基于混沌粒子群算法的项目调度干扰问题研究*

潘逢山，叶春明，姚远远
（上海理工大学 管理学院，上海 200093）

摘　要：针对资源受限项目调度问题中的干扰情况进行了界定，面向几种干扰问题建立了相应的资源受限项目调度干扰模型和混沌粒子群求解算法，对项目网络图干扰、任务干扰和资源干扰三种干扰问题进行仿真计算，验证了算法和模型的有效性，为决策者在干扰事件发生后及时对原最优调度计划作出调整给出了方向。

关键词：项目调度；干扰管理；混沌粒子群算法；仿真
中图分类号：TP18　　**文献标志码**：A　　**文章编号**：1001-3695(2013)09-2648-08
doi:10.3969/j.issn.1001-3695.2013.09.023

Research of project scheduling problems with disruption based on chaotic particle swarm algorithm

PAN Feng-shan, YE Chun-ming, YAO Yuan-yuan
(*Business School*, *University of Shanghai for Science & Technology*, *Shanghai* 200093, *China*)

Abstract: This paper defined the interference in the resource-constrained project scheduling problem (RCPSP) and established the corresponding resource-constrained project scheduling models and chaotic particle swarm algorithms for solving interference of several questions. Simulation of the project network disruption, task disruption and resources disruption from the RCPSP has been done to verify the validity of the algorithms and models. And it shows the way for the decision makers how to adjust the original optimal scheduling scheme timely after the disruption events happened.

Key words: project scheduling; disruption management; chaotic particle swarm algorithm; simulation

0　引言

对于企业来说，项目工期、成本等都是十分重要的影响因素，它们关系到企业的核心利益。在日常生产活动中，往往有内外部因素对原来制定的计划产生干扰，可能会对原目标工期产生影响，损害企业利益。因此将干扰因素考虑进 RCPSP 是符合实际生产活动的，对 RCPSP 的发展研究也有着更加深远的意义。

关于干扰的研究早在 20 世纪 70～80 年代就已经开始，但是直到 90 年代，干扰管理这一概念才明确提出。Yu 等人[1]将干扰管理定义为在计划开始阶段，用优化模型和求解算法得出一个好的运行计划；计划实施中，由于内外部不确定因素导致干扰事件的发生，使原计划变得不可行，需要实时地产生新计划，新计划要考虑到原来的优化目标，同时又要使干扰带来的负作用最小化。干扰管理需要针对各种实际问题和干扰事件的性质，建立相应的干扰管理模型和有效的求解算法，快速、及时地给出处理扰动的最优调整计划。这个调整计划不是针对扰动发生后的状态完全彻底地重新进行建模和优化，而是以此状态为基础，通过对原方案进行局部优化调整，快速生成对系统扰动最小的调整方案。

Yu 等人[1]首先将干扰管理应用在航空领域，随着干扰管理方法在航空领域的成功应用，很多学者对其产生了浓厚的兴趣。干扰管理目前已经广泛应用于许多领域[2]，主要包括航空运输、物流配送、供应链管理、机器调度及项目管理等。资源受限项目调度问题(resource-constrained project scheduling problem，RCPSP)是一个经典的 NP-hard 问题，其广泛应用于工程调度、车间调度等问题的模型建立中。目前国内外学者针对干扰管理在项目调度应用方面主要进行了以下研究：陈庭贵等人[3]建立了多因素干扰的项目调度干扰管理模型，并结合设计结构矩阵提出一种求解该问题的改进局部搜索方法，最后通过数值实例对所提的模型与方法进行了验证；刘志霞[4]研究了任务类干扰事件对多模式资源受限项目调度问题(MRCPSP)的影响，分别从客户满意程度、项目总工期以及扰动成本三方面对 MRCPSP 扰动问题进行扰动辨识和度量，构建干扰管理模型并设计一种改进的遗传算法求解，且针对 PSPLIB 中的基准问题进行了验证；陈卫明[5]对动态环境下产品开发项目的优化调度问题进行建模和仿真，并对其智能求解方法进行了研究；Lambrechts 等人[6]分析了不可预测的资源短缺对任务持续时间的影响，提出一种在项目调度中插入空闲时间的方法来保护原调度计划免受资源短缺的干扰。由于一个调度方案可通过不同的资源流动网络图执行，其在项目执行过程中对干扰的抵抗力各不相同，Klimek 等人[7]对资源受限项目调度中的资源分配问题进行了描述，提出了一个评估资源流动网络图的标准。为了提出有效方法产生强鲁棒性的调度方案使其免受不可控因素引起的干扰影响，Hazir 等人[8]研究了一个多执行模式的离散时间/成本权衡项目调度问题，介绍了一个替代

收稿日期：2013-01-18；**修回日期**：2013-03-13　**基金项目**：国家自然科学基金资助项目(71271138)；国家教育部人文社会科学规划基金资助项目(10YJA630187)；上海市教育委员会科研创新项目(12ZS133)；上海市一流学科建设项目(S1205YLXK)

作者简介：潘逢山(1962-)，男，山东邹平人，博士研究生，主要研究方向为工业工程(yechm6464@163.com)；叶春明(1964-)，男，安徽宣城人，副院长，教授，博士，主要研究方向为工业工程、生产调度等；姚远远(1989-)，女，河南灵宝人，硕士研究生，主要研究方向为工业工程、生产调度.

措施准确估计调度方案的鲁棒性,并提出一个两阶段的鲁棒性调度算法。Zhu 等人[9]提出了有限资源的项目调度的干扰管理整数规划模型及其求解方法,研究了使干扰事件对项目调度恢复成本影响最小的解决方案。针对项目管理中经常出现的干扰等不可控因素(如返工)导致项目不能如期完工,Al-Fawzan 等人[10]提出了调度鲁棒性概念和一个双目标资源受限项目调度模型及其禁忌搜索算法,有效地解决了项目管理中的干扰问题。以上关于干扰管理在资源受限项目调度方面的研究多集中在单执行模式资源受限项目调度问题,多执行模式方面的研究不多,关于干扰因素的建模也不完善。目前,智能算法采用的主要是优先规则的编码方式来解决 RCPSP。Kolisch[11]提出串行调度方案和并行调度方案来求解资源受限项目调度,并通过实证研究得出了性能良好的优先规则和计算方法。Zhang 等人[12]首次用粒子群算法来求解资源受限的项目调度问题,并研究了基于任务列表和优先数的两种粒子表示方法。Zhang 等人[13]应用粒子群算法求解资源受限项目的最小工期问题,以任务优先权作为粒子,并用并行调度方案求解,通过实证取得了比较令人满意的结果。Chen 等人[14]提出一种改进的粒子群算法求解 RCPSP 的最小工期,加大了局部搜索能力,并对关键路径进行研究,加快产生出切实可行的解决方案。Zhang 等人[15]用粒子群算法求解工期—成本—质量的多目标调度问题,并且针对质量目标的模糊性提出一种模糊多目标的粒子群算法求解该类问题。用于求解模型的优化算法也多集中在局部搜索、遗传和标准粒子群等算法。

干扰管理是近年来学者关注的焦点,它可帮助管理者以最小的系统扰动为目标重新制定最优的调整策略,算法设计也面临着不断更新的需要。本文从 RCPSP 和干扰管理出发,将两者结合建立一个混合问题模型,对单执行模式单资源的 RCPSP 和复杂的多模式多资源的 RCPSP 的三种干扰问题分别进行了实例仿真研究,并采用混沌粒子群算法求解,结果比较令人满意,验证了算法和模型的有效性,在理论方面为干扰调度提供了可行的解决方案,同时对企业实际生产活动也有着深远意义。

