广义优先关系约束下项目资源均衡的改进蝙蝠算法

李洪波[1], 熊 励[1], 刘寅斌[1], 魏文超[2*]

(1. 上海大学管理学院, 上海 200444;
2. 北京交通大学经济管理学院, 北京 100044)

摘要: 针对带有广义优先关系的项目资源均衡问题, 设计了一个改进的蝙蝠算法. 改进了蝙蝠位置和种群的更新机制, 提出了新的进度计划编码, 解码和局部改进方法. 利用正交试验设计探讨了算法参数设置. 基于公开的基准数据集, 通过大规模计算实验验证了所提算法的有效性. 对于活动数量不超过 50 个的中小规模项目, 所提算法可在 1 s 内求得接近于最优解的满意解; 对于活动数量多达 1 000 个的大规模项目, 求得满意解的时间不超过 5 min; 当项目截止日期较长时, 所提算法优于目前最好的元启发算法.

关键词: 项目调度; 资源均衡; 广义优先关系; 蝙蝠算法
中图分类号: C935; O223 **文献标识码**: A **文章编号**: 1000−5781(2019)05−0709−12

doi: 10.13383/j.cnki.jse.2019.05.012

Improved bat algorithm for project resource balance under generalized priority relation constraint

Li Hongbo[1], Xiong Li[1], Liu Yinbin[1], Wei Wenchao[2*]

(1. School of Management, Shanghai University, Shanghai 200444, China;
2. School of Economics and Management, Beijing Jiaotong University, Beijing 100044, China)

Abstract: This paper presents an improved bat algorithm (BA) for resource leveling with generalized precedence relations. The updating mechanism for bat location and population parameters is improved. New encoding, decoding and local improvement methods are applied to schedules. This paper uses an experiment to determine the appropriate BA parameters. Computational results on public benchmark test sets demonstrate the effectiveness of the BA. For small and medium instances with no more than 50 activities, the BA can obtain satisfactory solutions within 1 second; While for large instances with up to 1 000 activities, this time is less than 5 minutes. The BA outperforms the state-of-the-art heuristic algorithms when the project deadline is not tight.

Key words: project scheduling; resource leveling; generalized precedence relations; bat algorithm

收稿日期: 2016−11−30; 修订日期: 2017−06−29.
基金项目: 国家自然科学基金资助项目(71602106; 71572104; 71801013); 教育部人文社会科学研究青年基金资助项目(15YJCZH077); 上海市教委科研创新重点资助项目(14ZS085); 上海高校青年教师培养资助计划资助项目(ZZSD16025); 中国博士后科学基金资助项目(2015M571542).
*通信作者.

1 引 言

在项目周期内均衡分配资源, 可以避免资源使用量出现大幅波动, 从而提升项目执行效率, 降低项目成本. 这就导致了项目调度中的一类经典问题——资源均衡问题(resource leveling problem, RLP)[1]. 学术界已经提出了大量的精确和非精确算法求解 RLP[2−8]. 鉴于 RLP 的 NP 难特性[9], 对于大规模问题, 精确算法甚至难以在合理时间内求出可行解, 这时, 非精确的启发式算法和元启发算法成为仅有的选择.

近些年来, 含有最小和最大时间延迟的 RLP 得到了越来越多的关注, 导致了带有广义优先关系[10]的 RLP(RLP with generalized precedence relations, RLP-GPR). RLP-GPR 是经典 RLP 的泛化. 求解 RLP-GPR 的精确算法主要有分支定界算法[11,12], 和混合整数规划[13,14]. Rieck 等[13]的混合整数规划算法是目前表现最好的精确算法, 可以求解最多含有 50 个活动的项目. 在启发式算法方面, Neumann 等[11,15]设计了多项式时间启发式算法和基于优先规则的启发式算法. RLP-GPR的元启发算法主要有禁忌搜索算法[11]、迭代贪婪算法[16]、模拟退火算法[17] 等, 其中 Ballestin 等[16]的迭代贪婪算法是目前表现最好的元启发算法. 目前针对 RLP 的元启发算法, 大多直接对 RLP 的解(即进度计划, 表示为活动开始时间向量)进行操作, 即没有引入编码环节, 这种做法容易产生大量违反优先关系的解[18,19]. 本文算法通过引入新的编码与解码机制克服这一不足.

作为一种新兴的求解最优化问题的群体智能算法, 蝙蝠算法模仿了生物界中蝙蝠的回声定位行为. 在蝙蝠算法中, 每只蝙蝠的位置表示最优化问题的一个解, 猎物的位置代表问题的最优解, 蝙蝠利用回声定位改变其飞行位置(回声定位中涉及的基本参数有: 蝙蝠的飞行速度、超声波的频率、超声波的发射速率以及超声波的响度等), 从而不断向猎物(最优解)靠近. 自从 Yang[20] 于 2010 年提出以来, 蝙蝠算法已开始在一些研究领域获得成功应用. 例如, 在非线性函数优化领域, 针对多个经典基准问题实例的优化结果显示, 达到指定优化精度时, 相比遗传算法和粒子群优化, 蝙蝠算法具有更快的收敛速度, 其平均所用函数评估次数是其他两种算法的 1/4 左右[20]; 在混合流水车间调度领域, 基准数据集上的实验结果显示, 相比遗传算法和粒子群优化, 蝙蝠算法分别对 66 % 和 34 % 的算例获得了更优的结果[21].

本文首次将蝙蝠算法应用于项目资源均衡优化, 针对带有广义优先关系的资源均衡问题, 设计了一个改进的蝙蝠算法, 并取得了优于现有求解该问题的元启发算法的结果. 本文主要引入了三个新的机制来增强蝙蝠算法: 1) 改进了蝙蝠位置更新的方式, 并加入了回声定位参数的自适应调节机制. 2) 提出了新的进度计划编码与解码机制, 该机制确保了算法迭代过程中产生的解始终是可行的, 从而提升了算法寻优效率. 3) 设计了一个两阶段局部改进算法, 将其融入蝙蝠算法的寻优过程, 以进一步改善蝙蝠算法的优化效果. 最后, 本文利用正交试验设计方法确定了所提算法的参数组合, 并基于公开的大量基准测试数据集, 利用大规模计算实验以及跟已有最好的元启发算法的比较, 验证了所提算法的有效性和先进性.

2 广义优先关系约束下的资源均衡

2.1 问题描述

一个项目由加权有向含环图 $G = (N, A)$表示, 其中 N 为节点的集合, 表示活动, $N = \{0, 1, \ldots, n, n+1\}$; A 为有向弧的集合, 表示活动之间的广义优先关系(GPR), $A \subseteq N \times N$, 将各活动从 0 到 $n+1$ 编号. 活动 0 和 $n+1$ 是虚活动, 分别代表项目的开始和结束. 虚活动的工期为 0, 且不消耗任何资源. 每个非虚活动 i 的工期用整数 d_i 表示, 活动 i 的整数开始时间记为 s_i, 项目的工期不能超过截止日期 $\bar{d}$, 即 $s_{n+1} \leqslant \bar{d}$.

