

基于随机活动工期的资源约束项目鲁棒性调度优化

何正文, 刘人境, 徐渝

(西安交通大学 管理学院, 西安 710049)

摘要 项目进度计划的鲁棒性对于不确定条件下项目的顺利实施具有重要影响。作者研究具有随机活动工期的资源约束项目鲁棒性调度问题, 目标是在可更新资源和项目工期约束下安排活动的开始时间, 以实现项目进度计划鲁棒性的最大化。首先对所研究问题进行界定并用一个示例对其进行说明。随后构建问题的优化模型, 设计禁忌搜索、多重迭代和随机生成三种启发式算法。最后在随机生成的标准算例集合上对算法进行测试, 分析项目活动数、项目工期和资源强度等参数对算法绩效的影响, 并用一个算例对研究进行说明, 得到如下结论: 禁忌搜索的满意解质量明显高于其他两种算法; 当资源强度或项目工期增大时, 平均目标函数值上升, 禁忌搜索的求解优势增强。研究成果可为不确定条件下项目进度计划的制定提供决策支持。

关键词 资源约束项目调度; 优化模型; 启发式算法; 随机活动工期; 鲁棒性

Robust scheduling optimization for resource-constrained project based on random duration of activities

HE Zheng-wen, LIU Ren-jing, XU Yu

(School of Management, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract The robustness of project schedule plays an important role for the smooth execution of project under uncertain conditions. The authors studied the resource-constrained project robust scheduling problem with random duration of activities. The objective was to arrange the start time of activities so as to maximize the robustness of project schedule under the constraints of renewable resources and project duration. The studied problem was identified and illustrated by using an example at first. Then the optimization model was constructed and three heuristic algorithms, including tabu search, multiple iteration, and random generation, were developed. Ultimately, the algorithms were tested on a standard instance set generated randomly. The influences of activity number, project duration, and resource strength on the algorithms' performance were analyzed. An instance was utilized to illustrate the research in the paper and the following conclusions were drawn: The solution quality of the tabu search is remarkably higher than that of the other two algorithms; as the resource strength or the project duration increase, the average value of objective function climbs and the quality advantage of the desirable solution of the tabu search augments. The research in this paper can provide decision supports for the preparation of project schedule under uncertain conditions.

Keywords resource-constrained project scheduling; optimization model; heuristic algorithm; random duration of activities; robustness

1 引言

经典的项目调度问题(project scheduling problem)研究多属于确定型的, 即在假定内外部环境均不会发生变化的条件下, 寻求满足给定目标要求的最优(或满意)进度计划安排^[1-2]。然而在现实中, 绝大多数项目都不同程度地存在一定的不确定性, 如果在制定进度计划时不将其这一特性考虑在内, 那么在随后的项目实施过程中, 计划便会因不确定因素的影响而频繁地发生调整, 从而失去其有效性进而给项目的组织和协调带

收稿日期: 2010-11-19

资助项目: 国家自然科学基金(70971105); 陕西省自然科学基金(2009JM9001); 中央高校基本科研业务费专项资金

作者简介: 何正文(1967-), 副教授, 博士, 研究方向: 项目管理及优化, E-mail: zhengwenhe@mail.xjtu.edu.cn.

来混乱^[3].

鲁棒性 (robustness) 是指项目进度计划在内外部环境发生变化的条件下, 保持其稳定性的能力^[4]. 显然, 对于那些不确定性较大的项目来说, 其进度计划必须具有较高的鲁棒性, 这样它才能有效地抵御内外部诸多干扰因素的影响, 充分发挥其对项目实施的指导作用. 关于不确定条件下的项目鲁棒性调度问题, 国外一些学者已开始对其进行研究. Herroelen 和 Leus^[5] 设计并评价了几种能够生成稳定性基准进度计划的启发式算法, 目标是在存在单个扰动条件下最小化活动开始时间的期望加权偏差. Van de Vonder 等^[6] 利用仿真实验研究了质量鲁棒性 (指项目工期的稳定性) 和解鲁棒性 (指解在执行中的稳定性) 之间的权衡关系, 回答了时间缓冲是集中使用还是分散使用的问题. Van de Vonder 等^[7] 开发了一种解鲁棒性最大化启发式算法, 仿真分析了资源约束下的计划稳定性和工期之间的权衡关系. Al-Fawzan 和 Haouari^[8] 同时考虑工期最小化和鲁棒性最大化, 构建了双目标资源约束项目调度模型并设计了禁忌搜索启发式算法. Lambrechts 等^[9] 开发了一种禁忌搜索启发式算法以生成鲁棒性项目基准进度计划, 目标是最大化进度计划的解鲁棒性. Van de Vonder 等^[10] 通过一个实验, 比较分析了多种鲁棒性项目调度启发式算法在不同情形下的绩效. 然而, 与经典的确定型项目调度问题研究相比, 关于不确定条件下项目鲁棒性调度问题的研究目前仍然较为缺乏, 亟待进一步的深入和扩展^[11].

鉴于各种不确定因素的影响最终都会体现在活动工期的变化性上这一事实, 本文研究随机活动工期下的资源约束项目鲁棒性调度问题, 目标是在可更新资源 (renewable resource, 即可重复使用的资源如人力、设备、场地等) 和项目工期的约束下, 安排活动的开始时间以最大化项目进度计划的鲁棒性. 据作者所知, 该问题迄今尚无学者进行过深入研究. 在论文的后续部分, 作者首先界定所研究问题并用一个示例对其进行说明; 随后构建问题的优化模型; 针对其 NP-hard 属性设计启发式求解算法; 在随机生成的标准算例上对算法进行测试并用一个算例对研究进行说明; 最后总结全文并给出研究结论.

