

# 消除二次资源冲突的鲁棒性双目标关键链项目调度优化

田宝峰, 张静文\*, 史至瑶

(西北工业大学 管理学院, 陕西 西安 710072)

**摘要:** 传统关键链方法无法解决插入输入缓冲引起的二次资源冲突困境,更不能表述和建模调度方案的鲁棒性,这极大地限制了它在项目进度管理中的广泛应用。本文从鲁棒调度和双目标优化两个角度拓展和创新了传统关键链方法。首先,针对最棘手的二次资源冲突困境,从鲁棒优化视角提出基于局部重调度的二次资源冲突消除策略,进而设计基于消除策略的鲁棒性测度指标;其次,构建同时优化项目工期和调度方案鲁棒性的双目标关键链项目调度模型,并设计混合差分进化算法求解。在获取基准调度计划阶段为克服现有的关键链识别方法的不足,设计基于鲁棒性资源流网络的关键链识别算法并将其嵌入差分进化主算法中。最后,设计并运行大规模数值测试实验,输出数据的统计结果验证了关键链识别算法和二次资源冲突消除策略的有效性,同时也表明了本文设计混合差分进化算法的优越性。

**关键词:** 二次资源冲突; 鲁棒性关键链; 双目标; 差分进化算法

**中图分类号:** C935; F224.33

**文献标识码:** A

**文章编号:** 1004-6062(2024)02-0166-014

**DOI:** 10.13587/j.cnki.jieem.2024.02.012

## 0 引言

在经济全球化和信息技术突飞猛进的背景下,许多现代工程项目已趋于大型化、复杂化和多元化,项目内外部的不确定和风险日益增大,面临的实施环境复杂多变。例如,2020年始料未及并席卷全球的新冠疫情直接导致大量工程项目中人力、设备及材料的短缺甚至供应中断。体现到微观层面,多种不确定因素总是干扰着项目进度过程,最终可能导致工期拖延、范围变更、资源的间断性短缺或闲置、更多的在制品库存,或者频繁地重新调度而增加系统的紧张度;尤其对于现代大型复杂工程项目,其进度过程对不确定事件的扰动更加敏感和脆弱。经典的资源约束型项目调度问题(resource-constrained project scheduling problem, RCPSP)不适用于复杂多变的实施环境。因此,不确定型项目进度管理已成为项目调度理论研究的热点之一<sup>[1-2]</sup>。实践中,不确定型 RCPSP 具有很强的工程背景,涉及诸多工业和生活领域,例如工程管理、软件开发、港口调度及飞机航线制定等<sup>[3]</sup>。

关键链方法(critical chain method, CCM)是一种相对较新的项目进度管理技术<sup>[4]</sup>,通过在初始的基准调度方案中插入时间缓冲以消除活动工期波动对进度执行的扰动,因此一些学者将 CCM 归类为

一种不确定型项目调度理论<sup>[5]</sup>。同时,缓冲机制的设置表明 CCM 更是属于一种鲁棒不确定型项目调度理论。为表述方便,本文后续内容中将 Goldratt 提出的关键链方法雏形称为传统 CCM。关键链(critical chain, CC)指由优先关系和资源限量都可行的活动集合形成的一条虚拟链条。CCM 中包含两类时间缓冲,输入缓冲(feeding buffer, FB)和项目缓冲(project buffer, PB),FB 和 PB 都有两个属性:尺寸和位置。理论上,嵌入多个 FB 到基准调度计划可获得鲁棒性的二次调度计划,但是在插入每个 FB 时通常导致严重的二次冲突场景(优先关系冲突或/和资源冲突),只能通过重新调度获得可行的二次调度计划。处理二次资源冲突非常棘手,因此被称为传统 CCM 中的“二次资源冲突困境”<sup>[6]</sup>。自传统 CCM 提出以来,学者们在 CCM 建模和求解、时间缓冲区尺寸计算方法<sup>[7-8]</sup>、缓冲的消耗及管理<sup>[9-10]</sup>、CCM 的实践应用<sup>[11-14]</sup>等方面产生较多的成果<sup>[15]</sup>。

建模和求解关键链项目调度问题的本质是首先求解经典 RCPSP,其次在获得的最优基准调度计划中识别出关键链和非关键链。Van De Vonder 等<sup>[16]</sup>采用分支定界算法获得关键链的基准计划,而模型求解主要集中于开发高效的启发式智能优化

**收稿日期:** 2021-08-30

**基金项目:** 国家自然科学基金项目(71971173);陕西省自然科学基金项目(2020-JM146);西北工业大学文美文科交叉学科方向培育项目(21GH031128)

**\* 通讯作者:** 张静文(1976—),女,陕西韩城人;西北工业大学管理学院教授,博士生导师;研究方向:项目调度、服务运作管理。

算法方面。Peng 和 Huang<sup>[12]</sup>设计差分进化算法求解关键链调度模型。传统 CCM 的最优解指工期最短的基准调度计划,而寻找活动网络中的关键链和非关键链则更复杂。Tukel 等<sup>[8]</sup>计算资源约束下活动的自由时差来识别关键链,同时还提出了考虑资源利用因子的 APRT (adaptive procedure with resource tightness) 和引入网络复杂性的 APD (adaptive procedure with density) 两种缓冲区尺寸计算方法。然而,目前鲜有学者对插入 FB 引起的二次资源冲突进行研究。部分学者将输入缓冲看作一个不消耗资源的新活动以避免二次资源冲突<sup>[16]</sup>。这种处理方式值得商榷,因为非关键活动工期延长时,必然会消耗资源,显然这种方式与缓冲的本质不相符。一些学者取 FB 与自由时差的较小值作为输入缓冲尺寸以避免二次资源冲突<sup>[12]</sup>,这显然是对关键链调度方案鲁棒性的减弱,与关键链调度的本质背道而驰。目前有少量学者考虑了二次资源冲突重调度策略,Zhao 等<sup>[17]</sup>通过迭代削减缓冲尺寸的方式消除二次资源冲突,该种方式也是对关键链调度方案鲁棒性的减弱。

综上,传统 CCM 仅提出了设置缓冲的思想和定性的步骤,未涉及输入缓冲插入时引起的二次冲突困境及在进度实施中缓冲的动态消耗等关键性问题;现有研究多考虑项目工期目标,忽略了能体现关键链重要特质的鲁棒性指标。同时,对于二次冲突困境问题,较少有学者从量化建模角度展开研究。而且,已有的鲁棒性项目调度研究中很少涉及对关键链鲁棒性的定量表述和建模。上述不足直接限制了传统 CCM 在实践项目进度管理中的广泛应用。鉴于此,本文以传统 CCM 体现出的鲁棒调度优化思想为切入点,提出解决二次冲突困境的局部重调度策略,并设计基于消除策略的鲁棒性指标来量化调度方案的鲁棒性;进而研究同时考虑项目工期和鲁棒性指标的双目标鲁棒性关键链项目调度问题 (robust and bi-objective critical chain project scheduling problem, RB-CCPSP) 的建模和求解。研究工作从鲁棒调度和双目标建模优化两个角度,创新了传统 CCM 并拓展了现有的关键链项目调度理论。

## 1 问题描述及关键链识别

### 1.1 二次资源冲突的表述

传统 CCM 的应用步骤为:第一,寻找项目工期最短的计划作为基准调度方案,进而识别出关键链和多条非关键,从而确定出各个 FB 位置与尺寸。通过求解经典 RCPSP 获得基准调度方案,本质是在满足逻辑关系和资源限量约束下最优地解决了资

源竞争(第一次资源冲突)。第二,每条非关键末端汇入关键链的入口处插入各自所带的 FB,在关键链末端(项目结束处)插入唯一的 PB。然而,将多个 FB 插入时在基准调度方案中的某些活动之间引起优先关系冲突和资源竞争(第二次资源冲突),而优先关系冲突只能后移紧后活动的开始时间,这进一步恶化了 FB 插入点及一定时段内的二次资源冲突态势,因此传统 CCM 中的二次冲突问题本质是如何解决“二次资源冲突困境”。消除了二次资源冲突且包含多个 FB 及 PB 的调度计划称为二次调度计划(相对于基准调度计划)。定量化表达二次资源冲突困境并设计鲁棒性指标,是建模和求解双目标鲁棒性关键链项目调度问题的前提。

传统关键链的概念是一种狭义上的关键链,仅指存在于最优基准调度方案(项目工期最短)中那些制约项目工期的活动集合。本文同时考虑项目工期和鲁棒性两个指标,所以首先将传统关键链的概念拓展为一般含义的关键链,即对所有可行的基准调度方案都识别其中的关键链和非关键链,对于每个基准调度方案输入时间缓冲并消除二次资源冲突,进而计算其对应的鲁棒性指标,采用项目工期和鲁棒性两个绩效指标评价每个调度方案的优劣。沿袭经典 RCPSP 的符号和标记,项目活动之间的逻辑关系采用节点式(activity-on-node, AoN)网络  $G=(V,E)$  表达,其中  $V=\{1,2,\dots,J\}$  为节点(活动)集合,节点 1 和节点  $J$  表示开始和结束两个虚工作。有向弧  $(i,j) \in E$  表示工艺优先关系。活动执行中不能中断也不能抢先,活动之间为结束-开始型优先关系。项目实施中共需  $K$  种可更新资源,第  $k$  种资源的可用量为  $R_k$ ,  $r_{jk}$  表示活动  $j$  在单位工期内对第  $k$  种资源的需求量。表达二次资源冲突困境和建模 RB-CCPSP 的参数及符号见表 1。