假设一共存在 K 种可更新资源. 活动 i 在执行时, 每个时段对第 k 种资源的需求量为固定的整数 $r_{ik}(k = 1, 2, \ldots, K)$. 在第 t 个时间段内, u_{kt} 表示项目对第 k 种资源的总需求量, $u_{kt} = \sum_{i \in E_t} r_{ik}$, 其

中 E_t 是时段 t 上所有正在执行的活动集合, $E_t=\{i|s_i\leqslant t<s_i+d_i\}$. 假设活动一旦开始执行就不允许中途暂停.

相比项目调度领域广泛采用的 CPM 优先关系(即每项活动只有在其所有紧前活动均完工后才可开始), GPR 囊括了更多优先关系类型(包括活动之间的“开始—开始”, “开始—完成”, “ 完成—开始”, “ 完成—完成”关系, 并且允许这些关系之间包含最小和最大时间延迟)[9]. 上述四类关系均可以统一转换为“开始—开始”关系[22], 因此本文只考虑活动“开始—开始”之间的最小和最大时间延迟. 对于GPR诸多性质的深入讨论, 可以参考文献[9,23].

对于项目网络中的每个弧$\langle i,j\rangle\in A$ 赋予一个权重 δ_{ij}, 它表示活动 i 和 j 开始时间之间的延迟, 即“开始—开始”型时间延迟. 具体地, 如果活动 i 和 j 的开始时间之间存在最小时间延迟, 则 $\delta_{ij}\geqslant 0$, 这意味着活动 i 开始 δ_{ij} 个时间单位以后活动 j 才可以开始, 该情况可表示为 $s_j-s_i\geqslant\delta_{ij}$; 如果活动 i 和 j 开始时间之间存在最大时间延迟, 则 $\delta_{ij}<0$, 这意味着活动 i 必须在活动 j 开始后的 $-\delta_{ij}$ 个时间单位内开始, 该情况可表示为 $s_i-s_j\leqslant-\delta_{ij}$, 由此, 最小和最大时间延迟可用下述不等式统一表示, 即

$$s_j-s_i\geqslant\delta_{ij},\ \ \forall\langle i,j\rangle\in A. \tag{1}$$

2.2 优化模型

RLP-GPR的任务是在满足广义优先关系约束和项目截止日期约束的前提下, 制定一个项目基线进度计划 $S=(s_0,s_1,\ldots,s_{n+1})$, 从而最小化资源使用量的波动程度(即使得整个项目周期内资源的使用情况尽可能均衡). RLP-GPR 的优化模型为

$$\text{Min}\ \sum_{k=1}^{K}\sum_{t=1}^{\bar{d}}c_k u_{kt}^2, \tag{2}$$

s.t.

$$s_j-s_i\geqslant\delta_{ij},\ \ \forall\langle i,j\rangle\in A, \tag{3}$$

$$s_0=0, \tag{4}$$

$$s_{n+1}\leqslant\bar{d}, \tag{5}$$

$$s_i\geqslant 0,\forall i\in N, \tag{6}$$

$$\sum_{i\in E_t}r_{ik}\leqslant u_{kt}, k=1,2,\ldots,K\,;t=1,2,\ldots,\bar{d}, \tag{7}$$

其中目标函数(2)是经典的资源均衡目标函数——最小化资源使用量平方的加权和. 权重 c_k 表示项目执行时第 k 种资源使用量的波动所导致的单位惩罚成本. 式(3)∼式(6)是时序约束, 其中式(3)表示 GPR, 式(4)令项目在0时刻开始, 式(5)是项目截止日期约束, 式(6)确保活动的开始时间非负. 式(7)用于计算资源使用量 u_{kt}, RLP-GPR 的可行域非空, 当且仅当项目网络中任意环的长度是非负的[22]. 通过引入基于离散时间的辅助变量、并把目标函数线性化, 上述优化模型可以转换为混合整数线性规划模型进行求解[13]. Neumann 等[9]已经证明, RLP-GPR 是强 NP 难的.

给定时序约束, 活动 i 的最早开始时间等于从活动0 到 i 的最长路径长度, 即 $\text{es}_i=l_{0i}$; 活动 i 的最晚开始时间等于项目截止日期减去活动 i 到 $n+1$ 的最长路径长度, 即 $\text{ls}_i=\bar{d}-l_{i,n+1}$, 另外, 称 $\text{ESS}=(\text{es}_0,\text{es}_1,\ldots,\text{es}_{n+1})$ 为最早开始进度计划, $\text{LSS}=(\text{ls}_0,\text{ls}_1,\ldots,\text{ls}_{n+1})$ 为最晚开始进度计划. 如果仅有一部分活动完成了调度(即赋予了开始时间), 则这些活动属于已调度活动集合 N', 已调度活动 $i\in N'$ 的开始时间 $s_i\geqslant 0$ 组成的向量称为部分进度计划 S', 给定一个部分进度计划 $S'=(s_i)_{i\in N'}$, 未调度活动 $j\in N\backslash N'$ 的最早开始时间可按下式计算, 即

$$\text{es}_j(S')=\max\{l_{0j},\max_{i\in N'}(s_i+l_{ij})\}, \tag{8}$$

其中 $l_{ij}=-\infty$, 如果 j 与 i 之间不可达.

活动 j 的最晚开始时间为

$$\mathrm{ls}_j(S')=\min\{\bar{d}-l_{j,n+1},\min_{i\in N'}(s_i-l_{ji})\}. \tag{9}$$

3 改进的蝙蝠算法

为了实现对 RLP-GPR 问题的高效求解, 本文对常规的蝙蝠算法主要做了如下改进: 1) 改进了基于局部随机游走的蝙蝠位置更新机制, 利用最优蝙蝠位置(而不是常规蝙蝠算法中群体的平均位置)实现局部更新, 以提升算法的深度(强化)搜索能力; 2) 在种群更新中, 引入了基于自适应的超声波参数更新机制, 从而赋予算法自动调整寻优范围的能力; 3) 针对 RLP-GPR 的特点, 设计了"随机-位移键"编码及其解码方法, 确保了蝙蝠算法始终在可行解中进行高效寻优; 4) 从时差角度提出了专门的两阶段局部改进算法, 将其融入蝙蝠算法的寻优过程, 以进一步提升解的质量.

3.1 算法流程

本文所提蝙蝠算法的基本流程如下:

步骤 1 种群初始化. 令 $\boldsymbol{x}^*$ 存储当前最好的蝙蝠位置, O^* 为相应的目标函数值, $O^*\leftarrow+\infty$, 计数变量 gen 记录种群的迭代次数, gen $\leftarrow$ 0, 随机产生 POP 个种群, 对于种群中的每只蝙蝠 i, 初始化其相关参数(见 3.1.1 节); 调用 RLSGS(见 3.2 节)将蝙蝠位置 $\boldsymbol{x}_i$ 解码为进度计划, 并按式(2)计算对应的目标函数值 O_i; 如果 $O_i<O^*$, 则 $O^*\leftarrow O_i$, $\boldsymbol{x}^*\leftarrow\boldsymbol{x}_i$.