2 问题界定

本文采用基于活动 (activity-based) 的研究方法^[12], 即将项目表示为一个 AoN (Activity-on-Node) 网络, 其中节点代表活动, 箭线代表活动之间的逻辑关系. 考虑一个由 N 个活动构成的项目, 出于网络表述的需要, 额外添加两个虚活动: 活动 0 和活动 $N+1$, 分别表示项目的开始和结束. 项目的实施需要 K 种可更新资源, 第 k ($k=1, 2, \dots, K$) 种可更新资源的可用量为 R_k . 活动 n ($n=0, 1, \dots, N+1$) 执行时在单位时间里对第 k 种可更新资源的需求量为 r_{nk} , 由于不确定因素的影响, 其工期 d_n 为一均值和标准差分别为 $\mu(d_n)$ 和 $\sigma(d_n)$ 的随机变量. 注意, 前述两个虚活动对每种可更新资源的需求量及其工期均恒为 0. 计划的项目工期为 D . 需要指出的是, 由于活动工期的随机性, 项目工期 D 实质上为一柔性约束, 即该约束在项目进度计划制定时必须考虑并遵守, 但在计划的执行过程中却有可能被突破.

现假定在制定项目进度计划时, 基于工期均值 $\mu(d_n)$ 将活动 n 的开始时间安排为 s_n . 在项目的实施过程中, 由于各活动的实际工期并不一定等于其均值, 因此活动的实际开始时间也必然会随之发生变化, 进而导致进度计划的稳定性遭到破坏. 为了降低这种情况出现的可能性, 实际中的项目管理者通常会在每个活动上留出一定的时间冗余, 借此吸收或减轻活动工期的变化对计划稳定性的影响. 鉴于这一事实, 在活动 n ($n=1, 2, \dots, N$) 的计划完成时间 (即 $s_n + \mu(d_n)$) 之后设置一段时间缓冲 B_n , 由此提高进度计划的稳定性. B_n 的设置通过活动开始时间 s_n 的安排来完成, 给定各活动的计划开始时间 s_n , B_n 可按下式计算:

$$B_n = \min_{m \in U_n} \{s_m\} - [s_n + \mu(d_n)], \quad n = 1, 2, \dots, N.$$

其中, U_n 为活动 n 的所有紧后活动的集合. 从上式可以看出, 当活动 n 的实际工期偏离 $\mu(d_n)$ 时, 只要其真正的偏离幅度不超过 B_n , 那么活动 n 的紧后活动 m 的计划开始时间 s_m 就无需进行调整, 从而保证后续进度计划不受该偏离的影响.

对于一个给定项目, 各活动的时间缓冲越大, 进度计划的鲁棒性就越高. 然而, 由于存在项目工期 D 的约束, 活动的时间缓冲不可能无限制地随意设置. 因此, 必须将有限的时间缓冲合理地安排到活动上去, 以有效提高项目进度计划的鲁棒性. 为了完成这一任务, 需要为每个活动定义一个权重系数 ξ_n ($n=1, 2, \dots, N$) 并根据 ξ_n 设置 B_n 的大小. 显然, 活动 n 的 $\sigma(d_n)$ 越大, 其工期的变化性就越高, 需要的时间缓冲也越大. 为此, 基于 $\sigma(d_n)$ 将活动的权重系数 $\sigma(d_n)$ 定义如下: $\xi_n = \sigma(d_n) / \sum_{i=1}^N \sigma(d_i)$. 在项目的实施过程中, 变化性高的活动通常对进度计划稳定性的影响较大. 所以, 从另外一个视角来看, ξ_n 实际上反映了活动 n 上的单

位时间缓冲对进度计划鲁棒性的贡献。当活动 n 上的时间缓冲为 B_n 时, 其对进度计划鲁棒性的总贡献即可用 $\xi_n B_n$ 度量。基于上述理由, 把整个项目进度计划所拥有的总鲁棒性 $Robu$, 定义为所有活动上的时间缓冲对项目进度计划鲁棒性的贡献总和, 即 $Robu = \sum_{n=1}^N (\xi_n B_n)$ 。毋庸置疑, 当项目的总时间缓冲较高且在各活动之间进行了合理分配时, 则进度计划的 $Robu$ 就会比较高; 反之, 进度计划的 $Robu$ 就较低, 易受活动工期变化的影响而频繁地发生调整。在上述讨论的基础上, 最终把本文所研究问题界定为: 在可更新资源可用量 R_k 及项目工期 D 的约束下, 基于随机工期的均值 $\mu(d_n)$ 和标准差 $\sigma(d_n)$ 安排活动的开始时间 s_n , 以最大化项目进度计划的鲁棒性 $Robu$ 。

3 示例

用图 1 所示的一个示例项目对 2 中所界定的问题进行说明。该示例项目共包含 6 个活动, 其中 0 和 5 分别为虚的开始和结束活动, 1、2、3 和 4 为非虚活动。存在一种可更新资源限制, 其可用量 R_1 为 10。假定各活动的工期服从正态分布, 其均值、方差及活动对可更新资源的需求量均标注于图中。项目工期 D 为 20。

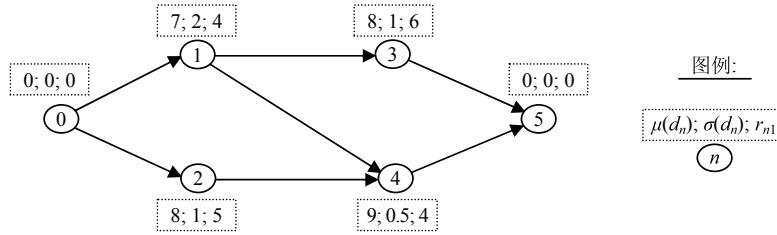


图 1 示例项目的 AoN 网络图

对于该示例项目, 首先以项目完成时间最早为目标, 在可更新资源及项目工期约束下, 基于活动工期的均值给出一个可行进度计划安排 (如图 2(a) 所示): $s_0 = 0, s_1 = 0, s_2 = 0, s_3 = 8, s_4 = 8, s_5 = 17$ 。在该进度计划安排下, 各非虚活动的时间缓冲为: $B_1 = 1, B_2 = 0, B_3 = 1, B_4 = 0$ 。根据活动工期的标准差, 可以方便地计算出各非虚活动的权重系数为: $\xi_1 = 0.45, \xi_2 = 0.22, \xi_3 = 0.22, \xi_4 = 0.11$, 由此可得上述进度计划的鲁棒性: $Robu = 0.67$ 。作为对比, 现以鲁棒性最大化为目标, 给出另一可行进度计划安排 (如图 2(b) 所示): $s_0 = 0, s_1 = 0, s_2 = 0, s_3 = 10, s_4 = 10, s_5 = 20$ 。在新的进度计划安排下, 各非虚活动的时间缓冲为: $B_1 = 3, B_2 = 2, B_3 = 2, B_4 = 1$ 。结合非虚活动的权重系数, 可计算出新进度计划的鲁棒性为: $Robu = 2.34$ 。

