考虑资源置信度的跨企业项目鲁棒性调度算法

徐汉川 [1] 徐晓飞 [1]

摘 要 资源不确定性高和调度鲁棒性要求高是跨企业项目调度问题的重要特征, 本文采用资源置信度度量资源的不确定性, 建立了考虑资源置信度约束的跨企业项目鲁棒性优化调度模型, 设计了路径重连求解算法. 算法以路径重连机制搜索解空间, 以嵌入的启发式时间缓冲插入算法快速生成鲁棒性调度, 并可通过局部增强搜索算法进一步优化调度的鲁棒性. 本文应用项目调度标准问题库 PSPLIB 中大量问题实例进行了仿真实验, 同两个当前具有代表性的鲁棒性项目调度算法进行了比较, 实验结果表明了文中算法的有效性与优势.

关键词 跨企业项目, 项目鲁棒性调度, 资源置信度, 路径重连算法

引用格式 徐汉川, 徐晓飞. 考虑资源置信度的跨企业项目鲁棒性调度算法. 自动化学报, 2013, **39**(12): 2176−2185

DOI 10.3724/SP.J.1004.2013.02176

Resource-confidence-considered Robust Project Scheduling Algorithm for Cross-enterprise Project

XU Han-Chuan[1] XU Xiao-Fei[1]

Abstract Higher resources uncertainty and requirement of schedule robustness are key characteristics of cross-enterprise project scheduling. For this problem, some conceptions of resource confidence are defined to measure the resource uncertainty, and a robustness optimization model for cross-enterprise project scheduling is developed which considers the special constraint of resource confidence. A path relinking based project scheduling approach is presented. Firstly, the approach searches the solution space employing path relinking technique. Then, it embeds a heuristic time buffer insertion algorithm to quickly generate robust project schedule. If needed, a local enhanced search algorithm can be used to further improve the schedule robustness. The proposed approach has been tested on the standard instances in PSPLIB, and compared with two well-known competitive algorithms. Experimental results have shown that the new proposed approach is highly effective and efficient for cross-enterprise project scheduling.

Key words Cross-enterprise project, robust project scheduling, resource confidence, path relinking algorithm

Citation Xu Han-Chuan, Xu Xiao-Fei. Resource-confidence-considered robust project scheduling algorithm for cross-enterprise project. *Acta Automatica Sinica*, 2013, **39**(12): 2176−2185

收稿日期 2012-01-05 录用日期 2012-09-05
Manuscript received January 5, 2012; accepted September 5, 2012
国家高技术研究发展计划 (863 计划) (2012AA040902, 2012AA040904), 国家自然科学基金 (71171066), 欧盟第七框架项目 (295130) 资助
Supported by National High Technology Research and Development Program of China (863 Program) (2012AA040902, 2012AA040904), National Natural Science Foundation of China (71171066), and FP7-PEOPLE-2011-IRSES (295130)
本文责任编委 王红卫
Recommended by Associate Editor WANG Hong-Wei
1. 哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001
1. School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001

随着经济全球化趋势和信息技术的发展, 以动态联盟、敏捷虚拟企业和网络化制造等为组织形式的跨企业项目管理模式已经成为了项目管理发展的必然趋势. 跨企业项目的调度问题, 除了具有典型资源受限项目调度问题 (Resource constrained project scheduling problem, RCPSP)[1] 的特征外, 在资源约束和优化目标方面存在自身的特殊性, 主要体现在: 1) 资源的不确定性高. 在单企业项目中, 项目使用的资源隶属于项目承担企业, 企业对资源信息掌握准确, 资源的可控性强、不确定性低. 而跨企业项目所用资源分属于不同的企业, 协作企业具有高度自治性, 有自己的生产运作方式和决策机制, 同时也会承担多个项目, 其自身资源的阶段性紧张, 经常会出现无法按时按量提供资源的情况, 导致跨企业项目所用资源不确定性高, 资源的不确定性是跨企业项目计划和调度过程中不容忽视的因素. 2) 调度的鲁棒性要求高. 跨企业项目计划是各协作企业制定各自生产和采购计划的基准, 执行过程中一旦发生变动, 会带来巨大的变更和协调成本, 甚至导致项目的拖期. 不同于传统的工期最短、成本最低或资源均衡利用等目标, 跨企业项目调度的重要目标是在保证项目交货期前提下, 使项目调度的鲁棒性最大, 即提高跨企业项目调度的稳定性和抵抗不确定因素干扰的能力. 因此, 研究考虑资源不确定性的跨企业项目鲁棒性调度方法是非常有意义的.

现有针对不确定性环境下的鲁棒性项目调度问题的研究, 主要关注项目鲁棒性的度量、时间缓冲量的计算、时间缓冲插入的位置, 以及相应的优化算法等方面[2]. 在鲁棒性度量方面, Al-Fawzan 等[3] 采用正、反向调度结合的方法计算资源约束下各活动的松弛时间之和, 以此度量调度的鲁棒性, 提出了工期最短和鲁棒性最大的双目标禁忌搜索算法. Kobylański 等[4] 采用各活动自由时差与活动工期的比率来度量调度的鲁棒性, 并改进了文献 [3] 中的算法. Lambrechts 等[5] 综合考虑了活动自由时差和活动执行偏差成本, 提出一种基于时间缓冲效用函数的鲁棒性度量模型, 部分解决了基于活动开始时间偏差的各类模型只能通过仿真实验寻优的问题. Herroelen 等[2] 则综合了各类度量方法, 将项目调度的鲁棒性分为质量鲁棒性 (Quality robust) 和解鲁棒性 (Solution robust), 前者确保项目按期完工, 后者确保各活动按计划开始时间执行. 在时间缓冲的计算和插入位置的研究方面, Tavares 等[6] 为各活动设置统一的浮动因子来计算插入时间缓冲的大小. Herroelen 等[7−8] 将各活动浮动因子的设置同活动的执行偏差成本相关联, 设计了启发式的缓冲区插入设置方法 (Adapted float factor, ADFF), 并对缓冲区应该加到项目后部, 还是应该分散插入到各活动间进行了深入讨论. 在其基础上, Vonder 等[9] 通过引入资源流网络的方法 (Resource flow-dependent float factor, RFDFF) 解决了 ADFF 方法在插入时间缓冲后可能引起的资源冲突问题. 刘士新等[10−11] 则讨论了关键链方法中插入缓冲的机制和算法. Vonder 等[12] 对比了当前几种较好的鲁棒性项目调度算法, 提出了插入时间缓冲的启发式算法 (Starting time criticality, STC), 同时提出了改进的深度搜索策略和禁忌搜索算法. 现有相关方法中, 考虑的多为活动工期不确定性, Lambrechts 等[13] 所提的项目调度算法是少数考虑了资源不确定性的研究, 但其并不适合跨企业项目.

通过上述分析可以看出, 当前相关研究侧重于解鲁棒性或质量鲁棒性其中之一, 缺乏二者的综合考虑, 缺乏对资源不确定性的研究, 所提各类方法并不适合跨企业项目调度. 由此, 本文提出一种考虑资源置信度的跨企业项目调度算法, 首先给出资源置信度的概念来度量资源的不确定性, 然后建立了考虑资源置信的鲁棒性项目调度模型. 进而给出了基于路径重连的求解算法, 对路径重连过程中嵌入的启发式算法和增强搜索算法进行了深入讨论. 最后给出了实例和对比分析.

1　问题定义

本节首先介绍资源置信度的概念, 然后讨论了调度鲁棒性的度量, 进而给出了问题模型.

1.1　资源置信度的定义

为度量跨企业项目中资源的不确定性, 本文采用资源置信度的概念来度量资源在一段时间内可按承诺量提供的可靠程度, 令 $a_k[t_1,t_2]$ 为资源 k 在时间段 $[t_1,t_2]$ 伙伴企业承诺的可用量, $r_k[t_1,t_2]$ 为资源 k 在时间段 $[t_1,t_2]$ 实际提供量, 资源置信度可定义如下:

定义 1. 在跨企业项目中, 对事件 $r_k[t_1,t_2] \geq a_k[t_1,t_2]$ 发生的可靠程度的度量, 称为资源 k 在时间段 $[t_1,t_2]$ 内的置信度, 记为 $rc_k[t_1,t_2]$, $rc_k[t_1,t_2] \in [0,1]$.

