基于鲁棒性的关键链二次资源冲突消除策略①

张静文，乔传卓，刘耕涛
(西北工业大学管理学院，西安 710072)

摘要： 以关键链方法中的二次资源冲突困境为切入点，从鲁棒调度优化角度提出一种解决策略. 首先，采用定量化建模对问题进行数学描述和表达，剖析插入输入缓冲引起二次资源冲突的原理，进而采用场景分析法从复杂的冲突表象中分解出四种基本的冲突场景构成要素. 其次，基于鲁棒调度优化理论，探究各种冲突子问题的有效对策并归类，据此开发出一种消除二次资源冲突的局部重调度启发式协调策略；根据策略设计基于两次调度进程和两类缓冲动态消耗的鲁棒性指标，采用鲁棒性关键链项目调度问题输出鲁棒性最大的调度方案. 再次，设计仿真程序和三个测试指标：项目实际按期完工率、活动开始时间偏差绝对值之和及偏差绝对值的方差；基于 ProGen 随机地生成测试算例集合进行数值实验. 结果发现，以鲁棒性调度方案为依据安排项目的实施过程，三个统计指标值都优于以传统关键链调度方案为依据时相对应的指标值. 结论表明：基于鲁棒调度优化的二次资源冲突消除策略及设计的关键链鲁棒性指标在项目实施中具有较好的稳定性效果.

关键词： 二次资源冲突；局部重调度策略；鲁棒性指标；关键链方法；数值实验
中图分类号： C935；F224.33 **文献标识码：** A **文章编号：** 1007－9807(2017)03－0105－14

0 引 言

经典的项目调度问题(project scheduling problem)研究多属于确定型的[1-2]，然而现实中多数项目在执行中通常面临较高的不确定性和风险；体现到微观层面上，各类不确定因素总是干扰着项目的执行过程：如活动工期延误或提前、资源可用量波动、原材料滞后、新任务到达或取消、交付期发生改变、天气变化等，这些随机因素的出现偏离了预期的假设，严重破坏了原调度方案的最优性甚至可行性[3-4]. 关键链方法(critical chain method，CCM)是 Goldratt 提出的一种较新的项目进度管理技术[5]，设想在初始的基准调度计划中设置缓冲以消除不确定因素对进展过程的扰动，因此一些学者将 CCM 归类为一种不确定型项目调度理论[6]. 为表述方便，本文后续内容中将 Goldratt 提出的关键链雏形称为传统 CCM.

应用 CCM 时，理论上将输入缓冲插入基准调度计划就可获得二次调度计划，且二次调度计划能够吸收活动工期一定程度的波动；然而，当插入输入缓冲时，通常在基准调度计划中引起二次资源竞争冲突，这种现象被称为传统 CCM 中的“二次资源冲突困境”. 目前，此问题已成为制约传统 CCM 在项目管理实践中广泛应用的瓶颈[7-8]. 鲁棒性(robustness)项目调度作为一种不确定型项目管理理论，指进度计划在内外部环境变化时仍保持稳定性的能力或易调整的灵活性. 缓冲机制表明 CCM 也是一种鲁棒性项目调度理论[9]，因此

① 收稿日期：2015－07－19；修订日期：2016－07－14.
基金项目：国家自然科学基金资助项目(71572148)；中国博士后科学基金资助项目(2015M580875；2016T90947)；航空科学基金资助项目(2015ZG53080)；陕西省科学基金资助项目(2015JM7368；2014P23)；西北工业大学研究生创意创新种子基金资助项目(Z2016177). 本文入选“第十三届全国青年管理科学与系统科学学术会议(2015 年，西安)优秀论文.
作者简介：张静文(1976—)，女，陕西韩城人，博士，教授. Email：zhangjingwen@nwpu.edu.cn

"二次资源冲突困境"可归类为鲁棒性项目调度研究中的一个子问题. 对于不确定环境下的项目进度过程,"二次资源冲突困境"具有普遍的现实背景,其解决方案对项目实施过程具有重要的现实意义. 对环境不确定性较大的项目来说,只有鲁棒性较高的进度方案才能有效抵御内外部诸多干扰因素的影响,最大程度地发挥进度计划对项目实施过程的指导作用. 如果在计划阶段未考虑到不确定性因素出现或没有采取有效的预案,那么在随后的项目执行中,计划便会因变化的干扰而频繁地调整,从而失去其有效性进而给组织和协调项目带来混乱. 通过在进度计划中预留缓冲的方式,尽可能地减轻或消除不确定因素的影响,使得项目顺畅地实施从而确保实现预期的项目目标,这是一种常用的提高计划鲁棒性的方法. 因此,"二次资源冲突困境"也是鲁棒性项目调度理论中需要解决的一个理论问题,目前鲁棒性调度已成为不确定型项目进度管理中的一个研究热点[10-11].

近年来,学者对 CCM 的研究集中于模型求解算法、缓冲区尺寸的计算以及关键链的应用等几个方面. Herroelen 等[12] 采用分支定界算法获得关键链的基准调度计划,但是模型求解主要集中于开发高效的启发式智能优化算法方面. 刘士新等[8] 提出一种基于启发式规则的关键链调度方法; Peng 等[13] 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[14] 通过计算资源约束下活动的自由时差来识别关键链. 对于缓冲区尺寸的计算,代表性的方法有 50% 剪切法、根方差法、Tukel 等[14] 提出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) 两种方法. 目前,根方差法、APRT 和 APD 方法被普遍认可,其他一些学者[15-16] 针对不同的情形,提出了改进的缓冲区尺寸计算方法.

活动时差通常是网络计划技术中进度方案拥有灵活性的根源[17],CCM 中的缓冲实质是将部分活动的时差集中放置并统一调度,所以 CCM 中蕴含了鲁棒调度优化思想. 然而,传统 CCM 仅以项目工期最短为目标,对如何设置缓冲仅驻留在概念应用层面,忽略了能体现其重要特质的鲁棒性指标. 虽然后续研究者继承和拓展了传统 CCM,但是由于关键链项目管理的具体实施非常复杂,包括两次调度过程,且第二次调度中的资源冲突困境很难处理,因此很少有学者从定量化建模角度研究 CCM 中设置缓冲机制所产生的鲁棒性效果. 本文作者前期已定性地提出了解决二次资源冲突的一种思路[18],但是对于导致问题的根源及解决方法没有给出定量化的理论表达. 因此,本文采用定量化建模对问题进行数学描述,剖析各种冲突场景并开发出消除二次资源冲突的启发式局部重调度策略,进而对基于策略的关键链调度方案的鲁棒性指标进行验证.

1　关键链方法中的二次资源冲突困境剖析

沿袭经典资源约束型项目调度问题(resource-constrained project scheduling problem, 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 $V=\{1,2,\cdots,J\}$ 为节点(活动) 集合,节点 1 和节点 J 分别表示唯一的开始和唯一的结束两个虚活动. 有向弧 $(i,j)\in E$ 表示活动 i 为活动 j 的紧前活动,活动 j 为活动 i 的紧后活动. 用 $P(j)$ 表示活动 j 的紧前活动集合,$S(j)$ 表示活动 j 的紧后活动集合. 活动执行中不能中断也不能抢先,活动之间为结束 - 开始型的优先关系. 项目实施中共需 K 种可更新资源,第 k 种资源的可用量为 $R_k(1\leqslant k\leqslant K)$; $r_{jk}(1\leqslant j\leqslant J,1\leqslant k\leqslant K)$ 表示活动 j 在单位工期上对第 k 种资源的需求量.