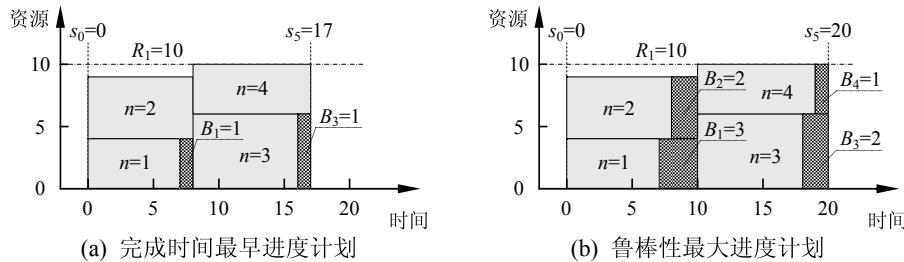


图 2 示例项目的进度计划安排

为了验证 $Robu$ 与进度计划稳定性之间的内在关系, 现基于活动工期的正态分布, 对上述两个进度计划安排的执行过程进行仿真。在仿真运行中, 只要有活动的开始时间晚于其计划时间, 则称进度计划发生了调整; 如果虚结束活动的开始时间晚于其计划时间, 则称项目未按计划完成。仿真次数设定为 10000 次, 得到的结果如下: 完成时间最早进度计划共有 8538 次发生了调整, 项目有 7980 次未按计划完成; 鲁棒性最大进度计划共有 1296 次发生了调整, 项目有 739 次未按计划完成。由此可以推论, 如果将项目进度计划由前者改为后者去实施, 那么进度计划发生调整及项目未按计划完成的概率, 将会由 85.4% 和 79.8% 分别显著下降至 13.0% 和 7.4%, 亦即项目进度计划的稳定性将会得到大幅度地提升。上述结果表明, 在随机活动工期条件下, 尽管图 2(a) 所示的进度计划目标是尽可能早地完成项目, 但由于其鲁棒性较低, 结果不仅难于实现预期目标, 而且极有可能造成进度计划在执行过程的调整, 进而引起资源配置及安排的变更并产生额外费用。相反, 如果执行图 2(b) 所示的进度计划, 那么, 由于该进度计划拥有较高的鲁棒性, 项目的实施过程将会平稳顺利得多, 反而更有利于项目的按时完成及预期目标的实现。

4 模型构建

根据上述对所研究问题的界定和说明, 构建基于随机活动工期的资源约束项目鲁棒性调度优化模型表述如下:

$$\text{Max} \quad Robu = \sum_{n=1}^N \left\{ \left[\sigma(d_n) / \sum_{i=1}^N \sigma(d_i) \right] [\min_{m \in U_n}(s_m) - (s_n + \mu(d_n))] \right\} \quad (1)$$

$$\text{s.t.} \quad s_0 = 0 \quad (2)$$

$$s_n + \mu(d_n) \leq s_m \quad m \in U_n; \quad n = 0, 1, \dots, N \quad (3)$$

$$s_{N+1} \leq D \quad (4)$$

$$\sum_{n \in V^T} r_{nk} \leq R_k \quad T = 0, 1, \dots, D; \quad k = 1, 2, \dots, K \quad (5)$$

$$s_n \text{ 为非负整数} \quad n = 0, 1, \dots, N+1 \quad (6)$$

其中, V_T 为在 T 时刻正在进行的活动的集合.

上述优化模型为一整数规划优化模型. 其中, 目标要求式 (1) 最大化项目进度计划的鲁棒 $Robu$. 约束条件式 (2) 将虚活动 0 的计划开始时间 s_0 (这也是整个项目的计划开始时间) 定义为 0 时刻; 式 (3) 为网络优先关系约束, 确保活动 n 的计划开始时间 s_n 与其工期均值 $\mu(d_n)$ 之和不晚于其紧后活动 m 的计划开始时间 s_m ; 式 (4) 为项目工期约束, 保证虚结束活动 $N+1$ 的计划开始时间 s_{N+1} (这也是整个项目的计划完成时间) 不超过项目工期 D ; 式 (5) 为可更新资源约束, 使得在项目实施过程中的任意一个时刻 T ($T = 0, 1, \dots, D$), 所有正在进行活动对第 k ($k = 1, 2, \dots, K$) 种可更新资源的需求总量 $\sum_{n \in V^T} r_{nk}$ 不超过该种资源的可用量 R_k ; 式 (6) 为决策变量的定义域约束. 通过上述优化模型, 便可借助活动计划开始时间 s_n 的安排, 将时间缓冲合理地分配到工期变化性较大的活动上, 以取得项目进度计划鲁棒性的最大化, 从而尽最大可能地保证项目的平稳实施及预期目标的顺利实现.

5 算法设计

本文所研究问题为一活动工期为随机变量的、鲁棒性目标资源约束项目调度问题, 它可视为经典的资源约束项目调度问题 RCPSP(resource-constrained project scheduling problem) 向不确定条件下的一种扩展. 由于 RCPSP 已被证明为 NP-hard 问题^[13], 所以, 本文所研究问题也必然为一 NP-hard 问题. 鉴于上述原因, 作者采用启发式算法求解该问题. 本节首先设计问题求解的禁忌搜索启发式算法. 随后, 给出另外两种启发式算法(即多重迭代和随机生成), 并在下一节的测试中对这三种算法的绩效进行对比分析.

5.1 初始可行解构造

为了方便文章后续表述, 定义如下决策向量 S 表示问题的可行解: $S = (s_n; n = 0, 1, \dots, N+1)$. 问题的初始可行解 S^0 通过下述步骤随机生成:

步骤 1 在不考虑可更新资源约束式 (5) 的条件下, 基于活动工期均值 $\mu(d_n)$ 、项目工期 D 和项目网络结构, 利用关键路径法 CPM(critical path method) 计算活动 n 的时间窗 $[E_n, L_n]$. 其中, E_n 和 L_n 分别表示活动 n 的最早和最晚开始时间.