从项目活动执行的角度看, 活动在执行过程中所用各资源的置信度体现了活动执行受资源不确定性影响的可能性, 也体现了活动按计划成功执行的可能性. 为反映活动所用资源的置信度对活动工期可靠程度的影响, 给出如下定义.

定义 2. 项目活动 i 在时间段 $[t_1,t_2]$ 执行时, 基于其所使用/依赖资源的置信度得出的活动工期的可靠程度, 称为活动执行的资源依赖置信度, 简称活动执行置信度, 记为 $\overline{rc^i}[t_1,t_2]$, $\overline{rc^i}[t_1,t_2] \in [0,1]$.

活动的执行受到多种因素的影响, 本文中的活动执行置信度特指活动执行所依赖资源的置信度对活动执行的影响.

活动执行时需要多种关键资源和非关键资源, 每种资源在活动执行区间内置信度不同, 资源对活动执行的重要程度也不同, 其中具有最高重要度的资源的置信度反映了活动执行置信度.

规则 1. 活动执行置信度确定规则: 活动执行置信度采用所用资源中具有最高重要度的资源的置信度度量, 当存在多个重要度相同的资源时, 则取决于其中资源置信度的最小值.

活动执行置信度计算公式为

$$\overline{rc^i}[t_1,t_2] = \min\{rc_k[t_1,t_2] | k \in R_i, \omega_k^i = \max_{j \in R_i}\{\omega_j^i\}\} \quad (1)$$

其中, R_i 为活动 i 所需资源的集合, ω_k^i 为资源 k 对于活动 i 的重要度, $\sum_{k \in R_i} \omega_k^i = 1$.

1.2　调度鲁棒性的度量

项目调度的鲁棒性是指当项目执行过程中存在不确定因素时, 项目调度仍然可行, 即调度具有抵抗不确定性因素影响的能力. 项目调度鲁棒性可分为保护工期的质量鲁棒性和保护活动开始时间的解鲁棒性[2]. 按期完工是项目的基本目标, 同时, 跨企业项目由多自主企业协作执行的特点, 决定了基准计划的变动会带来巨大的协调和变更成本, 因此在跨企业项目中, 解的鲁棒性同质量鲁棒性一样重要.

项目调度的鲁棒性是通过设置活动的松弛时间来实现的, 令 ES_i 和 LS_i 分别代表活动 i 在满足资源约束情况下的最早和最晚可行开始时间, 则活动 i 的松弛时间 $FS_i = LS_i$

$-\ ES_i$. 令 s_i 为活动 i 的计划开始时间, 为保证调度的鲁棒性, 设定活动不会提前开始执行, 则有 $ES_i = s_i$, $FS_i = LS_i - s_i$. 活动具有的松弛时间是对其后续活动执行的一种保护, 活动具有的松弛时间越大, 则其后续活动受到该活动异常执行的干扰可能性就越小. 考虑到松弛时间的保护效果并不是随着松弛时间的增加而线性递增, 采用收益递减的效用函数进行表示[5], 对于活动 (i), 其松弛时间的保护效果为 $\sum_{j=1}^{LS_i-s_i} \mathrm{e}^{-j}$. 另外, 活动的重要程度不同, 活动松弛时间起到的保护效果也不同, 定义 w_i 为活动 i 的执行偏差成本, 代表由活动的实际开始时间同计划开始时间偏差带来的计划变更和协调成本. 某活动后续相关活动的偏差成本越大, 则该活动每单位松弛时间起到的保护效果越大. 在保护后续活动开始时间的同时, 也要尽量保护当前活动本身的执行, 即应该使当前活动处于所用资源的置信度高的时间窗口内. 基于以上分析, 提出如下所示的项目调度鲁棒性度量公式:

$$RM = \sum_{i=1}^{n} \left\{ \overline{rc^i}[s_i, s_i + d_i] \cdot CW_i \cdot \sum_{j=1}^{LS_i - s_i} \mathrm{e}^{-j} \right\} \tag{2}$$

其中, CW_i 为活动 i 及其紧后活动执行时间偏差成本之和[5, 13], $CW_i = w_i + \sum_{j \in succ_i} w_j$, 目标函数 RM 最大化时, $CW_i \cdot \sum_{j=1}^{LS_i - s_i} \mathrm{e}^{-j}$ 最大化活动 i 松弛时间对后续活动的保护效果, $\overline{rc^i}[s_i, s_i + d_i]$ 作为系数使活动的执行时间窗口处于资源置信度高的区间内. 实际应用中, 可为项目的结束活动 n 设置较大的 w_n, 以保护项目工期, 提高调度的质量鲁棒性, 关于此点将在第 3 节中详细讨论.

1.3 问题模型

跨企业项目调度可归结为 $PSm, \sigma, \rho|prec|RM$ 调度问题[1], 其含义是项目具有 m 类可用资源, 每类可用资源具有 σ 个处理单元, 项目的每个活动至多需要这些资源的 ρ 个处理单元, 各活动之间具有时序约束关系. 要求在满足时序和资源约束的前提下, 确定项目各活动在各类资源上的处理时间安排, 使得鲁棒性最优.

跨企业项目可分解为 n 个活动, 整个项目的结构可表示为一个有向无环 (Activity-on-the-node, AON) 网络图 $G = (N, A)$, 其中 $N = 1, \cdots, n$ 为活动的集合, $A \subseteq N \times N$ 为边的集合, 表明了活动间时序关系. 活动 1 和 n 称为虚活动 (不消耗资源且执行时间为 0), 分别代表项目的开始和结束. 项目执行过程中共需要 K 种可更新资源, 每种资源在不同时段具有各自的资源置信度. 活动 i 的完成需要第 k $(k = 1, \cdots, K)$ 种资源数量为 r_{ik}, 活动工期为 d_i, 项目的交货期为 D. 项目调度的目标是确定各活动的开始时间 $S = (s_1, \cdots, s_n)$, 以保证在满足资源数量、活动时序关系、交货期和资源置信度等约束下, 产生的调度具有最大的鲁棒性.

本文的跨企业项目调度问题, 设定如下: 资源均为可更新资源; 活动为单执行模式; 活动执行不可被抢占中断; 活动不允许提前开始执行; 同类资源由同一企业提供.

考虑资源置信度的项目鲁棒性调度问题模型 RCRPSPM 可描述为

$$\max RM \tag{3}$$

$$\text{s.t.} \quad s_i + d_i \le s_j, \ \forall j \in succ_i \tag{4}$$

$$\sum_{i:i \in A_t} r_{ik} \le a_k, \ \forall t, \ k = 1, \cdots, K \tag{5}$$

$$\overline{rc^i}[s_i, s_i + d_i] \ge \eta_i, \ \ i = 1, \cdots, n \tag{6}$$

$$s_n \le D \tag{7}$$

$$s_i \ge 0, \ \ s_i \text{ 是整数}, \ i = 1, \cdots, n \tag{8}$$

其中, $succ_i$ 为活动 i 的紧后活动集合, a_k 为资源 k 的可用量, η_i 为活动 i 的执行置信度阈值, A_t 为时段 t 内正在进行的活动集合.

在该模型中, s_i $(i = 1, \cdots, n)$ 为决策变量, 目标函数 (3) 表示调度的目标为最大化鲁棒性, 约束 (4) ∼ (7) 分别为活动时序关系约束、资源数量约束、资源置信度约束和项目工期约束.

2 路径重连算法 TLPR

2.1 算法思想

本文问题的求解涉及三个关键子问题:

子问题 1. 如何确定最优活动调度顺序和资源分配顺序;

子问题 2. 如何确定活动的计划时间, 使活动所用资源的置信度尽可能高, 以规避资源不确定性对活动执行的影响;

子问题 3. 采用何种策略将有限的松弛时间在各活动间分配, 以使项目调度的鲁棒性最大化.

