柔性资源受限多项目调度的混沌粒子群算法研究*

陈君兰，叶春明
（上海理工大学，上海 200093）

摘 要： 为解决柔性资源受限多项目的调度问题，有效实行资源分配和工作时间安排，采用混沌粒子群算法结合混合优先规则，形成优先规则序列。针对多项目问题，避免了传统方法将多个项目合并为一个项目，而是形成一个链表在项目的各工序间进行选择调度，并在初始化中嵌入混沌理论，在迭代过程中使用并行算法，有效避免了算法易陷入局部最优解的可能。改编标准库的多模式算例，对比多种算法下的结果，验证了该方法在求解该问题的可行性和有效性，对于项目管理中柔性资源受限问题具有实际应用价值。

关键词： 柔性资源；粒子群；多项目调度；混沌
中图分类号： TP391；TP301.6 **文献标志码：** A **文章编号：** 1001-3695(2013)01-0117-04
doi:10.3969/j.issn.1001-3695.2013.01.028

Chaos particle swarm optimization on flexible-resource constrained multi-project scheduling research

CHEN Jun-lan, YE Chun-ming
(*University of Shanghai for Science & Technology*, *Shanghai* 200093, *China*)

Abstract: In order to solve the flexible-resource constrained multi-project scheduling problem and effectively solve the resource distribution and start time of tasks, this paper used CPSO (chaos particle swarm optimization) to solve this problem, combined with multi-work's priorities ways. And for multi-project scheduling problem, using the way to form a chain of all tasks in all projects and then chose from top to bottom to separately schedule instead of taking multi-project into a big project. It used chaos to initiate the regional group and during the process used both PSO and chaos to updated the group, and left the best result and it's proved better way to solve the problem from escaping the group best result. Finally, psplib database of this problem was modified under this problem. Comparing with other two algorithms, it proves the possibility and effect of this method in solving this problem. Therefore, this method has its practical application value for the flexible-resource constrained project scheduling problem.

Key words: flexible-resource; particle swarm optimization; multi-project scheduling; chaos

0 引言

柔性资源受限的项目调度问题是资源受限项目调度问题中的一种。柔性资源受限的项目调度问题可描述为：存在具有不同技能的有限资源，这些资源被称为柔性资源，而完成一项工序则需要不同的技能，为了在资源不冲突的情况下，合理安排各个工序，使整个项目的最终完成时间最短。对于柔性资源受限问题的研究源于柔性资源受限的车间调度问题，后运用在项目调度研究上，而目前使用较多的方法是建立一个简单的0-1资源技能矩阵。例如喻小光等人[1]通过建立两级映射模型表达任务—能力—资源的关系求解了资源均衡情况下的项目调度问题；黄敏镁等人[2]通过建立任务—技能矩阵和资源—技能矩阵，使用改进的遗传算法，采用拓扑排序和最大流理论对产品开发项目实例进行了研究。

多项目问题指同时对多个项目进行调度。由于在项目调度问题中，经常会出现多个项目共用相同资源，并几乎同时进行的情况发生，使得任务之间的资源冲突加剧，所以在解决多项目问题时，通常将多个项目合并为一个项目考虑，但是当项目数较多时，就会使计算繁琐，运算效率低下。因而很多学者开始研究新的方法以解决多项目问题。例如应瑛等人[3]建立了拍卖模型，通过让工序在每个调度阶段对资源竞价来确定资源的优先分配和工序的优先调度；徐赐军等人[4]根据资源维拉技术消解资源冲突，并在此基础上根据最小冲突强度和最小延迟构建动态优先规则。

柔性资源的多项目调度问题已逐渐受到关注，罗荣桂等人[5]对国内外柔性资源及多项目调度问题进行了研究，分析了拓展资源受限的多项目问题到柔性资源，并对该方向研究和发展作出了展望。将柔性资源受限与多项目调度相结合正是本文的研究重点，这类问题更符合目前项目调度问题，尤其是产品开发项目中的实际情况，具有更好的实际意义。