1　RCPSP 受干扰情况界定

项目在执行过程中不可能一帆风顺,常常会遇到各种各样的干扰,如项目的网络图调整、任务延迟、资源短缺等干扰事件。这类干扰事件往往会对原来的调度计划产生影响而造成项目工期延迟,由于任务执行顺序是一个网状结构,可能某一个任务的延迟而导致整个后续任务的连续延迟。如图 1 所示,若任务 4 由于某种原因导致延迟,可能会影响到任务 6、7 的项目进度。

在一般情况下,项目中的不确定性事件主要有两种形式。一种是经常性随机事件,这类事件发生的几率很大,也是项目执行过程中普遍存在的。例如,项目执行过程中的短暂资源短缺、质量不合格而返工等。另一种则是极小概率极大影响的随机事件,这类事件发生概率非常小,但是影响十分巨大,有可能对整个项目产生毁灭性影响。例如,洪水、地震、火山爆发等自然灾害。这两种事件都难以预测和准备,一旦发生,都会对项目产生不同程度的影响。由于第二种随机事件影响太大,恢复起来十分困难,不属于干扰管理研究范畴,所以本文只对第一种干扰事件进行研究和探讨。

1.1　项目网络图干扰

项目网络图是由任务和任务之间的紧前约束关系组成。紧前约束关系决定了各个任务执行的先后顺序,并且能够用一张箭线、节点组成的图表示出来。在一个项目的执行过程中,如果客户要求发生改变,这时可能需要调整生产工艺,如增加或者删减一个或者多个任务,也可能调整某几个任务之间的执行顺序来达到同样的目的,如图 2 和 3 所示。例如,客户对某个产品有了新的要求,项目工程师就要根据客户要求改变工艺,加上一些新的任务来满足客户的需求。但是,这样就会改变原项目的网络图,对项目产生干扰,这便是项目网络图干扰。

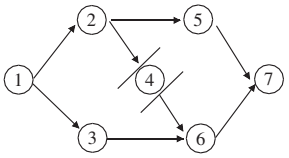

图 1　项目执行中发生干扰示意图

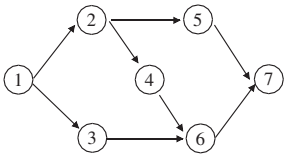

图 2　原项目网络图

增加、删减任务干扰(A_N,P_N)可表示为增加或者删除一个任务集合 A_N 及其对应的紧前关系约束 P_N。紧前关系约束的干扰(P_A,P_R)可表示为改变后的项目网络图需要满足新紧前关系约束集合 P_A,而不用再满足以前的紧前关系约束集合 P_R。

1.2　任务干扰

在实际项目执行过程中,任务常常会受到它的工期和资源使用量两方面的因素影响。资源使用量干扰主要是资源短缺和资源过剩造成的干扰,如人力资源短缺或者富余对项目原调度计划产生的影响。对于任务工期缩短和资源过剩两种情况,可以不用调整原计划就能达到预定目标。干扰管理研究的是对原计划扰动最小的方案,这两种情况对原计划未造成扰动,所以不属于本文研究范畴。任务延期干扰和资源短缺干扰是主要研究对象。

任务延期干扰可表示为任务 $i\in A$ 需要花费的时间比计划的时间要长 ψ_i 个单位。资源短缺干扰可表示为任务 $i\in A$ 在执行过程中,消耗资源 k 的量比原计划消耗的要多 γ_{ik} 个单位。

1.3　资源干扰

在项目的整个生命周期内,可能会有一段时间出现某种资源短缺。项目中可更新资源受整个周期内每个时间段的限制,只要其中某段时间出现短缺就会影响到后面所有任务的工期。比如项目中的一台机器设备在某段时间内无法正常运行了,就会造成后面需要这台机器设备的任务延期。不可更新资源受其在整个项目周期内的资源总量影响,只要总量不出现短缺,就不会对各个时间段资源需求量造成影响。这里的资源约束也是指资源短缺造成的约束,与上一节的资源干扰是不同的。上一节所讲的资源短缺干扰是某个任务增加了资源需求而造成的资源短缺干扰,而本节里的资源干扰是由于某种资源的总量短缺而引起的需要该资源的所有后续任务的干扰。资源短缺是项目管理中最常见的一种干扰,如机器设备故障、技术人员突然离职等。

资源干扰可表示为在某段时间内,资源 k 的资源量减少了 $\rho\pi_k$。

2　RCPSP 受干扰模型的建立

2.1　模型符号变量说明

D:任务总数;

i:任务序号,取值为 $1,2,\cdots,D$,其中任务 1 和任务 D 是虚工序;

K_k:可更新资源总数;
K_n:不可更新资源总数;
K:资源总数,$K = K_k + K_n$;
k:可更新资源序号,取值为 1,2,…,K_k;
n:不可更新资源序号,取值为 1,2…,K_n;
Tf_i:任务 i 完成时间;
m:执行模式;
M_i:任务的执行模式集合;
td_{im}:任务 i 在执行模式 m 下的持续时间;
EF_{im}:任务 i 在执行模式 m 下的最早完成时间;
LF_{im}:任务 i 在执行模式 m 下的最迟完成时间;
mc_{im}:任务 i 改变模式引起的费用;
rc_k:资源 k 使用量增加引起的费用;
TD:原调度计划的最优工期;
x_{imt}:0-1 变量,任务 i 选择模式 m 在时刻 t 完成则取 1,反之则为 0;
y_k:0-1 变量,如果资源 k 被选择则取 1,反之为 0;
dc:每延期一天所支付的罚金;
r_{imk}:任务选择模式 m 消耗资源 k 的资源量;
πR_k^p:可更新资源 k 的总量;
πR_n^v:不可更新资源 n 的总量;
$\rho\pi_k$:可更新资源 k 受干扰减少的数量;
$\rho\pi_n$:不可更新资源 n 的受干扰减少的数量。

2.2 目标函数

目前,研究资源受限项目调度问题的目标函数比较多,常见的有工期最小、资源均衡、成本最低和质量最大等。鉴于本文是研究干扰所带来的影响,故选取项目工期偏差函数和新调度变更成本函数。建立这两个函数所要研究的主要对象是项目中干扰发生时的未执行任务或者正在执行的任务,因为已执行的任务这些值为定值,可不作考虑。由未执行的任务和正在执行的任务组成一个新的任务集,在它的基础上建立如下所示的工期偏差函数和新调度变更成本函数。

1)项目工期偏差函数

工期偏差函数主要是用来衡量干扰后的新计划与原计划的差异,目标函数是取其最小值。

$$FT = \sum_{i=1}^{D} [\sum_{m}\sum_{t} t x_{imt} - Tf_i]^2 / D \tag{1}$$

2)新调度变更成本函数

由于干扰的影响,新调度可能会在原调度上作一些变更,这些变更都会引起成本增加,如任务模式变更的费用和任务延期的惩罚成本等。

(1)任务模式变更费用

新调度要保证系统扰动最小,可能需要更换执行模式。假设任务的执行模式发生变更时会产生一定的费用。通常,这个费用可以用增加的资源使用费用和模式变更费用来表示。

$$FC_1 = \sum_{k} rc_k \times y_k + \sum_{i}\sum_{m} mc_{im}\sum_{t} x_{imt} \tag{2}$$