步骤 2 蝙蝠种群迭代. 如果算法停止条件未满足, 则 gen $\leftarrow$ gen + 1, 执行步骤 3∼步骤 5; 否则, 转到步骤 6.

步骤 3 蝙蝠位置更新(见 3.1.2 节); 解码并计算新的蝙蝠位置 $\boldsymbol{x}_i'$ 对应的目标值 O_i' .

步骤 4 种群更新(见 3.1.3 节).

步骤 5 更新当前最好的蝙蝠位置 $\boldsymbol{x}^*$ 及其目标值 O^*; 转到步骤 2.

步骤 6 调用两阶段局部改进算法对当前最好的进度计划进行改进(见 3.3 节); 输出改进后的进度计划.

在进入步骤 3∼步骤 5 的迭代过程前, 需确定迭代停止条件. 本文采用项目调度中普遍接受的"生成进度计划的最大数量"作为停止条件[24].

3. 1. 1 种群初始化

初始蝙蝠种群是随机产生的. 给定蝙蝠种群, 其规模等于种群中蝙蝠的数量 POP. 对于种群中的第 i 只蝙蝠 B_i ($i=1,\ 2,\ \ldots,$ POP), 用一个五元组表示: $B_i=(\boldsymbol{x}_i,\boldsymbol{v}_i,f_i,\gamma_i,\beta_i)$, 其中 1) $n+2$ 维向量 $\boldsymbol{x}_i$ 为第 i 只蝙蝠的位置, $\boldsymbol{x}_i=(x_{i,0},x_{i,1},x_{i,2},\ldots,x_{i,n},x_{i,n+1})$, $x_{i,j}\in[0,1]$, $j=0,1,2,\ldots,n+1$ (n 为项目中非虚活动的数量). 向量 $\boldsymbol{x}_i$ 对应问题的候选解, 即编码后的进度计划(编码方法见3.2节), 其初始取值随机产生. 2) $n+2$ 维向量 $\boldsymbol{v}_i$ 表示第 i 只蝙蝠的飞行速度, 其初始取值为 $\mathbf{0}$. 3) 标量 f_i 为第 i 只蝙蝠发出的超声波频率, $f_i\in[f_{\min},f_{\max}]$, 其初始取值为 $f_{\max}$. 4) 标量 γ_i 为第 i 只蝙蝠发射超声波的速率, $\gamma_i\in[\gamma_{\min},\gamma_{\max}]$, 其初始取值为 $\gamma_{\min}$. 5) 标量 β_i 为第 i 只蝙蝠发出超声波的响度, $\beta_i\in[\beta_{\min},\beta_{\max}]$, 其初始取值为 $\beta_{\max}$. 在上述参数中, 本文取 $f_{\min}=0$, $\gamma_{\max}=1$, $\beta_{\min}=0$; 剩余参数 $f_{\max},\gamma_{\min},\beta_{\max}$ 的取值需要用户指定(本文通过正交试验设计的方法确定合适取值,见 4.2节).

蝙蝠 i 发出的超声波属性由 (f_i,γ_i,β_i) 描述. 蝙蝠根据猎物的大小调整超声波自身的频率 f_i, 随着对猎物的接近, 超声波的发射速率 γ_i 由小变大, 响度 β_i 则逐渐降低. 算法通过调整 f_i 来适应不同解的粒度, γ_i 用于平衡搜索的分散策略(diversification)与强化策略(intensification), β_i 则控制着种群的更新进程.

3.1.2 蝙蝠位置更新

在算法迭代过程中, 通过对蝙蝠位置 $\boldsymbol{x}_i$ 的更新, 实现蝙蝠群体逐渐向猎物(即最优解)靠近. 蝙蝠位置

有两种更新方式: 随机飞行和局部随机游走. 给定蝙蝠 i, 将随机数 $\alpha \in [0,1]$ 与超声波发射速率 γ_i 比较, 若 $\alpha \geqslant \gamma_i$, 则通过随机飞行更新位置; 否则通过局部随机游走对位置更新.

1) 随机飞行. 利用随机飞行更新蝙蝠 i 位置的规则为[25]

$$f_i = f_{\min} + \alpha(f_{\max} - f_{\min}), \tag{10}$$

$$\boldsymbol{v}_i' = \boldsymbol{v}_i + f_i(\boldsymbol{x}_i - \boldsymbol{x}^*), \tag{11}$$

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + \boldsymbol{v}_i', \tag{12}$$

其中 $\boldsymbol{v}_i'$ 为蝙蝠 i 的新的速度, $\boldsymbol{x}_i'$ 为蝙蝠 i 的新的位置. 如果新生成的 $\boldsymbol{x}_i'$ 中有元素的取值范围在区间[0, 1]之外, 则将取值小于 0 的元素重置为 0, 大于 1 的元素重置为 1(下文的局部随机游走也采用同样处理方式).

2) 局部随机游走. 给定蝙蝠 i, 随机选取当前最好位置附近的一个值, 实现蝙蝠 i 位置的更新

$$\boldsymbol{x}_i' = \boldsymbol{x}^* + 0.001\boldsymbol{\eta}, \tag{13}$$

其中 $\boldsymbol{\eta}$ 为服从标准正态分布的随机数.

3.1.3 种群更新

1) 蝙蝠位置的替换. 给定一个蝙蝠 i, 如果其新位置 $\boldsymbol{x}_i'$ 对应的目标值小于等于原目标值, 并且随机数 $\alpha \leqslant \beta_i$, 则用新的位置替换当前位置. 这一机制表明, 并非目标值有了改进, 就一定接受新的解, 而是通过超声波响度 β_i 来控制接受新解的概率, 从而在一定程度上避免陷入局部最优.

2) 超声波相关参数更新. 如果决定接受新的蝙蝠位置, 则该蝙蝠 i 用于下一次迭代的超声波参数(速率 γ_i 和响度 β_i) 也随之更新

$$\gamma_i' = \psi\gamma_i, \tag{14}$$

$$\beta_i' = \beta_{\max}\left(1 - \exp\left(-\chi \mathrm{gen}\right)\right), \tag{15}$$

其中 γ_i' 为更新后的超声波速率, ψ 为调节参数(本文中 $\psi = 0.95$); β_i' 为更新后的超声波响度, χ 为调节参数(本文中 $\chi = 0.99$). 更新后的参数确保在随后的迭代中尽量令蝙蝠向更好的位置靠近.

此外, 本文算法中还加入了超声波参数更新的自适应机制: 当连续两次算法迭代中当前最好目标函数值 O^* 均无改进时, 将 γ_i 和 β_i 重置为初始值.