项目调度计划 $S=(s_1,s_2,\cdots,s_J)$,是由每个活动的开始时间构成的开始时间序列. 关键链指在满足资源约束和活动间相互依赖关系下,制约项目工期的最长任务链. 对于某个基准调度计划 S,用 CC 表示关键链上的关键活动集合,用 NC_m 表示某条非关键链 m ($m=1,2,\cdots,M$) 上的非关键活动集合. 用 PB 表示唯一的项目缓冲(project buffer, PB),PB 被置于关键链的末端(项目结束处). 用 $FB^m_{i_mj_m}\{(i_m,j_m)\in E$ 且 $[i_m\in NC_m)\cap$

$(j_m \in CC)]\}$ 表示非关键链 m 的输入缓冲(feeding buffer, FB),有位置和大小(尺寸)两个属性. $FB^m_{i_m j_m}$ 位于非关键链 m 末端的非关键活动 i_m 和 m 汇入关键链入口处的关键活动 j_m 之间,活动 i_m 和活动 j_m 分别称为与 $FB^m_{i_m j_m}$ 紧密关联的非关键活动和关键活动. 对于某个 S 来说,不同的 FB 位于不同的位置处(网络结构中和 S 中的插入处),非关键活动 i_m 和关键活动 j_m 的下标"m"强调了其位置. 用 $ff^c_{i_m}$ 表示非关键活动 i_m 在 S 中的自由时差(free floats under the resource-constrained case). 用 d_j 表示活动 $j(j=1,2,\cdots,J)$ 的期望工期,s_j 和 f_j 分别表示活动 j 在 S 中的开始和结束时间,因此 f_J 表示 S 对应的项目工期. CCM 包括两个阶段:第一阶段,根据活动间的逻辑关系和资源限量,消除活动间的资源冲突(第一次资源冲突)得到基准调度计划 S,识别出 CC 并寻找各条非关键链 $NC_m(m=1,2,\cdots,M)$,确定出各个 $FB^m_{i_m j_m}$ 的位置,进而计算 PB 和各个 $FB^m_{i_m j_m}$ 的尺寸;第二阶段,插入各个 $FB^m_{i_m j_m}$ 和 PB,获得嵌入两类缓冲(FB 和 PB)的二次调度计划 S'. 然而,在第二阶段将各个 $FB^m_{i_m j_m}$ 插入到 S 中时,往往会破坏 S 中原本可行的资源配置状态,导致 S 中发生对有限资源的竞争(即二次资源冲突困境). 已有文献[7-8]也指出了插入 FB 通常会引起二次资源冲突,但是由于二次资源冲突表现形式的复杂性,很少有学者深入探究此问题的根源及解决方案.

传统 CCM 以基准调度计划 S 的 f_J 最短为目标,确定关键链和非关键链,对于如何将 FB 插入 S 仅给出定性地阐述. 现实中,工期最短下的基准调度方案不一定具有好的抵御外界环境变化的鲁棒性,所以传统的关键链概念是一种狭隘意义上的关键链. 据此,本文首先从鲁棒性视角将传统的关键链拓展为广义的关键链概念,即对任何 S 来说,都包含有关键链和非关键链. 在一个可行的调度计划中,制约项目工期的活动构成关键链活动集合,其余的活动构成非关键活动集合. 对于不同的 S,由于关键链不一样,确定出的 FB 位置和尺寸、PB 的尺寸也不同,进而 FB 在 S 中的分布状态(插入位置)也有差异,因此不同的 S 将具有不同鲁棒性. 如何量化不同调度方案的鲁棒性是研究鲁棒性关键链调度的关键和难点.

实际上,消除二次资源冲突是 CCM 在使用过程中最棘手的环节. 为解决此问题,首先需分析插入 FB 引起资源冲突的原因及冲突的类型. 通常将 $FB_{i_m j_m}$ 作为非关键活动 i_m 的紧后活动插入时,将导致插入点之后发生资源冲突,冲突可能涉及与 $FB_{i_m j_m}$ 紧密关联的关键活动 j_m,其他的关键活动及非关键活动,并且将出现三种类型的冲突:第一,仅有优先关系冲突;第二,仅有资源竞争冲突;第三,优先关系和资源竞争同时发生冲突. 发生优先关系冲突时,必须后移非关键活动 i_m 的紧后活动,消除冲突的困难主要体现为资源冲突,因此后面表述中的冲突指资源冲突. 为了更直观地阐述,采用一个简单的例子说明冲突的情形,如图 1 所示. 每个活动所需资源数量均为 1,项目执行共需 $K=2$ 种资源,即 $k=1$ 和 2,资源限量为 $R_1=1$ 和 $R_2=1$. (d_j, r_k) 表示活动 j 的(工期,需求资源的种类). 图 1 中用虚线连接起来的活动链表示关键链.

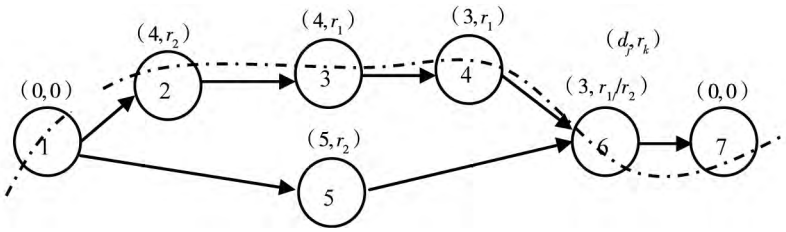

图 1 包含 5 个实活动的 AoN 项目网络

Fig. 1 An AoN project network containing five real activities

图 2 给出了 CCM 的实施过程,其中上部分给出了基准调度计划 S,下部分展示了插入 FB 后引起冲突的情形. 用 Act. 表示活动(activity),S 确定的关键链为 Act. 2→Act. 3→Act. 4→Act. 6,如图 1 中和图 2 中用虚线连接起来的活动链. 非关键链为 Act. 5,在非关键 Act. 5 汇入关键链的入口处 Act. 6 之间插入 $FB_{5,6}$. 当插入 $FB_{5,6}=3$ 时,因 $ff^c_5=1$ 从而将导致在 Act. 5 和 Act. 6 之间发生冲

突,具体可区分两种情形: 第一,当 Act. 6 占用的资源为 r_1 时,在 Act. 5 和 Act. 6 之间仅引起优先关系冲突; 第二,当 Act. 6 占用资源 r_2 时,在 Act. 5 和 Act. 6 之间同时引起优先关系冲突和资源竞争冲突. 因此,为了消除插入 $FB_{5,6}$ 导致的冲突,需要后移 Act. 6 的开始时间 s_6,如图 2 中右下方的虚线所示. 后移 Act. 6 的 s_6 将引起 S 中处于 Act. 5 结束之后的部分需要重新调整.

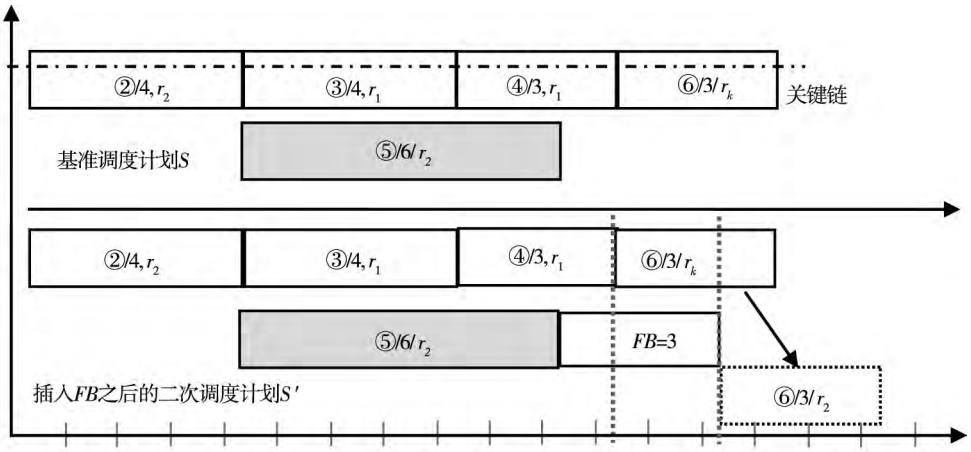

图 2　插入 *FB* 后导致优先关系冲突和(或) 资源冲突

Fig. 2 The precedence and/or the resource conflicts resulted from inserting *FB*

项目网络中通常存在多条非关键链,在每条非关键链汇入关键链的入口处设置各自的 FB,各个 FB 插入 S 中的时刻点不同. 图 3 中给出了二次资源冲突困境的一般化表示,即在 S 中的资源分配状态下各个 FB 预插入 S 中的时刻点. $FB^m_{i_mj_m}$ 在 S 中的初始插入时刻点为 f_{i_m}; $FB^{(m-1)}_{i_{(m-1)}j_{(m-1)}}$ 表示插入时刻点早于 $FB^m_{i_mj_m}$ 且与 $FB^m_{i_mj_m}$ 邻接的 FB,图 3 中处于最左端的 $FB^1_{i_1j_1}$ 的插入时刻点最早. 将多个 FB 依次插入 S 中时,导致的资源冲突表象较难描述. 为更清楚分析冲突的本质,本文将二次资源冲突的情形分为两个递阶的层次,具体如下.