(2)任务延期的惩罚成本

当工期延期后,根据合同会产生一个惩罚成本,即当 $Tf_D > TD$ 时:

$$FC_2 = (Tf_D - TD) \times dc \tag{3}$$

将两个成本综合考虑,得到一个总的成本函数,如式(4)所示。

$$FC = FC_1 + FC_2 \tag{4}$$

各个目标函数已确定,现需要将其混合建立一个综合的目标函数。但是,由于各个目标函数值具有不同的单位和量纲,所以在进行综合之前先将这些目标无量纲化,即仅用数值的大小来反映各目标属性值的优劣。式中的值是将各目标函数通过无量纲化所得到,具体公式如下:

$$\begin{cases} FT^* = (FT - FT_{\min})/(FT_{\max} - FT_{\min}) & (5) \\ FC^* = (FC - FC_{\min})/(FC_{\max} - FC_{\min}) & (6) \end{cases}$$

将工期与成本整合到一个目标函数中,需要给两个目标函数各赋一个权重,这个权重的大小根据目标函数的重要性来确定。因此,可以得到式(7)所示的目标函数。

$$\min\{F = \omega_T \times FT^* + \omega_C \times FC^*\} \tag{7}$$

$$\omega_T + \omega_C = 1 \tag{8}$$

2.3 约束条件

$$\sum_{i=1}^{D}\sum_{m=1}^{M_i}\sum_{t=EF_{im}}^{LF_{im}} x_{imt} = 1 \tag{9}$$

$$\sum_{i=1}^{D}\sum_{m=1}^{M_i}\sum_{t=EF_{im}}^{LF_{im}} t \times x_{imt} \leqslant \sum_{i=1}^{D}\sum_{m=1}^{M_i}\sum_{t=EF_{im}}^{LF_{im}} t \times x_{jmt} - td_{jm} \quad i \in P_j \tag{10}$$

$$\sum_{i=1}^{D}\sum_{m=1}^{M_i}\sum_{q=\max\{Tf_i, EF_{im}\}}^{\min\{Tf_i+td_{im}-1, LF_{im}\}} r_{imk} \times x_{jmq} \leqslant \pi R_k - \rho\pi_k \tag{11}$$

$$\sum_{i=1}^{D}\sum_{m=1}^{M_i}\sum_{t=EF_{im}}^{LF_{im}} r_{imn} \times x_{imt} \leqslant \pi R_n - \rho\pi_n \tag{12}$$

其中:式(9)表示每个任务都有唯一的一个完成时间与之对应;式(10)表示任务之间的紧前关系约束;式(11)表示项目中的可更新资源约束;式(12)表示项目中的不可更新资源约束。

3 混沌粒子群算法求解 RCPSP 步骤

3.1 混沌粒子群算法求解步骤

混沌粒子群算法主要是将混沌优化与粒子群算法并行化,以当前粒子变量为基础,通过对当前粒子个体产生混沌扰动形成混沌序列,把产生的混沌序列中的最优位置粒子随机替代当前粒子群中粒子的位置,从而提高粒子性能,使得群体的进化速度加快,帮助其跳出局部极值区间,提高了算法的收敛速度和精度。

这种通过将粒子进行混沌优化使每一代群体的质量提高一步的方法,无疑会更有助于算法后续的搜索过程。具体算法步骤如下:

a)混沌初始化形成粒子的位置与速度,计算初始粒子个体极值和粒子群全局极值。选取最优的数个混沌变量作为粒子初始种群。

b)以目前粒子位置作为混沌变量,利用典型的混沌系统[16] Logistic 映射:$x_{n+1} = \mu x_n(1 - x_n)$,$n = 0,1,\cdots$更新混沌序列,并计算、评价适应度值 F。

c)以粒子初始种群为基础更新粒子群,利用粒子群位置和速度公式分别更新粒子位置和速度,并计算、评价规则下满足约束条件的适应度值 F。

d)比较适应度值 F,选取较优的为个体最优值。

e)更新粒子局部最优和全局最优值。

f)若达到最大迭代次数,则返回全局最优解 gbest;否则跳至步骤 b)。

公式($x_{n+1} = \mu x_n(1 - x_n)$,$n = 0,1,\cdots$)中:$\mu$ 为控制参数;x 为状态变量;x_n 记录第 n 代的状态,$x_0 \in [0,1]$ 为初始值,其主要描述的是第 n 代与第 $n+1$ 代的关系;纵坐标上的区间 $0 \leqslant x \leqslant 1$,横坐标上的区间 $0 \leqslant \mu \leqslant 4$。当 $\mu < \mu_\infty = 3.569945\cdots\cdots$时,是系统的周期运动区;当 $\mu = 4$ 时,$0 \leqslant x \leqslant 1$ 有明显的混沌特征,即完全处于混沌状态;$\mu_\infty \leqslant \mu \leqslant 4$ 是系统的混沌运动区,混沌系统 Logistic 倍周期分岔过程如图 4 所示。

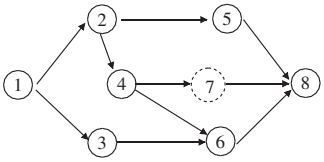

图 3　添加任务后的网络图

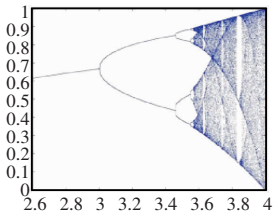

图 4　Logistic 混沌系统分岔图

3.2　混沌粒子群算法求解 RCPSP 具体步骤

混沌粒子群算法求解 RCPSP 具体步骤如下所示:

a)RCPSP 的混沌粒子群算法编码。对于一个含有 D 个任务的项目,设虚开始任务为 1,虚结束任务为 D,虚任务在项目执行过程中不消耗资源和时间,其他实际任务按照小到大、从左到右编号,实际任务编号要大于其所有紧前任务的编号。同理,实际任务编号要小于其所有紧后任务的编号,并行任务的编号可以随机化。以粒子群的搜索空间维数来代表项目中的任务数,则空间维度为 D。采用基于优先规则的粒子编码方式,其基本思路是:对于一个粒子 x_i,它每个维度的值 x_{ij} 都是[0,1]的随机实数,它的大小决定了第 j 维的优先顺序,值越大代表该任务越先执行。v_i 为粒子的更新速度,取[-1,1]的实数。

b)不符紧前关系的粒子处理。为了便于计算,现作如下设定:虚开始任务的优先值为 $x_{i1}=1$,虚结束任务的优先值为 $x_{iD}=0$,其余的任务都在(0,1)中取随机值。x_{ij} 越大,表示第 i 个粒子第 j 个任务的优先权越大,在项目执行过程中会优先执行,对于优先值相同的任务,取编号小的任务先执行。由于粒子编码采用的是随机实数,有的任务优先值可能会不满足紧前紧后关系约束,因此在程序设计中加入一个冒泡排序算法来调整粒子的值。其具体思路是:比较当前任务优先值和它的紧前任务优先值的大小,如果小于紧前任务,则无须调整;如果大于紧前任务,则交换两者的优先值。这样处理完所有任务后,就得到一个严格按照紧前约束的优先值数组 x'_i。x'_i 是和 x_i 一一对应的,在计算适应值时使用的是 x'_i。

c)确定任务的最早和最晚开始时间。

(a)确定任务最早开始时间。在不考虑资源约束的情况下,根据 PERT/CPM 图从第 1 个任务开始递推,第 1 个任务的最早开始时间为 0,其紧后任务的开始时间也为 0,然后后面的任务最早开始时间取其所有紧前任务最早开始时间加紧前任务持续时间的最大值,即