3.2 进度计划的编码与解码

本文提出了一种新的进度计划编码方式: "随机–位移键"(random-shift key)编码. 在该编码中, 用两个向量表示一个进度计划: 一个向量为随机键向量 $\mathbf{RK}$, 另一个为位移键向量 $\mathbf{SK}$, 即 $(\mathbf{RK}, \mathbf{SK}) = (\mathrm{rk}_{\mathbf{0}}, \mathrm{rk}_{\mathbf{1}}, \ldots \mathrm{rk}_{\mathbf{n+1}}; \mathrm{sk}_{\mathbf{0}}, \mathrm{sk}_{\mathbf{1}}, \ldots \mathrm{sk}_{\mathbf{n+1}})$, 其中 rk_i 和 $\mathrm{sk}_i (i = 0, 1, \ldots, n+1)$ 都是介于 0 到 1 之间的实数. rk_i 表示活动 i 的优先值. sk_i 表示活动 i 的开始时间相对于其最早可能的开始时间偏离的程度; 换句话说, 活动 i 的最早开始时间与最晚开始时间会形成一个时间窗, 而 sk_i 就表示活动 i 的开始时间减去其最早可能开始时间后的差值占该时间窗的比例. 在蝙蝠算法的术语中, 将编码后进度计划称为蝙蝠的位置, 因此 $\boldsymbol{x} = (\mathbf{RK}, \mathbf{SK})$.

本文设计了一个资源均衡进度生成机制(resource leveling schedule generation scheme, RLSGS), 用于将蝙蝠位置解码为进度计划. 其基本过程为给定一个蝙蝠位置 $(\mathbf{RK}, \mathbf{SK})$, 从中选择最大 $\mathbf{RK}$ 值对应的未调度活动 i $\left(\text{即 } i = \arg\min_i \{\mathrm{rk}_i | i \in N \backslash N'\}\right)$, 将活动 i 的开始时间设置为 $s_i = \mathrm{es}_i'(S) + \left\lfloor \mathrm{sk}_i \times [\mathrm{ls}_i'(S) - \mathrm{es}_i'(S)] \right\rfloor$, 然后更新剩余的每一个未调度活动 $j (j \in N \backslash N')$的最早开始时间 $\mathrm{es}_j(S')$ 和最晚开始时间 $\mathrm{ls}_j(S')$, 即令 $\mathrm{es}_j(S') = \max\{\mathrm{es}_j(S'), s_i + l_{ij}\}$, $\mathrm{ls}_j(S') = \min\{\mathrm{ls}_j(S'), s_i - l_{ji}\}$. 重复上述过程, 直到所有活动都分配了开始时间. 其中如果活动 i 与 j 之间不存在路径, 则令它们之间的最长路径长度 $l_{ij} = -\infty$,

利用上述编码和解码方法, 可以确保蝙蝠算法在迭代过程中, 产生的解始终是可行的, 从而避免了对不可行解的搜索, 提升了蝙蝠算法的寻优效率.

3.3 两阶段局部改进算法

为了进一步提升本文蝙蝠算法的寻优能力, 基于局部搜索的思想, 设计了一个两阶段局部改进算法(two-pass local improvement method, TLIM), 将其融入本文的蝙蝠算法. TLIM 与局部随机游走的目标类似, 只是进一步考虑了资源均衡问题的特征, 从时差的角度实现目标函数值的进一步优化.

TLIM 的基本流程为给定一个进度计划 S, 按照各活动在 S 中的时间, 从早到晚依次选择活动. 对于每个选中的活动 i, 在保持其他活动开始时间不变的前提下(这确保不违背优先关系和项目工期约束), 从 i 的所有可行开始时间中选择一个新的时间 s_i', 使得资源均衡程度指标 PM(PM 用于衡量一个进度计划的均衡程度, 其值等于目标函数(2)的值) 的取值最小, 从而确保调整后的进度计划对应的目标值有所改进或至少不会恶化. 所有活动的开始时间都调整完毕后, 即得到一个新的进度计划 S', 此时第一阶段完成. 第二阶段的流程与第一阶段类似, 区别在于按 S' 中给出的时间降序选择活动.

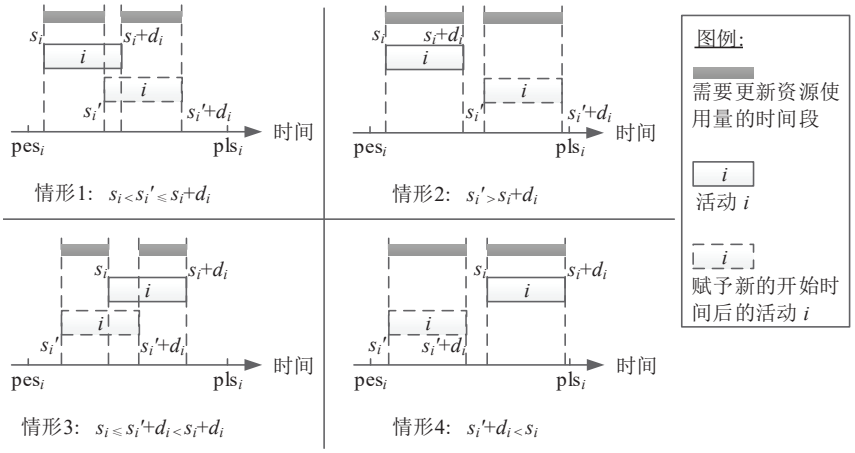

图 1 一个活动的开始时间变更后可能产生的四种情形

Fig. 1 Four possible cases after an activity's start time is changed

在 TLIM 中, 当一个活动的开始时间被改变后, 无需从头重新计算新的进度计划对应的目标函数值. 这是因为, 目标函数(2)实质上是对每个时间段上资源使用量的函数分段求和. 当一个活动被调度到一个新的开始时间后, 只可能出现图 1 中四种情形中的一种. 在图 1 中, 灰色矩形对应资源使用量发生变化的时间段 PERI, 该时间段之外的资源使用量没有发生变化, 因此计算新的目标值时只需考虑灰色矩形时间段的资源变动, 其他时间段对应的局部目标值没有发生变化. 基于此, 在 TLIM 中, 均衡程度指标 PM 采用局部更新: $\text{PM}^{\text{new}} = \text{PM}^{\text{old}} - O^{\text{old}} + O^{\text{new}}$, 其中 $O^{\text{old}}(O^{\text{new}})$表示在活动 i 调度到新的时间之前(之后), 时间段 PERI 对应的局部目标函数值. 通过上述做法, 避免了每个活动开始时间改变后需要重新从头计算目标函数值, 从而节省了计算时间.

4 计算实验

本文的蝙蝠算法在 MATLAB R2014a 中编程实现, 计算实验在 Intel Core i5 3.20 GHz CPU, Windows 7 64-bit 的计算机上实施. 本节首先介绍计算实验中使用的基准数据集; 然后利用试验设计的方法(design of experiment, DOE)探讨不同的算法参数对优化结果的影响, 进而确定合适的蝙蝠算法参数组合; 最后给出具体实验结果以及与文献中已有的最好的元启发算法的对比实验结果.

4.1 基准数据集

本文的实验在两个公开的基准数据集 T_{A} 和 T_{UBO} 上进行. 这两个数据集都是用项目调度问题实例

生成软件 ProGen/max[26]生成的. 本文所用数据集的主要参数设置如表 1 所示, 其中 RT(restrictiveness of Thesen)反映了优先关系约束的强弱程度, RT $\in [0,1]$. RT $=0$意味着项目网络完全并行, RT $=0$ 意味着项目网络完全串行[26]. 资源因子(resource factor, RF)反映了一个活动所需要资源种类的多少, RF $\in [0,1]$. RF 越大说明活动所需资源类别越多. 资源强度(resource strength, RS) 反映了资源的稀缺程度, RS $\in [0,1]$. RS 越大表明资源供应越充足.