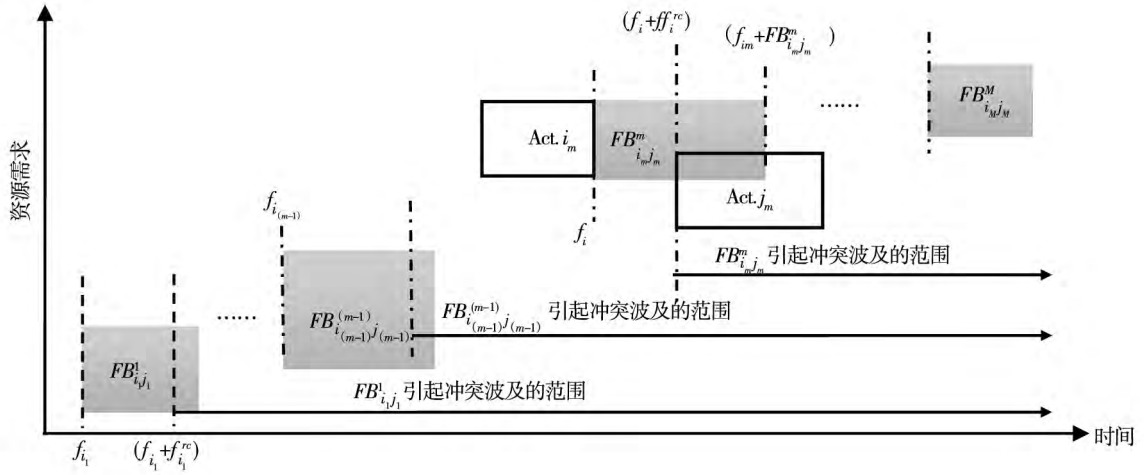

图 3　插入多个 *FBs* 引起的二次资源冲突

Fig. 3 The second resource conflicts arising from adding multiple *FBs*

1) 单个 FB 导致的局部二次资源冲突

首先从简单的情形入手,仅考虑将某一个 FB 插入 S 中. 以插入 $FB^m_{i_mj_m}$ 为例,发生冲突的临界条件为 $FB^m_{i_mj_m} > ff^{rc}_{i_m}$,冲突发生的起始时刻为 $t = f_{i_m} + ff^{rc}_{i_m}$. 若保持 S 中的资源分配状态不变,则在时段 $[f_{i_m} + ff^{rc}_{i_m}, f_{i_m} + FB^m_{i_mj_m}]$ 发生局部资源竞争冲突. 为了消除此时段内的冲突,需要调整后续活动的进度安排,即资源冲突将从时刻 t 开始向后蔓延,

发生在开始时间晚于 t 的活动之间,因此插入单个 $FB^m_{i_m j_m}$ 在 S 中首先会引起局部资源冲突.

2) 多个 FB 导致的多次二次资源冲突之间的影响

如图 3 中所示,当插入 $FB^m_{i_m j_m}$ 时,插入时刻点更早的 $FB^g_{i_g j_g}$($g = 1,2,\cdots,m-1$) 已被依次插入到 S 中,因此在 S 中确定的 $FB^m_{i_m j_m}$ 的初始插入时刻点 f_{i_m} 及之后的资源占用状态已经被改变了($m-1$)次;同时 $FB^m_{i_m j_m}$ 的插入将使得在他之后的各个 $FB^l_{i_l j_l}$($l = m+1,\cdots,M$) 在 S 中确定的初始插入时刻点之后的资源占用状态又被改变一次. 在图 3 中表明,为了消除 $FB^m_{i_m j_m}$ 引起的资源冲突,波及的活动是从冲突起始时刻至项目的结束处. 所以,当向 S 中依次插入多个 FB 时,引起多次资源冲突的情形已不同于仅单独将某个 FB 插入到 S 中时的局部资源冲突场景,这表明多个 FB 导致的二次资源冲突之间有影响. 但是,各个 FB 的属性(位置和尺寸)都是基于 S 中最初的资源分配状态确定的;并且,只要消除一个 FB 导致的冲突,将引起当前插入时刻点之后的所有 FB 插入时刻点的资源占用状态被改变一次,即不断地在调整基准调度计划. 由此可见,解决 CCM 中的"二次资源冲突困境"的确是一项复杂和困难的任务.

2 解决二次资源冲突困境的策略及鲁棒性指标

2.1 消除二次资源冲突的鲁棒性策略

从解决二次资源冲突的角度,以非关键活动 i_m 为参照,本文分析在时刻 t 冲突涉及的活动类型及组合. 在网络结构中,i_m 的紧后活动中包括有关键活动或(和)非关键活动;在 S 中,在 i_m 结束后的最早时刻开始的活动既有 i_m 的紧后活动(关键或非关键),也有 i_m 的非紧后活动(包括关键和非关键). 因此,根据冲突涉及的活动与 i_m 的关系(紧后或非紧后)、是关键或非关键活动这两种区分方式,现将冲突的复杂场景分解为 4 种基本的冲突构成要素.

i) 冲突涉及到了 i_m 的紧后关键活动;

ii) 冲突涉及到了 i_m 的紧后非关键活动;

iii) 冲突涉及到了 i_m 的非紧后关键活动;

iv) 冲突涉及到了 i_m 的非紧后非关键活动;

对实际中的工程项目,尽管冲突发生时涉及的活动类型和数量复杂多样,但是根据上述的 4 种基本构成要素,在最早发生冲突的时段内,涉及到的任何冲突场景(scenario)都由 4 种基本要素中的某一种或多种构成,理论上的组合情形共有 $C_4^1 + C_4^2 + C_4^3 + C_4^4 = 15$ 种,具体总结于表 1.

表 1 插入 *FB* 导致二次资源冲突时涉及的所有场景

Table 1 All scenarios in the second resource conflicts because of inserting *FB*

Scenario1: i	Scenario6: i + iii	Scenario11: i + ii + iii
Scenario2: ii X	Scenario7: i + iv	Scenario12: i + ii + iv
Scenario3: iii	Scenario8: ii + iii	Scenario13: i + iii + iv
Scenario4: iv X	Scenario9: ii + iv X	Scenario14: ii + iii + iv
Scenario5: i + ii	Scenario10: iii + iv	Scenario15: i + ii + iii + iv

其中每种场景下的含义表示上述 4 种基本子情形中的一种或某几种的组合,如"Scenario1: i"表示冲突发生时,仅涉及到非关键活动 i_m 的紧后关键活动;"Scenario: 15: i + ii + iii + iv"表示冲突发生时,同时涉及到 i_m 的紧后关键活动、紧后非关键活动、非紧后关键活动和非紧后非关键活动.

定理 在冲突发生的时段,上述的 Scenario2、Scenario4 和 Scenario9 不存在.

证明 Scenario2、Scenario4 和 Scenario9 这三种场景表示在发生冲突的时段 $[f_{i_m} + ff^{rc}_{i_m}, f_{i_m} + FB^m_{i_m j_m}]$,仅涉及到了非关键活动. 在基准调度计划 S 中,关键链是由一组没有时差的活动构成的活动序列(活动之间没有时间空隙). 如果在冲突的时段,仅仅涉及到了非关键活动,由于非关键活动有时差,表明关键链在这些活动的结束处被中断,这与关键链的定义(本质含义)相背离. 因此定理成立. 证毕.