$$\mathrm{tes}_j=\max_{i\in \mathrm{IP}_j}\{\mathrm{tes}_i+td_i\} \tag{13}$$

(b)确定任务最晚开始时间。将所有任务的持续时间相加可以求得最后任务的最晚开始时间 TD,然后可以倒推出其紧前任务的最晚开始时间为 TD,其他任务的最晚开始时间取其紧后任务的最早开始时间减该任务的持续时间的最小值,即

$$\mathrm{tes}_j=\min_{i\in \mathrm{IS}_j}\{\mathrm{tes}_i-td_j\} \tag{14}$$

d)粒子初始化。粒子速度和位置初始化,通过随机数生成函数给粒子速度和位置各赋一个 0-1 之间的实数数组。然后转步骤 b)。

e)计算粒子适应度。本节中优化目标是项目工期最小,即最后的虚结束任务的实际开始时间最小,则适应度函数可以取虚结束任务的实际开始时间。若粒子在单位时间内的最大资源消耗量大于资源限制量,即不满足资源限制约束,此时粒子的适应度取 $+\infty$ 或者任意一个大于任务最迟结束时间的值,这个值会在比较中被淘汰出局;若粒子在单位时间内的最大资源消耗量小于等于资源限制量,即满足资源约束,此时粒子的适应度就是实际求得的数值。

下面以图 5 所示的项目调度问题说明如何将粒子通过串行调度[12]生成方案转换成项目工期。

随机产生一个粒子 $x_i=(1,0.6837,0.9012,0.7032,0.5011,0)$,观察发现,任务 4 的优先值不满足紧前关系,通过上述步骤 b)调整后得 $x'_i=(1,0.7032,0.9012,0.6837,0.5011,0)$。这样就产生了一个可行的调度方案 1-3-2-4-5-6。通过串行调度思想,先做任务 3,然后比较剩余资源和紧随其后的任务 2 需要消耗的资源大小。如果前者小于后者,则等任务 3 完成后再开始任务 2;反之,则任务 2 和 3 同时开始,再比较 0-TD 时间内,每段时间内资源剩余量和下一个任务消耗资源量大小,来确定下个任务的实际开始时间。依此反复,直到最后得到虚结束任务的开始时间。上面 x'_i 对应的调度如图 6 所示,可以得到一个工期为 6 的调度。

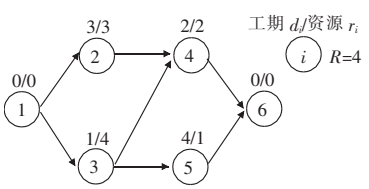

图 5　项目网络图

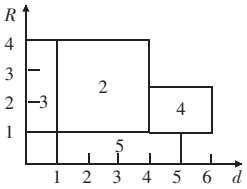

图 6　x'_i 对应的调度

f)判断粒子个体最优位置。比较粒子适应度计算结果,对比该粒子历史最优位置。若适应度更大则取新的适应度作为该粒子的个体最优位置;反之,则保留个体最优位置信息。

g)判断粒子群全局最优位置。比较每个粒子的个体最优位置,取适应度最大的粒子作为种群最优粒子,最后求得的该值便是资源受限项目调度的最短工期。

h)更新粒子。粒子采用更新方程对粒子速度和位置进行更新。更新后,如果非虚任务的 $x_{ij}\geqslant 1$ 或者 $x_{ij}\leqslant 0$,则 $x_{ij}=\mathrm{rand}$,rand 是产生 0-1 随机实数的函数。然后转步骤 b)消除紧前约束,最后得到一个 x'_i,将其代入串行调度求适应值。如果 $v_{ij}>1$,则 $v_{ij}=1$;如果 $v_{ij}<-1$,则 $v_{ij}=-1$。

4　带干扰的单模式单可更新资源项目调度问题仿真

本文将对带干扰的 RCPSP 算例进行仿真计算求解,分别从网络图干扰、任务干扰和资源干扰三种干扰情况对 RCPSP 进行简单的实例仿真。由于干扰发生具有随机性并且规模较大,难以全面分析,因此下文将选取几个确定的、具有代表性的问题进行仿真研究。首先将对项目中只有一种执行模式和一种可更新资源的干扰问题展开研究,采用文献[1]中所给出的例子,对其进行仿真计算,求解最优工期问题。

4.1　单模式单资源项目调度问题描述

文献[1]中的算例可以描述为:一个项目有 10 个任务,整个项目执行过程中只用到一种可更新资源,该资源总量为 10 个单位,即在项目执行过程中的每个时刻只有 10 个单位的资源可供使用,怎么安排各个任务的调度顺序和开始时间才能使总工期最小。原始的项目网络图如图 7 所示。图 7 中各个任务的作业时间、紧后逻辑关系约束和资源需求量如表 1 所示。求解该问题得到原始调度的最优工期为 32,原始最优调度的甘特图如图 8 所示,原计划最优任务调度顺序如表 2 所示。

4.3 ~4.5 节均以该调度算例为研究对象。

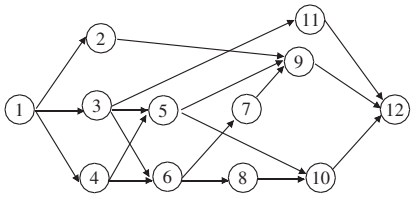

图 7 单资源项目调度的原始网络图

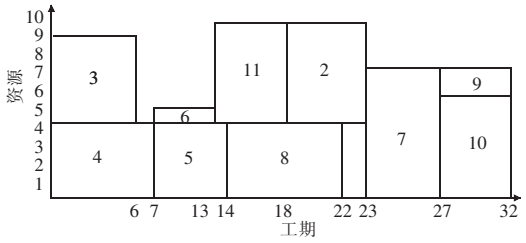

图 8 原始最优调度甘特图

表 1 单资源作业信息表

作业号	作业时间	紧后工序号	资源需求数量
1	0	2,3,4	0
2	5	9	6
3	6	5,6,11	5
4	7	5,6	4
5	7	9,10	4
6	6	7,8	1
7	4	9	7
8	8	10	4
9	3	12	1
10	5	12	6
11	5	12	6
12	0	0	0

表 2 原计划最优任务调度时间表

作业号	开始时间	结束时间
1	0	0
4	0	7
3	0	6
6	7	13
5	7	14
11	13	18
8	14	22
2	18	23
7	23	27
10	27	32
9	27	30
12	32	32

4.2 实验设计原理

a)首先依据资源作业信息表分别画出原始项目网络图、原始最优调度甘特图,以及原计划最优任务调度时间表。

b)在干扰情况发生后,根据问题的实际情况确定需要考虑的干扰问题目标函数(本文考虑工期作为目标函数),并画出干扰发生后的项目网络图。

c)将发生干扰时系统中正在执行的和没有执行的任务组成一个新的集合,然后对新集合重新调度,分别得出新任务集合网络图和新任务集合作业信息表。

d)通过混沌粒子群算法求解干扰后系统扰动最小的调度方案,采用 MATLAB7.0 编程,对混沌粒子群算法进行参数设置,根据运行若干次后得到的最优结果,绘制出干扰后的新最优调度方案甘特图。

e)将干扰发生后的最优调度与原调度的开始时间和结束时间绘制成表格形式,并分别求出两个开始时间之间的平均方差和两个结束时间之间的平均方差,得到干扰前后调度时间对照表,并对该表实验结果进行分析。

