值往往为对应均值的 3 – 6 倍;究其原因:sAOR 仅对受影响的工序实施调度修复,遇到不同的重调度情景会使得受扰工序的数目发生变化,进而导致长短有别的重调度耗时长度.

正如所料,方法 sTR_GOS 的耗时明显多于另外两者;不仅耗时均值超过 1 s,且在极端情形下会花销近 10 s 的时间. 另外,它的耗时还受到了因素 OT 与 SIZE 层次变化的显著影响,表现为:MUAD 干扰出现得越早,算例规模越大,则 sTR_GOS 的重调度耗时越长.

4.3.5 特定情景下重调度结果的统计启示

通过对各指标结果进行全面、深入地分析,针对特定的重调度情景可发现以下有关重调度方法取舍的统计性发现与启示:

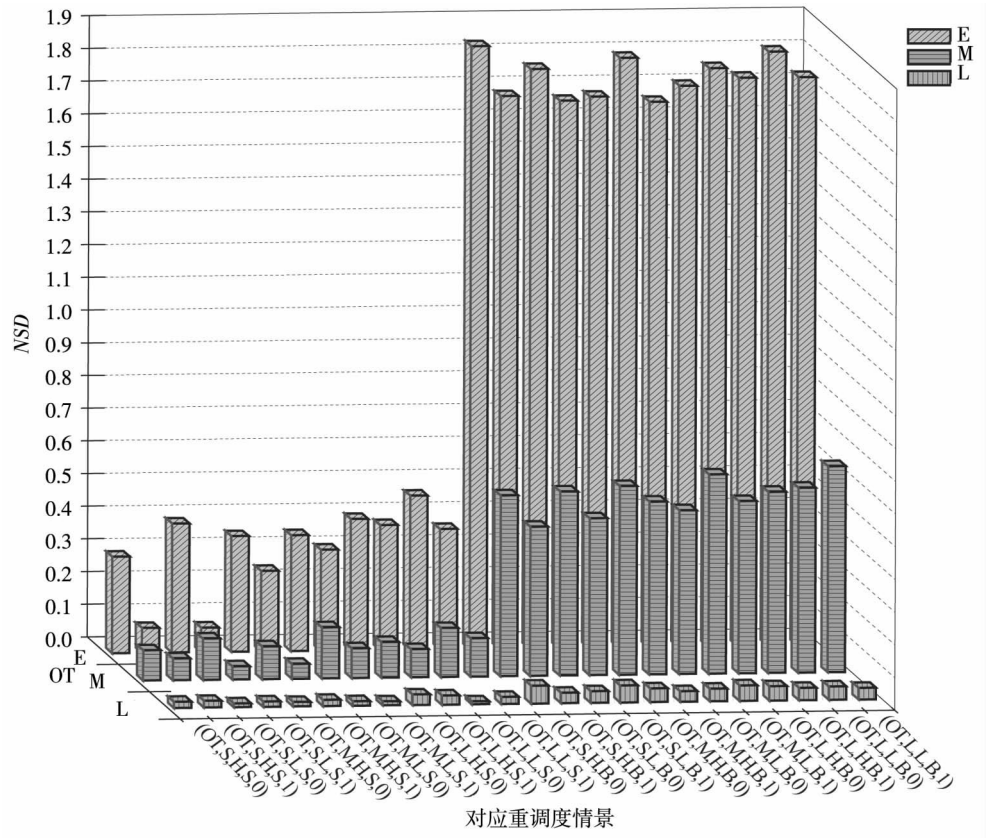

图 4　各情景下方法 sTR_GOS 在指标 *NSD* 上的均值统计结果

Fig. 4 The means on *NSD* of rescheduling methods sTR_GOS under various scenarios

表 10　三种重调度方法对指标 *T* 的统计结果 (单位: s)

Table 10 Statistical results on *T* for three rescheduling methods (unit: second)