本文在解决多项目问题时，单独考虑多个项目，通过建立一个项目选择的数组，从而解决多项目调度的问题。在此基础

收稿日期：2012-05-10；修回日期：2012-07-02 基金项目：国家教育部人文社会科学规划基金资助项目(10YJA630187)；高等学校博士点基金资助项目(20093120110008)；上海市重点学科建设资助项目(S30504)；上海市教育委员会科研创新资助项目(12ZS133)；上海市研究生创新基金资助项目(JWCXSL1102)

作者简介：陈君兰(1988-)，女，上海人，硕士，主要研究方向为项目管理(tracychen0602@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、企业战略、企业生产计划与控制、生产调度、企业资源计划(ERP)、供应链管理及企业信息化.

上，结合文献[1]的方法，改进了矩阵，使其体现出资源对技能的熟练度。

1 问题描述

假设共有 P 个项目，每个项目用 $k(k=1,2,\cdots,P)$ 表示，每个项目中的工序总数用 N_k 表示，项目 k 由工序集 J_k 组成，该项目中每道工序用 $j_k(j_k=1,2,\cdots,N_k)$ 表示。由于每道工序都有固定执行时间，因此该项目某一工序的执行时间用 TD_{j_k} 表示，每道工序的开始时间用 TS_{j_k} 表示。由于无论是多项目还是单项目，工序间都有相应的先后约束关系，因此用 IP_{j_k} 表示第 k 个项目的第 j 道工序的紧前工序集合，用 IS_{j_k} 表示第 k 个项目的第 j 道工序的紧后工序集合。假设所有项目都需在 SUM 前完成，即 SUM 为总项目的工期上限，则 ES_{j_k} 表示第 k 个项目的第 j 道工序的最早开始时间，LS_{j_k} 表示第 k 个项目的第 j 道工序的最晚开始时间；假设所有项目共用到技能种类 v 种，每项技能用 s 表示($s=1,2,\cdots,v$)，r_{j_ks} 表示第 k 个项目的第 j 道工序对技能 s 的需求量，因此由 r 构成技能需求矩阵 R；同样改进文献[1]的方法，从而建立技能供给矩阵，假设共有资源数量 H，每项资源用 $h(h=1,2,\cdots,H)$ 表示，用 ρ_{hs} 表示资源对技能 s 的熟练度，因此由 ρ 构成了技能供给矩阵，这个矩阵表示了某一资源对某项技能的熟练度，每项资源所具有所有技能的熟练度的和为 1，即生成的技能供给矩阵每一行的和为 1。数学模型表示如下：

$$\rho_{hs}=\begin{cases}0 & \text{资源 } h \text{ 不具有技能 } s\\ (0,1] & \text{资源 } h \text{ 对技能 } s \text{ 的熟练度}\end{cases} \tag{1}$$

$$\varphi_{hs}=\begin{cases}0 & \rho_{hs}=0\\ 1 & \rho_{hs}\neq 0\end{cases} \tag{2}$$

$$\min(\max\ TS_{kN_k}) \qquad k=1,2,\cdots,P \tag{3}$$

$$TS_{j_k}-TS_{i_k}\geqslant TD_{i_k} \qquad i_k\in IP_{j_k} \tag{4}$$

$$\sum_{h=1}^{H}x_{ht}\times\varphi_{hs}\geqslant r_{j_ks}, t\in[TS_{j_k},TS_{j_k}+TD_{j_k}]$$
$$s=1,2,\cdots,v;\ j_k=1,2,\cdots,N_k \tag{5}$$

$$\sum_{k=1}^{P}\sum_{s=1}^{v}r_{j_ks}\leqslant\sum_{h=1}^{m}x_{ht} \quad j_k\in A_t, t\in[TS_{j_k},TS_{j_k}+TD_{j_k}] \tag{6}$$

$$\sum_{s=1}^{v}\sum_{t=TS_{j_k}}^{TS_{j_k}+TD_{j_k}}r_{j_ks}\leqslant R_s \tag{7}$$

$$x_{ht}=\begin{cases}1 & t \text{ 时刻资源 } h \text{ 可用}\\ 0 & t \text{ 时刻资源 } h \text{ 不可用}\end{cases} \tag{8}$$