### 1.2 基于鲁棒性资源流网络的关键链与非关键链识别

#### 1.2.1 鲁棒性资源流网络构建

关键链和非关键链的识别是 CCM 应用的基础。作为基准调度方案中资源和工艺关系均可行的最长链路,关键链上的活动存在工艺和资源双重制约关系。资源流网络在满足工艺关系的前提下,强调活动之间的资源制约关系,与关键链的本质契合。由于资源流向的不同,同一基准调度计划可能对多种资源流网络图,而不同的资源流网络,在项目执行过程中鲁棒性存在差异<sup>[18]</sup>。通过合理分配资源生成鲁棒性资源流网络能有效地提高缓冲对于基准调度计划的保护效果<sup>[19]</sup>。因此本文基于鲁棒性资源流网络(robust resource flow network, rRFN)识别关键链和非关键链。

表 1 参数说明表  
Table 1 Symbol description

| 参数                                  | 说明                                                                                                                                                                        |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $S$                                 | 基准调度计划                                                                                                                                                                    |
| $S'$                                | 在 $S$ 中插入时间缓冲并消除二次资源冲突后的二次调度计划                                                                                                                                            |
| $s_j$                               | 基准调度计划 $S$ 中活动 $j$ 的(最早)开始时间                                                                                                                                              |
| $f_j$                               | 基准调度计划 $S$ 中活动 $j$ 的(最早)结束时间                                                                                                                                              |
| $ls_j^{rc}$                         | 满足资源限量条件下,在 $S$ 中活动 $j$ 的最晚开始时间                                                                                                                                           |
| $lf_j^{rc}$                         | 满足资源限量条件下,在 $S$ 中活动 $j$ 的最晚结束时间                                                                                                                                           |
| $ff_j^{rc}$                         | 满足资源限量条件下,在 $S$ 中活动 $j$ 的自由时差                                                                                                                                             |
| $CC$                                | 关键活动集合, $CC = \{CA^1, CA^2, \dots, CA^m, \dots, CA^M\}$ , $m = 1, 2, \dots, M$ , 其中 $M$ 为关键活动个数                                                                           |
| $NC_q$                              | 第 $q$ 条非关键链上的非关键活动集合, 即, $NC_q = \{na_1^q, na_2^q, \dots, na_{ NC_q }^q\}$ , $q = 1, 2, \dots, Q$ , 其中 $Q$ 为非关键链条数                                                        |
| $na_{ NC_q }^q$                     | 第 $q$ 条非关键链上的最后一个非关键活动(也称之为非关键的末端活动), 关键链 $q$ 上共包含 $ NC_q $ 个非关键活动                                                                                                        |
| $s'_j$                              | 消除二次资源冲突后, 在 $S'$ 中活动 $j$ 的开始时间                                                                                                                                           |
| $f'_j$                              | 消除二次资源冲突后, 在 $S'$ 中活动 $j$ 的结束时间                                                                                                                                           |
| $ff_j^{rc'}$                        | 消除二次资源冲突后, 在 $S'$ 中活动 $j$ 的自由时差                                                                                                                                           |
| $CA^q$                              | 第 $q$ 条非关键链的对应的汇入关键工作, $CA^q \in CC$                                                                                                                                      |
| $ff_{NC_q}$                         | 第 $q$ 条非关键链的自由时差                                                                                                                                                          |
| $ff_{NC_q}'$                        | 消除二次资源冲突后, 第 $q$ 条非关键链的自由时差                                                                                                                                               |
| $FB_q$                              | 第 $q$ 条非关键链对应的输入缓冲                                                                                                                                                        |
| $s_{na_{ NC_q }^q}^{na_{ NC_q }^q}$ | 在基准调度计划中, 输入缓冲 $FB_q$ 的输入时刻点, 即非关键链 $q$ 上最后一个非关键活动 $na_{ NC_q }^q$ 在 $S$ 中的开始时间。根据每个 $s_{na_{ NC_q }^q}^{na_{ NC_q }^q}$ 表示的时刻点, 由小到大依次对 $Q$ 条非关键链编号 $q = 1, 2, \dots, Q$ |
| $s_{FB_q}'$                         | 二次调度计划中, $FB_q$ 的插入时刻点                                                                                                                                                    |
| $r_{FB_q}^k$                        | 第 $q$ 条非关键链的输入缓冲对于第 $k$ 种资源的需求量                                                                                                                                           |
| $d_{FB_q}$                          | 第 $q$ 条非关键链的输入缓冲尺寸                                                                                                                                                        |
| $d_{FB_q}'$                         | 第 $q$ 条非关键链位于末节点后的输入缓冲尺寸                                                                                                                                                  |

在资源流网络中, 同一资源弧上的活动存在资源制约关系<sup>[20]</sup>。资源流紧前工作延误, 必然会由于资源的制约影响资源流紧后工作的开始。资源流网络图中活动之间的箭线  $|E'|$  越少, 资源约束造成的活动之间的相互制约越少, 调度方案鲁棒性越强, 因此本文以最少  $|E'|$  为原则构建鲁棒资源流网络, 限于篇幅, 具体过程本文不再赘述。

### 1.2.2 关键链及非关键链识别

CCM 中, 关键活动延误必然会影响项目工期<sup>[4]</sup>。由于自由时差的传递性, 具有传递自由时差的工作虽然自由时差为 0, 其延迟却不会延长项目工期。因此, 在识别关键工作时, 首先要排除具有自由时差和传递自由时差工作等必为非关键工作的活动。识别关键链和非关键链过程如表 2 所示。

表 2 基于 rRFN 的关键链识别算法  
Table 2 Critical chain identification algorithm based on rRFN

| 识别关键链和非关键链                               |                                                                                                                                                                  |
|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Step1: 初步识别出必为非关键工作的活动(具有自由时差和传递自由时差的活动) |                                                                                                                                                                  |
| 关键链识别                                    | 由活动 1 开始沿资源流活动网络识别出关键工作。当存在并行链路时, 优先选择单位时间资源用量最大工作 ( $\max \sum_{k=1}^K r_{jk}$ ) 为关键工作, 若单位时间资源用量相等, 则选择其优先关系紧后工作为关键工作。一次识别出唯一确定的关键链, 关键链上的工作即为关键工作, 其余工作为非关键工作 |
| 非关键链识别                                   | Step1: 沿着关键链正向逐个识别以该关键活动为汇入节点的非关键链。从某一关键工作 $CC_m$ 开始, 识别出其资源流紧前工作, 直至紧前工作全部为关键工作或其非关键链上的工作                                                                       |
|                                          | Step2: 判断所有的关键工作是否已经全部检查完, 若未检查完, 则转入 Step1, 继续检查下一关键工作 $CC_{m+1}$ 。若已识别完, 则转入 Step3                                                                             |
|                                          | Step3: 输出非关键链及其对应的关键工作入口节点                                                                                                                                       |



## 2 鲁棒性指标设计及双目标模型构建

CCM 中, FB 用于吸收非关键链延迟对关键活动的影响, 提高调度方案鲁棒性。FB 的尺寸越大, 关键链调度方案吸收非关键链延迟的能力越强。本文在保证 FB 尺寸不减小的前提下, 设计了基于局部重调度的鲁棒性二次资源冲突消除策略 (second resource conflicts eliminating policy, SRCEP)。

### 2.1 二次资源冲突困境分析

第  $q$  条非关键链上最后一个非关键活动为  $na^q_{|NC_q|}$  ( $q=1, 2, \dots, Q$ ), 在基准调度计划  $S$  中的结束时间为  $f_{na^q_{|NC_q|}}$ 。  $FB_q$  是非关键链  $q$  确定的输入缓冲, 其尺寸为  $d_{FB_q}$ 。在  $S$  中,  $FB_q$  的插入时刻点即为  $f_{na^q_{|NC_q|}}$ , 插入  $FB_q$  引起二次冲突的时段为  $[f_{na^q_{|NC_q|}} + ff_{na^q_{|NC_q|}}^{rc}, f_{na^q_{|NC_q|}} + d'_{FB_q}]$ 。多个 FB 逐个插入, 处理插入时刻点 (称为当前 FB 的插入时刻点) 较早时段内的二次资源冲突时, 通常影响当前插入时刻点之后 FB 的插入状态 (指时刻点及资源配置方案)。即当插入  $FB_l$  时, 影响到其后续共计  $(Q+1)$  个 FB 的插入状态; 当插入  $FB_l$  时将影响到  $FB_q$  ( $q=l+1, \dots, Q$ ) 的插入状态。为避免多次且无用地重调度涉及的活动, 同时使插入所有 FB 后的二次调度计划  $S'$  与  $S$  的资源配置状态尽可能接近, 基于鲁棒调度优化思想 (解的鲁棒性) 本文提出一种局部重调度策略来消除二次资源冲突。即解决当前 FB 插入引起的二次资源冲突时, 仅重调度当前缓冲插入时刻点至下一个缓冲插入时刻点涉及的活动。设当前  $FB_q$  的插入时刻点  $t_{FB_q} = f_{na^q_{|NC_q|}}$ , 下一个  $FB_{q+1}$  的插入时刻点  $t_{FB_{q+1}} = f_{na^{q+1}_{|NC_{q+1}|}}$ 。在时刻  $t_{FB_q}$ ,  $A^{q+1}_q$  表示此刻参与局部重调度的活动集合, 则参与局部重调度的活动  $j$  满足条件  $t_{FB_q} + ff_{na^q_{|NC_q|}}^{rc} \leq s_j \leq t_{FB_{q+1}} + ff_{na^{q+1}_{|NC_{q+1}|}}^{rc}$ 。以  $FB_q$  为分析对象, 根据  $FB_q$  与冲突时段内关键活动之间的干涉关系, 基于资源流网络将重调度时段内的冲突分为三种类型:

(1) 与关键活动发生优先关系冲突, 指  $FB_q$  与汇入关键活动  $CA^q$  存在优先关系, 即  $(FB_q, CA^q) \in E$ 。

(2) 与关键活动发生资源竞争冲突。即  $\exists k$  满足  $r_{FB_q, k} + r_{CA^q, k} > R_k$  时,  $FB_q$  与冲突时段内的关键活动至少一种资源需求量之和已超出其限量。