在表 1 中,Scenario2、Scenario4 和 Scenario9 三种场景后都标注了"X",表示在实际中不存在. 根据上面的分析,冲突的场景数减少到

12 种,探究每种场景下解决资源冲突的策略,从鲁棒性调度优化角度开发出一种解决二次资源冲突的启发式协调策略. 整体思路为: 从项目起始时间开始,根据时间进度从前向后依次插入每个 FB; 当发生冲突时,为了避免频繁地重调度后续的进度计划,同时为了尽可能维持每个 FB 插入时刻点的初始资源分配状态,本文提出仅解决当前 FB 插入处到下一个邻接 FB 插入处之间的冲突,即采用局部重调度的思路,启发式的局部重调度协调策略消除二次资源冲突的过程描述如下.

步骤 1　存储基准调度计划 S 的时间参数,包括各个活动的 s_j 和 f_j ($j=1,2,\cdots,J$) 、项目基准工期 f_J、各个 $FB^m_{i_mj_m}$ ($m=1,2,\cdots,M$) 在 S(初始的资源分配状态) 中的插入时刻点 f_{i_m}. 在插入时刻点较早的 FB 过程中,为消除二次资源冲突会影响到后续 FB 的插入时刻点和尺寸. $FB^m_{i_mj_m}$ 在二次调度过程中被调整后表示为 $FB^m_{i_mj_m}{}'$,用 f'_{i_m} 表示在二次调度过程中,前 $(m-1)$ 个 FB 插入后确定的非关键活动 i_m 的结束时间,也即 $FB^m_{i_mj_m}{}'$ 的插入时刻点. 当 $m=1$ 时,$f'_{i_1}=f_{i_1}$,$FB^1_{i_1j_1}{}' = FB^1_{i_1j_1}$.

步骤 2　根据插入时刻点由小到大,从第一个 FB 开始,依次插入各个 FB,会出现以下三种情形.

情形 1　$FB^m_{i_mj_m}{}' \leqslant ff^c_{i_m}$,则插入 $FB^m_{i_mj_m}{}'$ 不会引起二次资源冲突; 直接转步骤 3; 否则,进入情形 2.

情形 2　根据 f'_{i_m},计算插入 $FB^m_{i_mj_m}{}'$ 可能导致冲突的起始时段为 $[f'_{i_m}+ff^c_{i_m}, f'_{i_m}+FB^m_{i_mj_m}{}']$,找出在此时段内开始的活动集合,用 A 表示; A_t 表示在此时段内的某个时间区间内正在进行的活动集合. 进一步判断

$$r_{i_mk}+\sum_{l\subset A_t} r_{lk} \leqslant R_k;\ t\in [f'_{i_m}+ff^c_{i_m}, f'_{i_m}+FB^m_{i_mj_m}]$$

and $k=1,2,\cdots,K$; 是否满足?

若满足,则插入 $FB^m_{i_mj_m}$ 不会引起二次资源冲突,直接转步骤 3; 否则,进入情形 3.

情形 3　当引起二次资源冲突时,调整当前缓冲插入处至下一个缓冲插入处的活动以消除冲突. 在二次调度过程中,设 $FB^m_{i_mj_m}{}'$ 为当前要插入的 FB,$FB^{(m+1)}_{i_{(m+1)}j_{(m+1)}}{}'$ 为紧接 $FB^m_{i_mj_m}{}'$ 的下一个 FB; $s'_{i_{(m+1)}}$ 为前 $(m-1)$ 个 FB 插入后,与 $FB^{(m+1)}_{i_{(m+1)}j_{(m+1)}}{}'$ 紧密关联的非关键活动 $i_{(m+1)}$ 的开始时间,则 $[f'_{i_m}+ff^c_{i_m}, s'_{i_{(m+1)}}]$ 为当前 $FB^m_{i_mj_m}{}'$ 和下一个 $FB^{(m+1)}_{i_{(m+1)}j_{(m+1)}}{}'$ 插入处确定的时段,重新调度此时段内的活动来消除二次资源冲突. 检查集合 A 中每个活动的性质(是 i_m 的紧后关键活动、紧后非关键活动、非紧后关键活动还是非紧后非关键活动). 检查完毕后 12 种场景被区分为两种情形.

Subcase3.1　对于 Scenario5、7、8、10、11、12、13、14 和 15 共 9 种场景来说,冲突时段内涉及的活动中包含了非关键活动(包括 i_m 的紧后和非紧后的非关键活动). 重新调度时段 $[f'_{i_m}+ff^c_{i_m}, s'_{i_{(m+1)}}]$ 中的活动时,首先保证关键活动的开始时间不变,通过向后移动非关键活动来调整资源需求总量以消除冲突,后移量为 $(FB^m_{i_mj_m}{}' - ff^c_{i_m})$; 同时,若存在与被后移的非关键活动紧密关联的 FB 时,其尺寸应被减少 $(FB^m_{i_mj_m}{}' - ff^c_{i_m})$. 若仅后移非关键活动仍无法解决资源冲突,则需要将冲突起始时刻点之后的所有活动整体后推来消除冲突,解决方法参照 Subcase3.2.

Subcase3.2　对于 Scenario1、3 和 6 这三种场景来说,冲突发生时仅涉及到关键活动(包括 i_m 的紧后和非紧后),此时必须后推关键活动的开始时间,后推量为 $(FB^m_{i_mj_m}{}' - ff^c_{i_m})$,因此消除冲突的策略是整体后移冲突时刻点 $t=f'_{i_m}+ff^c_{i_m}$ 之后的调度计划. 由于 PB 设置在项目结束处用于吸收关键活动的拖延,故此时 PB 被消耗掉 $(FB^m_{i_mj_m}{}' - ff^c_{i_m})$; 当多次后推关键活动导致 PB 被消蚀完后,后续再有关键活动后移表明项目工期增加.

步骤 3　将 $FB^m_{i_mj_m}{}'$ 插入进度计划后,更新进度计划中位于 $FB^m_{i_mj_m}$ 之后的活动的 s'_i、f'_i 及 $FB^x_{i_xj_x}{}'$ ($x=m+1,m+2,\cdots,M$) 的尺寸,更新 PB 的尺寸和项目工期 f'_J.

步骤 4　令 $m=m+1$,判断 $m\leqslant M$,若满足,则转步骤 5; 否则,转步骤 2.

步骤 5　输出嵌入所有 FB 后的二次调度计划 S'(各 $FB^m_{i_mj_m}{}'$ 的尺寸,PB 的尺寸及 f'_J).

2.2　基于二次资源冲突消除策略的鲁棒性指标

根据 CCM 的两次调度进程,主要考虑第二次调度中两类缓冲(FB 和 PB) 的动态消耗过程,针对上述的局部重调度启发式协调策略,一种度量关键链调度方案的鲁棒性指标(robust measure,

RM) 被提出，见式(1)

$$RM_s = \frac{\sum_{m=1}^{M} \max(FB_{i_m j_m}^{m\prime}, ff_{i_m}^{rc})}{f_J'} + \frac{\max[0, \{PB - \max(0, (f_J' - f_J))\}]}{\sum_{j \in CC} d_j} \quad (1)$$

其中$\{(i_m, j_m) \in E$ and $[i_m \in NC_m) \cap (j_m \in CC)]\}$，在式(1)中，$RM_s$表示根据某个 S 和其相应的二次调度计划 S' 获得的鲁棒性指标. RM_s由两部分构成: FB 和 PB.

FB 对鲁棒性的贡献体现在式(1)中的第一部分，即二次调度中各 $FB_{i_m j_m}^{m\prime}$ 之和与项目工期之比，此部分包含了两个信息: 第一，在依次插入各个 FB 时，从前向后调整计划的过程中，消除插入时刻点较早的 FB 导致的冲突时，可能会消耗后续插入时刻点较晚的 FB，这一方面会使后续插入 FB 时导致资源冲突的次数减少; 另一方面表明，当不确定因素出现时，距离其出现时刻点最近的 FB 即可以消除其影响，以尽可能减轻其影响在进度中向后蔓延. 越靠近项目的结束，不确定性因素对项目的影响越小，因此后面的缓冲尺寸变小，本质上是后面的 FB 将消除不确定因素的能力部分地传递给了前面的 FB，通过 FB 之间的相互作用共同吸收项目执行中不同阶段的工期波动. 第二，当某个 $FB_{i_m j_m}^{m}$ 的尺寸小于其紧前 i_m 的 $ff_{i_m}^{rc}$ 时，尽管此时不会出现资源冲突，但是 $ff_{i_m}^{rc}$ 形成的时间缝隙存在，此时采用 $ff_{i_m}^{rc}$ 作为 $FB_{i_m j_m}^{m\prime}$ 的尺寸. PB 对鲁棒性的作用体现在式(1)中的第二部分. PB 是由 S 确定的，但在二次调度中为了解决资源冲突，在某些 FB 插入的时刻点需要后推 FB 之后的关键活动的开始时间，此时 PB 被消耗，因此第二部分即是剩余的 PB.