如果不考虑资源置信度约束, 本文问题退化为以项目鲁棒性最大化为调度目标的单模式资源约束项目调度问题, 为 NP-hard 问题[14], 因此本文问题亦为 NP-hard 问题. 针对本文问题优化目标和约束的特点, 设计了以路径重连算法 (Path relinking) 为搜索框架, 同时嵌入启发式时间缓冲插入算法 (Heuristic time buffer insertion algorithm, HTBIA) 和局部增强搜索算法 (Local enhanced search algorithm, LESA) 的求解算法 (HTBIA and LESA-based path relinking, TLPR).

针对子问题 1, 通过路径重连技术搜索解空间, 寻找最优的活动调度顺序, 对每个活动优先调度列表, 采用改进的串行项目调度方案进行解码, 生成以项目工期最短为目标的初始调度 S^0. 针对子问题 2 和子问题 3, 依据资源置信度和各活动的执行偏差成本, 通过算法 HTBIA 在调度 S^0 中插入时间缓冲, 调节各活动的松弛时间, 生成鲁棒性调度 S^B. HTBIA 算法可以快速地生成鲁棒性调度, 但不能够保证调度鲁棒性的最优化, 因此采用算法 LESA 对调度 S^B 进行鲁棒性优化, 得到针对当前活动列表的优化调度 S^R. 如果尚未达到中止条件, 则继续进行路径重连过程. 算法 TLPR 的总体思想如图 1 所示.

2.2 算法 TLPR 详细过程

2.2.1 算法整体步骤

路径重连算法是一种新兴的演化算法, 其采用基于种群的全局搜索策略, 运用 "分散 – 收敛 – 集聚" 的智能迭代机制, 在参考集中形成高质量和多样性的解, 并通过子集合并和参考集更新方法, 求取问题的全局最优解或满意解[15]. 由于其参考集的记忆能力, 使其可以动态跟踪当前的搜索情况, 以调整搜索策略, 减少了搜索过程的随机性, 目前在各类组合优化问题的求解中得到了广泛应用.

算法 TLPR 的流程如图 2 所示, 其步骤如下:

输入. RCRPSPM 模型的各参数和约束.

输出. 模型的优化解 S^R.

1) 初始化: 定义种群容量 N_p, 参考集容量 N_r, 最大迭代次数 $maxNum$;

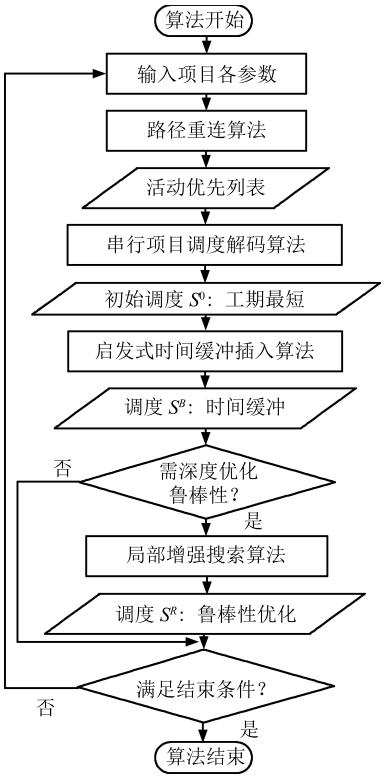

图 1　TLPR 算法总体思想

Fig. 1　Main idea of the TLPR algorithm

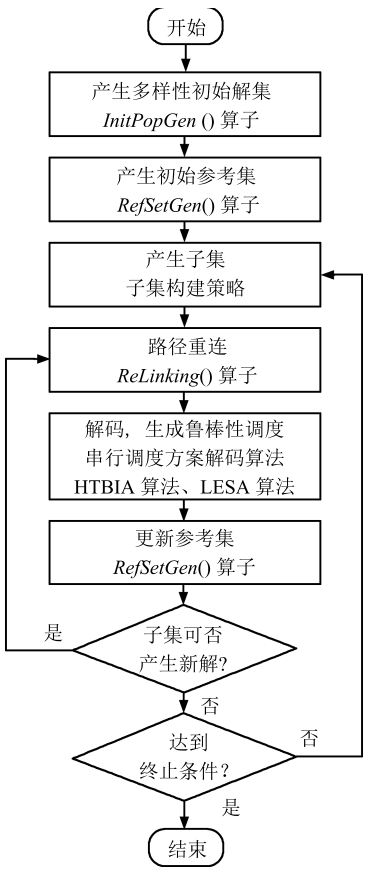

图 2　路径重连算法的基本流程

Fig. 2　Flow chart of the path relinking algorithm

2) 调用 $InitPopGen$ 算子产生初始种群 $InitP$;

3) 调用 $RefSetGen(InitP)$ 算子从 $InitP$ 中生成参考解集 $RefSet$;

4) 调用解码算法、HTBIA 算法和 LESA 算法, 得到 $RefSet$ 中的最好解, 存入 S^* 中;

5) **for** $Num = 1$ to $maxNum$ **do**;

6)　$NextP = \phi$;

7)　利用子集构建策略从 $RefSet$ 中生成有序解对集合 $PairSet$;

8)　**for** 每个解对 $(S', S'') \in PairSet$ **do**;

9)　　调用路径重连算子生成新解集 $NewSet = ReLinking(S', S'')$;

10)　　$NextP = NextP \cup NewSet$;

11)　**end for**;

12)　调用解码算法、HTBIA 算法和 LESA 算法, 计算 $NextP$ 中所有解的目标函数值;

13)　**if** $f(S^*) > \max\{f(S)|S \in NextP\}$, **then**

14)　　$NextP = NextP \cup S^*$;

15)　**else**

16)　　$S^* = S^{\max}$, 其中 $f(S^{\max}) = \max\{f(S)|S \in NextP\}$;

17)　**end if**;

18)　保留 $NextP$ 中前 N_p 个最好解, 删除其余解, 更新参考集;

19) **end for**;

20) $S^R \leftarrow S^*$, 输出解 S^R.

2.2.2　解的表达与评价

本文采用活动时序列表 AL 作为解编码方式, 在 AL 中没有活动会排列在其前导活动之前. 通过串行调度方案可将 AL 解码为一个可行调度, 解码时按照活动在 AL 中出现的先后顺序依据活动尽早开始的原则进行调度, 调度需要同时满足资源数量、时序关系和资源置信度的约束, 通过串行调度方案产生的调度结果均为积极调度计划 (Active schedule), 得到的是各活动的最早开始时间.

对于调度结果 S, 用 $f(S)$ 代表利用式 (2) 计算得到的目标值, 用来评价解的优劣. 为计算 $f(S)$, 需要确定各活动的松弛时间, 活动最晚完成时间 LF_i 应使在 $[ES_i, LF_i]$ 内资源约束始终满足, 则活动 i 在 LF_i 之前完成则不会影响到紧后活动的执行. 由于存在资源约束, LF_i 不一定等于 i 所有紧后活动最早开始时间的最小值. 本文采用倒排算法, 按照各活动最早完成时间降序依次计算各活动资源约束下的松弛时间, 详细算法略.

2.2.3　初始种群产生与参考集构建

初始种群的产生算子定义为 $InitPopGen$, 为保证初始种群的多样性, 通过随机生成的方法产生 N_p 个解构成初始种群, 确保每个解的 AL 满足活动时序关系约束.

参考集构建算子定义为 $RefSetGen(Pop)$, 参考集 $RefSet$ 由当前多样性解集中选取的 N_r 个相互间有一定距离的精英解构成, 精英解保证了解的质量, 解之间的距离保证了解的多样性. 解之间的距离主要体现在活动调度顺序差异度上, 对于解 S_u 和 S_v, 其对应活动列表 AL_u 和 AL_v 中第 q 个位置的差异度计算方法如下所示.

$$d_{uv}(q) = \begin{cases} 1, & \text{若 } AL_u(q) \neq AL_v(q) \\ 0, & \text{若 } AL_u(q) = AL_v(q) \end{cases} \tag{9}$$

解 S_u 和 S_v 间的距离 $D(S_u, S_v)$ 计算方法如式 (10) 所示.

$$D(S_u, S_v) = \sum_{q=1}^{n} d_{uv}(q) \tag{10}$$

解 S 与参考集 $RefSet$ 间的距离定义为 S 与 $RefSet$ 中所有解的最小距离 $D(S, RefSet)$, 有 $D(S, RefSet) = \min\{D(S, r)|r \in RefSet\}$. 为保证参考集中解的多样性, 定义距离阈值 $minD$, 种群中的解只有与 $RefSet$ 中的解距离大于 $minD$ 时才会被加入到 $RefSet$ 中.