4.3 项目网络图干扰

根据 4.2 节实验设计原理,以项目网络图干扰为例进行具体阐述。项目在执行过程中,由于客户要求发生更改,需要对原调度进行调整,在任务 2 和 9 之间添加一个任务 a,其作业时间为 5,资源需求量为 4。项目网络图发生干扰后的网络图如图 9 所示。

由于该问题只涉及到一种执行模式和一种可更新资源,发生干扰时,资源增加的费用为定值,可不用考虑,产生的费用主要与工期有关,所以该干扰问题目标函数只需要考虑新工期与原工期的偏差。

根据新增任务 a 的逻辑约束关系,必须在任务 2 完成后才能开始执行。再由表 2 的调度顺序可知,任务扰动最小的最优方案由任务 2 后面的所有任务形成的新调度决定。干扰后由未执行任务和正在执行的任务所形成的新任务集合的网络图如图 10 所示,新任务集合的作业信息如表 3 所示。其中,任务 m 为新任务集合的虚拟开始任务,其作业时间和资源需求量都为 0,开始时间为 23。

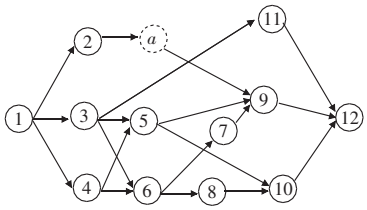

图 9 单资源项目调度增加任务后的网络图

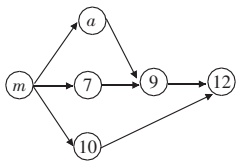

图 10 新任务集合的网络图

表 3 新任务集合作业信息表

作业号	作业时间	紧后任务号	资源需求数量	原调度开始时间
m	0	a,7,10	0	23
7	4	9	7	23
a	5	9	4	NULL
10	5	12	6	27
9	3	12	1	27
12	0	0	0	32

采用 MATLAB7.0 编程,粒子群算法参数设置为:学习因子 $c_1 = c_2 = 1.49445$,种群规模 popsize = 20,进化次数 maxgen = 50;$w_{max} = 0.9$,$w_{min} = 0.4$,根据运行 50 次后得到的最优结果,绘制出如图 11 所示的最优调度方案甘特图。

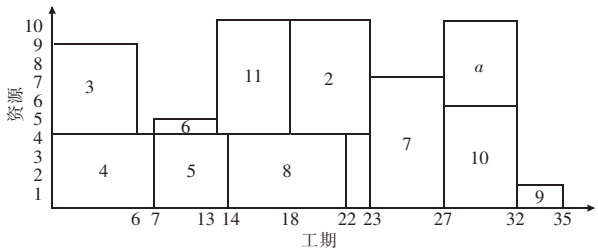

图 11 网络图干扰后新调度甘特图

将网络图中增加任务 a 后的最优调度与原调度的开始时间和结束时间绘制成表格形式,并分别求出两个开始时间之间的平均方差和两个结束时间之间的平均方差,得到如表 4 所示的对照表。

表 4 网络图干扰前后调度时间对比

任务号	原调度开始时间	原调度结束时间	新调度开始时间	新调度结束时间
1	0	0	0	0
2	18	23	18	23
3	0	6	0	6
4	0	7	0	7
5	7	14	7	14
6	7	13	7	13
7	23	27	23	27
8	14	22	14	22
9	27	30	32	35
10	27	32	27	32
11	13	18	13	18
12	32	32	35	35
a			27	32
S_1	S_2			
2.833	2.833			

表 4 中,S_1 表示新调度中任务 1 ~ 12 的开始时间与原调度中对应的开始时间之间的平均方差,S_2 表示新调度中任务 1 ~ 12 的结束时间与原调度中对应的结束时间之间的平均方差。S_1、S_2 值越大说明新调度计划与原调度计划之间工期的差异越大,即工期系统的扰动越大;如果它们的值等于 0,就说明新调度计划与原调度计划的工期无差异,工期系统扰动最小或者是无扰动。

对比表 4 中的新最优调度与原最优调度,新调度方案工期为 35,比原方案多 3 个单位,新方案总共有 2 个任务的调度顺序发生调整,开始时间方差 S_1 与结束时间方差 S_2 均为 2.833,比较接近 0,是程序运行 50 次得到的最小值,符合干扰最小的要求。通过上述例子的仿真实验发现在任务 2 和 9 之间增加一个任务 a 对系统的扰动比较小,要恢复到原最优调度计划或者接近原最优调度计划,实现起来较容易。

4.4　任务干扰

关于任务干扰将主要研究现实中存在的最为普遍的任务延期干扰,即在项目调度过程中,某个或者某些任务延期而导致整个调度延期的问题。假设由于任务 6 质量不合格需要返工,其作业时间由 6 增加到 9。由图 7 可知,当任务 6 发生延期干扰时,肯定会对其后面所有的任务产生影响。

上述任务延期干扰可等同于在任务 6 后面增加一个任务 6#,任务 6#的作业时间为 3,即任务 6 延期的时间,消耗资源为 1,其任务延期干扰后的网络图省略。这与上节网络图干扰中增加任务引起的干扰相似,不同的是任务 6#的优先级要高于原最优调度计划中任务 6 后面的所有任务。

以任务 6 正常完工时刻为原点,将发生任务延期干扰后未执行的任务和正在执行的任务形成一个新的任务集合,新集合网络图如图 12 所示。该图中,5#表示在任务 6 发生延期时刻还未完成的任务 5,其工期为 1,资源消耗量为 4。任务延期干扰的新任务集合的作业信息表省略,任务 m 为新任务集合的虚拟开始任务,其作业时间和资源需求量都为 0,开始时间为 13。

混沌粒子群算法参数设置同 4.3 节,根据运行 50 次后得到的最优结果,可以绘制出如图 13 所示的发生干扰后最优调度方案甘特图。将任务 6 延期后的新最优调度与原最优调度的开始时间和结束时间绘制成如表 5 所示的表格。

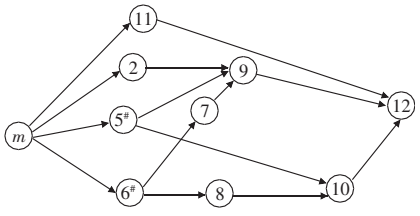

图 12　任务延期干扰新任务集合网络图

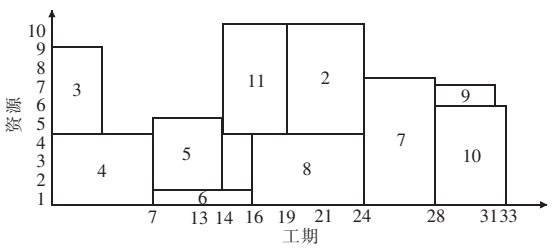

图 13　任务延期干扰后的新最优方案甘特图

表 5　任务延期干扰前后调度时间对比

任务号	原调度开始时间	原调度结束时间	新调度开始时间	新调度结束时间
1	0	0	0	0
2	18	23	19	24
3	0	6	0	6
4	0	7	0	7
5	7	14	7	14
6	7	13	7	16
7	23	27	24	28
8	14	22	16	24
9	27	30	28	31
10	27	32	28	33
11	13	18	14	19
12	32	32	33	33
S_1	S_2			
0.833	1.583			