其中：R_s 表示资源对技能 s 的总供给；A_t 表示($t-1,t$)时间段内正在进行的工序集合；r_{j_ks} 表示在 t 时刻第 k 个项目的第 j 道工序对技能 s 的需求量。式(3)为目标函数，式(4)为每个项目的前后关系时间约束，式(5)表示任意工序加工时间段内具有该工序所需技能且可用的资源数不小于该工序对技能的需求量，式(6)表示任意时段各种技能总需求量不能超过资源对技能的总供给量，式(7)表示资源对技能的总供给大于总需求，式(8)表示资源在 t 时刻是否可用。对于多模式的柔性资源受限多项目调度问题，只需在模型中考虑加入模式选择和模式约束(即只能有一种模式被选择)。

2 算法原理

本文采用并行混沌粒子群算法，并在初始化时先混沌序列化，求出较优的粒子作为初始粒子。本文应用 logistic 方法混沌化序列：

$$x_{n+1}=\mu x_n(1-x_n) \quad n=0,1,\cdots \tag{9}$$

其中：μ 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。

假设粒子 i 用 $x_i=(x_{i1},x_{i2},\cdots,x_{iD})$ 表示，其中 D 为粒子的维度，它经过代数进化后所经历的最好位置称为局部最优解，用 $p_i=(p_{i1},p_{i2},\cdots,p_{iD})$ 表示；而整个种群(由多个粒子构成)在经过代数进化后所遍历得到的最优解称为全局最优解，用 $p_g=(p_{g1},p_{g2},\cdots,p_{gD})$ 表示，每个粒子的速度用 $v_i=(v_{i1},v_{i2},\cdots,v_{iD})$ 表示。粒子速度和位置的更新方程改进如下：

$$v_{ij}^{t+1}=wv_{ij}^t+c_1r_1(p_{ij}^t-x_{ij}^t)+c_2r_2(p_{gj}^t-x_{ij}^t) \tag{10}$$

$$x_{ij}^{t+1}=x_{ij}^t+v_{ij}^{t+1} \tag{11}$$

其中：c_1、c_2 为学习因子，t 为代数。式(10)为速度更新方程，式(11)为位置更新方程。w 为惯性因子，一般而言，惯性因子的取值为 $w\in[0.8,1.2]$，多数实验证明在这个取值范围中，算法收敛速度更快。当 $w>1.2$ 时，算法则较多地陷入局部最优解，而随着迭代的增加，惯性因子也应不断减少[4]。所以取惯性因子为 0.8～1.2 递减。

混沌粒子群(CPSO)算法流程如下：

a) 混沌初始化形成粒子位置和粒子速度，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，利用式(9)更新混沌序列，并计算、评价适应度 F_1。

c) 以粒子初始种群为基础更新粒子群，利用式(10)和(11)分别更新粒子位置和速度，并计算和评价规则下满足柔性资源约束的适应度 F_2。

d) 比较两个适应度 F_1 和 F_2，保留较优的解。更新粒子局部最优和全局最优。

e) 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)。

3 求解问题的算法

对于多项目问题，一般的做法是将多个项目合并为一个项目计算，但是这会造成工序冗余、计算速度慢等一系列问题，因此本文还是将多个项目分开计算，使用一组数代表项目的选择，在多个项目中跳跃选择某一项目的某一工序进行调度。

假设共有项目 N 项，总工序数为 MAX，各项目用 n 表示，($n=1,2,\cdots,N$)，且每道工序可无空闲执行。

3.1 个体编码

本文解决多项目问题的主要思想是：先选择项目，后根据每个项目中工序的优先值，以先进先出为原则，填入优先规则链表。因此算法编码分为 x 和 y 两种粒子：粒子 x 记录工序随机优先级，用于计算每道工序的优先值；粒子 y 记录项目编号，用于选择项目。对工序编码时，根据项目数依次递增，工序号 = 项目内工序号 + 前几个项目的总工序数。