(3) 与正在进行中的非关键活动发生资源竞争冲突。在某种冲突情况下,  $FB_q$  与  $CA^q$  虽然不存在前两种资源冲突, 但是由于正在进行的非关键活动占用了部分资源, 使得  $FB_q$  与  $CA^q$  的可用资源量变少, 从而无法并行, 即  $\exists k$  满足  $r_{FB_q, k} + r_{CA^q, k} + r_{ik} > R_k$ , 其中  $i$  为正在进行中的非关键活动。

当存在冲突情境同时满足多种冲突类型时, 确定较小编号冲突类型为其冲突类型。

### 2.2 消除策略

通过以上分析设计出基于局部重调度的两阶段启发式二次资源冲突消除策略 SRCEP:

Stage1: 识别冲突类型并后移活动以消除二次冲突。当冲突为第 (1) 种类型或第 (2) 种类型时,  $CA^q$  须后移  $(d_{FB_q} - ff_{NC_q})$  个单位才能消除。当冲突类型为第 (3) 种时, 以  $i$  表示正在进行中的非关键活动, 则  $CA^q$  须后移  $\min \{f_i - s_{CA^q}, f_{NC_q} - s_{CA^q}\}$  个单位。无论哪种类型, 为了使得  $S'$  尽可能地维持  $S$  中的资源配置状态, 第一阶段采取将冲突时刻点之后的调度计划整体后移, 此方式可尽可能避免对后续插入 FB 时段内资源配置状态的影响和改变。关键活动后移, 导致项目 PB 缓冲被消耗。

Stage2: 将冲突时段非关键工作前移, 增大后续非关键链自由时差。经过 Stage1 的后移操作,  $FB_q$  在冲突时段内除已开始工作外, 已无任何并行工作, 故二次冲突已经消除。由前文分析可知, 虽然  $FB_q$  与  $CA^q$  存在资源流优先关系, 但是  $FB_q$  与  $[s'_{FB_q}, s'_{FB_{q+1}}]$  区间内开始的非关键活动并不存在资源流关系, 因此  $FB_q$  可与该区间内的某些非关键活动并行。通过连续前移非关键活动, 减小非关键活动的开始时间可增大其自由时差, 从而增加其所处非关键链的自由时差, 减小后续插入缓冲对于基准调度计划的破坏, 体现调度方案解在实施中的稳定性。前移时, 开始时间越小的活动优先权越高; 发生平局时优先左移所属的非关键链编号较小的非关键活动。

理论上, 设置在调度方案末端的 PB 可吸收关键链上活动的部分拖延。但是, 由于资源限量, 在基准调度方案和多个消除二次冲突的中间状态调度方案中, 关键活动拖延则破坏其所处时段的资源配置状态, 引起二次冲突。处于末端的 PB 仅在时间上吸收了项目工期的延长, 但是并不能直接消除关键活动拖延引起的二次冲突。在本文的二次资源冲突消除策略中, 在 FB 的插入时刻点后移调度方案, 本质上是将末端的整个 PB 分散到调度方案的各个阶段, 参与消除二次资源冲突。

以项目 1 为例, 该项目需要 R1 和 R2 两种资源, 资源限量分别为 13 和 15。项目网络如图 1 所示, 以工期最短调度方案为基准调度方案, 基于 rRFN 识别 CC 和 NC。采用 SRCEP 消除二次资源冲突, 根据插入时刻点, 从早到晚依次逐个插入各个 FB, 并采用本文的局部重调度策略消除每次插入 FB 引起的二次资源冲突, 最终获得嵌入所有 FB 和

唯一 PB 的二次调度计划,具体过程见表 3。如图 2 所示,二次调度计划的项目工期为 69,而基准调度计划的工期则为 54。

### 2.3 基于消除策略的鲁棒性指标设计

二次调度方案  $S'$  中,在某些插入 FB 时段,存在

一些时间区间中断,即在有些时段上没有安排活动。针对此,提出二次调度方案中的可用性中断的概念。

可用性中断:二次调度方案中,在不影响其他工作和缓冲开始时间的前提下,能吸收关键活动延

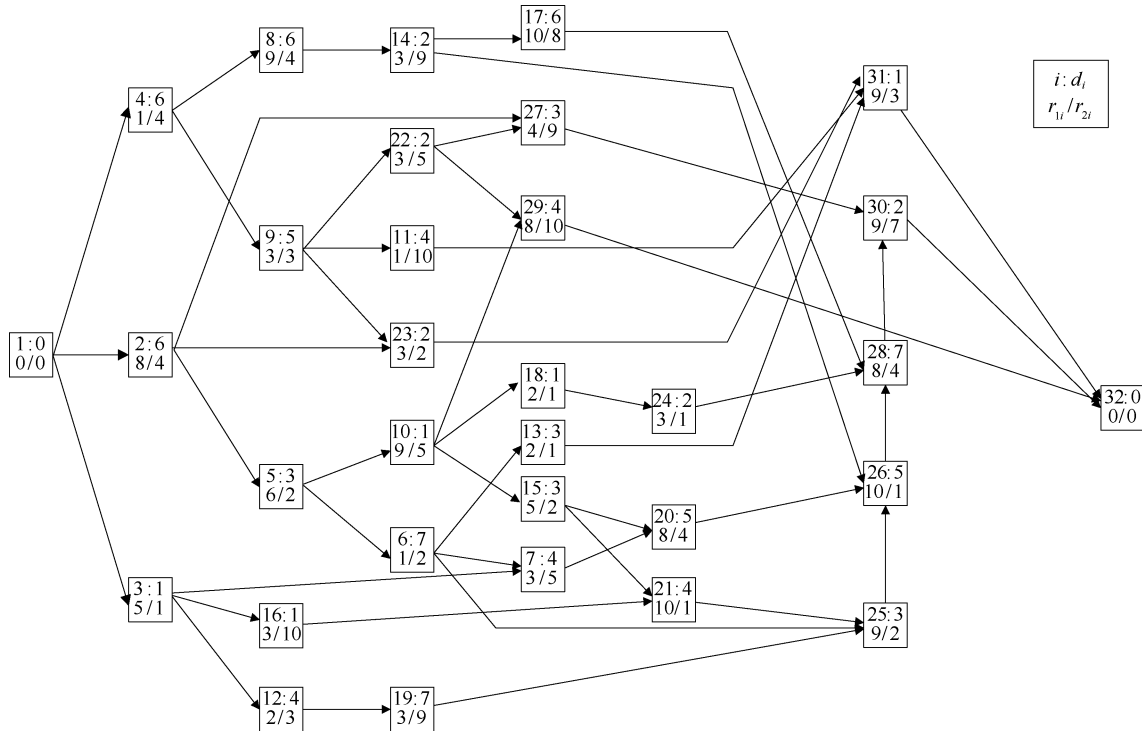
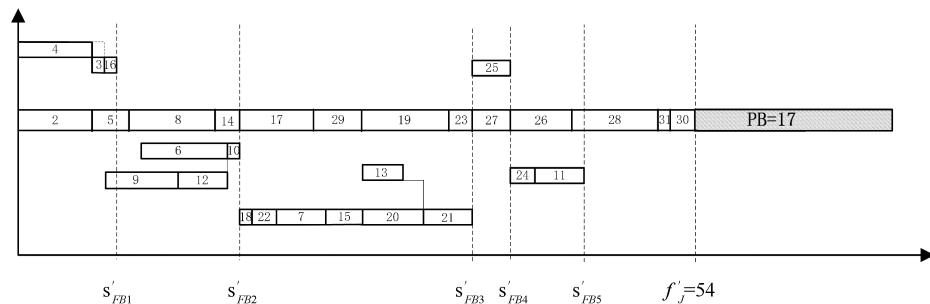
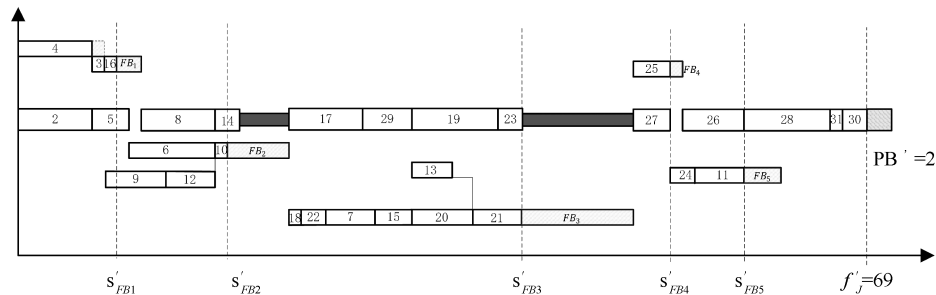


图 1 项目 1 网络

Figure 1 Network of project 1



(a) 未插入输入缓冲的基准调度计划



(b) 二次资源冲突消除后的关键链调度

图 2 二次冲突消除策略示意图

Figure 2 Baseline scheduling and critical chain scheduling after eliminating second resource conflicts