重调度情景		sRSR		sAOR				sTR_GOS			
		均值	标准差	均值	标准差	最小值	最大值	均值	标准差	最小值	最大值
OT	(E,*,*,*,*)	0.000	0.000	0.131	0.371	0.000	1.969	1.512	2.637	0.000	9.407
	(M,*,*,*,*)	0.000	0.000	0.003	0.013	0.000	0.250	1.086	1.925	0.000	6.609
	(L,*,*,*,*)	0.000	0.000	0.000	0.000	0.000	0.000	0.570	1.034	0.000	3.719
LD	(*,S,*,*,*)	0.000	0.000	0.021	0.133	0.000	1.907	1.091	2.096	0.000	9.407
	(*,M,*,*,*)	0.000	0.000	0.044	0.216	0.000	1.884	1.039	1.975	0.000	8.500
	(*,L,*,*,*)	0.000	0.000	0.068	0.289	0.000	1.969	1.039	1.970	0.000	9.250
RATE	(*,*,H,*,*)	0.000	0.000	0.036	0.195	0.000	1.969	1.054	2.012	0.000	9.250
	(*,*,L,*,*)	0.000	0.000	0.053	0.247	0.000	1.938	1.058	2.016	0.000	9.407
SIZE	(*,*,*,S,*)	0.000	0.000	0.000	0.000	0.000	0.000	0.013	0.016	0.000	0.072
	(*,*,*,B,*)	0.000	0.000	0.089	0.309	0.000	1.969	2.100	2.436	0.031	9.407
IM	(*,*,*,*,0)	0.000	0.000	0.052	0.245	0.000	1.969	1.047	1.996	0.000	8.500
	(*,*,*,*,1)	0.000	0.000	0.037	0.197	0.000	1.953	1.065	2.032	0.000	9.407
总体	(*,*,*,*,*)	0.000	0.000	0.044	0.223	0.000	1.969	1.056	2.014	0.000	9.407

1) 当处于重调度情景族(*,S,*,*,1) 时,表 7 中的相关统计结果已显示,方法 sAOR 在指标 *RM* 上与 sTR_GOS 相比无显著差异; 且它与 sRSR 一样同属定序重调度方法,具备最佳的序位稳定性. 方法 sAOR 在指标 *SRNTD* 上的表现具有比较优势,如图 5(a) 所示; 尽管在情景 (L, S, L, S, 1) 下 sTR_GOS

与 sAOR 表现较为接近，但表 9 中的相关结果已证实两者的 *SRNTD* 指标绩效仍存在显著差异. 此外，方法 sAOR 的干扰响应速度也相当快: 经统计，该情景族下它的平均耗时仅为 0.014 s，远小于方法 sTR_GOS 的 1.103 s; 且接近于方法 sRSR 的耗时表现，如图 5(b) 所示. 综上可得: 当 MUAD 干扰持续较短时间且采取“中断 — 可续”方式恢复被中断工序时，极力建议采取方法 sAOR 来实施重调度.

2) 对于重调度情景族(L，*，L，S，0) ，如表 9 所示，无统计证据说明方法 sTR_GOS 在指标 *SRNTD* 上与 sAOR 相比存在显著差别. 然而，在该情景下 sTR_GOS 在指标 *RM* 上表现得相对较优; 经统计，它的 *RM* 指标均值为 1.076，优于另两者的相应绩效，详见表 11(a) . 尽管 sTR_GOS 会造成序位变化，不过在该情景下它的 *NSD* 指标均值仅为 0.013，见表 11(b) ; 另据 *NSD* 的统计方式，sTR_GOS 在该情景下所引起的实际序位偏差均值不足 1，如此小幅的序位变化在运作实践中可以接受. 另外，方法 sTR_GOS 在情景(L，*，L，S，0) 下的平均耗时仅为 0.004 s，与另两种方法的耗时表现相当，见表 11(c) . 综上，当 MUAD 干扰发生在晚期、工时低幅缩减、算例较小且中断模式为“中断 — 不可续”时，从指标绩效的均衡性上看，宜采取 sTR_GOS 实施重调度.

