文章编号: 1001-0920(2008)06-0681-04

粒子群算法求解任务可拆分项目调度问题

邓林义[a,b], 林 焰[b]
(大连理工大学 a. 电子与信息工程学院, b. 船舶 CAD 工程中心, 辽宁 大连 116024)

摘 要: 首先针对任务可拆分的项目调度问题,提出一种带有局部搜索的粒子群算法 LSPSO;然后采用基于任务排列的粒子表示方法,将遗传算法中的定位交叉引入粒子的更新过程中,并采用局部搜索技术对更新后的粒子进行改进;最后对 Patterson 测试集中 110 个问题实例进行了测试,实验结果表明,算法 LSPSO 具有较快的速度,所给出的调度方案较优.

关键词: 项目调度; 资源受限; 粒子群算法; 可拆分任务
中图分类号: TP391 **文献标识码**: A

Particle swarm optimization for resource-constrained project scheduling problems with activity splitting

DENG Lin-yi[a,b], *LIN Yan*[b]
(a. School of Electronic and Information Engineering, b. Engineering Center of Ship CAD, Dalian University of Technology, Dalian 116024, China. Correspondent: DENG Lin-yi, E-mail: hsdly@163.com)

Abstract: A local search particle swarm optimization (LSPSO) is proposed to solve the resource constrained project scheduling problem (RCPSP) with activity splitting. The LSPSO makes use of a permutation based particle representation and an updating mechanism with one-point crossover. Then, a local search technique is adopted to improve the quality of the updated particles. Finally, the algorithm is tested on the instance set Patterson, and the results show that the LSPSO is an alternative and efficient optimization methodology for solving the RCPSP.

Key words: Project scheduling; Resource-constrained; Particle swarm optimization; Activity splitting

1 引 言

典型资源受限的项目调度问题(RCPSP)是指在满足项目先序约束和资源约束的条件下,安排任务的开始时间和完成时间,以达到项目工期最小. RCPSP 广泛存在于企业的设计和生产活动中. 理论上该问题属于 NP-hard 问题[1],吸引了国内外众多学者的广泛关注. 目前,针对 RCPSP 的优化方法主要可分为两类: 精确算法和启发式算法. 精确算法(如整数规划、分支限界、动态规划等)对于求解小规模的调度问题是可行的; 对于大规模的调度问题一般需借助于启发式方法进行求解,包括基于优先规则的启发式方法、采样算法和智能优化算法等[2].

传统的资源受限项目调度问题的前提之一是任务不可拆分,即每个任务只能被一次执行,在执行过程中,任务不可停顿,资源不可剥夺. 但在企业的实际项目调度中,许多任务是允许被拆分成若干次执行的. 为更好地研究企业实际情况,本文将任务可拆分的项目调度问题作为研究目标.

Buddhakulsomsiri[3,4]通过实验分析了在资源紧缺和资源丰富的两种情况下,任务的拆分调度对项目工期的影响. 结果表明,对于两种情况,任务拆分都可有效地缩短项目的工期,而且在资源紧缺的情况下更为明显. 2006 年,汪定伟等人[5]采用混合遗传算法对任务拆分的项目调度问题进行研究,对 Patterson 测试集中 110 个问题实例进行了测试,针对不同复杂程度的项目,当任务可拆分时项目工期可缩短 0～4 d,平均缩短 1.86 d.

粒子群算法(PSO)[6-8]是一种基于进化的群体智能算法. Zhang[9,10]于 2005 年采用粒子群算法对 RCPSP 进行研究,取得了较好的结果. 然而,粒子群

收稿日期: 2007-03-12; **修回日期**: 2007-07-17.
基金项目: 国家 863 计划项目(2003AA414060).
作者简介: 邓林义(1978—),女,河北衡水人,博士生,从事智能算法、项目计划与控制的研究; 林焰(1963—),男,福州人,教授,博士生导师,从事船舶与海洋工程的研究.

算法在任务可拆分 RCPSP 中的应用,还没有查阅到相关文献.

本文将粒子群算法应用于任务可拆分的 RCPSP 中,与遗传算法的计算结果相比,进一步提高了运算速度,缩短了项目工期.