对比表 5 中的新最优调度与原最优调度,新调度方案工期为 33,比原方案多 1 个单位时间,新方案共有 6 个任务的调度顺序发生调整,开始时间方差 S_1 与结束时间方差 S_2 分别为 0.833 和 1.583,非常接近 0,是程序运行 50 次得到的最小值,符合干扰最小的要求。上例表明任务 6 延期 3 天对系统造成的干扰对原调度计划影响很小,要恢复到原最优调度计划或者最接近原最优调度计划,实现起来较容易。

4.5　资源干扰

本节研究资源总量减少对项目调度造成的干扰。假设在 t = 10 时刻,项目中资源总量减少了 1 个单位,变成了 9 个单位。由图 8 可知,资源总量减少对后面的任务 2、8 和 11 会造成影响。

将 t = 10 时刻系统中正在执行的和还没有执行的任务组成一个新任务集合,它们之间的逻辑关系不发生改变,但是正在执行的任务 5 和 6 优先级最高,最先执行,新集合网络图见图 12。其中,5#是在资源干扰发生时刻还未完成的任务 5,其工期为 4,资源消耗量为 4。同理,6#是在资源干扰发生时刻还未完成的任务 6,其工期为 3,资源消耗量为 1。其资源干扰的新任务集合作业信息表省略。

算法参数设置同 4.3 节,根据运行 50 次后得到的最优结果,绘制出如图 14 所示的发生资源干扰后,最优调度方案对应的甘特图。将发生资源总量干扰后的新最优调度与原最优调度的开始时间和结束时间绘制成如表 6 所示的表格。

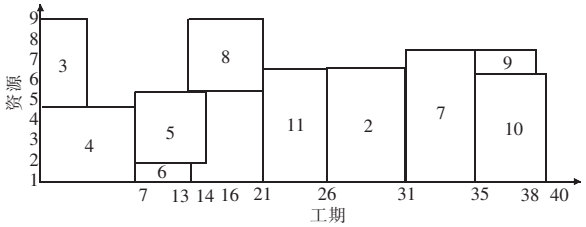

图 14　资源干扰后的最优调度甘特图

表 6　任务延期干扰前后调度时间对比

任务号	原调度开始时间	原调度结束时间	新调度开始时间	新调度结束时间
1	0	0	0	0
2	18	23	26	31
3	0	6	0	6
4	0	7	0	7
5	7	14	7	14
6	7	13	7	13
7	23	27	31	35
8	14	22	13	21
9	27	30	35	38
10	27	32	35	40
11	13	18	21	26
12	32	32	40	40
S_1	S_2			
32.083	32.083			

对比表 6 中的新最优调度与原最优调度,新调度方案工期为 40,比原方案多 8 个单位时间,新方案共有 7 个任务的调度顺序发生调整,开始时间方差 S_1 与结束时间方差 S_2 均为 32.083。这说明在 $t=10$ 时刻,在这个项目中资源总量减少 1 个单位对系统造成的干扰对原调度计划影响很大,大部分未执行的任务调度时间和调度顺序都要调整,要恢复到原最优调度计划或者最接近原最优调度计划,实现起来比较复杂。

综上所述,运用混沌 PSO 算法能够求解单资源单执行模式的项目网络图干扰、任务延期干扰,以及资源总量减少引起的干扰问题,并且效果比较理想。

5 带干扰的多模式多资源的项目调度问题仿真

第 4 章主要解决项目中只有一种执行模式和一种可更新资源的干扰问题,而在实际中可能会面临更加复杂的情况。例如,可能有的任务有多种执行模式,当选择不同的模式时会有不同的加工时间和资源消耗量,并且一般项目都会拥有众多资源,包括可更新的和不可更新的。因此在本章中,将研究多模式多资源受限的项目调度中的干扰问题,并分别从网络图干扰、任务干扰和资源干扰三个方面进行实例仿真。本章选取国际标准问题库(http://129.187.106.231/psplib/)中 J1015-5.SM 为仿真测试问题。

5.1 带干扰的多模式多资源的项目调度问题仿真

J1015-5.SM 问题可以描述为:一个项目有 10 个任务,各个任务的逻辑关系如图 15 所示,其中任务 1 和 12 为虚拟任务,是为了方便计算引进的任务,它们在执行过程中不占用时间和资源,如何选择执行模式并安排各个任务的调度顺序和开始时间才能使总工期和总成本达到最优组合?

整个项目执行过程中用到两种可更新资源和两种不可更新资源,其资源总量分别为 13、15、61、63,每个任务有 1 个或者多个可供选择的执行模式,不同执行模式所使用的执行时间和资源数量不同,各任务对应的具体数据如表 7 所示。以下 5.2~5.4 节均以该调度算例为研究对象。

表 7 J1015-5.SM 作业信息表

任务号	执行模式	工作时间	紧后任务号	资源量			
				K_1	K_2	N_1	N_2
1	1	0	2、3、4	0	0	0	0
2	1	2	8	0	7	9	5
	2	5		0	3	6	3
	3	7		2	0	4	3
3	1	1	10、11	0	7	4	6
	2	3		4	0	2	4
4	1	1	5、6、7	0	10	7	9
	2	2		7	0	7	9
5	1	2	8、10	3	0	5	8
	2	4		0	5	2	8
6	1	2	11	0	9	10	10
7	1	4	10	3	0	9	4
	2	6		0	4	8	3
8	1	2	9、11	0	4	8	9
	2	6		3	0	7	7
9	1	1	12	0	9	4	4
	2	3		10	0	4	4
10	1	2	12	0	6	6	8
	2	8		0	6	4	7
11	1	2	12	9	0	8	7
	2	4		8	0	5	5
12	1	0	—	0	0	0	0
资源单位成本				200	150	400	250

注:可更新资源 K_1、K_2 的总量分别为 13、15,不可更新资源 N_1、N_2 的总量分别为 61、63

原始调度最优解需要考虑工期和成本的综合影响,所以需要建立一个双目标函数。双目标函数建立的思路:先分别建立工期函数和成本函数,再将其无量纲化(归一化),各赋一个权重,权重的大小反映了两者的重要程度。根据上述思路可以建立如下模型。

$$FT=\sum_{m}\sum_{t}tx_{m1t} \tag{15}$$

$$FC=\sum_{i=1}^{D}\sum_{m=1}^{M}\sum_{k=1,2}c_k d_{ikm}R_{ikm}+\sum_{m=1}^{M}\sum_{k=3,4}c_k R_{km} \tag{16}$$

经过无量纲化之后,运用混沌粒子群算法求解该模型的最优解。算法参数设置为:学习因子 $c_1=c_2=1.49445$,粒子群种群规模 popsize=10,进化次数 maxgen=50。工期权重和成本权重的大小会对最优结果产生影响,当工期权重和成本权重取不同的值时,综合函数的最优值会有差异,权重取值的大小需要根据实际情况确定。在下面的实例仿真中,假设工期的权重比成本的权重大,取 $\omega_T=0.75$,$\omega_C=0.25$。

5.2 多模式调度的项目网络图干扰

本节仍然以网络图增加一个任务为研究对象,求解新的最优调度使干扰后的系统恢复到或者接近原最优调度状态。本节描述的项目在执行过程中,在 $t=4$ 时刻由于某种原因,需要对原网络图进行调整,在任务 7 和 10 之间添加一个任务 a,它只有一个执行模式,其作业时间为 3,四种资源的需求量分别为 4、3、1、0,项目网络图发生干扰后的网络图如图 16 所示。