从项目实施的动态过程分析，这种策略的鲁棒性效果体现为两点: 第一，项目实际进展中，非关键链上活动的拖延未超过其链路具有的自由时差时，即缓冲的实际消耗并未引起资源冲突时，虽然二次调度计划可能后移了紧随其后的关键活动的开始时间，但是此时不需要依据二次调度计划的时间开始关键活动，而根据其实际最早开始时间开始紧后的关键活动，这相当于节约了二次计划中的缓冲时间，其之后的活动调度计划整体前移，而不必对计划进行重新调度. 第二，起到了将 PB 分散到调度计划过程中的效果. 在项目实施中，当某个关键活动拖延时，其拖延通过链条的传递作用将影响之后的每个关键活动，如果仍保持之后的非关键活动的开始时间不变，则很可能引起后面调度计划中多处出现资源冲突. 然而，通过上面消除冲突的策略，在二次调度中将某些关键活动后移的同时也相当于给被后移的关键活动的紧前关键活动设置了 FB; 当前面的紧前活动有拖延时，此处设置的 FB 可以吸收这种拖延，而由于后推关键活动导致的资源冲突问题在实施二次调度计划过程中已经考虑到了，最终达到项目实际进展中对调度计划偏离尽可能小的稳定性目的. 因此，如此设置 FB 和消除二次资源冲突的策略，相当于将 PB 分散到了调度计划中的一些关键活动之后.

2.3 以鲁棒性为目标的关键链项目调度问题

为区别于传统 CCM，以式(1)为目标形成鲁棒性关键链项目调度新问题(critical chain project scheduling problem with robust objective，CCPSP-R)，本文作者前期的研究[18]仔细探究了 CCPSP-R 的建模过程和模型的结构特征，并设计了专用的启发式求解算法. 求解 CCPSP-R 模型，输出鲁棒性最大的调度方案是本篇论文的研究基础.

为更明确 CCPSP-R，现对其与基本 RCPSP(资源有限工期最短，活动仅有一种执行模式)和资源均衡问题(工期固定下的资源均衡)做出区分. 1) CCPSP-R 与基本 RCPSP 的区别体现为两个层面: 首先是传统 CCM 与 RCPSP 的异同，对此详见文献[19]; 其次，CCPSP-R 与传统 CCM 的差别，传统 CCM 以基准调度计划对应的项目工期最短为目标，因此未涉及解决二次资源冲突问题; 但是 CCPSP-R 同时考虑基准调度计划和插入输入缓冲后的二次调度计划，以结合两次调度进程的鲁棒性为目标，因此 CCPSP-R 是对传统 CCM 中蕴含的鲁棒性内涵的定量化表述和建模. 三个问题之间的递进关系是: 基本 RCPSP→传统 CCM→CCPSP-R. 基本 RCPSP 属于确定型问题，而传统 CCM 和 CCPSP-R 被归类为不确定型项目调度问题(活动工期不确定因而需要考虑应对不确定性的措施); 然而基本 RCPSP 为 CCM 的模型提供了基础框架; CCPSP-R 是对传统 CCM 基于鲁棒调度

优化理论的拓展. 2) CCPSP-R 与资源均衡问题差异较大,前者以调度方案中资源的均衡分配状态为目标,以项目工期为约束条件,由于没有了资源限量约束因而其求解难度要远低于 RCPSP. 此外,基本 RCPSP 和资源均衡问题都属确定型问题,CCPSP-R 根植于传统 CCM,因此 CCPSP-R 属于不确定型项目调度问题.

在实践中,CCPSP-R 对项目管理者具有重要的启示: 当预期项目的实施环境不稳定时,在制定进度计划阶段,不宜选择项目工期最短情形下对应的调度方案,而应该以鲁棒性较大的调度方案作为实施方案. 根据实际活动工期的波动并依据鲁棒性大的调度方案指导项目的执行,以期尽可能少地调整进度方案(次数和程度),达到实际进度过程尽可能地接近计划进度的稳定性(鲁棒性)目标.

采用文献[18]中的专用算法求解 CCPSP-R 模型,输出使式(1)的鲁棒性最大时的基准调度计划和二次调度方案,将对应的二次调度方案称为鲁棒性关键链调度方案,记为 $S_{RM}^{\max}$. 文献[18]基于一种静态的研究思路,未能涉及鲁棒性进度方案在项目实施中的动态效果,本文是对文献[18]的拓展和深化,采用概率分布描述随机的活动工期,研究基于二次资源冲突消除策略的最优鲁棒性进度方案在项目实施中的动态稳定性效果.

3　基于策略的鲁棒性指标测试

通过仿真实验来研究消除二次资源冲突的策略和鲁棒性关键链指标的有效性. 实验设计包括三部分: 第一,测试算例的产生及参数配置; 第二,仿真算法和测试指标的设计; 第三,输出数据的统计分析.

3.1　算例产生及参数配置

采用项目调度问题发生器 ProGen (project generator)[20] 随机地生成算例集来测试上一节的鲁棒性指标的动态效果. ProGen 生成算例的参数配置如表 2 所示,根据表 2 中的参数组合,按全因子实验设计的参数有三个: 项目包含的实活动数 J、网络结构的复杂性系数 NC (Networks complexity)、活动工期的标准差 σ_j. 参数 J 的取值为 5 种,NC 的取值为 3 种; 对每个算例,σ_j 取值分 3 种情形: 第一,项目中所有活动工期的标准差都较小,表示项目面临风险较低的情形; 第二,活动工期的标准介于较大和较小之间,对应项目面临风险程度一般的情形; 第三,所有活动工期的标准差都较大,表示项目面临风险较高的情形. 在每种参数组合下生成算例个数为 10 个,由此得到 $5\times3\times3\times10=450$ 个测试算例.

表 2　产生测试算例的参数配置

Table 2 Parameter configurations to generate test instances

ProGen 的参数	取值	备注
实活动数 J	12、16、18、20 及 30	
网络结构的复杂性系数 NC	1.5、1.8 和 2.1	
资源种类 K	2	
资源需求量 r_{j1}, r_{j2}	服从 [2, 8] 均匀分布,随机产生	整数
活动的期望工期 d_j	服从 [2, 12] 均匀分布,随机产生	整数
活动工期的标准差 σ_j	$\lambda\times d_j$,其中 $\lambda=0.3$、0.6 和 0.9	
在每种参数组合下生成的算例个数	10	

沿袭传统 CCM 的假设,活动工期服从正态分布. 插入 FB 在多数情形下都导致二次资源冲突,因此对于 FB,选择基于资源利用因子的 APRT 方法计算其尺寸. PB 的设置对 S 中的资源占用状态无影响,因此选择根方差法计算其尺寸. 基于活动的期望工期构建 CCPSP-R 模型并求解; 对每一个解,首先确定出 S、$FB_{i_mj_m}^m$ (插入位置及尺寸; $m=1, 2, \cdots, M$) 和 PB(尺寸); 其次调用消除二次资源冲突的策略,插入 PB 和各个 $FB_{i_mj_m}^m$,获得二次调度计划 S'、各个 $FB_{i_mj_m}^{m\prime}$ 和 PB; 再次计算解对应的目标值 RM_s,输出鲁棒性最大时的二次调度方案 $S_{RM}^{\max}$.