2.2.4 子集构建策略

子集 $PairSet$ 的构建采用两两有序组合的方法从 $RefSet$ 中创建, 同时确保任意两个有序解间至少存在一个不同解, $PairSet = \{(S', S'')|S', S'' \in RefSet, D(S', S'') > 2\}$.

2.2.5 路径重连算子

路径重连算子定义为 $ReLinking(S', S'')$, 其中 S' 称为初始解, S'' 称为向导解. 路径重连的过程是新解产生的过程, 新解从初始解和向导解继承优秀属性, 每步转换生成一个新解, 越早生成的解与初始解越相似, 越晚生成的解与向导解越相似, 所有新解构成一条从初始解到向导解的路径, 在路径上往往能够找到比初始解和向导解更好的解.

重连算子 $ReLinking(S', S'')$ 算法步骤如下:

1) 设置新解集合 $NewSet = \phi$;

2) 如果 S' 和 S'' 中活动时序相同, 则转 8), 否则找到 S' 和 S'' 第一个不同活动的位置 p;

3) 找到 $AL_{S''}(p)$ 在 S' 中的位置 q, 交换 $AL_{S'}(p)$ 与 $AL_{S'}(q)$, 产生新解 $NewS$, 如果 $NewS$ 满足时序约束, 转 7);

4) **while** $NewS$ 不满足时序约束;

5) 找到 AL_{NewS} 中第一个违反时序约束的活动 i, 找到 i 所在位置之后的紧前活动 j, 交换活动 i 和 j 的位置;

6) **end while**;

7) $NewSet = NewSet \cup NewS$, $S' = NewS$, 转 2);

8) 输出 $NewSet$.

2.3 启发式时间缓冲插入算法 HTBIA

串行解码算法生成的调度 S^0 是以活动尽早开始为原则, 以项目工期最短为目标的调度, 并没有考虑调度的鲁棒性. 为生成鲁棒性调度, 以 S^0 为初始调度, 在活动开始时间与其紧前活动最晚完成时间之间插入时间缓冲, 通过时间缓冲来吸收掉前序活动执行异常的影响. 由于交货期的限制, 能够插入的时间缓冲有限, 因此应将缓冲分配给最应该受到保护的活动. 对于活动 i ($i = 1, \cdots, n$), 在判断其应受保护的重要程度时, 主要考虑三方面因素:

1) 活动 i 的紧前活动所用资源的置信度. 紧前活动所使用资源的置信度越低, 其执行受到资源不确定性影响的可能性越大, 则影响到活动 i 执行的可能性越大.

2) 紧前活动已有松弛时间的多少. 紧前活动已有松弛时间越多, 其执行异常影响到活动 i 的可能性越小, 则对活动 i 的保护越大, 反之亦然.

3) 活动的执行偏差成本. 偏差成本越大, 则活动越应受到保护.

基于以上考虑, 活动 i 应受的保护程度 (简称为活动应受保护度) 定义为 $rcid(i)$, 其计算方法如下所示:

$$rcid(i) = w_i \sum_{j \in pred(i)} \frac{1}{\overline{rc^j}[s_j, s_j + d_j] \cdot \sum\limits_{k=1}^{FS_j} \mathrm{e}^{-k}} \tag{11}$$

以 "活动应受保护度越大越优先" 为启发式规则, 设计了启发式时间缓冲插入算法 HTBIA.

算法 HTBIA 步骤如下:

输入. 初始调度 S^0, RCRPSPM 模型的各参数和约束.

输出. 鲁棒性调度 $S^B = (s_1^B, \cdots, s_n^B)$.

1) 初始: $S^B \leftarrow S^0$, $f^* = f(S^B)$;

2) 计算各活动的 $rcid(i)$;

3) 生成活动队列 $Q = N\backslash\{1\}$, 对 Q 中活动按 $rcid$ 值降序排列;

4) $s_{Q(1)}^B = s_{Q(1)}^B + 1$;

5) 更新 $Q(1)$ 活动所有后序相关活动, 在满足资源约束下开始时间后移, 形成新解 S';

6) **if** $s'_n \le D$ and $f(S') > f^*$**then**

7) $S^B \leftarrow S'$, $f^* = f(S')$, 转 2);

8) **else**

9) $s_{Q(1)}^B = s_{Q(1)}^B - 1$, $Q = Q\backslash\{Q(1)\}$;

10) **if** $Q = \phi$ **then**

11) 退出;

12) **else**

13) 转到 7);

14) **end if**;

15) **end if**.

步骤 3) ~ (5) 按照 $rcid$ 降序排列生成活动列表 Q, 为最应受保护的活动增加时间缓冲 1, 并更新调度; 在步骤 6) ~ 15) 中, 如果新调度 S' 满足工期约束且使目标函数值增加, 则进行下次迭代; 否则还原调度, 处理 Q 中下个活动. 如所有活动均处理结束, 说明不能再增加调度的鲁棒性, 算法结束.

2.4 局部增强搜索算法 LESA

算法 HTBIA 依据启发式规则快速生成鲁棒性调度, 但启发式算法并不能保证获得最优解或近似最优解, 因此本节提出局部增强搜索算法 LESA 来进一步优化调度鲁棒性.

算法 LESA 以算法 HTBIA 的结果 S^B 为起点, 首先按各活动在调度 S^B 中开始时间的降序排列, 生成队列 L, 然后逐一对 L 中各活动进行局部增强搜索. 对于当前处理的活动 $L(i)$, 在满足资源约束和不影响其他活动开始时间的前提下, 找到活动 $L(i)$ 的可行开始时间窗口 $[ES_{L(i)}, LS_{L(i)}]$, $[ES_{L(i)}, LS_{L(i)}]$ 中的各开始时间构成了局部的邻域解集, 计算邻域中的最优解作为新调度中 $L(i)$ 的开始时间. 依次处理 L 中所有活动, 当所有活动处理完后, 如果解没有改进则算法结束, 否则重新进行下次迭代. 该算法并没有逐一遍历所有活动的可行开始时间, 只是对当前活动进行了局部增强搜索, 然后以更新后的调度为起点进行下步操作. 这种快速下降机制降低了计算的复杂性, 提高了算法的效率.

算法 LESA 步骤如下:

输入. 鲁棒调度 $S^B = (s_1^B, \cdots, s_n^B)$, RCRPSPM 模型的各参数和约束.

输出. 优化解 $S^R = (s_1^R, \cdots, s_n^R)$.

1) 初始化: $S^R \leftarrow S^B$, $s_1^R = 0$, $s_n^R = D$;

2) **repeat**

3) 按调度 S^R 中各活动完成时间降序生成活动队列 L;

4) **for** $i = 1$ to $n-1$ **do**
5) 单步右移, 确定活动最晚开始时间 $LS_{L(i)}$;
6) 单步左移, 确定活动最早开始时间 $ES_{L(i)}$;
7) 在 $[ES_{L(i)}, LS_{L(i)}]$ 内通过循环枚举找到使 RM 最大的时间 t'(多个时取最大值);
8) 更新调度, 令 $s^R_{L(i)} = t'$, 保持其余活动开始时间不变;
9) **end for**;
10) **until** 目标函数无改进;
11) 输出调度 S^R, 算法结束.

2.5 算法复杂度分析

算法 TLPR 的计算过程主要包括新解的产生、邻域搜索和候选解评估.

由解的编码方案可知, 每个解包含 n 个活动, 在两个解路径重连的路径中, 产生新解数目的上限为 n, 对于具有 N_r 个精英解的参考集, 算法 TLPR 的每次迭代都会产生 $N_r^2 - N_r$ 个解对, 总迭代次数记为 M, 则需要评估的候选解上限为 $Mn(N_r^2 - N_r)$.