表 1 基准数据集 T_{A} 和 T_{UBO} 的主要参数设置

Table 1 Parameter settings for test sets T_{A} and T_{UBO}

参数	数据集 T_{A}	数据集 T_{UBO}
n	10, 15, 20, 30, 50	10, 20, 50, 100, 200, 500, 1 000
RT	0.3, 0.6	0.25, 0.5, 0.75
RF	1.0 (当 $K=1$) 0.7 (当 $K=3$) 0.6 (当 $K=5$)	0.75
RS	0.5	0.0, 0.25, 0.5
K	1, 3, 5	5
r_{ik}	从区间[1,5]均匀抽取	从区间[1,10]均匀抽取
d_i	从区间[1,10]均匀抽取	从区间[1,10]均匀抽取
c_k	从区间[1,5]均匀抽取	1

数据集 T_{A} 共包含 15 个子数据集, 每个子数据集中包含 40 个问题实例. 每个子数据集的命名方式为 rlp $-$ n $-$ K, 其中 n 为非虚活动数量, K 是资源种类的数量. T_{A} 包含的实例总数为 600, 该数据集已在文献[13, 14]的研究中使用. 该数据集中的实例, 大部分已经求出了最优解[13]. 在 T_{UBO} 中, 在参数 n, RT 和 RS 的每种取值组合下包含 10 个问题实例. T_{UBO} 包含的实例总数是 630, 该数据集已在文献[9, 12, 16, 27]的研究中使用.

4.2 蝙蝠算法参数设置探讨

利用正交试验设计方法确定蝙蝠算法的合适参数取值[28]. 用于试验设计的数据集选取自 T_{A} 中的 rlp $-$ 50 $-$ 1, rlp $-$ 50 $-$ 3 和 rlp $-$ 50 $-$ 5, 这三个数据子集是目前已知最优解的规模最大(即活动数量最多)的数据集.

本文的蝙蝠算法有四个关键参数: 种群规模(POP), 超声波最大频率($f_{\max}$), 超声波最低发射速率($\gamma_{\min}$)和超声波最大响度($\beta_{\max}$). 各参数均取 5 个水平, 如表 2 所示. 在本节随后的部分, 按照试验设计中的术语, 有时也将参数称为因素.

表 2 参数水平

Table 2 Parameter levels

水平	参数(因素)			
	POP	$f_{\max}$	$\gamma_{\min}$	$\beta_{\max}$
1	50	0.000 01	0.1	0.1
2	100	0.000 1	0.3	0.3
3	150	0.001	0.5	0.5
4	200	0.01	0.7	0.7
5	250	0.1	0.9	0.9

接下来定义平均响应变量 ARV $=\sum_{i=1}^{\Sigma}\dfrac{o_i-o_i^*}{o_i^*\Sigma}$, 其中 o_i 为 rlp $-$ 50 $-$ 1, rlp $-$ 50 $-$ 3 和 rlp $-$ 50 $-$ 5 的并集中, 第 i 个实例的目标函数值(该值由蝙蝠算法求得). o_i^* 是该实例的最优值. Σ 为三个数据集中实例数量之和. 然后进行规模为 $L_{25}(5^4)$ 正交试验. 蝙蝠算法的停止条件为限定产生的进度计划的最大数量为 5 000. 项目截止日期 $\bar{d}=l_{0,n+1}$, 正交表和所得的 ARV 取值见表 3, 各参数的极差和重要程度如表 4 所

示. 表 4 的第二至第六行给出的是给定一个因素, 其在不同水平下的平均 ARV 值. “极差”行给出的是每个因素的平均 ARV 的极差. “排名”行显示了不同因素的重要性排名. 各参数对算法性能的影响趋势如图 2 所示.

表 3 正交表和ARV取值

Table 3 Orthogonal array and ARV values

实验序号	参数(因素)				ARV
	POP	$f_{\max}$	$\gamma_{\min}$	$\beta_{\max}$	
1	1	1	1	1	0.071 290
2	1	2	2	2	0.068 445
3	1	3	3	3	0.066 144
4	1	4	4	4	0.061 776
5	1	5	5	5	0.060 105
6	2	1	2	3	0.063 225
7	2	2	3	4	0.062 985
8	2	3	4	5	0.059 501
9	2	4	5	1	0.056 817
10	2	5	1	2	0.067 113
11	3	1	3	5	0.061 461
12	3	2	4	1	0.056 110
13	3	3	5	2	0.053 267
14	3	4	1	3	0.066 398
15	3	5	2	4	0.062 012
16	4	1	4	2	0.054 119
17	4	2	5	3	0.052 564
18	4	3	1	4	0.061 337
19	4	4	2	5	0.059 890
20	4	5	3	1	0.057 126
21	5	1	5	4	0.048 826
22	5	2	1	5	0.059 297
23	5	3	2	1	0.054 848
24	5	4	3	2	0.053 771
25	5	5	4	3	0.051 057

表 4 各参数的响应值和重要程度

Table 4 Response table for ARV and rank for each factor

水平	POP	$f_{\max}$	$\gamma_{\min}$	$\beta_{\max}$
1	0.065 552	0.059 784	0.065 087	0.059 238
2	0.061 928	0.059 880	0.061 684	0.059 343
3	0.059 850	0.059 019	0.060 297	0.059 877
4	0.057 007	0.059 730	0.056 513	0.059 388
5	0.053 560	0.059 483	0.054 316	0.060 051
极差	0.011 992	0.000 861	0.010 771	0.000 813
排名	1	3	2	4

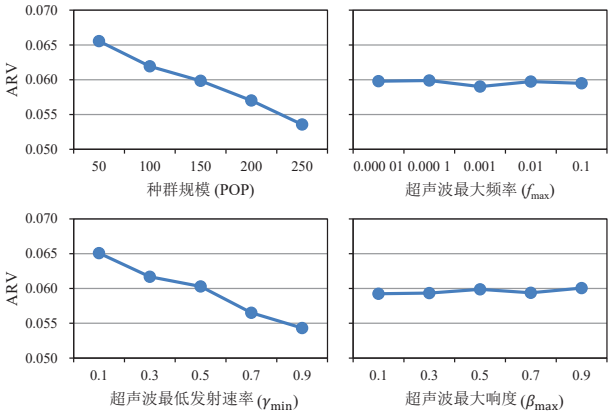

图 2 蝙蝠算法参数水平的趋势

Fig. 2 BA factor levels trend

观察表 4 和图 2 发现, 种群规模对算法性能的影响最大, 超声波发射速率次之, 超声波响度最小. 对于每个参数, 选择导致最小 ARV 值的水平. 因此, 本文蝙蝠算法的参数设置为 POP = 250(水平 5), $f_{\max} = 0.001$(水平 3), $\gamma_{\min} = 0.9$(水平 5), $\beta_{\max} = 0.1$(水平 1). 在随后的实验中将一直使用这一设置.