3.2 混沌初始化

随机生成满足资源量约束和技能约束的条件的技能供给矩阵，该矩阵由[0,1]间的小数构成，代表了资源拥有某一技能的能力。例如：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.6 & 0 & 0.2 \\ 0 & 0.3 & 0.4 & 0.3 \\ 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix}$$

其中: 列表示技能，行表示资源。因此由技能供给矩阵可知: 资源 1 具有技能 1，且熟练度为 1; 资源 2 具有技能 1、2、4，且熟练度分别为 0.2、0.6、0.2; 资源 3 具有技能 2、3、4，且熟练度分别为 0.3、0.4、0.3; 资源 4 具有技能 1、2、3、4，且熟练度分别为 0.1、0.2、0.3、0.4。

初始化所有工序开始时间为 0。混沌初始化时，对应每个项目的每一道工序，随机产生一组 n 维的 [0,1] 间的数作为混沌变量，并利用式(9) 更新混沌序列。其中虚开始工序对应 0，虚结束工序对应 1，random(a,b) 表示在 a、b 间随机生成一个小数，$x_s=(x_{s1},x_{s2},\cdots,x_{sn})$，粒子位置用 x_s 表示:

$$x_s=\text{random}(0,1) \tag{12}$$

粒子速度初始化时，采用式(12)，在 [-1,1] 间随机取值，$x_v=(x_{v1},x_{v2},\cdots,x_{vn})$，$x_v$ 表示粒子速度，

$$x_v=\text{random}(v_{\min},v_{\max}) \tag{13}$$

其中: $v_{\min}=-1$，$v_{\max}=1$。粒子 x 中记录了项目编号。

同样，混沌初始化，对于粒子 y_s 位置，所有项目的工序位置一起随机产生一组 $1\sim N$ 的矩阵，代表各项目编号，其中具有同一项目编号工序位置的总数应为该项目的工序总数，矩阵中共有数为所有项目的工序总和。

例如有三个项目，项目 1 有工序数为 4，项目 2 有工序数为 3，项目 3 有工序数为 6，则可生成 y_s = [1,2,1,3,3,2,1,1,2,3,3,3,3]，表示在这 13 道工序中，第一个安排项目 1 中的工序，第二个安排项目 2 中的工序，第三个安排项目 1 中的工序，第四个安排项目 3 中的工序，依此类推。

对于粒子 y_v 速度，随机生成 [-1,1] 间的一组数，生成过程中包括虚工序，利用式(10) 更新粒子 y_v 速度，并调整粒子 y_v，公式为

$$y_v^{t+1}=\begin{cases} v_{\min} & y_v^t<v_{\min} \\ y_v^t & v_{\min}<y_v^t<v_{\max} \\ v_{\max} & y_v^t>v_{\max} \end{cases} \tag{14}$$

$$\sigma_j=\frac{1}{1+\left(\dfrac{v_{\max}-y_v^{t+1}}{y_v^{t+1}-v_{\min}}\right)^2} \tag{15}$$

对所有的 σ_j 进行从小到大的排序，按照各项目的工序数，从第一个开始，对应某一项目的某道工序，将 y_s 中的这些个数的位置赋值为 1，依此类推。

例如仍是这三个项目，项目 1 有工序数为 4，项目 2 有工序数为 3，项目 3 有工序数为 6，项目 1 中的 4 道工序编号为 1，2，3，4，项目 2 中的 3 道工序编号为 5，6，7，项目 3 中的 6 道工序编号为 8，9，10，11，12，13。σ = [0.132，0.224，0.532，0.154. 0.312，0.123，0.217，0.281，0.191，0.121，0.341，0.261，0.421] 排序得到的结果为 [10，6，1，4，9，7，2，12，8，4，11，13，15，3]，则令工序编号为 10，6，1，4 的 4 道工序 $y_{sj}=1$，令工序编号为 9，7，2 的 3 道工序 $y_{sj}=2$，剩余工序编号 $y_{sj}=3$，最终更新得到的 y_s = [1，2，3，1，3，1，2，3，2，1，3，3，3]。