候选解的评估过程包括串行解码、活动松弛时间计算、HTBIA 和 LESA 四个算法, 各算法的时间复杂度分析如下: 1) 串行解码和活动松弛时间计算的复杂度均为 $\mathrm{O}(n^2K)$, K 为资源数目; 2) 算法 HTBIA 的每次迭代过程中, 所有活动按照 $rcid$ 值降序排序的时间复杂度为 $\mathrm{O}(n\log n)$, 插入时间缓冲的操作最多执行 n 次, 算法迭代次数上限为项目工期 D, 算法 HTBIA 的时间复杂度为 $\mathrm{O}(Dn\log n)$; 3) 算法 LESA 的每次迭代过程中, 对调度 S^R 中各活动排序的复杂度为 $\mathrm{O}(n\log n)$, 计算各活动当前的最早和最晚开始时间的复杂度均为 $\mathrm{O}(nKD)$, 算法迭代次数上限为 n, 算法 LESA 的时间复杂度为 $\mathrm{O}(n^2\max(\log n, KD))$. 综上, 算法 TLPR 评估候选解的时间复杂度为 $\mathrm{O}(n^2\max(\log n, KD))$.

3 算法实验与比较

3.1 实验环境

测试问题集采用项目调度标准问题库 PSPLIB[16] 中的四组单执行模式项目调度问题测试集, 即 J30、J60、J90 和 J120, 每组中均包含 480 个问题实例, 需要 4 种可更新资源. 对 PSPLIB 中各问题实例做如下修改: 采用 β 分布模拟资源各时段的资源置信度, 考虑到项目构建时会对伙伴资源置信度进行评估, 因此设置较高的资源置信度; 采用服从离散均匀分布 $\mathrm{U}\sim[1,6]$ 的随机数生成所有非虚活动的执行偏差成本.

3.2 路径重连参数设置

种群规模、参考解数量和距离阈值三个参数的选择是影响路径重连算法应用效果的关键因素, 本文通过对 J30 问题应用多组参数的不同组合实验结果来确定参数的最优配置. 经实验观察, 种群规模设置为 28.

参考解数量和距离阈值的设置采用同最好解平均偏差百分比来进行度量, 这里的最好解是指在各种参数组合下求得的最好解. 令 AVG_{N_r} 为 P 个测试问题在参考解数量为 N_r 时目标值偏离最好解的平均百分比, AVG_{minD} 为 P 个测试问题在距离阈值为 $minD$ 时目标值偏离最好解的平均百分比, 本文 $P = 480$. 计算公式为

$$AVG_{N_r} = \frac{1}{P}\sum_{p=1}^{P}\left(\frac{Obj_p^* - Obj_{p,N_r}}{Obj_p^*}\right)\times 100 \tag{12}$$

$$AVG_{minD} = \frac{1}{P}\sum_{p=1}^{P}\left(\frac{Obj_p^* - Obj_{p,minD}}{Obj_p^*}\right)\times 100 \tag{13}$$

图 3 和图 4 分别给出了参考解数量和距离阈值同解质量间的关系.

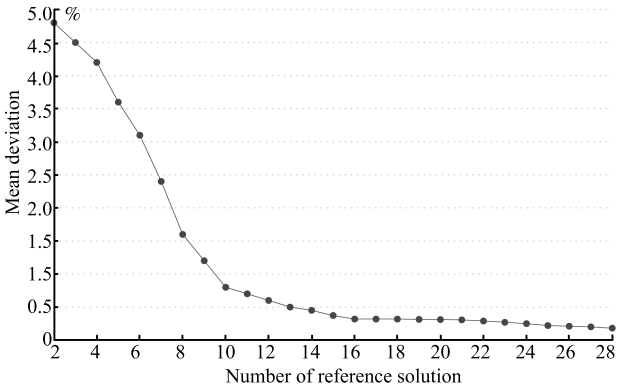

图 3　参考解数量对解质量的影响
Fig. 3　Influence on solution quality of number of reference solutions

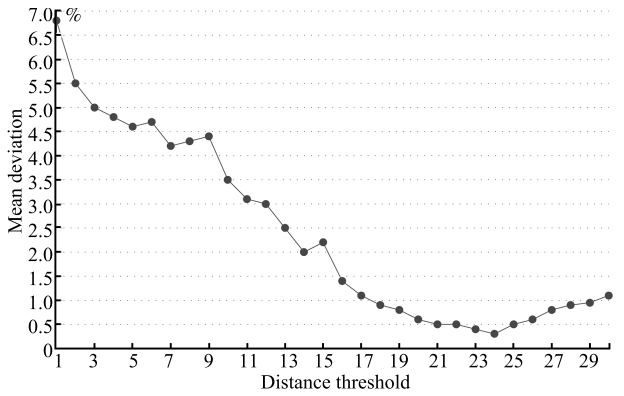

图 4　距离阈值对解质量的影响
Fig. 4　Influence on solution quality of distance threshold

从图 3 可见, 参考解数量较少会导致产生新解数量偏少, 限制了搜索范围; 数量过高时, 解质量无明显提升, 而由于需要进行路径重连的解对组合数量大幅度增加, 计算量显著增大, 降低了算法性能. 故参考解数量选择寻优能力较为稳定的区间, 取 16 ~ 20 最为合适. 从图 4 可见, 距离阈值较小时, 产生新解数量少, 缩小了寻优范围, 丢失优秀解的概率增大; 当距离阈值过大时, 许多优秀解由于距离阈值太大而无法被选入参考集, 进而丧失产生优秀解的机会. 实验结果表明距离阈值介于最大阈值 (即项目活动数) 的 70 % ~ 90 % 之间能取得比较理想的优化效果.

3.3 算法实例

采用 PSPLIB 问题库中的 J30_21 测试问题来给出 TLPR 的算法实例. 图 5 给出了以工期最短为目标, 无时

间缓冲的调度结果甘特图, 而采用 TLPR 算法, 鲁棒性调度结果甘特图如图 6 所示, 其中黑色矩形表示插入的时间缓冲.

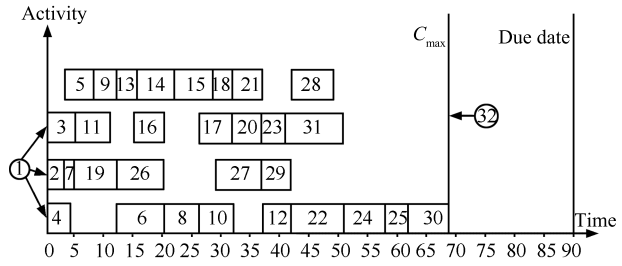

图 5 以工期最短为目标的调度结果甘特图

Fig. 5 Gantt chart of the scheduling result for minimizing project duration

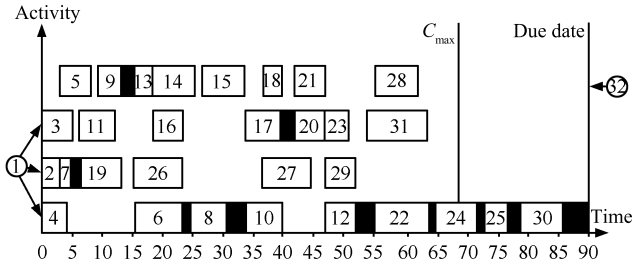

图 6 以鲁棒性为目标的调度结果甘特图

Fig. 6 Gantt chart of the scheduling result for maximizing project robustness

3.4 与相关方法的比较

3.4.1 对比方法的实现

据笔者所知, 目前相关研究中还没有与本文在优化目标和约束条件方面完全一致的模型, 考虑到本文方法的主要特点是在考虑资源置信度基础上, 通过插入时间缓冲来实现项目调度的鲁棒性, 为定量说明本文所提方法的有效性和优势, 选取了以插入时间缓冲方式实现调度鲁棒性的两个主要的项目调度方法 — 关键链方法 (Critical chain project management, CCPM) 和 RFDFF 方法, 同本文所提方法进行比较. 其中, CCPM 作为调度质量鲁棒性的代表, RFDFF 则作为调度解鲁棒性的代表, 同时为分析局部增强搜索 LESA 算法对鲁棒性的优化效果, 将不采用 LESA 算法的路径重连算法命名为 TLPR-NL.