步骤 2 在不违反网络优先关系约束式 (3) 的条件下, 为每个活动 n ($n = 0, 1, \dots, N+1$) 在时间窗 $[E_n, L_n]$ 中随机地安排一个开始时间 s_n , 由此形成问题的一个解向量 S^0 .

步骤 3 从时刻 0 至 s_{N+1} , 在每个时点上判断可更新资源约束式 (5) 是否得到满足(注意, 活动 n 的持续时间从 s_n 一直计算到 $\min_{m \in U_n}\{s_m\}$), 若是则已得到一个可行的初始解 S^0 ; 否则, 返回步骤 2 并重复上述操作直至得到一个可行的 S^0 为止.

5.2 邻点生成机理

当前可行解 S^1 的一个可行邻点 S^2 按如下步骤随机生成:

步骤 1 在当前可行解 S^1 中随机地选择一个活动(虚活动 0 除外), 将该活动的开始时间在其时间窗中随机地变化一个单位, 由此得到当前解的一个邻点 S^2 .

步骤 2 检查网络优先关系约束式(3)是否得到满足,若是则继续后续操作;否则,调整其他活动的开始时间直至式(3)得到满足为止.

步骤 3 按照与初始可行解构造的步骤 3 中相同的方法,判断可更新资源约束式(5)是否得到满足,若是则已得到一个可行的邻点 S^2 ;否则,返回步骤 1 并重复上述操作直至得到一个可行的 S^2 为止.

5.3 禁忌搜索启发式算法

根据禁忌搜索的基本原理,可设计算法的搜索步骤如下所述:

步骤 1 输入初始可行解 S^0 及其对应的目标函数值 $Robu^0$; 定义算法的终止条件,即探测可行解总数 Num_{stop} ; 初始化禁忌列表 TL ; 令计数器 $Num = 0$; 将当前解 S^1 及当前最好解 S^* 赋值为 S^0 : $S^1 = S^* = S^0$, $Robu^1 = Robu^* = Robu^0$. 其中, $Robu^1$ 和 $Robu^*$ 分别为 S^1 和 S^* 所对应的目标函数值.

步骤 2 生成 S^1 的一个可行邻点 S^2 , 计算在 S^2 下的目标函数值 $Robu^2$. 检查生成该邻点的移动是否位于禁忌列表 TL 中,若是转步骤 4;否则,转步骤 3.

步骤 3 令 $S^1 = S^2$, $Robu^1 = Robu^2$, $Num = Num + 1$, 更新禁忌列表 TL . 如果 $Robu^2 > Robu^*$, 进一步令 $S^* = S^2$, $Robu^* = Robu^2$, 转步骤 5;否则,直接转步骤 5.

步骤 4 如果 $Robu^2 > Robu^*$, 激活生成该邻点移动的禁忌状态,令 $S^1 = S^* = S^2$, $Robu^1 = Robu^* = Robu^2$, $Num = Num + 1$, 更新禁忌列表 TL , 转步骤 5;否则,转步骤 2.

步骤 5 判断 $Num \geq Num_{stop}$ 是否成立,若成立转步骤 6;否则,转步骤 2.

步骤 6 输出得到的满意解,即 S^* 和 $Robu^*$.

在上述搜索过程中, 禁忌列表 TL 按如下方式进行管理: 每当生成当前解的一个可行邻点时, 该生成操作所对应移动的逆向移动从底部加入到 TL 中. 同时, 最早进入列表的逆向移动从顶部移出 TL , TL 中的其余逆向移动向上递进一位. 所有位于 TL 中的移动都是被禁忌的,但当一个被禁忌的移动能够生成比当前最好解还要好的邻点时,其禁忌状态可依据激活准则被解除,从而使算法能够移动到该邻点上,以避免错失问题的最好解.

5.4 多重迭代和随机生成启发式算法

多重迭代和随机生成已被多位学者用作禁忌搜索的对比算法^[14-16]. 多重迭代从一个与禁忌搜索相同的初始可行解开始,仅选择目标函数值发生改进的邻点进行迭代.为了避免陷入局部最优,当目标函数值无法改进时,算法从另外一个随机生成的可行解再重新开始迭代. 在探测的可行解总数达到规定的数目时,算法终止并输出保存的最好解为满意解. 随机生成则更为简单,该算法不进行邻域搜索,仅随机产生规定数目的可行解,从中选出最好解作为问题的满意解.

对于本文所研究的问题,多重迭代启发式算法的具体搜索过程如下:

步骤 1 输入初始可行解 S^0 及其目标函数值 $Robu^0$; 定义算法终止条件,即探测可行解总数 Num_{stop} ; 设定目标函数值无改进时允许探测的累计邻点总数 $TNum$; 令计数器 $Num_1 = 0$, $Num_2 = 0$, 其中, Num_1 记录已探测可行解的数目, Num_2 记录目标函数值无改进的累计邻点数目; 将当前解 S^1 及当前最好解 S^* 赋值为 S^0 : $S^1 = S^* = S^0$, $Robu^1 = Robu^* = Robu^0$.

步骤 2 生成 S^1 的一个可行邻点 S^2 , 计算 S^2 所对应的目标函数值 $Robu^2$. 判断 $Robu^2 > Robu^1$ 是否成立,若成立转步骤 3;否则,转步骤 4.

步骤 3 令 $S^1 = S^2$, $Robu^1 = Robu^2$, $Num_1 = Num_1 + 1$, $Num_2 = 0$. 如果 $Robu^2 > Robu^*$, 进一步令 $S^* = S^2$, $Robu^* = Robu^2$. 转步骤 5.

步骤 4 令 $Num_1 = Num_1 + 1$, $Num_2 = Num_2 + 1$. 判断 $Num_2 \geq TNum$ 是否成立,若成立则重新生成一个可行解并计算其目标函数值,分别记为 S^1 和 $Robu^1$,令 $Num_1 = Num_1 + 1$, $Num_2 = 0$, 转步骤 5;否则,直接转步骤 5.