2 数学模型

任务可拆分的 RCPSP 可描述为: 一个项目包括 J 个任务,其中开始任务 1 和结束任务 J 是虚拟任务,不占用任何资源且工期为 0;任务 $j(j = 2,3,\cdots,J-1)$ 可拆分成多次执行,总耗时为 d_j 个时间单位; 任务与任务之间存在先序关系, P_j 表示任务 j 的先序任务集合, S_j 表示任务 j 的后续任务集合,任务 j 在其任一先序任务 $p(p \in P_j)$ 完成前不能开始; 项目共有 K 种可更新资源(每一时刻总量保持不变),第 k 种资源在任意时刻的总量为 $R_k(k = 1,2,\cdots,K)$,加工任务 j 所消耗的第 k 种资源的量为 r_{jk},任意时刻使用的各种资源量不能超过其总量. 任务可拆分的 RCPS 问题的目标就是要确定所有任务的开始时间和拆分方案,使项目在计划期 T 内的总工期最短. 在任务可拆分的 RCPSP 模型中还使用以下符号:

$$ES_j = \max\{EF_p \mid p \in P_j\},$$
$$\forall j \in \{2,3,\cdots,J\}, ES_1 = 0, EF_1 = 0.$$

其中: ES_j 是任务 j 的最早开始时间, EF_j 是任务 j 的最早结束时间.

$$LF_j = \min\{LS_s \mid s \in S_j\},$$
$$\forall j \in \{J-1, J-2, \cdots, 1\},$$
$$LF_J = T, LS_J = T.$$

其中: LS_j 是任务 j 的最晚开始时间, LF_j 是任务 j 的最晚完成时间.

x_{jt} 为 0-1 变量,即

$$x_{jt} = \begin{cases} 1, & \text{任务 } j \text{ 在时刻 } t \text{ 正在进行;} \\ 0, & \text{其他.} \end{cases}$$

任务 j 的开始时间为

$$e_j^S = \min_{ES_j \le t \le LS_j} \{t x_{jt} \mid x_{jt} = 1\},$$

任务 j 的完成时间为

$$e_j^E = \max_{EF_j \le t \le LF_j} \{t x_{jt} \mid x_{jt} = 1\}.$$

根据以上描述可给出以下模型:

$$\min e_j^E, \tag{1}$$

$$\text{s.t.} \sum_{t=ES_j}^{LF_j} x_{jt} = d_j, \ \forall j \in \{1,2,\cdots,J\}, \tag{2}$$

$$e_p^E \le e_j^S, \ \forall j \in \{1,2,\cdots,J\}, \forall p \in P_j, \tag{3}$$

$$\sum_{j=1}^{J} (r_{jk} \times x_{jt}) \le R_k, \ x_{jt} \in \{0,1\},$$
$$\forall k \in \{1,2,\cdots,K\}, \ t \in \{1,2,\cdots,T\}. \tag{4}$$

其中:式(1) 表示优化目标为项目工期最小; 约束(2) 表示任务拆分方式不影响任务的总工期; 约束(3) 表示每个任务必须满足先序约束; 约束(4) 表示在每个时刻使用的各种资源量不能大于其资源总量.

3 粒子群算法

粒子群算法的基本思想是通过群体中个体之间的协作和信息共享来寻找最优解. 其中,每个粒子表示调度问题的一个解,对应粒子在搜索空间中的一个位置,每个粒子通过速度决定它们的搜索方向和搜索范围;在整个寻优过程中,粒子通过追随群体的最优粒子(即全局最优) 和自身经历的个体最优位置(即个体极值) 来调节速度,以搜索最优解.

假设在一个 D 维的目标搜索空间中,POP 个粒子组成一个种群,每个粒子对应于 D 维的搜索空间中的一个点. 粒子 i 可表示为一个 D 维向量 $x_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$, $i = 1,2,\cdots,\text{POP}$. 将 x_i 带入目标函数可计算出该粒子的适应值,再根据适应值来衡量粒子 i 的优劣. 粒子 i 的速度也是一个 D 维向量,记为 $v_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. 粒子 i 当前搜索到的最优位置(个体极值) 记为 $p_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$,整个种群当前搜索到的最优位置(全局最优) 记为 $p_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$.