表 3 基于鲁棒调度的局部重调度消除二次资源冲突的过程  
Table 3 The process of eliminating secondary resource conflicts by SRCEP

| 序号                                         | 插入时刻、冲突时段及时长                                     | 冲突类型<br>重调度时段                                             | 消除策略以及对后续 FB 插入状态的影响<br>PB 的消耗                                                                                                                                                                                                                      |
|--------------------------------------------|--------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NC <sub>1</sub><br>FB <sub>1</sub> =3      | $t_{FB_1}=9$<br>(9,10]<br>$FB_1 - ff_{NC_1} = 1$ | 第三种 (由于非关键活动 9 执行消耗了部分资源)<br>[8,17]                       | Stage1 从 $t = 9$ 开始调度计划后移 1 个时间单位<br>Stage2: 非关键活动 6、10 和 12 前移 1 个时间单位, 导致 10 的自由时差增加 1 个时间单位 (缓解了 $FB_2$ 插入引起的冲突形势)<br>$FB_2$ 的插入时刻点不变, 但影响 $FB_2$ 插入时段的资源配置状态<br>将 $FB_3$ 、 $FB_4$ 和 $FB_5$ 的插入时间点推迟 1 时间单位, 资源配置状态与 S 中初始状态相同<br>消耗了 1 个时间单位 PB |
| NC <sub>2</sub><br>FB <sub>2</sub> =5      | $t_{FB_2} = 7$<br>(17,22]<br>4                   | 第二种 ( $FB_2$ 和关键活动 17 对 $r_1$ 的使用量之和超过 $R_1$ )<br>[18,44] | Stage1: 从 $t=18$ 开始调度计划向后移 4 个时间单位<br>Stage2: 无可前移的非关键活动<br>$FB_3$ 、 $FB_4$ 和 $FB_5$ 的插入时刻点在上一个状态下继续后移 4 个时间单位<br>消耗了 4 个时间单位 PB                                                                                                                    |
| NC <sub>3</sub><br>FB <sub>3</sub> =9      | $t_{FB_3} = 36, 37, 41$<br>(41,50]<br>9          | 第二种 ( $FB_3$ 和 27 对于 $r_1$ 的消耗量之和大于 $R_1$ )<br>[42,53]    | Stage1: 从 $t=41$ 开始调度计划向后移动 9 个时间单位<br>Stage2: 无可前移的非关键活动<br>$FB_4$ 和 $FB_5$ 的插入时刻点在上一个状态下继续后移 9 个时间单位<br>消耗了 1 个时间单位 PB                                                                                                                            |
| NC <sub>4</sub><br>FB <sub>4</sub> =1      | $t_{FB_4} = 39, 40, 44, 53$<br>(53,54]<br>1      | 第一种 ( $FB_4$ 与关键活动 26 发生优先关系)<br>[54,59]                  | Stage1: 从 $t=53$ 开始, 调度计划向后移动 1 个时间单位<br>Stage2: 非关键活动 24 和 11 前移 1 个时间单位 (11 增加 1 个单位的自由时差, 缓解了 $FB_5$ 插入引起的冲突)<br>消耗了 1 个时间单位 PB                                                                                                                  |
| NC <sub>5</sub><br>FB <sub>5</sub> =3<br>0 | $t_{FB_5} = 45, 46, 50, 59$<br>(62,66]<br>0      | 无冲突 ( $f_{11}^* > FB_5$ )<br>无须重调度                        | 无须消除                                                                                                                                                                                                                                                |

迟的中断为可用性中断 (available break-down,  $BD_{Available}^l$ )。可用性中断的大小为中断前关键活动可以后移的单位。例如图 2(b) 中阴影中断可以吸收关键链活动延迟, 因此可以看作是项目缓冲的一部分。然而, 可用性中断只能吸收中断之前关键活动的延迟, 而项目缓冲却可以吸收所有关键活动的延迟, 为测度不同可用性中断对鲁棒性贡献的差异性, 提出可用性中断的鲁棒性测度指标如下:

$$BD_{Available}^l = \frac{\sum_{i \in CC^l} d_i}{\sum_{j \in CC} d_j} \cdot length_l \quad (1)$$

其中,  $CC^l$  表示在二次调度方案  $S'$  中, 位于第  $l$  个中断之前的关键活动集合, 显然  $CC^l \subseteq CC$ ,  $CC$  表示关键活动集合,  $length_l$  表示第  $l$  个可用性中断的长度。以中断之前关键活动的活动工期之和与所有关键活动工期之和作为权值。由式 (1) 可以看出, 可用性中断的鲁棒性贡献由中断的位置和尺寸决定, 可用中断位置越靠后, 尺寸越大, 鲁棒性贡献越大。

在  $S'$  中, 处于  $NC_3$  上的活动 22 与关键活动 29 存在优先关系约束, 且 29 位于  $NC_3$  汇入关键链后接驳的关键活动 27 之前。当 22 拖延较小时, 可用通过  $FB_3$  吸收其拖延, 但拖延超过 3 个时间单位时, 其拖延在未被  $FB_3$  消除之前就先通过优先关系网络影响关键活动 29 的执行, 削弱了  $FB_3$  对关键链的保护效果。针对此情景, 提出不稳定活动集合的概念。

不稳定活动集合: 对于某条非关键链  $NC_q$ , 其汇入关键链时接驳的关键活动为  $CA^q$ , 但是  $NC_q$  上存在非关键活动  $i$ ,  $i$  之后的关键活动  $j$  满足  $s'_j < s'_{CA^q}$ 。此时若  $i$  拖延导致其结束时间超过  $s'_j$ ,  $i$  与  $j$  首先引起优先关系冲突, 也即  $i$  的拖延在未被  $FB_q$  消除之前就影响到关键链。这种潜在的冲突削弱了  $j$  之后  $FB_q$  的鲁棒性效果。出现这种场景的非关键活动  $i$  的集合称为  $NC_q$  上的不稳定性活动集合, 表述如下:

$$F_q = \{i | i \in NC_q, j \in CC \text{ and } (i, j) \in E, s'_j < s'_{CA^q}\} \quad (2)$$

显然  $F_q$  中集合元素越多,  $FB_q$  的鲁棒性越低,

因此设置输入缓冲鲁棒性系数  $\rho_q = 1 - |F_q|/|NC_q|$ 。同时,考虑缓冲效用值,设置各条非关键链输入缓冲鲁棒性上限值为 1。为消除非关键链条数对于鲁棒性测度的影响,取各条非关键链鲁棒值的平均值。

根据上述的两个定义和前述的局部重调度二次资源冲突消除策略,设计表达策略效果的关键链调度方案的鲁棒性指标如下:

$$RM = \frac{\sum_{q=1}^Q \min \left( 1, \rho_q \frac{\max \{d_{FB_q}, ff'_{NC_q}\}}{\sum_{i \in NC_q} d_i} \right)}{Q} + \frac{PB' + \sum_{l=1}^L BD_{Available}^l}{\sum_{j \in CC} d_j} \quad (3)$$

从式(3)可以看出,鲁棒性指标由两部分构成,第一部分由 FB 来体现,第二部分为 PB 对鲁棒性的贡献。在第一部分中  $\frac{\max \{d_{FB_q}, ff'_{NC_q}\}}{\sum_{i \in NC_q} d_i}$  表示非关键链

活动任务工期内每个单位工期分配的输入缓冲尺寸,单位工期分配到的输入缓冲尺寸越大,鲁棒性越强。第二部分为项目缓冲的鲁棒性贡献测度,  $PB'$  表示二次资源冲突消除后剩余的项目缓冲值,  $\frac{PB'}{\sum_{j \in CC} d_j}$  表示关键活动活动任务工期内每个单位工期分配到的项目缓冲尺寸,单位工期分配到的输入缓冲尺寸越大,鲁棒性越强。

#### 2.4 模型构建

定性分析,通常基准调度方案的项目工期越长,表明其中的非关键活动数量较多或者机动时间充裕,插入所有 FB 时引起二次资源冲突困境更易解决,调度方案解的鲁棒性越大。因此,工期和鲁棒性是一对相互矛盾的目标。现实中,项目经理在追求更短项目工期,也期望调度方案在实施中尽可能少地调整或者更易调整,因此项目工期和鲁棒性是同时关注的目标。鉴于此,基于鲁棒调度优化思想和鲁棒性指标,构建考虑二次冲突困境的鲁棒性关键链项目调度模型 RB-CCPSP 如下:

$$\begin{aligned} & \min f_j \\ & \max RM \\ & \text{s. t.} \\ & s_i + d_i \leq s_j, \quad \forall (i, j) \in E \quad (4) \\ & s'_i + d_i \leq s'_j, \quad \forall (i, j) \in E \quad (5) \\ & s'_{FBq} + d'_{FBq} \leq s'_j, \quad \forall (FB_q, j) \in E \quad (6) \\ & s'_{na^q|NC_q|} + d_{na^q|NC_q|} = s'_{FBq}, \quad \forall q \quad (7) \end{aligned}$$

$$\sum_{j \in A(t)} r_{jk} \leq R_k, \quad \forall k, \forall t, j = 1, 2, \dots, J, \quad (8)$$

$$r_{FB_q k} = r_{na^q|NC_q| k}, \quad \forall q, \forall k \quad (9)$$

$$s'_i, d'_{FBq}, d_{FBq} \geq 0 \quad (10)$$

在上述模型中,目标函数为最小化基准调度计划的项目工期和最大化二次资源冲突消除后调度方案的鲁棒性;式(4)为基准调度方案要满足活动优先关系约束;式(5)表示二次资源冲突消除后的调度方案仍要满足优先关系约束;式(6)表示消除优先关系冲突;式(7)表示输入缓冲要与该条关键链末节点工作连续执行,不能断开;式(8)表示基准调度计划和二次调度计划均要满足可更新资源约束,其中  $A(t)$  代表  $t$  时刻正在执行的活动集合;式(9)表示非关键链的输入缓冲对于各类资源的消耗量等于该非关键链的末节点资源消耗量。