CCPM 通过项目缓冲、输送缓冲和资源缓冲机制来消除不确定因素, 保证确定环境下的编制的项目计划在动态环境下的顺利执行. CCPM 在关键链的确定和各种缓冲的设置方面还有很多待深入研究的地方, 目前尚无公认的最优算法, 本文采用目前较优的文献 [10−11] 提出的关键链和缓冲区设置方法作为 CCPM 的主要实现算法. RFDFF[9] 是典型的考虑确定性活动执行时间, 通过将时间缓冲分散的插入到各活动之前来实现调度解鲁棒性的方法, 被众多研究作为度量调度解鲁棒性的标准评价方法.

3.4.2 模拟执行的设定

由于约束条件同本文方法考虑上的不同, 在进行仿真实验时适当修改了 CCPM 和 RFDFF 中的部分参数和约束条件.

项目交货期的设定方面, 以 CCPM 中关键链长度和 RFDFF 中最短工期的最大者作为项目最短工期, 记为 $C_{\max}$. 为测试交货期对调度鲁棒性的影响, 指定的交货期 D 分别设置为 $\lfloor 1.3 \times C_{\max} \rfloor$、$\lfloor 1.5 \times C_{\max} \rfloor$ 和 $\lfloor 1.8 \times C_{\max} \rfloor$, 下文分别简记为 1.3 C、1.5 C 和 1.8 C. 对于 CCPM 方法, 延长的 30 %、50 % 和 80 % 工期, 作为项目缓冲放置在项目执行的末尾.

资源不确定性的模拟方面, 分别按照低 (5 %)、中 (10 %)、高 (15 %) 三种比例在项目交货期范围 $[1, D]$ 内选取时段, 资源置信度越低的时段被选中的概率越大. 在这些时段内从四种资源中选取一种资源设置为不可用, 本文假定同一时段只允许一种资源不可用, 并采用服从离散均匀分布 U $\sim$ [1, 3] 的随机数模拟不可用持续的时段数. 低、中、高三种资源中断的程度分别简记为 L-RU、M-RU 和 H-RU. 对于由资源不可用导致的活动执行中断, 仿真实验时只考虑资源恢复后活动重新执行的情况, 如果出现调度不再可行的情况, 则采用 “右移” (Right-shifted) 的再调度策略.

对于项目调度的质量鲁棒性, 采用项目在指定工期内完工概率 (Timely project completion probability, TPCP) 进行度量; 对于解鲁棒性, 由于 CCPM 和 RFDFF 方法中均没有考虑资源置信度, 比较时无法使用式 (3) 中的目标函数, 为公平比较, 实验中采用文献 [17] 中提出的公式 $\sum_{i=1}^{n} w_i |s_i^* - s_i|$ 来度量解鲁棒性, 其中 s_i^* 和 s_i 分别为实际执行开始时间和计划开始时间, 即项目的总执行偏差成本 (Total activity deviation cost, TADC).

活动的执行偏差成本是 RFDFF、TLPR-NL 和 TLPR 三种方法设置时间缓冲时需要考虑的重要因素, 项目结束活动 n 的执行偏差成本决定了设置在项目执行末端的时间缓冲量, 意味着对项目交货期的保护程度. 定义 CWP 为活动 n 的执行偏差成本与其他所有活动执行偏差成本平均值的比例. 图 7 给出了针对 TLPR 算法, J30 测试集、资源中断程度取 M-RU、不同的工期松弛度下 CWP 取值对 TPCP 值的影响. 图 8 给出了 J30 测试集、1.5 C 的工期松弛、不同资源中断程度下 CWP 取值对 TADC 值的影响. 从图 7 中可见, 随着 CWP 值的增加, 三种工期松弛度下的 TPCP 值均有显著增加, 要让 TPCP 值达到 95 % 以上, 1.8 C、1.5 C 和 1.3 C 三种情况下 CWP 分别至少要达到 4、8 和 11. 从图 8 中可以看出, 资源中断程度越高项目的总执行偏差成本越大,

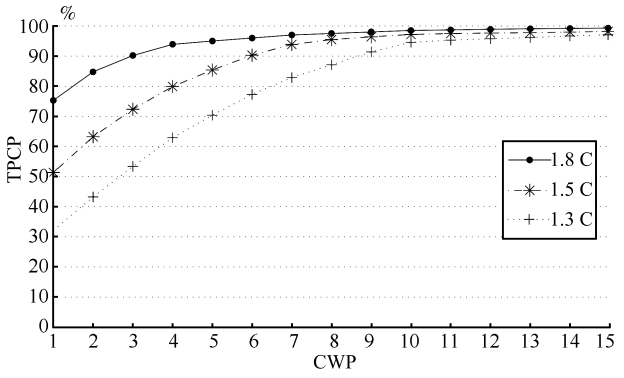

图 7 TLPR 算法中 CWP 取值对 TPCP 的影响

Fig. 7 Influence on TPCP value of different CWP values in TLPR algorithm

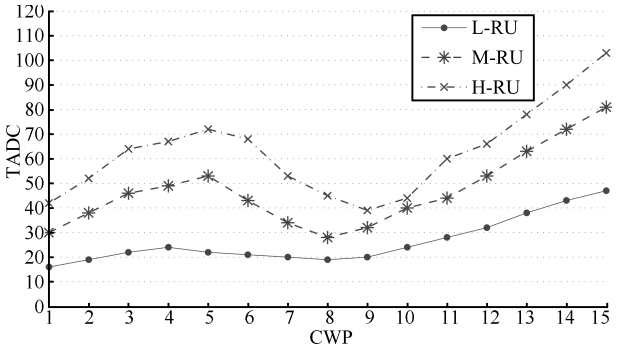

图 8　TLPR 算法中 CWP 取值对 TADC 的影响

Fig. 8　Influence on TADC value of different CWP values in TLPR algorithm

在 CWP 开始逐步增加的时候, 项目的 TADC 值逐步增大, 这是因为 TLPR 算法将越来越多的缓冲置于项目执行末端保护工期, 其余活动插入的时间缓冲变小, 造成 TADC 增大, 而由于 CWP 还不够大, 项目不能按时完工的概率仍然很高, 活动 n 的执行偏差成本大部分情况下仍会加入到 TADC 中. 当 CWP 增大到一定程度后, 虽然对其余活动的保护继续减少, 但是项目的完工概率迅速增大, 活动 n 的执行偏差成本不会加入到 TADC 中, 项目的 TADC 反而会呈现缓慢下降趋势; 但当 CWP 达到一定程度后, TPCP 值已经趋于上限, 再增加 CWP 值不会对 TPCP 值产生太大的效果, 但由于分派给其余活动缓冲量的减少, TADC 值会迅速增加.

因此, 在综合衡量 CWP 值对 TPCP 和 TADC 的影响情况下, 本文仿真实验中取 CWP = 8, 这样既能保证较高的 TPCP 值 (> 95 %), 同时 TADC 值也处于较低水平.

3.4.3　质量鲁棒性的分析与比较

表 1 中给出了四种方法在不同工期松弛程度和资源中断程度下, 在四种问题规模的测试集下的 TPCP 值.

从资源不确定性的角度看, 随着资源中断程度的增强, 项目执行受到的干扰增大, 四种方法的 TPCP 值均会下降. 当资源不确定性较低时 (L-RU), TLPR 和 TLPR-NL 要高于 RFDFF 的值, 但差距并不明显, 说明 RFDFF 与本文方法的时间缓冲机制均可以吸收大部分活动工期的微小增加. 但当资源不确定性较高时 (M-RU 和 H-RU), TLPR 和 TLPR-NL 相对于 RFDFF 的优势则较为明显, 而与 CCPM 方法的差距也迅速缩小, 当工期为 1.3 C 和资源不确定度为 H-RU 时, TLPR 和 TLPR-NL 的 TPCP 值基本接近了 CCPM 方法, 这主要得益于本文算法在插入时间缓冲时对资源置信度因素的特殊考虑.