PSO 算法根据下式对粒子进行更新:

$$v_{id} = w v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}), \tag{5}$$

$$x_{id} = x_{id} + v_{id}. \tag{6}$$

其中: $i = 1,2,\cdots,\text{POP}$, $d = 1,2,\cdots,D$;学习因子 c_1 和 c_2 是非负常数; r_1 和 r_2 是介于 $[0,1]$ 之间的随机数;惯性因子 w 是非负数; $v_{id} \in [-v_{\max}, v_{\max}]$, $v_{\max}$ 是常数,由用户设定. 迭代中止条件通常为算法达到最大迭代次数. 由于具有较少的待定参数,粒子群算法已在连续性问题中得到了广泛应用. 2000 年, Maurice[11] 首次将粒子群算法的更新公式进行修改,并将其应用到求解离散问题中,修改后的更新公式可表示为

$$v_{id} = w v_{id} \oplus c_1 r_1 (p_{id} - x_{id}) \oplus c_2 r_2 (p_{gd} - x_{id}), \tag{7}$$

$$x_{id} = x_{id} \oplus v_{id}. \tag{8}$$

4 PSO 求解任务可拆分的项目调度问题

4.1 粒子表示

采用基于任务序列的粒子表示方法,每个粒子对应一个先序可行的任务序列,任务在序列中的位置表示任务调度的优先顺序,粒子的维数对应项目的任务数,即 $D = J$,粒子的适应值对应其生成调度

的工期.如任务序列 x_i 表示粒子(x_{i1},x_{i2},…,x_{iJ}),其中第 j 个分量对应于任务 x_{ij},任务 x_{ij} 在粒子转化为调度方案时,其调度次序为 j.图 1 是 Patterson 测试集中一个问题实例的网络结构图,该项目的一个可行粒子为

$$x_1 = (0,1,3,4,2,9,5,7,10,8,6,11,12).$$

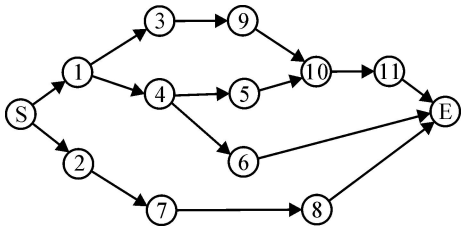

图 1　项目调度问题的例子

4.2　粒子更新

为了保持现有粒子的分段整体性,本文将遗传算法中的定位交叉[12]引入粒子的更新中.粒子通过与全局最优和个体极值进行定位交叉予以更新.令学习系数 $c_1 = c_2 = J$.从任务序列中随机选择位置点 $\mathrm{pos}_1(=c_1 r_1)$ 和 $\mathrm{pos}_2(=c_2 r_2)$ 对粒子进行更新.其中:pos_1 表示与个体极值进行定位交叉时的位置点,pos_2 表示与全局最优进行定位交叉时的位置点.粒子 x_i 的更新过程分为以下两步:

首先,x_i 在 pos_1 位置与个体极值 p_i 进行更新,具体方法如下:

1) 根据 r_1,在粒子 x_i 确定一个位置点 pos_1;x_i 中的前 pos_1 个任务组成的分段称为 S_1,其余部分称为 S_2;

2) 将 p_i 中与 S_2 相同的分量保持原有顺序提取出来,形成新的分段 S_3,其余的分量形成分段 S_4;

3) 将 S_1 中的任务和 S_3 中的任务合并为一个新的粒子 y_1,将 S_2 中的任务和 S_4 中的任务合并为一个新的粒子 y_2;

4) 计算 y_1 和 y_2 的适应值,令 y 为其中适应值小的粒子.

然后,采用类似的更新过程,对粒子 y 与全局最优 p_g 在位置 pos_2 进行定位交叉,得到两个新粒子 z_1 和 z_2.计算它们的适应值,取适应值小的粒子作为 x_i 更新后的新位置.