### 3 混合差分进化算法

当仅考虑最小化项目工期目标值时, RB-CCPSP 将退化为以最小化工期为目标的经典 RCPSP,而经典 RCPSP 已经被证明为 NP-hard 问题<sup>[21]</sup>,因此 RB-CCPSP 也为 NP-hard 问题。此外,相比经典 RCPSP, RB-CCPSP 的复杂性还体现在两方面:(1)在评估关键链进度计划鲁棒性时,需要考虑缓冲插入位置,缓冲尺寸,关键链和非关键链识别,二次冲突消除过程;(2)相比单目标优化问题 RCPSP,目标函数数量的增加也提高了 RB-CCPSP 的求解难度。综上所述, RB-CCPSP 可以看作经典 RCPSP 在调度鲁棒性和目标函数上的拓展,求解难度更高,因此可以归约为一类比经典 RCPSP 更复杂的 NP-hard 问题。基于此,本文将差分进化算法(differential evolution algorithm, DE)与变邻域搜索算法(variable neighborhood search, VNS)结合,设计混合差分进化算法(hybrid differential evolution algorithm, HDE)求解 RB-CCPSP。

#### 3.1 编码与解码

采用 Debels 等<sup>[22]</sup>提出的标准随机键编码(standardized random key, SRK)消除冗余搜索。为不错过最优解,采用串行进度生成机制解码生成基准调度计划和基准调度工期,进一步识别 CC 和 NC 并消除二次资源冲突,从而获得二次调度计划鲁棒性。

#### 3.2 变异和交叉算子

假设算法种群规模为  $PS$ ,最大进化代数  $G$ ,第  $t$  代种群中第  $i$  个个体为  $X_i^t$ 。随机初始化种群后对目标个体  $X_i^t = (X_{i1}^t, X_{i2}^t, \dots, X_{ij}^t)$  按式(11)进行变异操作生成变异矢量  $V_i^t$ ,其中  $a, b, c \in \{1, 2, \dots,$



$PS\}$  且  $a, b, c, i$  互不相等,  $F$  为缩放因子。

$$V_i = X_a^t + F(X_b^t - X_c^t) \quad (11)$$

在变异操作完成后,如式(12)所示,对  $V_i$  和  $X_i^t$  进行二项式交叉产生试验矢量  $U_i^t$ 。其中,  $CR \in [0, 1]$  为交叉概率值。 $rand(j)$  为随机生成的  $[0, 1]$  区间内第  $j$  个随机数,  $rc(i)$  为个体  $i$  在区间  $[2, J-1]$  中随机抽取的正整数,用于保证试验矢量  $U_i^t$  至少有一个维度分量来自于变异矢量  $V_i$ 。

$$u_{ij}^t = \begin{cases} v_{ij}^t, & rand(j) \leq CR \text{ 或 } j = rc(i) \\ x_{ij}^t, & \text{else} \end{cases} \quad (12)$$

### 3.3 选择操作

采用一对一贪婪选择操作,从目标个体和与之对应的试验矢量中选择较优个体进入下一代。当二者不存在相互支配关系时,设计如下选择算子。

引入非支配解集合  $ND$  存储进化至当前代的所有非支配解,  $ND$  集合中的解元素按照  $f_j$  的大小升序排列,用  $|ND|$  表示  $ND$  中元素个数。同时,引入相对质量系数  $n(X)$  表示  $ND$  中可以支配解  $X$  的非支配解数量;相反,若被  $X$  支配,则  $n(X)$  为负数。显然  $n(X)$  的值越小,解  $X$  越优。此时,选择  $n(X_i^t)$  和  $n(U_i^t)$  中较小者进入下一代。当  $n(X_i^t) = n(U_i^t) = 0$  时,采用拥挤距离  $Cd$  进一步测度其相对质量。以  $S_l$  表示  $ND$  集合中的第  $l$  ( $1 \leq l \leq |ND|$ ) 个解,则  $S_l$  的拥挤距离计算公式如式(13)所示。当  $n(X_i^t) = n(U_i^t) > 0$  时,说明二者均比  $ND$  中解的质量差,为提高种群多样性,选择  $U_i^t$  进入下一代。

$$Cd(S_l) = \begin{cases} +\infty & l = 1 \text{ 或 } |ND| \\ \frac{f_j(S_{l+1}) - f_j(S_{l-1})}{f_j(S_{|ND|}) - f_j(S_1)} & \\ + \frac{RM(S_{l+1}) - RM(S_{l-1})}{RM(S_{|ND|}) - RM(S_1)} & \\ l = 2, 3, \dots, |ND| - 1 \end{cases} \quad (13)$$

### 3.4 非支配解集 $ND$ 的更新

分析  $ND$  性质可知,若存在解  $S_i \notin ND$  的相对质量系数  $n(S_i) \leq 0$ ,则其必然不会被  $ND$  中的元素支

配。因此,本文更新  $ND$  操作为:将所有满足  $n(U_i^t) \leq 0$  的个体插入  $ND$  并删除  $ND$  中被新进入个体支配的解。

### 3.5 局部搜索策略

全局搜索结束后,设计 VNS 对优势个体进行局部搜索,旨在通过提高解的鲁棒性提升解的质量。

#### 3.5.1 邻域解的生成

邻域解的生成是 VNS 的关键环节。本文设计 Insert 和 Swap 邻域搜索算子如下。

插入操作  $Insert(\pi, i, n)$  是指将调度计划  $\pi$  中活动  $i$  随机键值  $SRK(i)$  变换为  $n$ ,在标准随机键编码中,所有活动按照活动最早开始时间的先后顺序由小到大确定随机键值。当  $n < SRK(i)$  时,相当于将  $i$  插入了原始位置之前的某个位置,当  $n > SRK(i)$  时,相当于将  $i$  插入了原始位置之后的某一位置。

交换操作  $Swap(\pi, i, l)$  是指将调度计划  $\pi$  中活动  $i, l$  的随机键值交换。

#### 3.5.2 局部搜索对象的确定

在确定了邻域解的生成策略以后,需要确定局部搜索活动对象。RB-CCPSP 中,调度方案的鲁棒性取决于输入缓冲和项目缓冲。对于确定的调度计划,输入缓冲的尺寸无法改变。而项目缓冲的消耗主要是由于关键链的中断,因此如何减小关键链的中断对于增大项目缓冲十分关键。而关键链中断时,通常可用性中断时段内的资源负荷相对较小,可以在二次冲突消除过程中非关键活动移动至该时段内进行,增大后续非关键链的自由时差,缩小项目缓冲的消耗。因此,本文选择最早可用中断之后开始的第一个非关键工作作为进行局部搜索对象  $i$ 。如图 3 所示,以项目 1 为例,选择活动 18 进行局部搜索。当将其 Insert 活动 7 之后时,  $ff_{NC_3}$  增加 4 个单位,  $PB$  消耗减少 4 个单位,增大了调度方案鲁棒性。

#### 3.5.3 调度提升

在确定工作  $i$  和邻域解空间之后,进入可变邻域搜索阶段。以  $k$  表示邻域解个数,邻域搜索阶段

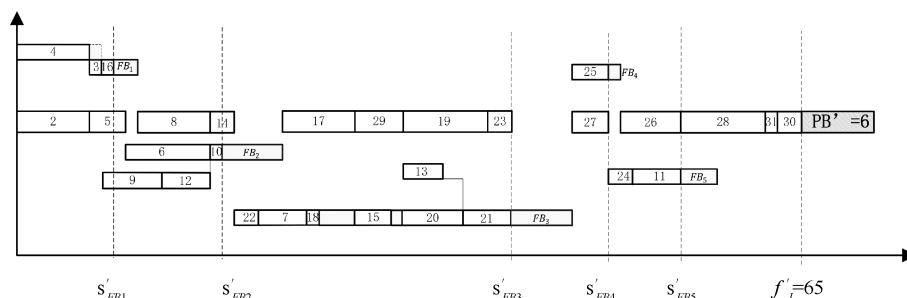


图 3 采用 VNS 局部搜索后的二次调度计划

Figure 3 Secondary scheduling after VNS



包含 Variable Neighborhood Descent (VND) 和 Shake 两个阶段。以  $V(\pi)$  表示调度方案  $\pi$  可用中断个数,以  $v(\pi)$  表示该调度方案进行 VNS 的次数。针对 RB-CCPSP,局部搜索过程如下。

VND:对于  $ND$  中  $v(\pi) < V(\pi)$  的调度方案  $\pi$ ,选取第  $v$  个可用中断后首个开始非关键工作作为局部搜索操作对象,生成邻域解,更新  $v(\pi) = v(\pi) + 1$ 。以  $k = 1$  开始,产生一新的调度方案  $\pi'$ ,确定  $f_j(\pi')$  和  $RM(\pi')$ 。判断  $\pi'$  与  $\pi$  的支配关系,若  $\pi > \pi'$ ,则继续下一个邻域搜索,令  $k = k + 1$ 。若  $\pi' > \pi$ ,则首先将  $\pi'$  插入  $ND$  中,更新  $v(\pi') = 1$ ,并删除  $\pi'$  支配的解,同时,令  $k = 1$ ,可变邻域搜索由  $\pi$  转移到  $\pi'$ 。若  $\pi'$  与  $\pi$  无相互支配关系,令  $k = k + 1$ ,并根据  $\pi'$  与  $ND$  中解的支配关系更新  $ND$ 。

Shake:若对于某一解  $\pi$  存在  $v(\pi) \geq V(\pi)$ ,说明该解已达到了局部最优,需要对其进行扰动已逃离局部最优。随机选择一关键活动进行局部搜索,并令  $v(\pi) = v(\pi) + 1$ 。