从工期松弛的影响来看, CCPM 方法对工期松弛程度要求较低, 30 % 工期的延长即可达到较高的 TPCP 值, 而 TLPR 和 TLPR-NL 算法在工期松弛 50 % 的情况下会取得比较好的 TPCP 值, RFDFF 则需要更长的工期松弛才能达到较好的保护程度.

从项目活动数的影响来看, CCPM 方法当活动数增大时 (J90、J120 测试集), 在 1.5 C 和 1.8 C 的情况下, TPCP 值非常高, 这可能是随着活动数增加, 关键链也随之加长, 进而设置的项目缓冲值也在增加, 使 TPCP 值增加明显, 但这有可能会导致对工期的过度保护. RFDFF、TLPR 和 TLPR-NL 三种方法的 TPCP 对活动数并不敏感,　四种测试集中在相同的工期松弛情况下, TPCP 变化不明显, 这可能是因为这三种方法的时间缓冲插入策略是分散插入在各活动之前的缘故.

TLPR 和 TLPR-NL 间的对比上, 在调度的质量鲁棒性方面, TLPR 要略优于 TLPR-NR, 但是差距并不大, 这主要是因为 TLPR 是在 TLPR-NL 基础上对各活动的时间缓冲设置进行了局部增强搜索后的再调整, 并没有针对结束活动 n 进行特殊处理.

总体上, CCPM 方法在质量鲁棒性方面是四种算法中较优的, 其次为本文提出的两种方法. CCPM 方法得益于采用了项目缓冲对工期的特殊保护机制, 将大部分缓冲设置在了项目执行后部, 而其余三种方法对项目工期的保护是根据结束活动 n 的执行偏差成本大小来启发式设置缓冲保护. 要特殊说明的是, 表 1 中的对比数据是在 CWP = 8 的情况下进

表 1　四种算法在各种参数组合下 TPCP 值的比较

Table 1　Comparison of TPCP value among the four algorithms with various parameters

测试集	项目工期	CCPM			RFDFF			TLPR-NL			TLPR		
		L-RU	M-RU	H-RU	L-RU	M-RU	H-RU	L-RU	M-RU	H-RU	L-RU	M-RU	H-RU
J30	1.3 C	93	88.15	83.54	91.52	81.87	76.23	91.86	85.57	81.71	92.13	86.42	82.92
J30	1.5 C	98.23	96.02	92.08	97.31	90.02	86.14	97.42	94.98	89.69	97.56	95.28	90.16
J30	1.8 C	99.96	99.79	98.13	99.13	96.13	89.98	99.24	98.44	95.93	99.13	98.36	96.25
J60	1.3 C	94.04	89.06	82.79	92.3	82.05	79.54	92.17	87.26	82.61	93.65	87.71	83.14
J60	1.5 C	98.31	97.15	92.58	95.9	92.79	84.71	96.81	95.63	91.26	97.33	96.14	91.47
J60	1.8 C	100	99.96	98.94	98.96	94.25	90.59	99.52	97.86	96.32	99.83	98.63	97.12
J90	1.3 C	95.06	92.6	86.44	93.26	82.56	78.57	93.14	87.9	85.27	93.38	88.59	85.97
J90	1.5 C	99.98	98.23	95.63	97.05	92.38	85.05	97.98	96.14	92.75	98.48	95.42	93.46
J90	1.8 C	100	100	99.92	98.78	95.39	91.62	99.14	98.78	97.04	99.77	99.12	97.56
J120	1.3 C	97	95.83	86.25	93.12	81.32	79.85	92.46	85.87	83.42	92.85	86.26	85.13
J120	1.5 C	100	99.19	97.44	97.63	89.71	85.26	98.52	95.33	92.41	98.63	96.98	92.28
J120	1.8 C	100	100	99.96	99.12	93.41	90.73	99.37	97.65	95.42	99.56	98.96	97.75

行的, 经过实验, 当 CWP 值进一步提高时, RFDFF、TLPR 和 TLPR-NL 的 TPCP 值均会进一步提高. 针对 J30 测试集, 当 CWP = 11 时, TLPR 和 TLPR-NL 基本上可获得同 CCPM 方法相当的完工概率, 这是因为当为活动 n 设置较大执行偏差成本时, TLPR 和 TLPR-NL 会更加关注对项目工期的保护, 但是从图 7 和图 8 可以看出, 过大的 CWP 会导致执行偏差成本增加, 因此项目管理者选择策略时需要在二者之间做出均衡考虑. 在现实情况中, 绝大部分项目所用资源应具有较高的置信度, 即资源中断的概率应该处于 L-RU 或 M-RU 情况下居多, 而本文提出的方法在 L-RU、M-RU 和 50 % 工期松弛的情况下基本都可以达到 95 % 以上的按期完工率. 在 H-RU 和较为松弛的工期约束下也可以达到 90 % 的按期完工率, 且可以通过提高 CWP 值进一步提高按期完工率.

3.4.4 解鲁棒性的分析与比较

采用 J30 和 J120 分别作为小规模和大规模问题的示例, 表 2 和表 3 分别给出了 CWP = 8 时四种算法在 J30 和 J120 测试集中各种参数下的总执行偏差成本 TADC 值. 其中的最好值 (Best)、最差值 (Worst) 和平均值 (Avg) 为各测试集中 480 个实例分别运行 10 次后取到的总平均值.

从资源不确定性的影响看, 本文方法在相同的工期松弛程度情况下, 解鲁棒性方面都明显优于 CCPM 和 RFDFF 方法. 从 TADC 的平均值对比看, TLPR-NL 为 CCPM 的 40 % ∼ 50 %, 为 RFDFF 方法的 70 % ∼ 80 %, 而 TLPR 则要更好, 这主要还是因为 TLPR-NL、TLPR 和 RFDFF 三种方法的时间缓冲策略是采用分散在各活动中, 对各活动的缓冲时间均有一定的保护, 而 CCPM 则是将大量的时间缓冲放置在了项目的最后来保护工期, 对非关键链上的项目开始时间保护不够. 另外, 本文方法相对于 RFDFF 方法而言, 资源不确定性越高的情况下 (H-RU) 性能优势越明显, 这主要是因为本文两种算法考虑到了资源置信度的约束, 在插入时间缓冲时有针对资源置信度的特殊处理.

从工期的松弛程度分析, 四种方法的 TADC 值都随着工期松弛程度的增加而减少, 这是因为对于以插入时间缓冲方式来提高调度鲁棒性的算法而言, 工期松弛程度越大, 则可以插入的时间缓冲越多, 对活动的保护越好.

从活动数量的影响看, 对比表 2 和表 3 中的数据, 可以看出 CCPM 在活动数大幅增加的情况下, 解鲁棒性性能下降的较为明显, 这可能是因为活动数越大, 关键链越长, 项目缓冲设置的也越大, 相对来说分配给各活动的缓冲增加并不多, 而其余三种方法对活动数的增加并不敏感,TADC 的增加基本上同活动数的增加成比例.

对比文本的两种方法, TLPR 在解鲁棒性方面要优于 TLPR-NL, TADC 值约有 20 % 左右的下降. 而且工期松弛程度越高, 优势越明显, 这是因为项目工期松弛时间越大, 则局部增强搜索的进一步优化鲁棒性的余地越大.

总之, 在解鲁棒性方面, 本文提出的 TLPR、TLPR-NL 方法要优于 RFDFF 和 CCPM, 特别是在高资源中断程度的情况下, 优势更为明显.