根据粒子 x 和 y 计算目标函数，从中选择较好的 popsize 个粒子作为粒子 x 和 y 的初始位置。

3.3　调度生成

调度生成的具体步骤如下:

a) 分别计算各项目工序的最短开始时间和最晚开始时间。各项目的期限为各项目内所有工序的工期总和。

b) 对各项目内的工序分别赋优先值。针对柔性资源受限问题的多项目调度问题，优先值赋值公式改为

$$\tau_{ij}=\left(TD_{ij}\frac{\sum_{k=1}^{K}r_{kij}}{H}\prod_{k=1}^{K}\frac{r_{kij}}{R_k}\right)^{x_s} \tag{16}$$

其中: $r_{kij}\neq 0$; $\dfrac{\sum_{k=1}^{K}r_{kij}}{H}$ 部分表示工序 j 总共使用的资源占总资源的比率; $\dfrac{r_{kij}}{R_k}$ 部分表示完成工序 j 需要有技能 k 的资源数占有技能 k 的资源总数的比率; H 为总共资源数; K 为技能种类; r_{kij} 表示项目 i 的第 j 道工序对技能 k 的需求量; R_k 表示技能 k 的总供给量。

在各项目内对 τ_{ij} 进行从大到小排序形成各项目的 ordertemp$_n$，因为资源占用率高的和工期大的工序适宜先调度，所以对 τ_{ij} 值大的工序先调度。

c) 根据粒子 y 从目标项目中选取工序按顺序依次存入优先规则链表内，并将存入优先规则链表的工序移出 ordertemp$_n$，其中的工序号依项目编号而重新编号，如项目 1 中的所有工序号不变，项目 2 中的所有工序号要加上项目 1 中工序的总数。仍以上述例子，项目 2 的工序号变更为 5、6、7，项目 3 的工序号变更为 8、9、10、11、12、13，各项目工序的优先值从大到小顺序为 ordertemp$_1$ = [1，3，2，4]，ordertemp$_2$ = [5，6，7]，ordertemp$_3$ = [8，11，9，10，12，13]。假设此时 y_s = [1，2，3，1，3，1，2，3，2，1，3，3，3]，则形成的优先规则链表 order = [1，5，8，3，11，2，6，9，7，4，10，12，13]。

d) 与单个项目的柔性资源受限项目调度问题相同，运用串行调度生成机制，按照所有项目的优先规则链表，order 进行工序时间分配。

3.4　种群更新

对于粒子 x，采用粒子群更新方程式(10)(11) 分别对粒子速度和位置更新，但由于粒子速度控制在 [-1,1] 间，因此对于超出边界的粒子速度取边界值。同时，由于粒子位置需控制在 [0,1] 间，因此对于超出边界的粒子位置取边界值。采用并行混沌粒子群算法，同时继续运用式(10) 更新粒子 x 的速度。

对于粒子 y，采用粒子群更新方程式(10) 更新粒子速度，并运用式(14) 对速度进行调整，利用式(15) 计算，并运用混沌初始化粒子 y 中提到的方法更新粒子 y 的位置。由于采用并行的混沌粒子群算法，同时继续运用式(10) 更新粒子 y 的速度。

4　实例研究

采用国际标准问题库中的多模式经典算例对算法求证，以 C1511.2.mm(项目 1) 和 C1511.3.mm(项目 2) (http://129.187.106.231/psplib/) 这两个项目的调度为例，将库中各工序对资源的需求量视为对技能的需求量，比较多个项目的资源最大值，将同一种资源的资源最大值中较大的那个视为该技能供给最大值，设定资源数为 50 个，则根据技能供给最大值随机生成技能供给矩阵，其数值在 [0,1] 之间，技能供给矩阵的行代表资源序号，列代表技能序号，其中的数值则代表技能熟练度。以表 1 的技能供给矩阵为例，测试算例。随机生成采用 MATLAB 7.0 进行算法编程，粒子群优化算法的参数设置如下: 种