对于某一解  $\pi$  当其存在  $v(\pi) = V(\pi) + 1$  或者其搜索的邻域解达到最大邻域解数量时,停止对该解进行局部搜索。

综上所述,HDE 算法的整体流程如图 4 所示。

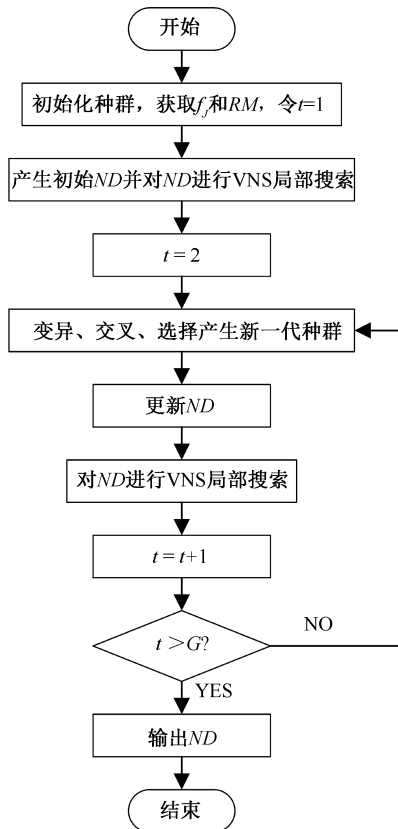


图 4 HDE 算法流程图

Figure 4 Framework of hybrid differential evolution algorithm

## 4 数值实验

从模型和算法两方面验证 RB-CCPSP 和 HDE 的有效性。本文所有测试均采用 Microsoft Visual Studio 2017 进行编程,所有算法的测试均在处理器为 Intel Core i7,主频为 3.20 GHz,内存为 8 G 的 PC 机上进行。

### 4.1 模型有效性实验设计

#### 4.1.1 实验设置

本部分主要验证 RB-CCPSP 的有效性。相对于传统的关键链 (CCPM),本文基于 rRFN 识别关键链和非关键链,同时设计了消除二次资源冲突的 SRCEP 机制。为证明基于 rRFN 识别关键链算法和 SRCEP 机制的有效性,设计了三组实验进行分析,如表 4 所示。

表 4 模型有效性实验设置

Table 4 Experiment for verifying the validity of the model

| 实验名称                                  | 实验设置             |      |          |
|---------------------------------------|------------------|------|----------|
|                                       | CC 和 NC 识别<br>传统 | rRFN | 二次<br>冲突 |
| 传统关键链项目调度模型 (CCPM)                    | ✓                | ×    | ×        |
| 鲁棒性关键链项目调度模型 (rCCPSP)                 | ×                | ✓    | ×        |
| 考虑消除二次资源冲突的鲁棒性双目标关键链项目调度模型 (RB-CCPSP) | ×                | ✓    | ✓        |

以 PSPLIB 中 J30 算例集作为测试集问题,首先采用分支定界算法计算出每个算例的最优工期及其对应的调度计划,以最优调度计划作为基准调度计划,分别进行 CCPM, rCCPSP 和 RB-CCPSP 三组实验,获得对应的二次调度计划。针对每组实验获得的二次调度计划,分别进行 1000 次蒙特卡洛仿真实验<sup>[23]</sup>。假设每个活动的活动时长服从右偏对数正态分布<sup>[24]</sup>,以基准算例中的活动时长为分布的期望值,对应正态分布的标准差 0.3, 0.6, 0.9, 以模拟在不同水平的不确定情形下调度计划的鲁棒性。在仿真过程中,按照接力赛机制安排活动开始时间,以二次调度计划中活动开始时间赋予每项活动一个优先级,开始时间越早,活动的优先级越高。当关键活动与非关键活动处于同一优先级时,优先安排关键活动。

以 CCPM 为参照组,主要从平均工期改善度  $I_{AM}$ 、活动开始时间偏差平均值改善度  $I_{AD}$ 、关键活动开始时间偏差平均值改善度  $I_{AD-C}$  和按期完工率改善度  $I_{RS}$  四方面对 rCCPSP、RB-CCPSP 进行评价,以 rCCPSP 为例,具体指标如式 (14) 所示,其中,  $index$  分别为  $I_{AM}$ 、 $I_{AD}$ 、 $I_{AD-C}$  和  $I_{RS}$ 。

$$I_{\text{index}} = \frac{\text{index}(\text{CCPM}) - \text{index}(\text{rCCPSP})}{\text{index}(\text{CCPM})} \times 100\% \quad (14)$$

#### 4.1.2 实验结果分析

以 CCPM 模型为参照,得到 rCCPSP、RB-CCPSP 在  $I_{AM}$ 、 $I_{AD}$ 、 $I_{AD,C}$  和  $I_{RS}$  四个指标上的实验结果如表 5 所示。从表 5 可以看出依据 RB-CCPSP 得到的调度计划在四个指标均得到了最大的改善。较小的平

均工期和较高的按期完成率可以提高项目按期完工的概率,较低的活动开始时间偏差平均值能减小调度计划不断调整的概率,同时较低的关键活动开始时间偏差说明非关键活动延期对于关键活动的影响较小,符合关键链项目进度管理的初衷。因此,本文所采用的 RB-CCPSP 在项目实施过程中更有利于保证调度计划的平稳执行,同时也可以提高项目按期完工的概率。

表 5 模型有效性实验结果对比  
Table 5 Comparison of rCCPSP and RB-CCPSP

| $\sigma$ | rCCPSP   |          |            |          | RB-CCPSP |          |            |          |
|----------|----------|----------|------------|----------|----------|----------|------------|----------|
|          | $I_{AM}$ | $I_{AD}$ | $I_{AD,C}$ | $I_{RS}$ | $I_{AM}$ | $I_{AD}$ | $I_{AD,C}$ | $I_{RS}$ |
| 0.3      | 2.54%    | 4.10%    | 15.98%     | 1.03%    | 4.84%    | 7.62%    | 24.34%     | 2.63%    |
| 0.6      | 5.09%    | 8.51%    | 23.07%     | 8.07%    | 7.48%    | 13.61%   | 38.90%     | 15.07%   |
| 0.9      | 7.22%    | 14.71%   | 27.25%     | 8.03%    | 13.40%   | 19.35%   | 35.32%     | 20.36%   |

同时,在表 5 中,我们可以看出 rCCPSP 模型在低、中、高标准差下  $I_{AM}$ 、 $I_{AD}$ 、 $I_{AD,C}$  和  $I_{RS}$  四个指标值均为正数,说明基于 rRFN 进行关键链和非关键链识别能够寻找出对项目工期制约最严重的关键活动,在项目实施过程中更可能保证项目计划的平稳进行。同理, RB-CCPSP 模型在各项指标上均大于 rCCPSP 模型,在关键链项目调度模型中加入 SRCEP 机制能更贴合关键链项目管理实施的实际情形,考虑二次资源冲突获得的调度计划具有更高的鲁棒性。

#### 4.2 算法有效性实验设计

为衡量本文改进多目标差分进化算法的有效性,本部分通过将 HDE 算法与当前最先进的多目标进化算法 NSGA-II、标准差分进化算法 DE 及 CPLEX 的表现进行分析比较。

##### 4.2.1 指标设计

以  $P(a)$  代表  $a$  算法对应的帕累托集合,为评价  $P(a)$  的优劣,引入参照集  $RS$ 。以  $RS$  表示精确的帕累托前沿或者近似最优帕累托前沿。由于 RB-CCPSP 的复杂性,难以获得精确的最优帕累托前沿,本文将所有非支配解中不被其他算法支配的解组成参照集  $RS$ ,具体操作为首先设置  $RS = P(\text{NSGA-II}) \cup P(\text{DE}) \cup P(\text{HDE})$ ,然后消除其中被其他解支配的解。基于  $RS$  设计如下评价指标。

(1) 非支配解个数  $N_{nd}$ :指最终集合  $P(a)$  中非支配解的数量,即  $N_{nd} = |P(a)|$ ,集合中非支配解的数量越多,可供选择的方案越多,算法搜索能力越好。

(2) 非支配比率  $R_{nd}$ :指算法对应的 pareto 集合中不受其他算法非支配解支配的比例,以集合

$G(a) = \{x \in P(a) \mid \forall y \in RS, G(a) = \{x \in P(a) \mid \forall y \in RS, y \text{ 不支配 } x\}$  表示算法  $a$  对应的 pareto 集合中不受其他算法非支配解支配的解集合,则  $R_{nd} = |G(a)| / |P(a)|$ 。显然,算法非支配解比例  $R_{nd}$  越大,说明算法求解质量越高。

(3) 与参照集平均距离  $AD$ :指参照集各点与算法对应的帕累托前沿的距离平均值。平均距离越小,算法的收敛性越好。

$$AD = \frac{\sum_{y \in RS} \min_{x \in P(a)} \text{dis}(x, y)}{|RS|} \quad (15)$$

其中,  $\text{dis}(x, y)$  为  $x, y$  两点的欧式距离。

(4) 鲁棒性改进比例  $I_{RM}$ :指基准调度均为最短工期条件下, HDE 算法求得调度计划相比 CPLEX 调度计划在鲁棒值上的改进程度。改进比例越大,说明 HDE 相比 CPLEX 求得解的质量越优。

$$I_{RM} = \frac{RM_{HDE} - RM_{CPLEX}}{RM_{CPLEX}} \times 100\% \quad (16)$$

(5) 最短工期偏差  $DEV$ :指以 HDE 算法求得解中最短工期  $f_j^{HDE}$  与基准调度计划工期下界  $f_j^{LB}$  的偏差程度。偏差越小,说明 HDE 对边界解的搜索能力越强,解的质量越高。

$$DEV = \frac{f_j^{HDE} - f_j^{LB}}{f_j^{LB}} \times 100\% \quad (17)$$

##### 4.2.2 实验结果分析

首先,采用  $N_{nd}$ 、 $R_{nd}$  和  $AD$  对 HDE、DE 和 NSGA-II 算法进行对比分析。本文令 NSGA-II 的交叉概率  $P_c = 0.8$ ,变异概率  $P_m = 0.05$ 。为保证算法执行环境的公平,三种算法的种群规模均为  $PS = 50$ ,最大进化代数  $G = 100$ ,总共获得 5000 个调度计划。DE 和 HDE 算法的缩放因子  $F = 1.25$ ,交叉概率值  $CR =$