步骤 5 判断 $Num_1 \geq Num_{stop}$ 是否成立,若成立则转步骤 6;否则,转步骤 2.

步骤 6 输出得到的满意解,即 S^* 和 $Robu^*$.

随机生成启发式算法的搜索步骤如下:

步骤 1 输入初始可行解 S^0 及其目标函数值 $Robu^0$; 定义算法终止条件,即探测可行解总数 Num_{stop} ; 令计数器 $Num = 0$; 将最好解 S^* 赋值为 S^0 : $S^* = S^0$, $Robu^* = Robu^0$.

步骤2 随机生成一个可行解 S^1 , 计算在 S^1 下的目标函数值 $Robu^1$, 令 $Num = Num + 1$. 判断 $Robu^1 > Robu^*$ 是否成立, 若成立则令 $S^* = S^1$, $Robu^* = Robu^1$. 转步骤 3.

步骤3 判断 $Num \geq Num_{stop}$ 是否成立, 若成立则转步骤 4; 否则, 转步骤 2.

步骤4 输出得到的满意解, 即 S^* 和 $Robu^*$.

6 算法测试

6.1 实验设计

在由项目调度问题算例生成器 ProGen^[17] 随机生成的标准算例集合上, 对上一节中所设计的三种启发式算法进行对比测试. ProGen 生成算例时的参数设置见表 1, 其中, 按全因子实验设计的参数有 3 个: 项目非虚活动数 N 、项目工期 D 和资源强度 RS (resource strength)^[18]. 参数 N 的取值为 4 种, D 和 RS 的取值均为 3 种, 在每种参数组合下生成的算例数为 10 个, 由此得到 $10 \times 4 \times 3 \times 3 = 360$ 个算例. 需要特别说明的是, 本文所研究问题存在项目工期和可更新资源的双重约束, 为了避免生成的算例没有可行解, 在算例生成过程中, 首先依据资源强度确定可更新资源的可用量, 然后利用串行 SGS(schedule generation scheme)^[19] 找到在该资源可用量约束下的项目最早完成时间 C_{max} , 最后以 C_{max} 为基准设置项目工期.

上述三种启发式算法采用 Visual Basic 6.0 编程, 在 CPU 主频为 1.6GHz、内存为 768MB 的个人计算机上运行. 对于某一给定算例 i , 分别使用三种启发式算法对其进行求解, 满意解的目标函数值记为 $Robu_i$, 算法的计算时间记为 CT_i . 令 $Robu_i^{best}$ 为三种算法所得到的最好满意解的目标函数值, 则称 RD_i ($RD_i = (Robu_i^{best} - Robu_i)/Robu_i^{best}$) 为满意解距离最好解的相对偏差. 定义如下 4 个指标以评价启发式算法的绩效:

- $ARD: ARD = \sum_{i=1}^I RD_i/I$, 满意解距离最好解的平均相对偏差.
- $MRD: MRD = \max_{1 \leq i \leq I} \{RD_i\}$, 满意解距离最好解的最大相对偏差.
- $ACT: ACT = \sum_{i=1}^I CT_i/I$, 算法的平均计算时间.
- $MCT: MCT = \max_{1 \leq i \leq I} \{CT_i\}$, 算法的最大计算时间.

其中, I 为某一算例子集中所包含的算例个数.

表 1 ProGen 的参数设置

ProGen 参数	取值
算例的非虚活动数 N	10, 20, 30 或 40
在某一非虚活动数下生成的算例数	10
算例的起始和终止活动数	从 2, 3 和 4 中随机选取
最大紧前和紧后活动数	4
活动工期的均值 $\mu(d_n)$	从 [10, 20] 中均匀地随机选取
活动工期的方差 $\sigma(d_n)$	从 [1, 5] 中均匀地随机选取
可更新资源的种类数 K	2
活动对资源 1 的需求量 r_{n1}	从 [1, 10] 中均匀地随机选取
活动对资源 2 的需求量 r_{n2}	从 [1, 10] 中均匀地随机选取
可更新资源的强度 RS	0.3, 0.5, 0.7
项目工期 D	$\rho \cdot C_{max}$, 其中 ρ 分别取为 1.1、1.3 和 1.5, C_{max} 为资源约束下的项目最早完成时间

6.2 算法参数

在对算法进行测试之前, 首先通过试验法为其中的相关参数确定一组相对满意的取值. 关于禁忌搜索, 存在如下两个关键参数: 探测可行解总数 Num_{stop} 和禁忌列表长度. 由于问题的可行解空间规模与项目活动数 N 之间存在一定的比例关系, 因此, 将 Num_{stop} 设置为 N 的某一倍数. 具体地, 先将禁忌列表长度设置为 20 不变, 在 Num_{stop} 分别取 $6000 \cdot N$ 、 $7000 \cdot N$ 、 $8000 \cdot N$ 、 $9000 \cdot N$ 和 $10000 \cdot N$ 下求解所有算例; 然后再将 Num_{stop} 设置为 $8000 \cdot N$ 不变, 在禁忌列表长度分别取 10、15、25 和 30 下求解所有算例, 得到的满意解的平均目标函数值及平均计算时间见表 2. 根据表 2 中的计算结果, 综合考虑平均目标函数值和平均计算时间随算法参数的变化情况, 将探测可行解总数设置为 $8000 \cdot N$, 将禁忌列表的长度选定为 20. 为使其他两种算法与禁忌搜索具有相同的终止条件, 其探测的可行解总数 Num_{stop} 也设置为 $8000 \cdot N$. 而对于多重迭代的算法参数 $TNum$, 基于表 3 中的计算结果将其设定为 15.