3.2　仿真程序流程

仿真过程中,根据鲁棒性调度方案 $S_{RM}^{\max}$ 安排

活动的实际进度. 对于项目的起始活动(gating tasks) 采用铁路调度规则(railway) ,其他的活动按照接力赛规则(roadrunner) 安排,据此实现项目实施过程的一次随机抽样,用 as_j 表示活动 j 的实际开始时间(actual start time) ,用 af_j 表示 j 的实际结束时间(actual finish time) . 仿真算法的具体过程如下.

步骤 1 基于均值 d_j 和标准差 σ_j,随机地产生每个活动的实际抽样工期 $ad_j(j=1,2,\cdots,J)$. 根据 $S_{RM}^{\max}$ 确定仿真测试算法中各活动的新索引号 priority [j]; 将合格活动集合 E_n 和局部调度计划的活动集合 PS_n 初始化为仅包含虚活动 1. 索引号序列为从 1 开始的自然数序列,定义为 priority [j] = {1, 2, …, J} ,索引号越小代表项目实施中活动的优先权越高. 其规则如下.

a) 起始虚活动 1 的索引号设为 1,结束虚活动的索引号设为 J;

b) $S_{RM}^{\max}$ 中计划开始时间 s_j' 越早的活动索引号越小;

c) $S_{RM}^{\max}$ 中 s_j' 相同时,关键活动优先执行.

步骤 2 根据各活动索引号表示的优先权值,在某个阶段 n,从 E_n 中选出具有最高优先权的活动 j,根据其所有紧前活动的实际结束时间,确定 j 可能的 $as_j = \max\{af_i, i \in P(j)\}$; 若此时刻安排 j 发生资源冲突,则继续更新 as_j 至最早有活动结束的时刻点,直到满足在时段 $[as_j, as_j + ad_j]$ 的资源余量满足活动 j 的 $r_{jk}(k=1,2,\cdots,K)$,此时 as_j 即确定,进一步可计算出 $af_j = as_j + ad_j$.

步骤 3 将活动 j 从 E_n 中移除并添加到局部调度集合 PS_n.

步骤 4 更新 E_n: 访问活动 j 的紧后活动集合 $S(j)$ 中的每个活动,将符合条件的紧后活动(其所有紧前活动都已进入 PS_n) 加入到 E_n.

步骤 5 判断结束虚活动 J 是否已进入 PS_n,即 PS_n 是否已成为一个全局调度计划活动集合,若是则计算出活动 J 的 af_J,即项目实际工期,并转入步骤 6; 反之则转入步骤 2.

步骤 6 输出本次抽样的项目进展信息,记录保存实验数据: $as_j(j=1,2,\cdots,J)$ 、本次抽样的 af_J 等信息,以便计算实验测试指标.

3.3 实验测试指标

根据 $S_{RM}^{\max}$ 的 s_j' 和本次抽样的 af_J,判断项目是否按期完工; 根据各活动的 as_j 和 s_j' $(j=1,2,\cdots,J)$ 计算本次抽样中活动开始时间偏差指标. 设抽样规模为 N,对每个项目随机抽样多次,从统计意义评价基于二次资源冲突消除策略的关键链鲁棒性指标在项目实施中的效果. 具体设计如下的三个统计指标.

1) 项目的按期完工率(ratio on schedule, RS) . 在某次抽样中,如果得到的 af_J 不超过 s_J',则表明项目按期完工,否则认为项目拖延. 用 RS^i 表示某个算例 i 在多次仿真中的按期完工率; 用 RS_{avg} 表示基于特定算例集合统计出的平均按期完工率.

2) 活动开始时间偏差的绝对值之和(sum of absolute deviation, AD) . 用 AD_n^i 表示算例 i 在第 n 次抽样时,活动的实际开始时间与 $S_{RM}^{\max}$ 中的计划开始时间的偏差值之和. AD_n^i 计算公式如下

$$AD_n^i = \sum_{j=1}^{J} |as_j^i - s_j'|_n \tag{2}$$

用 AD_{avg}^i 和 $AD_{\max}^i$ 分别表示算例 i 在多次仿真中的平均偏差和与最大偏差和; 用 $AD_{\max}$ 和 AD_{avg} 分别表示基于特定算例集合统计出的最大偏差和与最大平均偏差和; $AD_{\max}$ 和 AD_{avg} 越小,表明项目实际进展过程与调度方案的偏差越小.

3) 活动开始时间偏差绝对值的方差(variance of absolute deviation, VD) . 用 VD_n^i 表示算例 i 在第 n 次抽样时,活动的 as_j 与 s_j' 偏差绝对值的方差,计算公式如下

$$VD_n^i = \frac{\sum_{j=1}^{J}\left(|as_j^i - s_j'|_n - \overline{|as_j^i - s_j'|_n}\right)^2}{(J-1)} \tag{3}$$

相应地,用 VD_{avg}^i 和 $VD_{\max}^i$ 分别表示算例 i 在多次仿真中,偏差绝对值方差的平均值和最大值; 用 VD_{avg} 和 $VD_{\max}$ 分别表示基于特定算例集合统计出的偏差绝对值方差的平均值和最大值. $VD_{\max}$ 和 VD_{avg} 越小,表示调度方案在实施过程中越稳定.

为对比 $S_{RM}^{\max}$ 与传统 CCM 的调度方案,模型求解中也保存了传统 CCM 对应的最优进度方案 $S_{f_J}^{\min}$,并且采用本文消除二次资源冲突的策略获

得其对应的二次调度方案 $S_{fJ}^{\min'}$，根据式(1)计算出传统 CCM 调度方案的鲁棒性值. 数值实验中，对传统 CCM 的调度方案也进行测试，根据前述的仿真算法流程，基于与 $S_{RM}^{\max}$ 同样的活动抽样工期，输出以 $S_{fJ}^{\min'}$ 为依据的项目实际进展过程并计算上述三个实验测试指标.

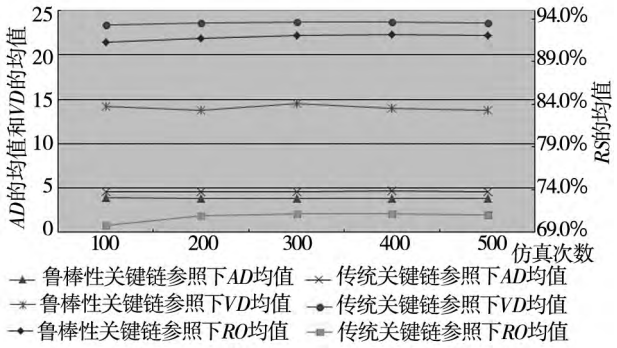

图 4　三个测试指标的均值随仿真次数增加的变化情况

Fig. 4 The means of three indicators change with the increase of simulation size

3.4　数据结果分析

仿真测试算法采用 Visual C++6.0 编码实现，在 CPU 主频 3.10GHz、内存为 4.00GB 的 Inter(R) Core(TM) i5 -2 400 的 PC 机上运行. 已有研究[21]表明，对于求解 $J=120$ 的随机 RCPSP 来说，当仿真次数 N 达到 200 时获得的项目期望工期即可接受. 以此为参照，本文采取序贯程序法确定合适的仿真次数: 设定初始 N 为 100，之后 N 每增加 50 即观察统计指标值的变化，当统计指标值随着 N 的增加趋于稳定时，N 值即可确定. 以 $J=18$ 的所有算例为训练样本，研究抽样规模对算法测试指标的影响. 图 4 给出了统计指标值随仿真次数变化的情况. 从图 4 中可以看出，仿真次数超过 200 时，各指标已基本趋于稳定. 通常活动网络规模越大，项目实现过程的场景数越多. 经过多次试验，最后确定出对 $J=12$、16、18、20 及 30 的算例，选取的 N 分别为 200、300、400、500 及 600.