4.3 主要实验结果

本节的实验主要考察所提蝙蝠算法在中小规模问题实例上的表现, 因此选择数据集 $T_{\rm A}$, 对于蝙蝠算法的停止条件, 采用 1 000 和 5 000 两种最大进度计划数量. 为了衡量蝙蝠算法的求解质量, 计算蝙蝠算法得到的解与最优解之间的平均偏差(与上一节相同, 这些最优解由 Rieck 等[13]求得).

表 5 给出了算法在数据集 $T_{\rm A}$ 上的计算结果. 表格第一列是子数据集的名称. 第二列是项目截止日期, 表示为关键路径长度 $l_{0,n+1}$ 的倍数, 即 $\bar{d} = \alpha l_{0,n+1}$, 表 5 的第三、四两列显示的是在不同停止条件下, 蝙蝠算法获得的解与最优解之间的平均偏差(这一偏差的计算原理同 ARV). 对于数据集 rlp − 30(当截止日期为 $\bar{d} = 1.5\,l_{0,n+1}$ 时), rlp − 50(当截止日期为 $\bar{d} = 1.1\,l_{0,n+1}$,$\bar{d} = 1.5\,l_{0,n+1}$ 时), 目前尚未求出最优解, 因此相应的结果无法在表 5 中报告. 表 5 的最后两列给出的是求解每个实例所需的平均 CPU 时间. 观察表 5 的实验结果可以发现, 对于中小规模的问题实例, 本文蝙蝠算法的求解质量是可以接受的. 就目标函数值而言, 本文蝙蝠算法求得的值与最优值之间的偏差在 1.02 % ∼ 8.83 %之间. 另外, 蝙蝠算法在两种停止条件下求得的目标值之间比较接近, 这说明蝙蝠算法的收敛速度较快, 如果对求解精确程度要求不高, 利用 1 000 个进度计划作为停止条件已经可以获得满意的结果.

表 5 蝙蝠算法得到的解与最优解之间的平均偏差(数据集 T_{A})

Table 5 Average deviations from the optimal objective function value for test set T_{A}

数据集	截止日期($\bar{d}$)	平均偏差(%) 最大计划数量 1 000	平均偏差(%) 最大计划数量 5 000	计算时间(秒) 最大计划数量 1 000	计算时间(秒) 最大计划数量 5 000	数据集	截止日期($\bar{d}$)	平均偏差(%) 最大计划数量 1 000	平均偏差(%) 最大计划数量 5 000	计算时间(秒) 最大计划数量 1 000	计算时间(秒) 最大计划数量 5 000
rlp-10-1	$1.0l_{0,n+1}$	1.09	1.08	0.04	0.21	rlp-20-1	$1.0l_{0,n+1}$	3.10	3.04	0.06	0.30
	$1.1l_{0,n+1}$	3.71	3.59	0.04	0.21		$1.1l_{0,n+1}$	6.37	6.12	0.06	0.30
	$1.5l_{0,n+1}$	4.88	4.85	0.04	0.22		$1.5l_{0,n+1}$	8.83	7.96	0.06	0.31
rlp-10-3	$1.0l_{0,n+1}$	1.39	1.34	0.05	0.23	rlp-20-3	$1.0l_{0,n+1}$	3.14	3.01	0.07	0.34
	$1.1l_{0,n+1}$	2.80	2.80	0.05	0.23		$1.1l_{0,n+1}$	5.97	5.47	0.07	0.34
	$1.5l_{0,n+1}$	4.55	4.44	0.05	0.24		$1.5l_{0,n+1}$	6.81	6.73	0.07	0.35
rlp-10-5	$1.0l_{0,n+1}$	1.16	1.02	0.05	0.25	rlp-20-5	$1.0l_{0,n+1}$	3.00	3.00	0.08	0.38
	$1.1l_{0,n+1}$	2.42	2.42	0.05	0.26		$1.1l_{0,n+1}$	5.20	5.24	0.08	0.38
	$1.5l_{0,n+1}$	3.63	3.27	0.05	0.26		$1.5l_{0,n+1}$	8.04	7.59	0.08	0.39
rlp-15-1	$1.0l_{0,n+1}$	1.84	1.84	0.05	0.25	rlp-30-1	$1.0l_{0,n+1}$	4.10	3.87	0.09	0.43
	$1.1l_{0,n+1}$	5.35	5.00	0.05	0.25		$1.1l_{0,n+1}$	6.86	6.37	0.09	0.43
	$1.5l_{0,n+1}$	8.77	7.86	0.05	0.25	rlp-30-3	$1.0l_{0,n+1}$	4.70	4.34	0.10	0.48
rlp-15-3	$1.0l_{0,n+1}$	2.22	2.15	0.06	0.28		$1.1l_{0,n+1}$	8.21	7.12	0.10	0.49
	$1.1l_{0,n+1}$	5.62	5.42	0.06	0.28	rlp-30-5	$1.0l_{0,n+1}$	3.93	3.64	0.11	0.54
	$1.5l_{0,n+1}$	5.68	5.30	0.06	0.29		$1.1l_{0,n+1}$	6.24	6.01	0.11	0.55
rlp-15-5	$1.0l_{0,n+1}$	1.54	1.48	0.06	0.31	rlp-50-1	$1.0l_{0,n+1}$	4.89	4.63	0.16	0.81
	$1.1l_{0,n+1}$	3.52	3.49	0.06	0.31	rlp-50-3	$1.0l_{0,n+1}$	5.73	5.15	0.18	0.90
	$1.5l_{0,n+1}$	4.37	4.29	0.07	0.32	rlp-50-5	$1.0l_{0,n+1}$	5.67	5.24	0.20	0.99

就计算时间而言, 在本实验设置下, 蝙蝠算法对中小规模问题的求解时间均未超过 1 s. 而且本文算法的计算时间主要受活动数量的影响, 资源数量、项目截止日期对计算时间的影响并不明显. 需要指出的是, Rieck 等[13]利用混合整数规划求解相同数据集的最优解时, 其使用的 CPU 为 2.66 GHz, 求解软件为 CPLEX, 对于包含 50 个活动的项目, 大部分实例的平均求解时间在 500 s∼1 000 s 之间. 这与本文算法的时间形成了鲜明对比: 表 5 最后三行是针对 50 个活动的项目的求解结果, 本文算法与最优解的差距仅仅在 5 % 左右, 但本文算法的速度比最优算法快 500∼1 000 倍. 因此, 在对求解实时性要求比较高的环境下, 本文算法是有竞争力的.

此外, 为了进一步反映所提算法的改进效果, 本文还通过实验比较了所提蝙蝠算法与常规蝙蝠算法. 对比实验在数据集 T_{A} 上进行, 算法产生 1 000 个进度计划后停止, 项目截止日期为关键路径长度. 常规蝙蝠算法没有加入本文所提算法的改进措施: 未采用随机-位移键及相应的解码方法(即算法直接作用于进度计划), 未采用回声定位参数的自适应调节机制, 未融入两阶段局部改进算法. 常规蝙蝠算法的初始解利用随机抽样启发式算法产生(类似于本文算法 2, 用随机数作为其输入参数); 对算法迭代过程中产生的不可行解, 将其目标函数值置为一个非常大的整数. 表 6 显示了本文算法相比常规蝙蝠算法在目标函数值上的改进比例(在 3.13 % ∼ 9.44 % 之间), 可见改进效果明显, 显示了本文所提算法的优越性.