表 2 禁忌搜索启发式算法在算法参数不同取值下的计算结果

探测可行解总数 Num_{stop}	平均目标函数值	平均计算时间 (s)	禁忌列表长度	平均目标函数值	平均计算时间 (s)
6000·N	27.30	34.01	10	32.57	63.12
7000·N	31.59	48.25	15	32.85	63.52
8000·N	33.16	64.38	20	33.16	64.38
9000·N	33.48	77.13	25	33.16	66.85
10000·N	33.77	93.00	30	33.19	69.72

表 3 多重迭代改进启发式算法在 $TNum$ 不同取值下的计算结果

$TNum$	平均目标函数值	平均计算时间 (s)	$TNum$	平均目标函数值	平均计算时间 (s)
5	30.21	45.42	20	31.01	45.58
10	31.13	45.43	25	30.86	45.33
15	31.21	45.73	30	30.99	45.87

6.3 测试结果

在上述算法参数设置下的测试结果见表 4。关于全部算例以及 N 、 RS 和 D 分别取不同值的算例子集，表 4 给出了三种启发式算法前述 4 个评价指标的计算结果。注意，表 4 中 ARD 、 MRD 、 ACT 和 MCT 的上标 “ T ”、“ M ” 和 “ R ” 分别代表禁忌搜索、多重迭代和随机生成的相应测试指标。

表 4 算法测试结果

算例集合	ARD^T (%)	MRD^T (%)	ACT^T (s)	MCT^T (s)	ARD^M (%)	MRD^M (%)	ACT^M (s)	MCT^M (s)	ARD^R (%)	MRD^R (%)	ACT^R (s)	MCT^R (s)
$N=10$	1.07	2.31	9.13	13.36	1.06	2.56	6.05	10.34	1.29	3.44	3.44	6.24
$N=20$	0.88	1.59	31.85	56.07	1.52	3.68	26.48	41.72	7.31	12.06	15.18	23.04
$N=30$	0.19	0.36	69.37	103.13	8.10	14.86	48.47	82.46	15.45	23.92	33.45	58.17
$N=40$	0.10	0.24	147.09	209.20	17.11	21.04	101.90	148.81	26.77	39.28	74.72	102.48
$RS=0.3$	0.81	2.31	94.16	209.20	5.16	10.95	65.09	148.81	9.29	22.36	46.74	102.48
$RS=0.5$	0.54	2.11	57.47	101.73	6.65	13.24	40.45	90.03	11.21	21.42	29.21	65.64
$RS=0.7$	0.36	1.79	41.48	83.52	9.03	21.04	31.63	71.79	17.60	39.28	19.11	46.76
$D=1.1 \cdot C_{max}$	0.63	2.31	85.03	209.20	5.58	15.32	64.54	148.81	9.78	23.12	45.14	102.48
$D=1.3 \cdot C_{max}$	0.55	1.91	59.42	161.06	6.74	18.47	42.31	104.03	12.44	30.67	29.71	80.77
$D=1.5 \cdot C_{max}$	0.51	1.19	48.61	143.25	8.53	21.04	30.34	81.97	15.93	39.28	20.21	71.04
全部算例	0.56	2.31	64.36	209.20	6.95	21.04	45.73	148.81	12.71	39.28	31.70	102.48

首先对于全部算例来说，禁忌搜索的 ARD 和 MRD 分别为 0.56% 和 2.31%，远小于多重迭代的 6.95% 和 21.04% 及随机生成的 12.71% 和 39.28%，说明禁忌搜索获得的满意解的平均质量明显高于其他两种启发式算法。其次，当算例活动数从 10 增加到 40 时， ARD 和 MRD 分别从 1.07% 和 2.31% 下降到 0.10% 和 0.24%，而多重迭代的这两个指标分别从 1.06% 和 2.56% 上升到 17.11% 和 21.04%，随机生成分别从 1.29% 和 3.44% 上升到 26.77% 和 39.28%，表明禁忌搜索的满意解质量随着问题规模的增大而上升，而其他两种启发式算法的满意解质量却随着问题规模的增大而下降。上述结果的原因分析如下：与多重迭代相比，禁忌搜索通过对禁忌列表的检查和更新，可以有效地避免算法重复探测某一可行解，从而提高算法的搜索效率。相比于多重迭代和禁忌搜索，随机生成则更为简单，它甚至不对可行解的邻域进行搜索，所以其得到的满意解质量最差。当算例活动数增大时，问题的可行解空间规模变大，由于禁忌搜索拥有更高的搜索效率，因此，在相同的终止条件下，它相对于其他两种启发式算法在满意解质量方面的优势便体现得更加明显。值得注意的是，由于需要花费额外的时间管理禁忌列表，禁忌搜索的 ACT 和 MCT 要大于其他两种算法的相应指标，且这两个指标均随算例活动数 N 的增加而增大。

当资源强度 RS 或项目工期 D 增大时，禁忌搜索的 ARD 和 MRD 单调下降，而多重迭代和随机生成的 ARD 和 MRD 单调上升，表明禁忌搜索相对于其他两种算法在满意解质量方面的优势扩大。这一现象可解释如下：随着 RS 的增大，两种可更新资源的可用量同步增加，使得问题的可行解数量增加、可行解空间规模变大，由于禁忌搜索具有搜索效率上的优势，所以其得到的满意解的相对质量较高。当 D 变大时，项目活动开始时间安排的方式数上升，同样会造成问题的可行解空间规模变大，因而产生与 RS 增大时相似的结果。此外，需要说明的是， RS （或 D ）的增大会导致资源（或项目工期）约束的放松，这使得算法在搜索过程中

可行解及其邻点的生成操作变得相对容易, 进而导致算法的计算时间缩短。因此, 三种算法的 ACT 和 MCT 指标均随 RS (或 D) 的增大而减小。

6.4 一个算例

用图 3 所示算例对本文研究进行说明。图中, 活动 0 和 19 分别为虚的开始和结束活动, 其余 18 个非虚活动的相关参数见表 5。存在两种资源约束: $R_1 = 20$, $R_2 = 23$, 项目工期 D 为 45。利用本文所设计的禁忌搜索启发式算法, 可求得该算例的满意进度计划如下:

$$S = (0, 0, 0, 10, 10, 6, 14, 14, 11, 0, 29, 16, 10, 21, 30, 15, 38, 45); Robu = 6.97.$$

在项目实施过程中, 两种资源使用量的变化情况如图 4 所示。资源 1 的最大使用量 20 出现在第 11、12 时刻, 而资源 2 的最大使用量 23 出现在第 15 时刻, 均未超过其可用量。