0.6。本文在基础实验中发现,当交叉概率  $F = 1.25$ ,  $CR = 0.6$  时,算法产生的非支配解数量更多,解的质量更优。NSGA-II、DE 以及 HDE 算法的比较结果如表 6 所示。

表 6 NSGA-II、DE 以及 HDE 算法性能比较  
Table 6 Comparison of NSGA-II, DE and HDE

|      | NSGA-II  |          |        | DE       |          |        | HDE(本文)  |          |        |
|------|----------|----------|--------|----------|----------|--------|----------|----------|--------|
|      | $N_{nd}$ | $R_{nd}$ | $AD$   | $N_{nd}$ | $R_{nd}$ | $AD$   | $N_{nd}$ | $R_{nd}$ | $AD$   |
| J30  | 11.4     | 16.70%   | 0.8764 | 12.34    | 8.95%    | 0.9655 | 14.17    | 84.52%   | 0.1621 |
| J60  | 16.25    | 12.07%   | 1.4430 | 17.87    | 12.05%   | 1.1777 | 19.26    | 74.63%   | 0.2889 |
| J90  | 19.93    | 26.87%   | 2.1774 | 20.34    | 5.00%    | 1.7309 | 22.89    | 78.13%   | 0.5309 |
| J120 | 24.76    | 7.63%    | 1.7626 | 24.23    | 2.27%    | 1.9949 | 27.38    | 87.82%   | 0.5376 |

由表 6 可以看出 HDE 算法在 J30、J60、J90、J120 基准算例的表现均优于 NSGA-II、DE 两种算法。在不同规模的算例中,  $N_{nd}(HDE)$  均大于 NSGA-II、DE 算法的非支配解个数  $N_{nd}$ , 说明 HDE 算法具有更好的搜索能力, 能够获得更多的帕累托解; 在  $R_{nd}$  方面, HDE 算法的非支配比率  $R_{nd}(HDE)$  始终大幅领先于 NSGA-II、DE 算法, HDE 算法在 J30 算例集中的  $R_{nd}(HDE)$  达到最大值 84.52%, 在 J60、J90 和 J120 算例集中,  $R_{nd}$  随着算例集规模的扩大而不断增大, 说明 HDE 相对于 NSGA-II、DE 两种算法对于大规模算例求解质量更好。在  $AD$  方面, HDE 算法的  $AD$  值始终保持在 0.6 以下, 低于 NSGA-II、DE 算法的  $AD$  值, 说明 HDE 算法具有较好的收敛性, NSGA-II 算法与 DE 算法在算法收敛性上差别不大。

对比分析 HDE 算法和 DE 算法的表现可以发现, HDE 算法相对于 DE 算法在非支配解个数  $N_{nd}$ 、非

支配比率  $R_{nd}$ 、与参照集平均距离  $AD$  三个指标上的表现都得到了明显的提升。除 J30 算例集外, HDE 算法在各项指标上的提升随算例规模的增大而增大, 在 J120 算例集达到最大。这说明随着算例规模的不断提升, VNS 局部搜索算法的局部搜索效果不断提升, 充分证明了 VNS 局部搜索算法的有效性。

其次, 为进一步说明 HDE 算法的优越性, 采用 NSGA-II、DE 以及 HDE 算法对 PSPLIB 数据库 J30 算例集中 J301-1 算例求得帕累托前沿如图 5 所示。由图 5 可以直观地看出 HDE 算法的性能明显优于 NSGA-II 算法和 DE 算法, 其 pareto 解相比其余两种算法处于明显的支配地位。随着基准工期的增加, 二次资源冲突消除后的调度方案鲁棒性不断增加, 说明随着基准调度工期的增加, 调度方案在项目实施过程中的稳定性不断增加。同时, 随着基准工期的增大, 帕累托解鲁棒性斜率逐渐变小, 说明随着工期变大, 调度方案鲁棒性增加变平缓。

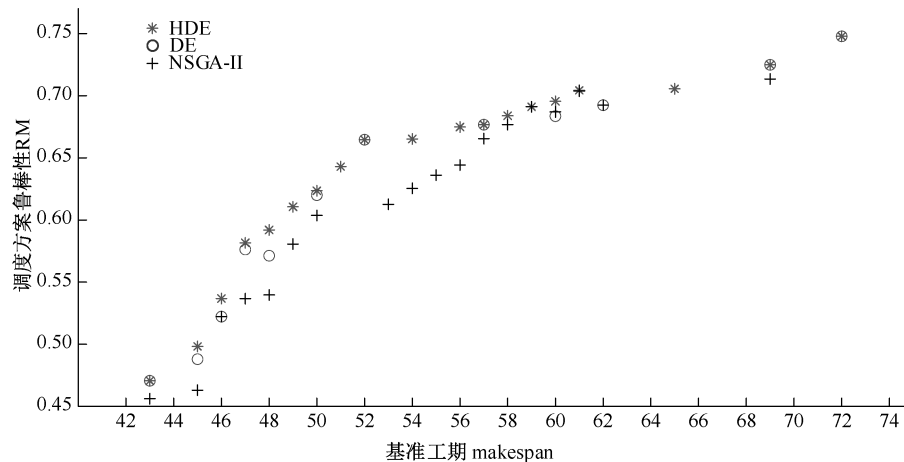


图 5 NSGA-II、DE 和 HDE 对应 pareto 前沿图  
Figure 5 Pareto frontier of NSGA-II, DE and HDE

再次, 基于 J30 算例, 采用 CPLEX 解得工期最短的调度计划, 通过关键链识别, 二次资源冲突消除等过程确定该进度计划下的鲁棒性值  $RM_{CPLEX}$ 。将  $RM_{CPLEX}$  与 HDE 算法求得相同工期进度计划下的鲁棒性值  $RM_{HDE}$  对比, 求得  $I_{RM}$  及实验结果如表 7 所

示。由表 7 可以看出, 在工期最短的基准调度计划上, HDE 在 87.18% 算例上优于 CPLEX, 且最大改进比例达到 107.67%, 这说明相比 CPLEX, HDE 算法能获得更优解。

最后, 以 J30、J60、J90 和 J120 为测试算例, 分析



HDE 在最短工期偏差  $DEV$  上的表现如表 8 所示。显然,对于 J30、J60 和 J90, HDE 算法可以针对大部分算例求得工期最短解,且平均偏差在 3% 以下;对于 J120 算例,虽然仅对 32.14% 算例求得工期最短

解,但是平均偏差仅为 5.06%, 仍然处于较低水平。在需要广泛搜索所有非支配解的前提下,对于边界解高效的搜索效率证明了 HDE 算法具有较高的搜索能力。

表 7 最小工期调度计划下 HDE 相比 CPLEX 鲁棒性改进结果表 (J30)

Table 7 Comparison between HDE and CPLEX in the RM of schedule with shortest makespan for J30

| $I_{RM}$ | 改善程度   |        |        |         |          |       | $I_{RM}^{Mean}$ | $I_{RM}^{Max}$ |
|----------|--------|--------|--------|---------|----------|-------|-----------------|----------------|
|          | <0%    | 0%     | 0%~20% | 20%~50% | 50%~100% | >100% |                 |                |
| 算例占比     | 12.82% | 33.33% | 30.77% | 12.82%  | 7.69%    | 2.56% | 18.20%          | 107.67%        |

表 8 HDE 算法求得最短工期与工期下界偏差统计表

Table 8 The deviation between the shortest makespan obtained by CPLEX and the lower bound of makespan

| 算例比例 | 偏差程度   |        |        |        |        | $DEV^{Mean}$ | $DEV^{Max}$ |
|------|--------|--------|--------|--------|--------|--------------|-------------|
|      | 0      | 0%~2%  | 2%~5%  | 5%~10% | >10%   |              |             |
| J30  | 70.83% | 18.75% | 6.25%  | 4.17%  | 0.00%  | 0.76%        | 7.06%       |
| J60  | 66.67% | 0.00%  | 16.67% | 16.67% | 0.00%  | 1.96%        | 9.82%       |
| J90  | 60.42% | 2.08%  | 12.50% | 8.33%  | 14.58% | 2.75%        | 12.82%      |
| J120 | 32.14% | 3.57%  | 21.43% | 25.00% | 17.86% | 5.06%        | 17.75%      |

综上所述, NSGA-II 算法、DE 算法在求解 RB-CCPSP 时,并没有明显的优劣关系,而本文设计的 HDE 算法相对于 NSGA-II 算法 DE 算法在所有规模的算例中均有显著优势。同时,通过与 CPLEX 及边界解的对比,证明 HDE 搜索能力较强,因此采用本文所设计的 HDE 算法求解 RB-CCPSP 具有显著的优越性。

## 5 结论

本文针对 CCM 中棘手的二次资源冲突困境,设计了基于局部重调度的鲁棒性二次资源冲突消除策略,并基于消除策略设计了考虑可用中断和不稳定集合的鲁棒性测度指标;在此基础上,以最小化项目工期和最大化鲁棒性为双目标构建 RB-CCPSP 模型;同时,针对传统 CCM 关键链和非关键链识别的不足,设计了基于 rRFN 的关键链和非关键链识别方法;针对 RB-CCPSP 问题,设计融入 VNS 局部搜索算法的 HDE 算法;最后,通过大规模数据实验分别验证了 RB-CCPSP 模型和 HDE 算法的有效性。实验结果表明:(1)考虑二次资源冲突的鲁棒性关键链项目调度模型在项目实施过程中更有利于保证调度计划的平稳执行,同时也可以提高项目按期完工的概率;(2)采用本文设计 HDE 算法能有效的求解双目标关键链项目调度模型,同时 VNS 局部搜索算法能有效地改善解的质量。

虽然目前学者们对 CCM 进行了大量的研究,但是在实际应用中仍然存在一些障碍。例如静态的缓冲设置方法难以满足项目具体实施的要求,随着项目的进行,缓冲尺寸应该随着项目执行的过程动