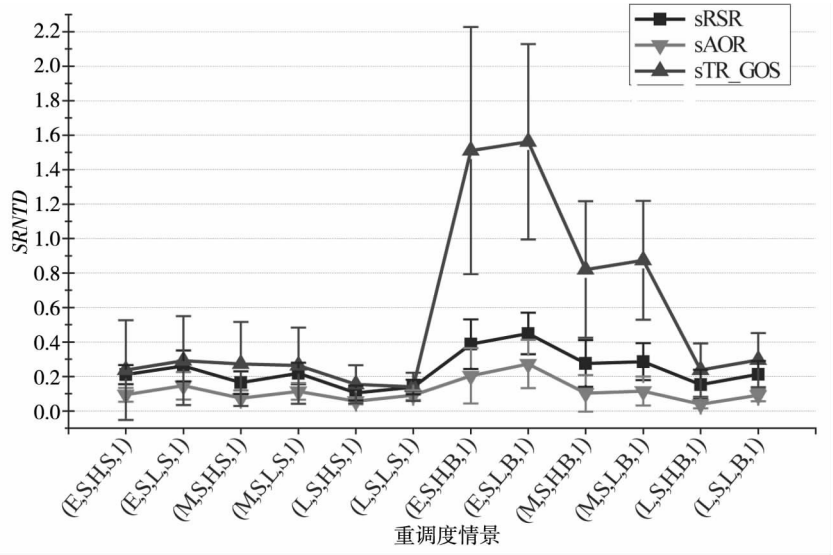

图 5(a)　情景(*，S，*，*，1) 下三种重调度方法在指标 *SRNTD* 上的均值及其标准差

Fig. 5(a) Means and standard deviations of three rescheduling methods on *SRNTD* under the (*，S，*，*，1) scenarios

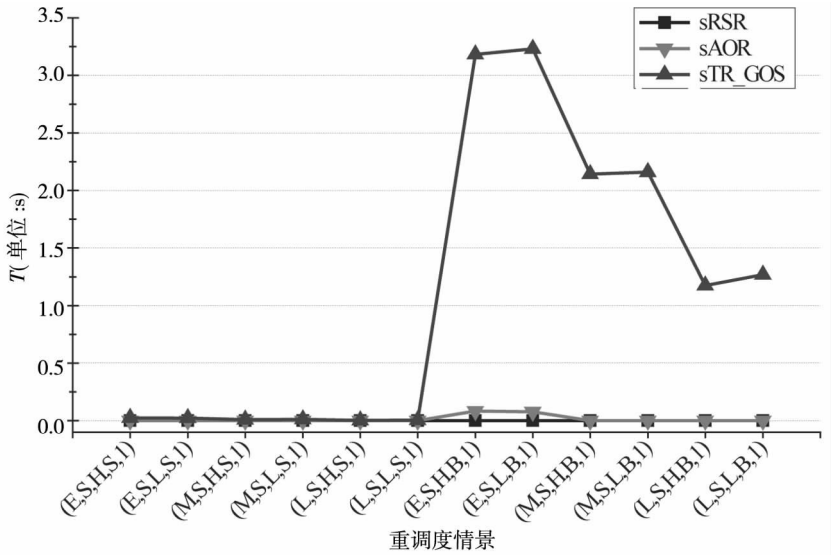

图 5(b)　情景(*，S，*，*，1) 下三种重调度方法在指标 *T* 上的均值结果

Fig. 5(b) Means of three rescheduling methods on *T* under the (*，S，*，*，1) scenarios

3) 当面对重调度情景族(E, *, *, B, *)时,方法 sTR_GOS 在时间表偏差指标 *SRNTD* 和序位偏差指标 *NSD* 上的比较劣势明显,见图3(b)和图4中的统计结果. 另外,结合图6中的 T 指标均值可看出,sTR_GOS 在干扰响应速度上也远不及方法 sRSR 和 sAOR. 因而,当 MUAD 干扰发生在早期且算例规模较大(即 $m \times n \geqslant 100$)时,不建议选用 sTR_GOS 来响应、处理 $\mathrm{MUAD}_{\mathrm{CPT}}$ 干扰.

表 11(a)　情景(L, *, L, S, 0)下三种重调度方法在指标 *RM* 上的均值结果

Table 11(a) Means of three rescheduling methods on *RM* under the (L, *, L, S, 0) scenarios

重调度情景	*RM*		
	sRSR	sAOR	sTR_GOS
(L, S, L, S, 0)	1.105	1.072	1.061
(L, M, L, S, 0)	1.122	1.102	1.090
(L, L, L, S, 0)	1.125	1.090	1.076
(L, *, L, S, 0)	**1.117**	**1.088**	**1.076**

表 11(b)　情景(L, *, L, S, 0)下三种重调度方法在指标 *NSD* 上的均值结果

Table 11(b) Means of three rescheduling methods on *NSD* under the (L, *, L, S, 0) scenarios

重调度情景	*NSD*		
	sRSR	sAOR	sTR_GOS
(L, S, L, S, 0)	0	0	0.012
(L, M, L, S, 0)	0	0	0.014
(L, L, L, S, 0)	0	0	0.013
(L, *, L, S, 0)	**0**	**0**	**0.013**