表 3　不同规模的活动网络的测试结果

Table 3 Test results according to the sizes of activity networks

J	鲁棒性关键链调度					传统 CCM				
	RS_{avg}	AD_{avg}	AD_{max}	VD_{avg}	VD_{max}	RS_{avg}	AD_{avg}	AD_{max}	VD_{avg}	VD_{max}
$J=12$	91.8%	3.2	18.9	13.1	44.1	76.9%	3.8	21.5	16.0	48.3
$J=16$	90.4%	3.4	19.7	17.9	60.3	75.2%	4.4	21.8	24.9	70.6
$J=18$	86.8%	3.9	20.3	20.6	64.2	74.7%	6.3	24.7	28.7	112.1
$J=20$	85.6%	4.1	24.1	28.6	118.2	73.6%	5.4	39.2	38.0	124.6
$J=30$	85.2%	6.7	34.4	41.5	132.8	68.3%	7.7	48.1	63.9	157.1
均值	87.9%	4.3	23.5	24.3	75.1	73.7%	5.5	31.1	34.3	102.5

表 4　复杂性不同的活动网络的测试结果

Table 4 Test results according to the complexities of activity networks

NC	鲁棒性关键链调度					传统 CCM				
	RS_{avg}	AD_{avg}	AD_{max}	VD_{avg}	VD_{max}	RS_{avg}	AD_{avg}	AD_{max}	VD_{avg}	VD_{max}
$NC=1.5$	90.4%	3.7	20.3	20.6	99.6	78.2%	4.9	21.5	30.7	113.2
$NC=1.8$	88.6%	4.3	24.2	24.5	106.8	77.3%	5.1	39.2	34.8	130.7
$NC=2.1$	84.7%	4.8	34.4	27.8	132.8	65.6%	6.5	48.1	37.4	157.1

表 5 三种活动工期标准差情形下的测试结果

Table 5 Test results under three standard deviations of activity durations

σ_j	鲁棒性关键链调度					传统 CCM				
	RS_{avg}	AD_{avg}	AD_{max}	VD_{avg}	VD_{max}	RS_{avg}	AD_{avg}	AD_{max}	VD_{avg}	VD_{max}
$0.3\times d_j$	92.2%	2.5	31.2	17.3	112.3	76.0%	5.2	47.1	31.3	154.9
$0.6\times d_j$	88.4%	4.1	33.6	24.3	125.9	73.5%	5.4	47.3	34.1	156.7
$0.9\times d_j$	83.3%	5.9	34.4	31.5	132.8	71.7%	6.2	48.1	37.4	157.1

对于每次抽样，分别以鲁棒性关键链调度计划 S_{RM}^{max} 和传统关键链调度计划 $S_{fJ}^{min'}$ 为依据获得两种实际项目实施过程，并记录相应的输出数据. 分析项目按期完工率 RS、活动开始时间的偏差绝对值之和 AD 及活动开始时间的偏差绝对值方差 VD 三个统计指标. 统计并处理输出数据见表 3、表 4 和表 5. 通过分析表中的数据可以发现以下结果.

1) 对不同规模 J 的活动网络(表 3)，以鲁棒性关键链调度计划为依据安排进度时，RS、AD 及 VD 三个指标都分别优于传统关键链调度计划依据下的三个指标的值. 首先从全部测试算例的均值指标分析，鲁棒性关键链法的 RS_{avg} 为 87.9%，AD_{avg} 为 4.3 和 VD_{avg} 为 24.3，都分别好于传统 CCM 对应的三个指标值 $RS_{avg}=73.7\%$、$AD_{avg}=5.5$ 和 $VD_{avg}=34.3$. 对不同规模的活动网络，根据三个指标的取值范围可以看出，鲁棒性关键链的 RS_{avg} 明显高于传统 CCM 的 RS_{avg}；鲁棒性关键链的 AD_{avg} 和 VD_{avg} 都分别低于传统 CCM 的 AD_{avg} 和 VD_{avg} 值. 例如，在 $J=12$ 时鲁棒性关键链的 RS_{avg} 达到最大，为 91.8%，AD_{avg} 达到最小为 3.2，VD_{avg} 也达到最小为 13.1，而此时相应的传统 CCM 的 RS_{avg} 则为 76.9%，AD_{avg} 为 3.8，VD_{avg} 为 16.0；鲁棒性关键链的 RS_{avg} 在 $J=30$ 时达到最小为 85.2%，AD_{avg} 达到最大值 6.7，VD_{avg} 同样达到最大值 41.5；而此时传统 CCM 对应的 RS_{avg} 也比 $J=30$ 时减小到 68.3%，AD_{avg} 增大到 7.7，VD 更是增大到 63.9. 对鲁棒性关键链来说，AD_{max} 和 VD_{max} 分别为 34.4 和 132.8(当 $J=30$)，而传统 CCM 的 AD_{max} 和 VD_{max} 分别为 48.1 和 157.1(当 $J=30$)，这表明对于最差情形下的 AD_{max} 和 VD_{max} 两个指标，鲁棒性关键链也不比传统 CCM 差.

2) 对复杂性 NC 不同的活动网络(表 4)，总体来看，三个统计指标的值也表明鲁棒性关键链比传统 CCM 好. 观察指标值变差的趋势：对鲁棒性关键链来说，随着 NC 的增大($NC=1.5\sim2.1$)，RS_{avg} 分别降低 1.8% 和 3.9%，AD_{avg} 分别增加 0.6 和 0.5，VD_{avg} 分别增加 3.9 和 3.3；对于传统 CCM，RS_{avg} 分别降低 0.9% 和 11.7%，AD_{avg} 分别增加 0.2 和 1.4，VD_{avg} 分别增加 4.1 和 8.6. 从数据的变化趋势可看出，NC 从 1.5 变化到 1.8 时，对于 RS_{avg} 的变化，鲁棒性关键链比传统 CCM 降低得多(1.8% 对 0.9%)；对于 AD_{avg}，鲁棒性关键链比传统 CCM 增加得多(0.6 对 0.2)；仅有 VD_{avg} 增加的较传统 CCM 稍少(3.9 对 4.1). 但是当 NC 从 1.8 变化到 2.1 时，鲁棒性关键链的 RS_{avg} 降低得比传统 CCM 的明显少(3.9% 对 11.7%)，AD_{avg} 增加的比传统 CCM 的少(0.5 对 1.4)，VD_{avg} 增加的较传统 CCM 也少(3.3 对 8.6). 对表 4 数据分析结果表明：尽管随着网络复杂性的提高，两种调度方案对应的三个统计指标值都变差，但是对于复杂性更高的活动网络，鲁棒性关键链指标值变差的程度要轻于传统 CCM. 通常，越复杂的项目，不确定性因素对进度过程的影响越大. 因此，鲁棒性关键链在复杂性较高项目实施中的效果，比在复杂性低的项目的实施效果更明显，这也表明鲁棒性指标在应对不确定因素上具有较好的效果.

3) 对于不同的活动工期标准差情形(表 5)，三个指标都随着标准差的增大而变差，但是鲁棒性关键链的指标值仍好于传统 CCM 的指标值，这种结果较容易理解. 实践中，各类不确定因素的

影响多数都体现到活动工期的波动上,缓冲最本质的作用就是吸收活动工期的拖延. 鲁棒性关键链追求项目执行过程的稳定性,在二次调度过程中给各个 FB 配置资源,形成了一种柔性的进度方案(缓冲的消耗取决于实际的活动工期);然而,传统 CCM 仅关注基准调度计划的工期最短,其中包含的非关键活动自由时差数最少,对应的是一种刚性的进度计划,不能很好地消除一些活动工期波动的情形. 因此,相对于传统 CCM,在同样的活动工期波动程度下,鲁棒性关键链在项目实施过程中能更好地吸收工期拖延. 然而,对于鲁棒性关键链来说,当 σ_j 从 $0.3\times d_j$ 增加到 $0.6\times d_j$ 时,RS_{avg}、AD_{avg} 和 VD_{avg} 三个统计指标分别减少了 3.8%、增加了 1.6% 和 7;当 σ_j 从 $0.6\times d_j$ 增加到 $0.9\times d_j$ 时,RS_{avg}、AD_{avg} 和 VD_{avg} 三个统计指标分别减少了 5.1%、增加了 1.8% 和 7.2,这说明了当工期波动程度很高时,鲁棒性关键链吸收工期波动的效果也减弱了.