态调整。因此,本文下一步的研究将集中于项目实施过程中的缓冲动态调整和项目实时监控方向。

## 参考文献

- [1] 王艳婷,何正文,刘人境. 随机多模式项目摄性调度与反应性调度的权衡[J]. 管理工程学报, 2018, 32(4): 158-166.  
WANG Y T, HE Z W, LIU R J. Trade-off between stochastic multi-mode proactive scheduling and reactive scheduling[J]. Journal of Industrial Engineering and Engineering Management, 2018, 32(4): 158-166.
- [2] SERVIRANCKX T, VANHOUCKE M. Strategies for project scheduling with alternative subgraphs under uncertainty: Similar and dissimilar sets of schedules[J]. European Journal of Operational Research, 2019, 279(1): 38-53.
- [3] 王凌,郑环宇,郑晓龙. 不确定资源受限项目调度研究综述[J]. 控制与决策, 2014, 29(4): 577-584.  
WANG L, ZHENG H Y, ZHENG X L. Survey on resource-constrained project scheduling under uncertainty[J]. Control and Decision, 2014, 29(4): 577-584.
- [4] GOLDRATT E M. Critical Chain: A business novel[M]. Great Barrington, MA: North River Press, 1997.
- [5] HERROELEN W, LEUS R. Project scheduling under uncertainty: Survey and research potentials[J]. European Journal of Operational Research, 2005, 165(2): 289-306.
- [6] 张静文,乔传卓,刘耕涛. 基于鲁棒性的关键链二次资源冲突消除策略[J]. 管理科学学报, 2017, 20(3): 106-119.  
ZHANG J W, QIAO C Z, LIU G T. Strategy to eliminate the second resource conflicts in critical chain method based on robustness[J]. Journal of Management Sciences in China, 2017, 20(3): 106-119.

- [7] NEWBOLD R C. Project management in the fast lane—applying the theory of constraints[M]. Boca Raton: The St Lucie Press, 1998.
- [8] TUKELO I, ROM W O, EKSIÖGLU S D. An investigation of buffer sizing techniques in critical chain scheduling[J]. *European Journal of Operational Research*, 2006, 172(2): 401-416.
- [9] ZHANG J, WAN D. Integrated buffer monitoring and control based on grey neural network[J]. *Journal of the Operational Research Society*, 2019, 70(3): 516-529.
- [10] MARTENS A, VANHOUCKE M. A buffer control method for top-down project control[J]. *European Journal of Operational Research*, 2017, 262(1): 274-286.
- [11] 张静文, 刘耕涛. 基于鲁棒性目标的关键链项目调度优化[J]. *系统工程学报*, 2015, 30(1): 135-144.  
ZHANG J W, LIU G T. Critical chain project scheduling problem with the robust objective[J]. *Journal of Systems Engineering*, 2015, 30(1): 135-144.
- [12] PENG W L, HUANG M. A critical chain project scheduling method based on a differential evolution algorithm[J]. *International Journal of Production Research*, 2014, 52(13): 3940-3949.
- [13] 田旻, 张光军, 刘人境. 基于改进关键链方法的MRCPSP的鲁棒性优化[J]. *系统工程学报*, 2019, 34(2): 277-288.  
TIAN M, ZHANG G J, LIU R J. Robust optimization of the MRCPSP based on improved critical chain method[J]. *Journal of Systems Engineering*, 2019, 34(2): 277-288.
- [14] TIAN M, LIU R J, ZHANG G J. Solving the resource-constrained multi-project scheduling problem with an improved critical chain method[J]. *Journal of the Operational Research Society*, 2020, 71(8): 1243-1258.
- [15] GHAFARI M, EMSLEY M W. Current status and future potential of the research on critical chain project management[J]. *Surveys in Operations Research and Management Science*, 2015, 20(2): 43-54.
- [16] VAN DE VONDER S, DEMEULEMEESTER E, HERROELEN W, et al. The trade-off between stability and makespan in resource-constrained project scheduling[J]. *International Journal of Production Research*, 2006, 44(2): 215-236.
- [17] ZHAO Y, CUI N F, TIAN W D. A two-stage approach for the critical chain project rescheduling[J]. *Annals of Operations Research*, 2020, 285(1-2): 67-95.
- [18] 梁洋洋, 崔南方. 基于资源流网络优化的鲁棒性项目调度[J]. *系统管理学报*, 2020, 29(2): 335-345.  
LIANG Y Y, CUI N F. Robust project scheduling based on optimization of resource flow network[J]. *Journal of Systems & Management*, 2020, 29(2): 335-345.
- [19] VAN DE VONDER S, DEMEULEMEESTER E, HERROELEN W. Proactive heuristic procedures for robust project scheduling: An experimental analysis[J]. *European Journal of Operational Research*, 2008, 189(3): 723-733.
- [20] ARTIGUES C, MICHELON P, REUSSER S. Insertion techniques for static and dynamic resource-constrained project scheduling[J]. *European Journal of Operational Research*, 2003, 149(2): 249-267.
- [21] BLAZEWICZ J, LENSTRA J K, KAN A H G R. Scheduling subject to resource constraints: Classification and complexity[J]. *Discrete Applied Mathematics*, 1983, 5(1): 11-24.
- [22] DEBELS D, DE REYCK B, LEUS R, et al. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling[J]. *European Journal of Operational Research*, 2006, 169(2): 638-653.
- [23] BALLESTIN F, LEUS R. Resource-constrained project scheduling for timely project completion with stochastic activity durations[J]. *Production and Operations Management*, 2009, 18(4): 459-474.
- [24] TRIETSCH D, MAZMANYAN L, GEVORGYAN L, et al. Modeling activity times by the Parkinson distribution with a lognormal core: Theory and validation[J]. *European Journal of Operational Research*, 2012, 216(2): 386-396.

## Robust and bi-objective critical chain project scheduling optimization with a second resource conflicts eliminating policy

TIAN Baofeng, ZHANG Jingwen\*, SHI Zhiyao

(School of Management, Northwestern Polytechnical University, Xi'an 710072, China)

**Abstract:** In the context of economic globalization and the rapid development of information technology, the progression of a project is more complex and unpredictable. Therefore, project management under uncertainty arises. As an efficient project scheduling method, Critical Chain Method (CCM) could reduce the impact of uncertain activity duration on projects by inserting buffers. Specifically, the principle of CCM is to insert feeding buffers wherever the non-critical chain joins the critical chain and to insert the project buffer at the end of the critical chain. However, the existing studies on CCM mostly focused on makespan minimization and the robustness which reflects the ability to cope with the uncertainty was ignored. Meanwhile, to avoid the second resource conflicts dilemma caused by

inserting feeding buffers, most previous studies assumed that the feeding buffer does not consume resources. This assumption is inconsistent with the nature of the buffer and greatly limited the application of the CCM in project management.

In order to solve the aforementioned problems effectively, this paper focuses on the robust optimization for critical chain project scheduling. The present study unfolds as follows. First, a local rescheduling strategy to solve the second resource conflicts dilemma is proposed, and a robustness index according to the rescheduling strategy is designed to measure the robustness of the project schedule. Then, a formulation of Robust and Bi-objective Critical Chain Project Scheduling Problem (RB-CCPSP) is proposed to minimize project duration and maximize robustness simultaneously. Finally, a hybrid differential evolution (HDE) algorithm is put forward to solve the formulation. This paper consists of the following four sections.

In the first section, application steps of the classical CCM are introduced and the formation mechanism of the second resource conflicts dilemma is qualitatively analyzed. Meanwhile, a robust resource flow network is constructed on the basis of heuristic rules, and critical chain and non-critical chains are identified based on this network.

In the second section, a quantitative analysis of the second resource conflicts dilemma is conducted, and the conflicts are classified according to the interference relationship between the feeding buffers and the critical activities. Furthermore, on the basis of above analysis, a two-stage heuristic second resource conflicts eliminating strategy based on local rescheduling is designed, and the robustness measurement of the second scheduling plan is put forward. By taking both the makespan minimization and robustness maximization as objectives, a robust critical chain project scheduling model considering the second resource conflicts dilemma is proposed.

In the third section, to cope with the NP-hardness and bi-objective characteristics of the problem, a hybrid differential evolution algorithm is designed by integrating the differential evolution algorithm and variable neighborhood search (VNS) approach. In the global search, considering the bi-objective characteristics, the selection operator and the non-dominant set updating operator are proposed. Then in the local search, the Insert and Swap operators are put forward.

In the fourth section, the validity of the model and the algorithm proposed in this study are verified through large scale numerical experiments. As for model effectiveness, compared with the classical critical chain project scheduling problem (CCPSP) and the robust critical chain project scheduling problem (rCCPSP), the robust critical chain project scheduling model considering second resource conflicts dilemma is more conducive to the smooth progression of the project and increase the probability of the project being completed on time. As for algorithm effectiveness, compared with NSGA-II and the classical differential evolution algorithm, the HDE algorithm could solve the bi-objective critical chain project scheduling model with better solution quality. In addition, the VNS could improve the solution significantly.

To sum up, the method taking the secondary resource conflicts dilemma into consideration put forward in the present study significantly improves the utility of CCM. Moreover, the robust and bi-objective critical chain project scheduling optimization model proposed in this study could generate different project schedules according to the specific uncertainty conditions. The present study could provide insights into decision-making about project management under uncertainty.

**Key words:** Second resource conflicts; Robust critical chain method; Bi-objective; Differential evolution algorithm

---

**Received Date:** 2021-08-30

**Funded Project:** The National Natural Science Foundation of China (71971173); The Natural Science Basic Research Plan in Shaanxi Province of China (2020JM-146); The Social Science Funds for Interdisciplinary Subject of Northwestern Polytechnical University (21GH031128)

\* Corresponding author