表 11(c)　情景(L, *, L, S, 0)下三种重调度方法在指标 *T* 上的均值结果

Table 11(c) Means of three rescheduling methods on *T* under the (L, *, L, S, 0) scenarios

重调度情景	*T*		
	sRSR	sAOR	sTR_GOS
(L, S, L, S, 0)	0.000	0.000	0.003
(L, M, L, S, 0)	0.000	0.000	0.003
(L, L, L, S, 0)	0.000	0.000	0.007
(L, *, L, S, 0)	**0.000**	**0.000**	**0.004**

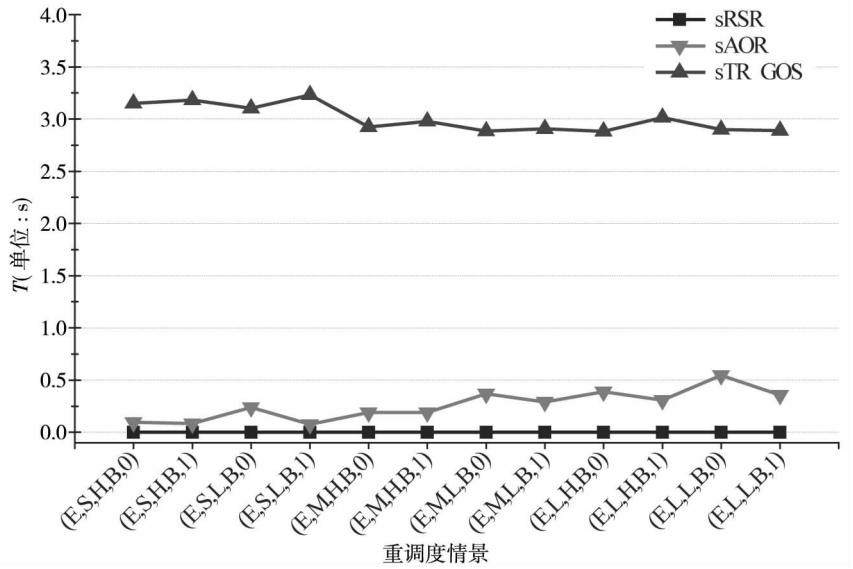

图6　情景(E, *, *, B, *)下三种重调度方法在指标 *T* 上的均值结果

Fig. 6 Means of three rescheduling methods on *T* under the (E, *, *, B, *) scenarios

5　结束语

本文针对“机器持续不可用”干扰及其并发出现的个别工时缩减现象,从右移、部分重调度及调度再生成三种典型重调度策略入手,以 $O_m \mid\mid C_{\max}$ 为基准问题,提出并实现了三种重调度方法,即 sRSR、sAOR 和 sTR_GOS. 通过大量模拟不同的重调度情景,观测了三种方法在相同情景下各自的重调度绩效,并利用统计分析的手段得出了一系列隐藏在实验数据中的重要结论与启示.

本文得出的总体性实验研究结论有: 在针对

完工期的性能指标上,方法 sTR_GOS 的表现显著优于 sAOR,而方法 sAOR 明显好于 sRSR;在结束时间偏差指标上,方法 sAOR 显著优于另外两者;两种定序重调度方法 sRSR 和 sAOR 具备最佳的序位稳定性,而 sTR_GOS 则会造成显著的序位变化;在重调度耗时上,sRSR 的干扰响应速度最快、最稳定,而方法 sTR_GOS 的耗时相对更多. 此外,针对特定的重调度情景,本文揭示了有关重调度方法适用性的统计结论,即:当出现短时长的 MUAD 干扰且采取"中断 — 可续"方式恢复被中断工序时,推荐采用方法 sAOR;当 MUAD 干扰发生在晚期、工时低幅缩减、算例小且以"中断 — 不可续"方式恢复中断时,建议采取 sTR_GOS 实施重调度;当 MUAD 干扰发生在早期且算例规模较大时,不建议选用方法 sTR_GOS.

需要指出的是,本文以 $MUAD_{CPT}$ 干扰单一出现为假定条件而展开研究;然而,在实际的车间运作环境中却普遍存在着各种干扰事件多次、反复出现的情形. 为了更贴近于生产运作的实际情况,进一步的研究有待于针对某干扰因素反复出现或多种干扰因素相继出现情形下的开放式车间重调度活动而展开.

参 考 文 献:

[1] Ouelhadj D, Petrovic S. A survey of dynamic scheduling in manufacturing systems [J]. Journal of Scheduling, 2009, 12(4): 417 – 431.