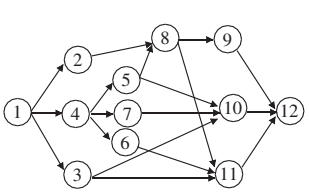

图 15 J1015-5.SM 网络图

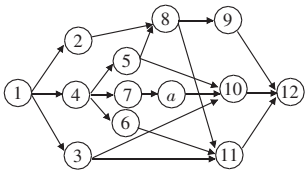

图 16 多模式项目调度增加任务后的网络图

该类问题的解决方法与 4.2 节的方法相似,首先将发生干扰时系统中正在执行的和没有执行的任务组成一个新的集合,然后对新集合重新调度。首先需要求原最优调度计划,取工期的权重 $\omega_T=0.75$,成本的权重 $\omega_C=0.25$,运用混沌粒子群算法求解得到原最优调度的最优模式、开始时间和结束时间,如表 8 所示。

表 8 原最优调度信息表

任务号	执行模式	开始时间	结束时间	任务号	执行模式	开始时间	结束时间
1	1	0	0	7	2	3	9
2	2	0	5	8	1	5	7
3	2	0	3	9	1	7	8
4	1	0	1	11	2	7	11
5	1	1	3	10	1	9	11
6	1	1	3	12	1	11	11

将开始时间大于等于 4 和结束时间大于 4 的任务组成一个集合 $P=\{2,7,a,8,9,11,10,12\}$;然后再在这些任务基础上重新调度,运用干扰模型求解最优调度方案。在 MATLAB 程序设计时,只需要将任务 1~7 的执行模式设定为表 8 中的最优模式,任务 1~7 的优先级粒子按照调度先后顺序设置为近似于 1 的实数,再对集合 P 里面未调度的任务进行重新调度。网络图中增加一个任务 a 后,取任务每延期 1 个单位的惩罚成本 $dc=3000$,模式变更成本 $mc=2000$ 时的最优调度,混沌粒子群参数设置同上。运行的结果如图 17 所示,其中 97 表示任务

a,从图 17 可以看出在 $t=4$ 时刻,增加任务 a 后的最优调度工期为 15,增加的成本为 32650。结合图 17 中的数据可以画出四种资源的甘特图,分别如图 18 ~ 21 所示。

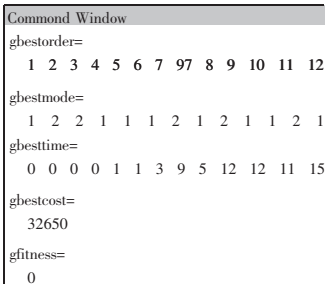

图 17　t=4 时刻增加任务 a 后的最优调度

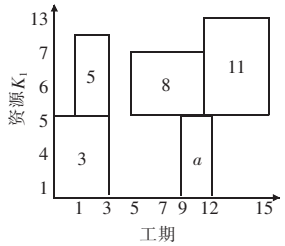

图 18　新最优调度中资源 K_1 对应的甘特图

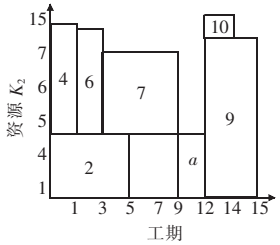

图 19　新最优调度中资源 K_2 对应的甘特图

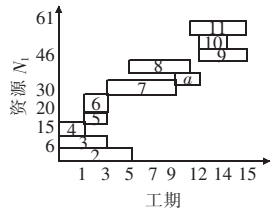

图 20　新最优调度中资源 N_1 对应的甘特图

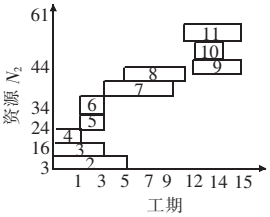

图 21　新最优调度中资源 N_2 对应的甘特图

综上所述,用混沌粒子群算法求得的这个解是可行的,增加一个工期为 3 的任务后,最优的综合目标值对应的工期从 11 变为 15,增加了 4 个单位,成本增加了 32650。通过增加迭代次数和多次运行,观察发现工期 15 是干扰后最优调度的最优工期,对原系统的扰动也最小,验证了混沌粒子群算法求解网络图干扰的有效性。

5.3　多模式调度的任务干扰

任务延期干扰是目前项目调度问题中出现最为频繁的一个问题,解决这类问题有十分重要的实际意义,所以本节将主要对项目调度中的任务延期干扰进行仿真研究。

假设在 $t=4$ 时刻,任务 7 未满足原计划进度,需要延期 3 个单位时间。这会对其后的未调度任务产生影响,原调度计划可能会发生改变,如果对原调度计划有影响就需要重新制定调度计划。

该问题解决思路与前面的一样,MATLAB 程序设计的方法同 5.2 节相似,取任务每延期 1 个单位的惩罚成本 $dc=3000$,模式变更成本 $mc=2000$ 时,算法参数设置同上。任务 7 延期后,程序的运行结果如图 22 所示。从该图中可以看出,在 $t=4$ 时刻,任务 7 延期 3 个单位的时间最优目标值对应的工期为 14,增加的成本为 32800。根据以上数据可以得出四种资源对应的甘特图,在此省略。

该问题研究结果的分析表明,任务 7 延期 3 个单位的时间后,新最优调度的总工期比原来的工期向后延期 3 个单位,经过多次运行后,发现工期 14 一直是粒子群算法求得的最优结果。通过分析验证,确认 14 是符合要求的,并且它就是该干扰问题的最优解,从而证实了混沌粒子群算法求解任务延期干扰问题的有效性。

5.4　多模式调度的资源干扰

本节主要研究资源总量减少对项目调度造成的干扰,假设在 $t=4$ 时刻,项目中可更新资源 K_2 的总量减少了 3 个单位,变成了 12 个单位,如何重新安排调度使干扰对系统的影响最小。

资源总量减少一般会对原调度产生影响。比如,在原调度计划中,在某个时刻系统是满负荷运行的,那么资源总量减少必定会使原调度不可行,这时就需要重新调度来满足要求。解决思路和前几节的相似,MATLAB 程序设计中的参数设置也与前几节一致,程序运行结果如图 23 所示。从该图中可以看出,在 $t=4$ 时刻,R_2 资源总量减少 3 个单位的最优目标值对应的工期为 12,增加的成本为 25000。其对应的四种资源甘特图在此省略。

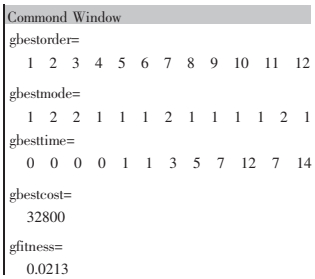

图 22　任务 7 延期的运行结果

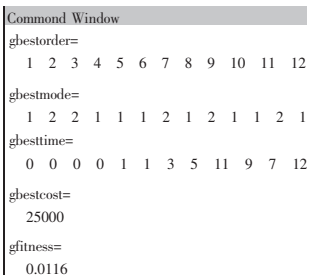

图 23　资源干扰程序运行结果

通过对运行结果图的分析发现,资源 K_2 总量减少 3 个单位后,新最优调度的总工期比原来的工期向后延期 1 个单位,经过验证,确认工期 12 的调度是符合要求的,这个解也是最接近原最优调度的一个解。通过多次实验发现,在可更新资源的总量变化幅度不是太大的情况下,系统恢复的效果非常显著。总之,混沌粒子群算法求解资源干扰效果是可行的,并且求解效果很好。

6　结束语

在本文中,首先从最简单的单执行模式单资源的 RCPSP 开始研究,分别对网络图干扰、任务干扰和资源干扰等三种干扰问题进行实例仿真,再通过混沌粒子群算法求解干扰后系统扰动最小的调度方案。然后仍然以三种干扰问题为切入点,再对复杂的多模式多资源的 RCPSP 进行仿真。通过实例仿真结果发现,混沌粒子群算法能够求解上述干扰问题,并且求解结果比较令人满意,从而验证了算法和模型的有效性。研究结论说明,先进的智能算法可以有效解决多模式多资源的项目调度的干扰管理问题,为干扰调度提供了可行的解决方案。进一步的工作将围绕各种干扰源的组合干扰和大规模的项目调度干扰管理问题来展开。

参考文献:

[1] YU Gang, QI Xiang-tong. Disruption management: framework, models and applications [M]. Singapore: World Scientific Publishing Co. Pte. Ltd., 2004: 1-50.