表 6 本文蝙蝠算法相比常规蝙蝠算法在数据集 T_{A} 上的改进比例

Table 6 Average relative improvements of the proposed BA over the original BA on set T_{A}

数据集	rlp-10-1	rlp-10-3	rlp-10-5	rlp-15-1	rlp-15-3	rlp-15-5	rlp-20-1	rlp-20-3	rlp-20-5	rlp-30-1	rlp-30-3	rlp-30-5	rlp-50-1	rlp-50-3	rlp-50-5
改进比例 / %	3.36	3.13	3.35	3.66	3.87	4.50	5.66	5.61	5.68	7.49	5.88	6.62	9.01	9.44	8.37

4.4 对比实验

本节对比实验的目的有二: 一方面, 考察所提蝙蝠算法在大规模问题实例上的表现; 另一方面, 将本文蝙蝠算法与现有文献中表现最好的求解 RLP-GPR 的元启发算法——迭代贪婪算法(IG)[16]做性能比较. 同时本文算法还将与一个启发式算法——多重开始优先规则算法(PR)[15]做比较, PR 算法主要用作算法比较

的基准. 对比结果在数据集 T_{UBO} 上运行相应算法获得, 项目截止日期 $\bar{d} = 1.50 l_{0,n+1}$,

为了确保对比的公平性, 本文做了如下设定: 1) 文献[16]并没有直接报告 IG 算法的计算结果, 而是以 PR 算法为基准, 报告了 PR 与 IG 算法之间的平均相对偏差(average relative deviations, ARD, 其计算方式见文献[16]). 鉴于此, 本文按照文献[16]的描述, 将 PR 算法在 MATLAB 中重新编程实现. 这样, 本文就可以通过计算蝙蝠算法与 PR 算法的相对偏差, 来间接实现与 IG 算法的比较. 2) 文献[16]中对 IG 使用的停止条件为 1 000 repeat-loop iterations(见文献[16]的 Algorithm 5). 通过分析可知, 在最好情形下, IG 算法的 1 000 次迭代对应 4 050 个进度计划; 在最坏情形下对应 $5\,050 + 2\,000\,n$ 个进度计划(其中 n 为给定问题实例中活动的数量). 为了确保本文蝙蝠算法与 IG 算法比较结果的公平性与说服力, 本文设定蝙蝠算法的停止条件为最多生成 4 050 个进度计划, 这意味着本文用蝙蝠算法最差情形下的表现与 IG 算法(可能)的最好表现做比较.

基于上述设定, 算法的对比结果如下. 当活动数量分别为 100, 200, 500, 1 000 时, PR 算法求得的目标函数值与蝙蝠算法目标值之间的平均相对偏差(即 ARD)为 17.72 % (2.43 s), 17.56 % (8.30 s), 17.66 % (49.46 s), 17.01 %(245.51 s), 括号内数字为每个实例的平均计算时间. 该 ARD 值大于 0 表明蝙蝠算法优于 PR 算法. 在蝙蝠算法和 IG 算法之间, 该 ARD 值越大表明算法的优化效果越好, PR 算法求得的目标函数值与 IG 算法目标值之间的平均相对偏差分别为 15.22 %, 16.19 %, 15.19 %, 13.49 % (该结果来自文献[16]的 Table 3), 可见本文的蝙蝠算法在项目截止日期较长时也优于 IG 算法. 对比实验的结果表明, 本文的蝙蝠算法在求解大规模 RLP-GPR 问题时是有竞争力的.

5 结束语

本文研究了求解广义优先关系约束下项目资源均衡问题的蝙蝠算法, 改进了蝙蝠位置和种群的更新方式, 提出了新的进度计划编码, 解码和局部改进方法. 利用正交试验设计确定了蝙蝠算法参数设置. 基于公开的基准数据集, 与当前最好的精确算法和元启发算法的对比实验结果表明, 本文算法是有效的和有竞争力的. 未来的研究将进一步改进本文算法, 并将其应用于项目管理中其他重要的调度和均衡问题[29−34].

参考文献:

[1] 李洪波, 熊 励, 刘寅斌. 项目资源均衡研究综述. 控制与决策, 2015, 30(5): 769–779.
Li H B, Xiong L, Liu Y B. A literature survey of project resource leveling. Control and Decision, 2015, 30(5): 769–779. (in Chinese)

[2] Bandelloni M, Tucci M, Rinaldi R. Optimal resource leveling using non-serial dyanamic programming. European Journal of Operational Research, 1994, 78(2): 162–177.

[3] Easa S M. Resource leveling in construction by optimization. Journal of Construction Engineering and Management, 1989, 115(2): 302–316.

[4] He L, Zhang L. Dynamic priority rule-based forward-backward heuristic algorithm for resource levelling problem in construction project. Journal of the Operational Research Society, 2013, 64(8): 1106–1117.

[5] 刘士新, 王梦光, 唐加福. 求解项目调度中资源水平问题的近似算法. 系统工程学报, 2002, 17(4): 296–302.
Liu S X, Wang M G, Tang J F. Approximate algorithm for solving resource leveling problems in project scheduling. Journal of Systems Engineering, 2002, 17(4): 296–302. (in Chinese)

[6] 张连营, 张金平, 王 亮. 工程项目资源均衡的遗传算法及其MATLAB 实现. 管理工程学报, 2004, 18(1): 52–55.
Zhang L Y, Zhang J P, Wang L. Genetic algorithms based on MATLAB of construction project resource leveling. Journal of Industrial Engineering/Engineering Management, 2004, 18(1): 52–55. (in Chinese)

[7] Ranjbar M. A path-relinking metaheuristic for the resource levelling problem. Journal of the Operational Research Society, 2013, 64(7): 1071–1078.

[8] 郭 研, 李 南, 李兴森. 多模式多资源均衡及基于动态种群的多目标微粒群算法. 控制与决策, 2013, 28(1): 131–136.
Guo Y, Li N, Li X S. Multi-mode multiple resources leveling and multi-objective particle swarm optimization with dynamic population. Control and Decision, 2013, 28(1): 131–136. (in Chinese)

[9] Neumann K, Schwindt C, Zimmermann J. Project Scheduling with Time Windows and Scarce Resources. Berlin: Springer-Verlag, 2003.

[10] 苏志雄, 乞建勋, 王　强. 求解广义优先关系下的项目最小费用问题. 管理科学学报, 2013, 16(11): 42–54.
Su Z X, Qi J X, Wang Q. An approach for the minimum cost problem of project under generalized precedence relations. Journal of Management Sciences in China, 2013, 16(11): 42–54. (in Chinese)

[11] Neumann K, Zimmermann J. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. European Journal of Operational Research, 2000, 127(2): 425–443.