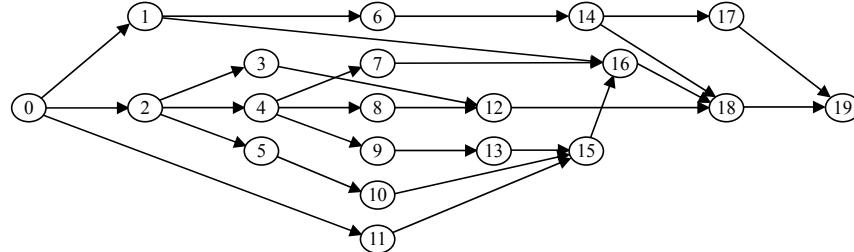


图 3 算例的 AoN 网络图

表 5 算例活动的相关参数

n	$\mu(d_n)$	$\sigma(d_n)$	r_{n1}	r_{n2}	n	$\mu(d_n)$	$\sigma(d_n)$	r_{n1}	r_{n2}	n	$\mu(d_n)$	$\sigma(d_n)$	r_{n1}	r_{n2}
1	1	2	4	0	7	2	2	5	6	13	5	1	3	4
2	9	3	7	1	8	9	2	0	5	14	4	1	2	3
3	6	3	3	3	9	1	2	4	8	15	8	3	9	1
4	3	3	6	1	10	3	4	9	0	16	8	1	6	7
5	1	1	2	2	11	5	4	2	1	17	6	3	6	9
6	4	0	1	5	12	9	1	7	6	18	7	2	4	5

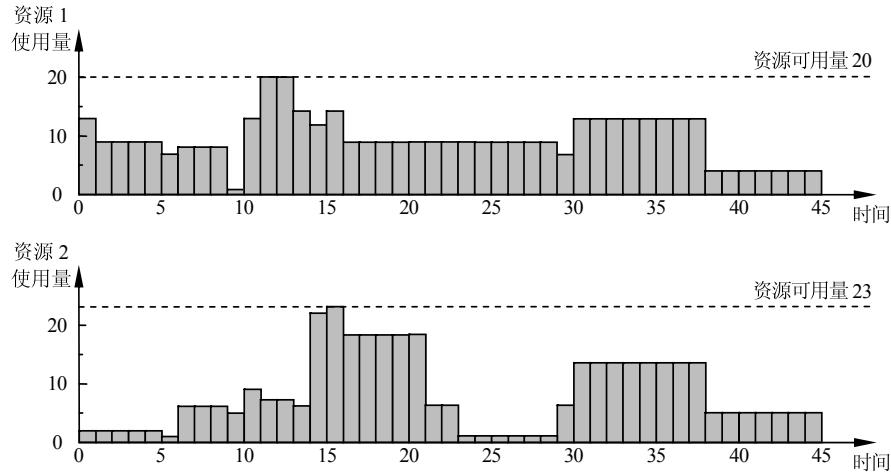


图 4 资源使用量在项目实施过程中的变化情况

在满意进度计划安排下, 各非虚活动的时间缓冲如下: $B_1=5, B_2=1, B_3=13, B_4=1, B_5=0, B_6=0, B_7=14, B_8=6, B_9=1, B_{10}=7, B_{11}=16, B_{12}=0, B_{13}=0, B_{14}=1, B_{15}=1, B_{16}=0, B_{17}=24, B_{18}=0$ 。以图 3 所示网络中的一条路径 “ $0 \rightarrow 1 \rightarrow 6 \rightarrow 14 \rightarrow 17 \rightarrow 19$ ” 为例, 说明时间缓冲是如何在活动上进行分配的。上述路径上各活动的 $\mu(d_n)$ 之和为 15, 给定项目工期 45, 该路径所拥有的时间缓冲为 $45-15=30$ 。在组成该路径的 4 个非虚活动中, 活动 17 的 $\sigma(d_n)$ 最高, 其次为活动 1、14, 活动 6 的 $\sigma(d_n)$ 最低。为了获得一个鲁棒性较大的进度计划安排, 按照 $\sigma(d_n)$ 的高低, 30 个单位的时间缓冲中, 有 24 个分配到活动 17 上、5 个分配到活动 1 上、1 个分配到活动 14 上, 活动 6 上没有得到任何时间缓冲。

需要指出的是, 时间缓冲的分配必须考虑路径之间的关联性。例如, 路径 “ $0 \rightarrow 1 \rightarrow 6 \rightarrow 14 \rightarrow 17 \rightarrow 19$ ” 上活动时间缓冲的分配, 便受到路径 “ $0 \rightarrow 1 \rightarrow 6 \rightarrow 14 \rightarrow 18 \rightarrow 19$ ” 及路径 “ $0 \rightarrow 1 \rightarrow 16 \rightarrow 18 \rightarrow 19$ ” 的

直接影响，并通过这两条路径与网络中的其他路径间接关联。此外，在进行时间缓冲分配时，相关活动开始时间的调整还受到资源可用量的限制，而且项目工期也会对各条路径上的时间缓冲产生重要影响。表6给出了在其它参数保持不变的条件下，资源可用量及项目工期对进度计划的总时间缓冲 ΣB_n 与鲁棒性 $Robu$ 的影响，根据表6绘制的 ΣB_n 与 $Robu$ 随资源可用量及项目工期的变化曲线见图5(注意，在表6和图5中， ΔR 为资源1和2的可用量分别在20和23基础上的同步增加幅度)。由图5可见，随着资源可用量的增加，进度计划的 ΣB_n 与 $Robu$ 均呈逐步上升趋势；而当项目工期延长时， ΣB_n 与 $Robu$ 近似线性地单调增加。这一结果的原因如下：资源可用量的增加会使活动开始时间的调整拥有更大自由度，因此总时间缓冲增加，且其在活动上的分配更加合理，进而导致计划的鲁棒性上升。项目工期的影响则更为直接，随着项目工期的延长，网络中各条路径上的时间缓冲均同步增加，所以，进度计划的总时间缓冲增加、鲁棒性上升。

表6 资源可用量及项目工期对进度计划鲁棒性的影响

ΔR	ΣB_n	$Robu$	D	ΣB_n	$Robu$
0	90	6.97	45	90	6.97
2	96	7.26	46	96	7.44
4	96	7.58	47	103	8.13
6	97	7.63	48	110	8.75
8	98	7.68	49	118	9.17