总体来看,从对上述 3 个表格的数据分析获知,以鲁棒性调度计划为依据安排项目进度的稳定性明显好于以传统关键链调度计划为依据的情形. 从统计指标角度,不论是以鲁棒性关键链调度计划为依据还是以传统关键链调度计划为依据实施项目,三个统计指标值都随着活动网络规模、网络复杂性和活动工期标准差的增大而变差. 具体来说: RS 随着活动网络规模、网络复杂性和活动工期标准差的增大而减小; AD 和 VD 都随着活动网络规模、网络复杂性和活动工期标准差的增大而增大.

4　结束语

从定量化建模和鲁棒调度优化的角度,剖析了关键链中的二次资源冲突困境问题并提出了有效的解决方案. 本文的主要贡献包括三个方面: 第一,从鲁棒调度优化角度开发了一种启发式的局部重调度策略,以消除二次资源冲突问题; 第二,基于开发的策略,设计了考虑两次调度进程和两类缓冲动态消耗的指标来度量关键链调度方案的鲁棒性; 第三,随机地产生标准算例集,设计仿真实验和测试指标,通过对大量随机算例的测试发现: 以鲁棒性关键链调度方案为依据安排进度时,项目的按期完工率、活动开始时间偏差绝对值之和及偏差绝对值的方差三个统计指标值都优于以传统 CCM 调度计划为依据实施项目时的情形. 研究结论表明在项目的实施过程中,本文消除二次资源冲突的启发式协调策略和设计的鲁棒性关键链指标表现出了较好的稳定性. 研究工作从新的理论视角拓展了传统 CCM,同时提高了 CCM 在项目进度管理实践中的适用性,对于项目经理科学地运用 CCM 提供理论依据. 本文形成鲁棒性关键链项目调度问题时,限于篇幅没有同时考虑鲁棒性、项目工期和项目成本等多个目标,后续将探索同时包含鲁棒性和项目绩效的多目标鲁棒性关键链项目调度优化问题.

参 考 文 献:

[1] Hartmann S, Briskon D. A survey of variants and extensions of the resource-constrained project scheduling problem [J]. European Journal of Operational Research, 2010, 207(1): 1 - 14.

[2] 何正文, 刘人境, 胡信步. 基于合同双方相互作用的项目调度优化 [J]. 管理科学学报, 2014, 17(8): 48 - 59.
He Zhengwen, Liu Renjing, Hu Xinbu. Project scheduling optimization based on interaction between two parties of contracts [J]. Journal of Management Sciences in China, 2014, 17(8): 48 - 59. (in Chinese)

[3] Zhu G, Bard J F, Yu G. Disruption management for resource-constrained project scheduling [J]. Journal of the Operational Research Society, 2005, 56(4): 365 - 381.

[4] Zhang J, Elmaghraby S E. The relevance of the "alphorn of uncertainty" to the financial management of projects under uncertainty [J]. European Journal of Operational Research, 2014, 238(1): 65 - 76.

[5] Goldratt E M. Critical Chain [M]. Great Barrington: The North River Press, 1997.

[6] Herroelen W, Leus R. Project scheduling under uncertainty: Survey and research potentials [J]. European Journal of Operational Research, 2005, 165(2): 289 – 306.

[7] Luong D L, Ario O. Fuzzy critical chain method for project scheduling under resource constraints and uncertainty [J]. International Journal of Project Management, 2008, 26(6): 688 – 698.

[8] 刘士新, 宋健海, 唐加福. 资源受限项目调度中缓冲区的设定方法 [J]. 系统工程学报, 2006, 21(4): 381 – 386.
Liu Shixin, Song Jianhai, Tang Jiafu. Approach for setting time buffers in resources-constrained project scheduling [J]. Journal of Systems Engineering, 2006, 21(4): 381 – 386. (in Chinese)

[9] Van de Vonder S, Demeulemeester E, Herroelen W. A classification of predictive-reactive project scheduling procedures [J]. Journal of Scheduling, 2007, 10(3): 195 – 207.

[10] Demeulemeester E, Herroelen W. Robust Project Scheduling [M]. Boston: Now Publishers Inc, 2011.

[11] 田文迪, 胡慕海, 崔南方. 不确定性环境下鲁棒性项目调度研究综述 [J]. 系统工程学报, 2014, 29(1): 136 – 144.
Tian Wendi, Hu Muhai, Cui Nanfang. Review of studies on robust project scheduling under uncertainty [J]. Journal of Systems Engineering, 2014, 29(1): 136 – 144. (in Chinese)

[12] Herroelen W, Leus R. On the merits and pitfalls of critical chain scheduling [J]. Journal of Operations Management, 2001, 19(5): 559 – 577.

[13] Peng W, Huang M. A critical chain project scheduling method based on a differential evolution algorithm [J]. International Journal of Production Research, 2015, 52(13): 3940 – 3949.

[14] Tukel O I, Rom WO, Eksioglu S D. An investigation of buffer sizing techniques in critical chain scheduling [J]. European Journal of Operational Research, 2006, 172 (2): 401 – 416.

[15] 施 骞, 王雅婷, 龚 婷. 项目缓冲设置方法及其评价指标改进 [J]. 系统工程理论与实践, 2012, 32(8): 1739 – 1746.
Shi Qian, Wang Yating, Gong Ting. An improved approach for project buffer sizing and evaluation [J]. Systems Engineering: Theory & Practice, 2012, 32(8): 1739 – 1746. (in Chinese)

[16] Bie L, Cui N, Zhang X. Buffer sizing approach with dependence assumption between activities in chain scheduling [J]. International Journal of Production Research, 2012, 50(24): 7343 – 7356.

[17] 李星梅, 乞建勋, 苏志雄. 自由时差定理与 k 阶次关键路线的求法 [J]. 管理科学学报, 2009, 12(2): 98 – 104.
Li Xingmei, Qi Jianxun, Su Zhixiong. Free float theorem and algorithm of seeking the k-the order critical path [J]. Journal of Management Sciences in China, 2009, 12(2): 98 – 104. (in Chinese)

[18] 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(2): 135 – 144.
Zhang Jingwen, Liu Gengtao. Critical chain project scheduling problem with the robust objective [J]. Journal of Systems Engineering, 2015, 30(2): 135 – 144. (in Chinese)

[19] 张静文, 李若楠. 关键链项目调度方法研究评述 [J]. 控制与决策, 2013, 28(9): 1281 – 1287.
Zhang Jingwen, Li Ruonan. Review of critical chain project scheduling method [J]. Control and Decision, 2013, 28(9): 1281 – 1287. (in Chinese)

[20] Kolisch R, Sprecher A. PSPLIB-A project scheduling problem library [J]. European Journal of Operational Research, 1997, 96(1): 205 – 216.

[21] 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.

Strategy to eliminate the second resource conflicts in critical chain method based on robustness

ZHANG Jing-wen, QIAO Chuan-zhuo, LIU Geng-tao

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

Abstract: An effective strategy is proposed to solve the dilemma of second resource conflicts in the critical chain method (CCM) from a novel perspective of robust optimization. Firstly, this dilemma is mathematically expressed to gain a quantitative model, and the logic of the dilemma resulted from insertions of feeding buffers is explored. Then scenario analysis is adopted and four basic components are decomposed from the complex conflict phenomena. Secondly, based on robust scheduling optimization, some effective policies are investigated and classified for all sub-problems of conflict situations, and a local rescheduling strategy is developed. Moreover, a robust index is designed based on the strategy, which combines the two scheduling progresses with dynamic consumptions of two kinds of buffers. The optimal schedule is an output when maximizing the robust index. Extensive numerical experiments and three performance indicators are constructed, which include the realized probability that a project finishes on schedule, the sum of the absolute deviations between the actual and the planned activity starting times and the variance of the absolute deviations for activity starting times. Extensive numerical experiments are also performed on the set of instances randomly generated by Progen. The experimental results show the values of three statistical indicators under the schedule of the robust CCM are better than those under the schedule of the traditional CCM. Finally, the conclusions indicate that the strategy to deal with the second resource conflicts and the robust measure exhibits good stable effects during project implementation.

Key words: the second resource conflicts; local rescheduling policy; robust measurement; critical chain method; numerical experiments