[12] Gather T, Zimmermann J, Bartels J H. Exact methods for the resource levelling problem. Journal of Scheduling, 2011, 14(6): 557–569.

[13] Rieck J, Zimmermann J, Gather T. Mixed-integer linear programming for resource leveling problems. European Journal of Operational Research, 2012, 221(1): 27–37.

[14] Kreter S, Rieck J, Zimmermann J. The total adjustment cost problem: applications, models, and solution algorithms. Journal of Scheduling, 2014, 17(2): 145–160.

[15] Neumann K, Zimmermann J. Resource levelling for projects with schedule-dependent time windows. European Journal of Operational Research, 1999, 117(3): 591–605.

[16] Ballestín F, Schwindt C, Zimmermann J. Resource leveling in make-to-order production: Modeling and heuristic solution method. International Journal of Operations Research, 2007, 4(1): 50–62.

[17] 庞南生, 纪昌明. 广义时序下活动多模式与离散型资源均衡优化. 系统工程学报, 2011, 26(4): 538–545.
Pang N S, Ji C M. Leveling optimization for multi-mode and discrete resource under generalized precedence relations. Journal of Systems Engineering, 2011, 26(4): 538–545. (in Chinese)

[18] Kolisch R, Hartmann S. Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis // Project Scheduling: Recent Models, Algorithms and Applications. Dordrecht: Kluwer Academic Publishers, 1999: 147–178.

[19] Demeulemeester E L, Herroelen W. Project Scheduling: A Research Handbook. Dordrecht: Kluwer Academic Publishers, 2002.

[20] Yang X S. A new metaheuristic bat-inspired algorithm // Nature Inspired Cooperative Strategies for Optimization. Berlin: Springer-Verlag, 2010: 65–74.

[21] Marichelvam M K, Prabaharan T, Yang X S, et al. Solving hybrid flow shop scheduling problems using bat algorithm. International Journal of Logistics Economics and Globalisation, 2013, 5(1): 15–29.

[22] Bartusch M, Mohring R H, Radermacher F J. Scheduling project networks with resource constraints and time windows. Annals of Operations Research, 1988, 16(1): 199–240.

[23] Neumann K, Schwindt C. Activity-on-node networks with minimal and maximal time lags and their application to make-to-order production. OR-Spektrum, 1997, 19(3): 205–217.

[24] Kolisch R, Hartmann S. Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research, 2006, 174(1): 23–37.

[25] Yang X S. Bat algorithm and cuckoo search: A tutorial// Artificial Intelligence, Evolutionary Computing and Metaheuristics. Berlin: Springer-Verlag, 2013: 421–434.

[26] Schwindt C. Generation of Resource Constrained Project Scheduling Problems Subject to Temporal Constraints. Karlsruhe: WIOR-Report 543 of University Karlsruhe, 1998: 1–24.

[27] Franck B, Neumann K, Schwindt C. Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. OR-Spektrum, 2001, 23(3): 297–324.

[28] Montgomery D C. Design and Analysis of Experiments. Hoboken: John Wiley & Sons, 2008.

[29] 张　松, 陈华平, 刘　建. 复杂时间约束的水利工程项目调度问题研究. 系统工程学报, 2016, 31(1): 135–144.
Zhang S, Chen H P, Liu J. Research on the complicated time-constrained project scheduling in water conservancy. Journal of Systems Engineering, 2016, 31(1): 135–144. (in Chinese)

[30] Li H, Demeulemeester E. A genetic algorithm for the robust resource leveling problem. Journal of Scheduling, 2016: 19(1): 43–60.

[31] 王伟鑫, 葛显龙, 王　旭, 等. 基于关键链的非抢占式多项目调度多属性优化. 系统工程学报, 2016, 31(5): 689–699.
Wang W X, Ge X L, Wang X, et al. Multi-attribute optimization for non-preemptive multi-project scheduling based on critical chain. Journal of Systems Engineering, 2016, 31(5): 689–699. (in Chinese)

[32] Li H, Xu Z, Xiong L, et al. Robust proactive project scheduling model for the stochastic discrete time/cost trade-off problem. Discrete Dynamics in Nature and Society, 2015, Article ID 586087.

[33] 李洪波, 徐 哲, 于 静. 基于 DSM 的研发项目流程多目标仿真优化. 系统工程理论实践, 2015, 35(1): 142–149.
Li H B, Xu Z, Yu J. Multi-objective simulation optimization for the process of R&D projects based on DSM. Systems Engineering: Theory & Practice, 2015, 35(1): 142–149. (in Chinese)

[34] 张静文, 刘耕涛. 基于鲁棒性目标的关键链项目调度优化. 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. Journal of Systems Engineering, 2015, 30(1): 135–144. (in Chinese)

作者简介:

李洪波(1985—), 男, 山东东营人, 博士, 副研究员, 研究方向: 计算智能, 项目调度等, Email: ishongboli@gmail.com;
熊 励(1966—), 女, 湖北武汉人, 博士, 教授, 博士生导师, 研究方向: 信息管理, 电子商务等, Email: xiongli8@shu.edu.cn;
刘寅斌(1974—), 男, 重庆人, 博士, 副教授, 研究方向: 信息管理, 电子商务等, Email: yinbinliu@126.com;
魏文超(1985—), 男, 河北沧州人, 博士, 讲师, 研究方向: 调度, 供应链管理等, Email: weiwenchao@bjtu.edu.cn.

****

(上接第 708 页)

[15] 韦 铁, 鲁若愚. 基于 Hotelling 改进模型的服务创新差异化竞争战略研究. 管理工程学报, 2013, 27(3): 69–73.
Wei T, Lu R Y. Differentiation strategy of service innovation based on Hotelling advanced model. Journal of Industrial Engineering and Engineering Management, 2013, 27 (3): 69–73. (in Chinese)

[16] 张新鑫, 侯文华, 申成霖. 消费者行为、市场进入威胁与战略性设计架构. 系统工程学报, 2016, 31(4): 441–450.
Zhang X X, Hou W H, Shen C L. Comsumer behavior, market entry threaten and strategic product-design architecture. Journal of Systems Engineering, 2016, 31(4): 441–450. (in Chinese)

[17] Annabelle G. Platforms, Markets and Innovation. UK: Edward Elgar Publishing, 2010.

[18] Campbell J D. Targeting informative messages to a network of consumers. Review of Network Economics, 2012, 11(3): 1–31.

[19] Aral S, Walker D. Creating social contagion through viral product design: A randomized trial of peer influence in networks. Management Science, 2011, 57(8): 1623–1639.

[20] Weyl E G. A price theory of multi-sided platforms. American Economy Review, 2010, 100(4): 1642–1672.

[21] Zhu F, Iansiti M. Entry into platform-based markets. Strategic Management Journal, 2012, 33(1): 88–106.

作者简介:

段文奇(1976—), 男, 湖南邵阳人, 博士, 教授, 研究方向: 复杂网络与平台管理; Email: wenqiduan@vip.126.com;
余俊平(1989—), 男, 江西抚州人, 硕士生, 研究方向: 平台企业管理; Email: 704497430@qq.com.