[2] Abumaizar R J, Svestka J A. Rescheduling job shops under random disruptions [J]. International Journal of Production Research, 1997, 35(7): 2065 – 2082.

[3] Vieira G E, Herrmann J W, Lin E. Rescheduling manufacturing systems: A framework of strategies, policies, and methods [J]. Journal of Scheduling, 2003, 6(1): 39 – 62.

[4] 胡祥培,孙丽君,王雅楠. 物流配送系统干扰管理模型研究 [J]. 管理科学学报, 2011, 14(1): 50 – 60.
Hu Xiangpei, Sun Lijun, Wang Yanan. A model for disruption management in urban distribution systems [J]. Journal of Management Sciences in China, 2011, 14(1): 50 – 60. (in Chinese)

[5] 王旭坪,阮俊虎,张 凯,等. 有模糊时间窗的车辆调度组合干扰管理研究 [J]. 管理科学学报, 2011, 14(6): 2 – 15.
Wang Xuping, Ruan Junhu, Zhang Kai, et al. Study on combinational disruption management for vehicle routing problem with fuzzy time windows [J]. Journal of Management Sciences in China, 2011, 14(6): 2 – 15. (in Chinese)

[6] Akturk M S, Atamturk A, Gurel S. Parallel machine match-up scheduling with manufacturing cost considerations [J]. Journal of Scheduling, 2010, 13(1): 95 – 110.

[7] Li R K, Shyu Y T, Adiga S. A heuristic rescheduling algorithm for computer-based production scheduling systems [J]. International Journal of Production Research, 1993, 31(8): 1815 – 1826.

[8] Dong Y H, Jang J. Production rescheduling for machine breakdown at a job shop [J]. International Journal of Production Research, 2012, 50(10): 2681 – 2691.

[9] Mason S J, Jin S, Wessels C M. Rescheduling strategies for minimizing total weighted tardiness in complex job shops [J]. International Journal of Production Research, 2004, 42(3): 613 – 628.

[10] Guo B, Nonaka Y. Rescheduling and optimization of schedules considering machine failures [J]. International Journal of Production Economics, 1999, 60 – 61(1): 503 – 513.

[11] Akturk M S, Gorgulu E. Match-up scheduling under a machine breakdown [J]. European Journal of Operational Research, 1999, 112(1): 81 – 97.

[12] Kurihara K, Owada Y, Li Y L, et al. A method of flow-shop re-scheduling dealing with variation of productive capacity [J]. Journal of Systemics, Cybernetics and Informatics, 2005, 3(1): 18 – 24.

[13] 上官春霞,周 泓,师瑞峰,等. 作业车间排序重调度问题及其改进修复约束满足算法 [J]. 计算机集成制造系统, 2008, 9(14): 1742 – 1751.
Shangguan Chunxia, Zhou Hong, Shi Ruifeng, et al. Flow shop rescheduling problem and its improved repair-based con-

straint satisfaction algorithm [J]. Computer Integrated Manufacturing Systems, 2008, 9(14): 1742 – 1751. (in Chinese)

[14] 李铁克, 肖拥军, 王柏琳. 基于局部性修复的 HFS 机器故障重调度 [J]. 管理工程学报, 2010, 24(3): 45 – 49.
Li Tieke, Xiao Yongjun, Wang Bailin. HFS rescheduling under machine failures based on local repair [J]. Journal of Industrial Engineering / Engineering Management, 2010, 24(3): 45 – 49. (in Chinese)

[15] Subramaniam V, Raheja A S. mAOR: A heuristic-based reactive repair mechanism for job shop schedules [J]. International Journal of Advanced Manufacturing Technology, 2003, 22(9 – 10): 669 – 680.

[16] Subramaniam V, Raheja A S, Rama Bhupal Reddy K. Reactive repair tool for job shop schedules [J]. International Journal of Production Research, 2005, 43(1): 1 – 23.

[17] Suwa H, Sandoh H. Capability of cumulative delay based reactive scheduling for job shops with machine breakdowns [J]. Computers & Industrial Engineering, 2007, 53(1): 63 – 78.