(下转第 2659 页)

从实验结果可以看出,GPU 版本的算法在性能上明显高于 CPU 版本,整体算法的加速比达到了 50 倍左右。可以看到,在构建预测矩阵的时候,算法的效率得到了最大的提高。而在最后构建推荐结果矩阵时,加速比却明显偏低,只有 2 倍左右,这是因为该步骤里面使用了复杂的 Top-*N* 算法,而 GPU 的处理单元是不擅长于复杂的逻辑运算。但因为该步骤所使用的时间在整个算法的运算时间中所占比例比较小,所以对整个算法效率的提高影响不大。

从召回率的实验结果(表 4)可以看出,无论是在计算时间还是在准确率方面,本文的算法相比 Puntheeranurak 等人[8]在 2011 年提出的方法有明显的优势。因为在数据集 *A* 和 *B* 中,某些用户对电影的评价总是偏向于给较低的分数而另一些用户则偏向于给较高的分数。本文设计的项目相关性矩阵、用户偏好向量、预测向量和推荐结果向量有利于得出这一特征信息。再者,这些矩阵与向量的代数结构比较容易实现于 CUDA 框架中,所以,本文提出的并行协同过滤算法在推荐准确率和处理时间上都有较优的性能。

4　结束语

为了解决协同过滤算法的介绍性问题,本文设计了一套基于 CUDA 的协同过滤算法。首先,定义了算法问题和参数意义。其次,提出了算法的设计思想,本文提出的算法主要是在基于项目的协同过滤算法的基础上进行设计的。然后,提出了算法的详细实现步骤,主要分为构建相关性矩阵、构建用户偏好向量、构建预测向量、构建推荐结果向量,共四个步骤。接着,本文介绍了如何使用 CUDA 实现这个并行算法。最后,本文从准确率和性能两个方面对这个算法进行测试,测试结果显示该算法有着较高的加速比。

参考文献:

[1] ADOMAVICIUS G, TUZHILIN A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions[J]. IEEE Trans on Knowledge and Data Engineering, 2005, 17(6): 734-749.

[2] 陈健,印鉴. 基于影响集的协作过滤推荐算法[J]. 软件学报, 2007, 18(7): 185-194.

[3] MELVILLE P, MOONEY R J, NAGARAJAN R. Content-boosted collaborative filtering for improved recommendations[C]//Proc of the 19th National Conference on Artificial Intelligence. Menlo Park: American Association for Artificial Intelligence, 2002: 187-192.

[4] CGEORGE T, MERUGU S. A scalable collaborative filtering framework based on co-clustering[C]//Proc of the 5th International Conference on Data Mining. Washington DC: IEEE Computer Society, 2005: 625-628.

[5] 吕成戍,王维国,丁永健. 基于 KNN-SVM 的混合协同过滤推荐算法[J]. 计算机应用研究, 2012, 29(5): 1707-1709.

[6] 吴月萍,杜奕. 基于人工鱼群算法的协同过滤推荐算法[J]. 计算机工程与设计, 2012, 33(5): 1852-1856.

[7] LI Rui-feng, ZHANG Yin, YU Hai-han. A social network-aware top-*N* recommender system using GPU[C]//Proc of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries. 2011: 287-296.

[8] PUNTHEERANURAK S, CHAIWTOOANUKOOL T. An item-based collaborative filtering method using item-based hybrid similarity[C]//Proc of the 2nd IEEE International Conference on Software Engineering and Service Science. 2011: 469-472.

[9] SONG Guo-hui, SUN Shu-tao, FAN W. Applying user interest on item-based recommender system[C]//Proc of International Conference on Computational Sciences and Optimization. 2012: 635-638.

[10] ZHENG Si-ting, HONG Wen-xing, ZHANG Ning, *et al*. Job recommender systems: a survey[C]//Proc of International Conference on Computer Science & Education. 2012: 920-924.

(上接第 2655 页)

[2] 胡祥培,丁秋雷,张漪,等. 干扰管理研究评述[J]. 管理科学, 2007, 20(2): 2-8.

[3] 陈庭贵,琚春华. 多干扰的资源约束项目调度问题[J]. 计算机集成制造系统, 2012, 18(11): 2409-2418.

[4] 刘志霞. 资源受限项目调度问题及其任务扰动的干扰管理研究[D]. 沈阳:沈阳工业大学, 2011.

[5] 陈卫明. 动态环境下产品开发项目调度问题及其求解研究[D]. 武汉:华中科技大学, 2011.

[6] LAMBRECHTS O, DEMEULEMEESTER E, HERROELEN W. Time slack-based techniques for robust project scheduling subject to resource uncertainty[J]. Annals of Operations Research, 2011, 186(1): 443-464.

[7] KLIMEK M, LEBKOWSKI P. Resource allocation for robust project scheduling[J]. Bulletin of the Polish Academy of Sciences-Technical Sciences, 2011, 59(1): 51-55.

[8] HAZIR O, HAOUARI M, EREL E. Robust scheduling and robustness measures for the discrete time/cost trade-off problem[J]. European Journal of Operational Research, 2010, 207(2): 633-643.

[9] ZHU Z, BARD J F, YU G. Disruption management for resource-constrained project scheduling[J]. Journal of the Operational Research Society, 2005, 56(4): 365-381.

[10] AL-FAWZAN M A, HAOUARI M. A bi-objective model for robust resource-constrained project scheduling[J]. International Journal of Production Economics, 2005, 96(2): 175-187.

[11] KOLISCH R. Serial and parallel resource-constrained project scheduling methods revisited: theory and computation[J]. European Journal of Operational Research, 1996, 90(2): 320-333.

[12] ZHANG Hong, LI Xiao-dong, LI Heng, *et al*. Particle swarm optimization-based schemes for resource-constrained project scheduling[J]. Automation in Construction, 2005, 14(3): 393-404.

[13] ZHANG Hong, LI Heng, TAM C M. Particle swarm optimization for resource-constrained project scheduling[J]. International Journal of Project Management, 2006, 24(1): 83-92.

[14] CHEN R M, WU C L, WANG C M, *et al*. Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB[J]. Expert Systems with Applications, 2010, 37(3): 1899-1910.

[15] ZHANG Hong, XING Feng. Fuzzy-multi-objective particle swarm optimization for time-cost-quality tradeoff in construction[J]. Automation in Construction, 2010, 19(8): 1067-1075.

[16] 王东升,曹磊. 混沌、分形及其应用[M]. 合肥:中国科学技术大学出版社, 1995.