群数 popsize = 30，学习因子 c_1、c_2 = 2，进化次数为 300 次，粒子速度取值为 [-1，1]，w 取值范围为 [0.8，1.2]。技能供给矩阵如表 1 所示。

表 1 技能供给矩阵

资源号	技能 1	技能 2	技能 3	技能 4	资源号	技能 1	技能 2	技能 3	技能 4
1	0.4	0.4	0.2	0	26	0.4	0	0.6	0
2	0	0.4	0.6	0	27	0	0	0.4	0.6
3	0	0.3	0.5	0.2	28	0.4	0	0.2	0.4
4	0	0	0.6	0.4	29	0	0.3	0.3	0.4
5	0	0.5	0.2	0.3	30	0	0	0.7	0.3
6	0.2	0	0.8	0	31	0.5	0	0.2	0.3
7	0	0	0.6	0.4	32	0.4	0.2	0.4	0
8	0.3	0.2	0.3	0.2	33	0.4	0	0	0.6
9	0	0	1	0	34	0.4	0	0.3	0.3
10	0.1	0.2	0.2	0.5	35	0	0.9	0.1	0
11	0	0	0.1	0.9	36	0	0	0.4	0.6
12	0.3	0	0.3	0.4	37	0	0	0.9	0.1
13	0.3	0.4	0.1	0.2	38	0.3	0	0.3	0.4
14	0.2	0.2	0.4	0.2	39	0	0	0.4	0.6
15	0	0.5	0.1	0.4	40	0	0.4	0.4	0.2
16	0	0.3	0.2	0.5	41	0	0	0.4	0.6
17	0	0.1	0.8	0.1	42	0	0	0.5	0.5
18	0.2	0.3	0.3	0.2	43	0	0	0	1
19	0.5	0	0.2	0.3	44	0.2	0	0.2	0.6
20	0	0	0.7	0.3	45	0.1	0.4	0.3	0.2
21	0.2	0.1	0.4	0.3	46	0.4	0	0.6	0
22	0	0.9	0.1	0	47	0	0	0.3	0.7
23	0	0	0.6	0.4	48	0	0	0.1	0.9
24	0.5	0	0.1	0.4	49	0.7	0	0.1	0.2
25	0	0	0.5	0.5	50	0.4	0.2	0.1	0.3

算例共有四种技能，拥有技能 1 的资源数为 24 个，拥有技能 2 的资源数为 21 个，拥有技能 3 的资源数为 50 个，拥有技能 4 的资源数为 44 个。计算的整个项目工期为 43 d，其中项目 1 的工期为 43 d，项目 2 的工期也为 43 d。

在这种技能供给矩阵情况下，分别运行遗传算法(GA)、传统 PSO 算法和本文提出的 CPSO 算法，本文选用文献 [7] 提出的遗传算法并作适当调整，种群数为 30，迭代次数为 300 次，变异概率 0.5；另选用文献 [6] 提出的粒子群算法并作适当调整，种群数为 30，迭代次数为 300 次，结果如表 2 所示。

表 2 GA、PSO、CPSO 运算结果比较

算法	与最优方案平均距离	达最优值次数百分比/%
GA	2.48	0
PSO	4.55	0
CPSO	1.43	38

由表 2 可知，只有 CPSO 算法达到了最优解 43d，而 GA 和 PSO 算法均没有达到最优解，且 CPSO 算法距离最优方案的平均距离小于 GA、PSO 算法距离最优方案的平均距离。由此分析可知，CPSO 算法在求解多模式的柔性资源受限多项目调度问题上优于其他两种算法，可见其有效性。三种算法运算后的收敛曲线如图 1 所示。