[18] 刘　琳, 谷寒雨, 席裕庚. 工件到达时间未知的动态车间滚动重调度 [J]. 机械工程学报, 2008, 44(5): 68 – 75.
Liu Lin, Gu Hanyu, Xi Yugeng. Rescheduling algorithm based on rolling horizon decomposition for a dynamic job shop with uncertain arriving time [J]. Chinese Journal of Mechanical Engineering, 2008, 44(5): 68 – 75. (in Chinese)

[19] Moratori P, Petrovic S, Vazquez-Rodriguez J A. Integrating rush orders into existent schedules for a complex job shop problem [J]. Applied Intelligence, 2010, 32(2): 205 – 215.

[20] Jain A K, Elmaraghy H A. Production scheduling/rescheduling in flexible manufacturing [J]. International Journal of Production Research, 1997, 35(1): 281 – 309.

[21] Wang S J, Xi L F, Zhou B H. Filtered-beam-search-based algorithm for dynamic rescheduling in FMS [J]. Robotics and Computer-Integrated Manufacturing, 2007, 23(4): 457 – 468.

[22] Fahmy S A, Balakrishnan S, ElMekkawy T Y. A generic deadlock-free reactive scheduling approach [J]. International Journal of Production Research, 2009, 47(20): 5657 – 5676.

[23] Zakaria Z, Petrovic S. Genetic algorithms for match-up rescheduling of the flexible manufacturing systems [J]. Computers & Industrial Engineering, 2012, 62(2): 670 – 686.

[24] Louis S J, Xu Z. Genetic algorithm for open shop scheduling and re-scheduling [C] // Proceedings of The 11th International Conference on Computers and Their Applications, California, USA, 1996: 99 – 102.

[25] Gonzalez T, Sahni S. Open shop scheduling to minimize finish time [J]. Journal of the Association for Computing Machinery, 1976, 23(4): 665 – 679.

[26] Liaw C F. A hybrid genetic algorithm for the open shop scheduling problem [J]. European Journal of Operational Research, 2000, 124(1): 28 – 42.

[27] Blum C. Beam-ACO — hybridizing ant colony optimization with beam search: An application to open shop scheduling [J]. Computers & Operations Research, 2005, 32(6): 1565 – 1591.

[28] Sha D Y, Hsu C Y. A new particle swarm optimization for the open shop scheduling problem [J]. Computers & Operations Research, 2008, 35(10): 3243 – 3261.

[29] Brasel H, Tautenhahn T, Werner F. Constructive heuristic algorithms for the open shop problem [J]. Computing, 1993, 51(2): 95 – 110.

[30] Gueret C, Prins C. Classical and new heuristics for the open-shop problem: A computational evaluation [J]. European Journal of Operational Research, 1998, 107(2): 306 – 314.

[31] Naderi B, Fatemi Ghomi S M T, Aminnayeri M, et al. A contribution and new heuristics for open shop scheduling [J]. Computers & Operations Research, 2010, 37(1): 213 – 221.

[32] 李　琳, 江志斌. 虚拟生产系统的自适应动态调度机理及算法 [J]. 计算机集成制造系统, 2006, 12(9): 1444 – 1452.
Li Lin, Jiang Zhibin. Self-adaptive dynamic scheduling mechanisms & algorithm of virtual production systems [J]. Computer Integrated Manufacturing Systems, 2006, 12(9): 1444 – 1452. (in Chinese)

[33] Taillard E. Benchmarks for basic scheduling problems [J]. European Journal of Operational Research, 1993, 64(2): 278 – 285.

Open shop rescheduling under a common disruptive condition

LIU Le, ZHOU Hong

School of Economics and Management, Beihang University, Beijing 100191, China

Abstract: This paper focuses on how to conduct effective and real-time rescheduling at an open shop subject to random Machine UnAvailability for a Duration (MUAD) and its concurrent Compressions of Processing Times (CPT). In this study, efficiency is measured by the makespan, while the stability measure is associated with the sequence deviation and ending time deviation. With three typical rescheduling strategies in the literature (i.e., right-shift rescheduling, affected operations rescheduling and total rescheduling), three specific approaches named sRSR, sAOR and sTR_GOS are proposed and implemented for the concerned MUAD_{CPT} disruption. In extensive experiments, by randomly generating various rescheduling scenarios, three initializations of previous schedules are examined and three rescheduling approaches are independently tested. The results statistically reveal that: 1) sAOR is highly recommended when the MUAD disruption lasts for a short duration and Interrupt-Resume mode is adopted; 2) sTR_GOS is relatively desirable in case of late MUAD disruption, low compression rate, small instance size and Interrupt-Repeat mode.

Key words: rescheduling; open shop; disruptions; stability; affected operations rescheduling