表 2 J30 测试集下四种方法 TADC 比较

Table 2 Comparison of TADC value between the four algorithms with the J30 instance set

资源不确定性	项目工期	CCPM			RFDFF			TLPR-NL			TLPR		
		Worst	Best	Avg	Worst	Best	Avg	Worst	Best	Avg	Worst	Best	Avg
L-RU	1.3 C	87	35	59	77	30	47	66	25	36	44	18	31
L-RU	1.5 C	70	26	41	61	23	31	49	20	27	32	13	19
L-RU	1.8 C	52	21	24	33	15	19	30	12	17	22	8	13
M-RU	1.3 C	105	50	71	95	39	60	82	31	51	58	24	38
M-RU	1.5 C	90	37	55	79	27	46	66	24	34	48	18	28
M-RU	1.8 C	75	24	40	66	21	34	53	15	27	37	12	19
H-RU	1.3 C	122	64	87	111	53	76	90	38	59	68	32	47
H-RU	1.5 C	111	45	74	96	35	62	77	27	47	56	23	39
H-RU	1.8 C	92	31	57	84	25	47	60	18	31	47	14	25

表 3 J120 测试集下四种方法 TADC 比较

Table 3 Comparison of TADC value between the four algorithms with the J120 instance set

资源不确定性	项目工期	CCPM			RFDFF			TLPR-NL			TLPR		
		Worst	Best	Avg	Worst	Best	Avg	Worst	Best	Avg	Worst	Best	Avg
L-RU	1.3 C	330	145	247	262	111	175	206	87	124	147	64	90
L-RU	1.5 C	262	111	172	206	86	121	150	72	96	112	47	67
L-RU	1.8 C	189	88	107	106	60	79	91	46	64	80	32	49
M-RU	1.3 C	399	203	292	326	143	223	258	105	172	192	82	104
M-RU	1.5 C	341	152	229	270	103	172	205	83	116	160	65	88
M-RU	1.8 C	280	102	171	222	79	130	164	56	93	125	44	69
H-RU	1.3 C	470	262	359	381	191	279	283	127	196	223	107	155
H-RU	1.5 C	422	183	308	329	129	229	242	93	158	184	79	115
H-RU	1.8 C	350	131	240	285	95	177	187	66	108	159	52	85

3.4.5 方法比较总结

从以上各节的分析可以看出, 得益于考虑资源置信度的时间缓冲插入机制, 本文所提方法 TLPR 和 TLPR-NL 在调度的解鲁棒性方面比相关方法 RFDFF 方法和 CCPM 方法有比较大的优势, 特别是在资源不确定性较高的环境下优势更加明显. 在质量鲁棒性方面, 本文所提方法同 CCPM 方法在按期完工率方面接近, 均强于 RFDFF 方法. 但是在相同的按期完工率下, 总执行偏差成本要远低于 CCPM 方法, 同时在保持较低的总执行偏差成本前提下, 本文方法仍能达到较高的按期完工率 ($> 95\,\%$), 并且在实际应用中可以通过调整活动 n 的执行偏差成本来进一步提高按期完工率. 采用局部增强搜索的 TLPR 方法在优化效果上要好于 TLPR-NL 方法, 但需要更多的计算时间, 项目管理者可以根据对计算速度和优化性能的均衡考虑进行选择. 综合来看, 本文所提方法考虑了项目资源的置信度, 兼顾了解鲁棒性和质量鲁棒性, 同时提供灵活的参数设置让项目管理者根据实际应用中的侧重进行相应配置, 有较好的性能和适用性.

4 结论

资源不确定性高和调度鲁棒性要求高是跨企业项目调度问题同传统项目调度问题的主要区别, 本文建立的考虑资源置信度的跨企业项目鲁棒性调度优化模型, 考虑了资源不确定性这一重要因素, 优化目标上兼顾了调度的解鲁棒性和质量鲁棒性. 以路径重连方法为求解框架, 嵌入的启发式时间缓冲插入算法能够依据活动的执行偏差成本、现有缓冲状态等特征, 合理、快速地生成鲁棒性调度, 进而可通过局部增强搜索算法对调度鲁棒性进行深入优化. 通过与 RFDFF 方法和关键链方法的比较, 本文方法在资源不确定性高的环境下, 解鲁棒性方面有较大的优势, 在保持较低的执行偏差成本情况下, 质量鲁棒性同 CCPM 相近, 优于 RFDFF 方法, 并可通过调整结束活动的执行偏差成本来进一步提高质量鲁棒性.

今后的研究工作将放在资源不确定性较高的调度环境中的再调度方法和策略方面, 将再调度方法同鲁棒性调度相结合, 提出更好的跨企业项目计划与调度方案.

References

1 Brucker P, Drexl A, Möhring A, Neumann K, Pesch E. Resource-constrained project scheduling: notation, classification, models, and methods. *European Journal of Operational Research*, 1999, **112**(1): 3−41

2 Herroelen W, Leus R. Project scheduling under uncertainty: survey and research potentials. *European Journal of Operational Research*, 2005, **165**(2): 289−306

3 Al-Fawzan M A, Haouari M. A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics*, 2005, **96**(2): 175−187

4 Kobylański P, Kuchta D. A note on the paper by M. A. Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling. *International Journal of Production Economics*, 2007, **107**(2): 496−501

5 Lambrechts O, Demeulemeester E, Herroelen W. A tabu search procedure for developing robust predictive project schedules. *International Journal of Production Economics*, 2008, **111**(2): 493−508

6 Tavares L V, Ferreira J A A, Coelho J S. On the optimal management of project risk. *European Journal of Operational Research*, 1998, **107**(2): 451−469

7 Herroelen W, Leus R. The construction of stable project baseline schedules. *European Journal of Operational Research*, 2004, **156**(3): 550−565

8 Vonder S, Demeulemeester E, Herroelen W, Leus R. The use of buffers in project management: the trade-off between stability and makespan. *International Journal of Production Economics*, 2005, **97**(2): 227−240

9 Vonder S, Demeulemeester E, Hemeulemeester W, Leus R. The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research*, 2006, **44**(2): 215−236

10 Liu Shi-Xin, Song Jian-Hai, Tang Jia-Fu. Approach for setting time buffers in resources-constrained project scheduling. *Journal of Systems Engineering*, 2006, **21**(4): 381−386
(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. 系统工程学报, 2006, **21**(4): 381−386)

11 Liu Shi-Xin, Song Jian-Hai, Tang Jia-Fu. Critical chain based approach for resource-constrained project scheduling. *Acta Automatica Sinica*, 2006, **32**(1): 60−66
(刘士新, 宋健海, 唐加福. 基于关键链的资源受限项目调度新方法. 自动化学报, 2006, **32**(1): 60−66)

12 Vonder S, Demeulemeester E, Herroelen W. Proactive heuristic procedures for robust project scheduling: an experimental analysis. *European Journal of Operational Research*, 2008, **189**(3): 723−733

13 Lambrechts O, Demeulemeester E, Herroelen W. Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of Scheduling*, 2008, **11**(2): 121−136

14 Leus R, Herroelen W. The complexity of machine scheduling for stability with a single disrupted Job. *Operations Research Letters*, 2005, **33**(2): 151−156

15 Wang Xiao-Qing, Tang Jia-Fu, Han Yi. Advances in scatter search. *Journal of System Simulation*, 2009, **21**(11): 3155−3160
(王晓晴, 唐加福, 韩毅. 分散搜索算法研究进展. 系统仿真学报, **21**(11): 3155−3160)

16 Kolisch R, Sprecher A. PSPLIB — a project scheduling problem library: OR software — ORSEP operations research software exchange program. *European Journal of Operational Research*, 1997, **96**(1): 205−216

17 Herroelen W, Leus R. Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, 2004, **42**(8): 1599−1620

徐汉川 哈尔滨工业大学计算机科学与技术学院讲师. 主要研究方向为服务计算, 项目计划与调度优化. 本文通信作者.
E-mail: xhc@hit.edu.cn
(**XU Han-Chuan** Lecturer at the School of Computer Science and Technology, Harbin Institute of Technology. His research interest covers services computing, project planning and scheduling. Corresponding author of this paper.)

徐晓飞 哈尔滨工业大学计算机科学与技术学院教授. 主要研究方向为服务工程与服务计算, 企业计算与互操作. E-mail: xiaofei@hit.edu.cn
(**XU Xiao-Fei** Professor at the School of Computer Science and Technology, Harbin Institute of Technology. His research interest covers services computing and services engineering, enterprise computing and enterprise interoperability.)