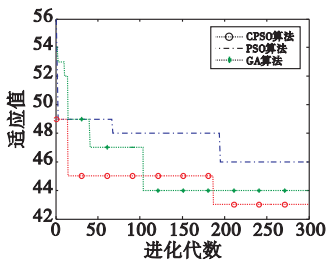

图1 GA、PSO、CPSO算法收敛图

由图 1 可以看出，CPSO 算法在求解多模式柔性资源受限多项目问题中具有较好的效果，且可行解更接近最优解。PSO 算法或 GA 在求解该问题时，在 300 代以内均无法达到 CPSO 所能达到的最优解。因此 CPSO 具有更好的收敛性，且能有效避免 PSO 算法易陷入局部最优解的缺点。

5 结束语

本文给出了柔性资源受限多项目调度问题的具体模型，通过混合优先规则的方法，结合式(16)中的 x 粒子，进行混沌粒子群算法迭代，而不同于将多个项目视为一个大项目的方法，本文设计出一个链表，记录该阶段所选取的项目，并对该项目中由优先规则排列出的即将调度的工序进行调度，同时设计出两种粒子，一种用在项目内排列优先值，也通过混沌粒子群算法对链表进行迭代更新。由于多项目问题中工序较多，所以初始化混沌序列，从中选取较优粒子作为初始粒子，同时考虑到粒子群算法易陷入局部最优。本文运用并行的混沌粒子群算法计算，比较混沌序列后和粒子群更新序列后适应值的大小，保留产生较优适应值的粒子。最后通过对国际标准库中的经典问题进行改编，分别用 PSO、GA、CPSO 算法进行算例测试，发现 CPSO 算法较传统 PSO 算法具有更好的搜索能力和收敛性，验证了该算法的有效性。

参考文献：

[1] 喻小光，战德臣，聂兰顺，等. 柔性资源约束的资源水平项目调度问题 [J]. 计算机集成制造系统，2010，16(9)：1967-1975.
[2] 黄敏镁，罗荣桂. 柔性资源约束下的产品开发项目优化调度研究 [J]. 管理工程学报，2010，24(4)：143-147.
[3] 应瑛，寿涌毅. 基于组合拍卖方法的资源受限多项目调度 [J]. 计算机集成制造系统，2009，15(11)：2160-2165.
[4] 徐赐军，李爱平，刘雪梅. 基于资源推拉技术的多项目调度算法 [J]. 计算机集成制造系统，2010，16(6)：1246-1254.
[5] 罗荣桂，杨世宏，吴兵，等. 柔性资源受限的多项目调度问题研究 [J]. 武汉理工大学学报，2006，19(6)：843-846.
[6] 曾建潮，介婧，崔志华. 微粒群算法 [M]. 北京：科学出版社，2004.
[7] 刘士新. 项目优化调度理论与方法 [M]. 北京：机械工业出版社，2006.
[8] 苗东升. 系统科学大学讲稿 [M]. 北京：中国人民大学出版社，2007.
[9] 叶春明，潘登，潘逢山. 基于混沌粒子群算法的关键链项目进度管理研究 [J]. 计算机应用研究，2011，28(3)：890-894.
[10] KRÜGER D，SCHOOL A. Managing and modeling general resource transfers in (multi-) project scheduling [J]. OR Spectrum，2010，32(2)：369-393.
[11] ZHANG Lian-ying，SUN Ruo-xin. An improvement of resource-constrained multi-project scheduling model based on priority-rule based heuristics [C] //Proc of International Conference on Service Systems and Service Management. 2011：1-5.
[12] LIU Shi-xin，TUKEL O I，ROM W. Flexible scheduling approach for resource-constrained project scheduling problems [C] //Proc of the 7th World Congress on Intelligent Control and Automation. 2008：3522-3526.
[13] ZDAMAR L，ULUSOY G. A local constraint based analysis approach to project scheduling under general resource constraints [J]. European Journal of Operational Research，1994，79(2)：287-298.
[14] KENNEDY J，EBERHART R C. Particle swarm optimization [C] //Proc of IEEE International Conference on Neural Networks. 1995：1942-1948.