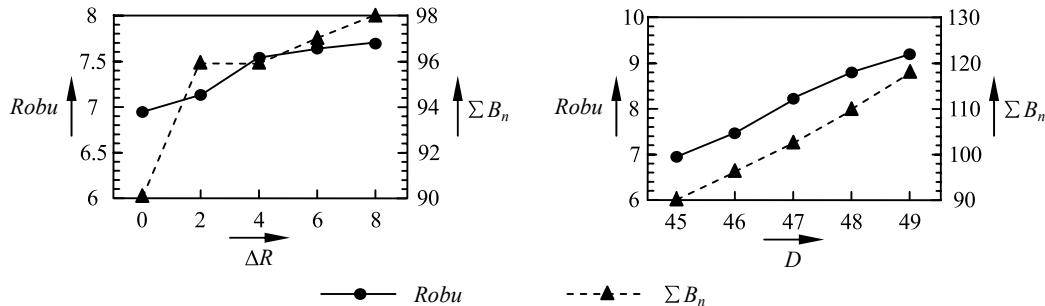


图5 计划鲁棒性随资源可用量与项目工期的变化曲线

7 结论

本文研究了随机活动工期下的资源约束项目鲁棒性调度问题。作者首先对研究问题进行界定，其中活动工期为均值和方差已知的随机变量，鲁棒性定义为所有活动的时间缓冲与其权重系数的乘积之和，目标是在可更新资源和项目工期的约束下，安排活动的开始时间以实现时间缓冲的合理分配，进而确保项目进度计划鲁棒性的最大化。随后用一个示例项目对问题进行说明，通过对该示例项目两种可行进度安排的仿真分析，验证了进度计划的鲁棒性对项目顺利实施及目标实现的重要性。在此基础上构建了问题的优化模型，针对其NP-hard属性设计了禁忌搜索、多重迭代和随机生成三种启发式算法。最后在随机生成的标准算例集合上对算法进行了比较测试，基于计算结果讨论了项目活动数、项目工期和资源强度等参数对算法绩效的影响，并通过一个算例对研究进行了说明，得到如下结论：

- 禁忌搜索获得的满意解质量明显高于其他两种启发式算法，且这种优势随着问题规模的增加而增大。
- 当资源强度或项目工期增大时，禁忌搜索相对于其他两种算法在满意解质量方面的优势随之提高。
- 项目进度计划的鲁棒性随着资源可用量的增大或项目工期的延长而上升。

本文的研究可以为不确定条件下项目进度计划的制定提供量化决策支持。

参考文献

- [1] Kolisch R, Padman R. An integrated survey of deterministic project scheduling[J]. Omega, 2001, 29(3): 249–272.
- [2] 刘士新, 王梦光, 唐加福. 资源受限工程调度问题的优化方法综述 [J]. 控制与决策, 2001, 16(S1): 647–651.
- Liu S X, Wang M G, Tang J F. The optimization algorithms for solving resource-constrained project scheduling problem: A review[J]. Control and Decision, 2001, 16(S1): 647–651.
- [3] 汪嘉昊, 孙永广, 吴宗鑫. 时间和费用具有不确定性的优化进度计划 [J]. 系统工程理论与实践, 2002, 22(1): 93–98.
- Wang J M, Sun Y G, Wu Z X. Optimal scheduling under probabilistic time and cost[J]. Systems Engineering —

- Theory & Practice, 2002, 22(1): 93–98.
- [4] 寿涌毅, 王伟. 基于鲁棒优化模型的项目调度策略遗传算法 [J]. 管理工程学报, 2009, 23(4): 148–152.
Shou Y Y, Wang W. A robust optimization model based genetic algorithm for project scheduling policies[J]. Journal of Industrial Engineering and Engineering Management, 2009, 23(4): 148–152.
- [5] Herroelen W, Leus R. The construction of stable project baseline schedules[J]. European Journal of Operational Research, 2004, 156: 550–565.
- [6] Van de Vonder S, Demeulemeester E, Herroelen W, et al. The use of buffers in project management: The trade-off between stability and makespan[J]. International Journal of Production Economics, 2005, 97: 227–240.
- [7] 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.
- [8] Al-Fawzan M A, Haouari M. A bi-objective model for robust resource-constrained project scheduling[J]. International Journal of Production Economics, 2005, 96: 175–187.
- [9] Lambrechts O, Demeulemeester E, Herroelen W. A tabu search procedure for developing robust predictive project schedules[J]. International Journal of Production Economics, 2008, 111: 493–508.
- [10] 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: 723–733.
- [11] Herroelen W, Leus R. Project scheduling under uncertainty: Survey and research potentials[J]. European Journal of Operational Research, 2005, 165: 289–306.
- [12] Elmaghraby S. Activity nets: A guided tour through some recent developments[J]. European Journal of Operational Research, 1995, 82(3): 383–408.
- [13] Blazewicz J, Lenstra J K, Rinnooy Kan A H G. Scheduling subject to resource constraints: Classification and complexity[J]. Discrete Applied Mathematics, 1983, 5: 11–24.
- [14] Mika M, Waligóra G, Weglarz J. Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times[J]. European Journal of Operational Research, 2008, 187(3): 1238–1250.
- [15] Waligóra G. Discrete-continuous project scheduling with discounted cash flows — A tabu search approach[J]. Computers and Operations Research, 2008, 35: 2141–2153.
- [16] He Z W, Wang N M, Jia T, et al. Simulated annealing and tabu search for multi-mode project payment scheduling[J]. European Journal of Operational Research, 2009, 198(3): 688–696.
- [17] Kolisch R, Sprecher A. PSPLIB — A project scheduling problem library[J]. European Journal of Operational Research, 1997, 96: 205–216.
- [18] Kolisch R, Sprecher A, Drexl A. Characterization and generation of a general class of resource-constrained project scheduling problems[J]. Management Science, 1995, 41: 1693–1703.
- [19] Kolisch R. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation[J]. European Journal of Operational Research, 1996, 90: 320–333.