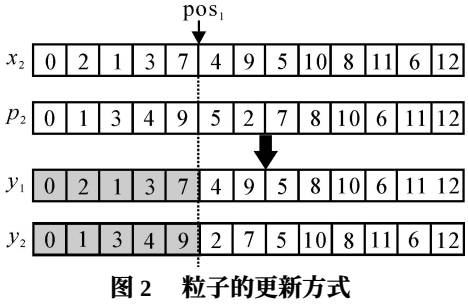

图 2　粒子的更新方式

图 2 给出了一个粒子与其个体极值进行定位交叉的例子.

4.3　局部搜索技术

在进化过程中,为了避免种群陷入局部极值,本文引入一种基于任务序列的局部搜索技术对更新后的粒子 x_i 进行改进.具体步骤如下:

Step1:随机生成两个位置点 p_1,$p_2 \in [1,J]$,设这两个位置点的任务分别为 x_{ip_1},x_{ip_2}.

Step2:如果交换任务 x_{ip_1} 和 x_{ip_2} 的位置而不破坏先序关系,则转 Step3;否则转 Step1.

Step3:交换任务 x_{ip_1} 和 x_{ip_2} 的位置,得到 x_i 的新位置,结束.

图 3 为一个对粒子进行局部搜索的例子.

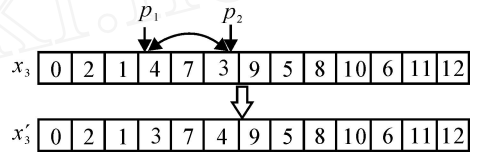

图 3　粒子的局部搜索

4.4　LSPSO 算法框架

LSPSO 算法如下:

Step1:初始化.初始 POP 个粒子和最大迭代次数 T_S.

Step2:对整个种群进行评价,确定每个粒子的个体极值 p_i 和整个种群的全局最优 p_g.

Step3:对种群中的每个粒子进行更新:定位交叉更新和局部搜索更新.

Step4:对整个种群进行评价,确定粒子的个体极值 p_i 和整个种群中的全局最优 p_g.

Step5:判断是否满足最大迭代次数,如果满足,则转 Step6;否则转 Step3.

Step6:结束.

4.5　调度方案的生成

在 LSPSO 算法中,一个粒子代表一个可行解.本文采用串行调度生成方案[13]将粒子的可行解转化为可行的调度方案.在调度方案的生成过程中,每次处理一个任务,处理过程中任务可进行多次拆分,每个拆分单元在满足先序约束和资源约束的前提下,开始时间要尽量早.为了描述生成算法,定义如下符号:S 为已处理完的任务集合;E 为先序任务已处理完,等待处理的任务集合;A_t 为在时刻 t 正在处理的任务集合;$e_j^s(h)$ 为任务 j 第 h 个拆分单元的开始时间;$d_j(h)$ 为任务 j 第 h 个拆分单元的工作时间.

具体算法描述如下:

Step1:初始化.$S = \Phi, E = \Phi, i = 0$.

Step2:更新 $E, E = \{j \mid j \notin S, P_j \in S, j = 1,$

2, …, J}.

Step3: 根据粒子中任务序列的顺序,选择第 i 个被调度的任务 j,令 $h = 0$.

Step4: 根据 S 中已调度任务顺序的完成时间和资源使用情况,确定任务 j 第 h 个拆分的最早开始时间 $e_j^S(h)$ 及相应的工期 $d_j(h)$.

Step5: 判断 d_j 是否等于 $\sum_{m=1}^{h} d_j(m)$,如果相等,则转 Step6;否则,转 Step7.

Step6: 计算任务 j 的开始时间 e_j^S 和完成时间 e_j^E,更新 $S \leftarrow S \cup \{j\}$,判断 $|S|$ 是否等于 J,如果相等,则转 Step8;否则,令 $i = i + 1$,转 Step2.

Step7: $h = h + 1$,转 Step4.

Step8: 结束.

5 算 例

为了测试算法的有效性,本文对 Patterson 测试集的 110 个问题实例[14] 进行了测试,其任务数为 7 ~ 51. 图 4 是图 1 所示问题实例带有工期和资源的项目网络结构图. 图中节点上面的数字表示该任务的工期,下面的 3 个数字分别表示该任务对 3 种资源的消耗量,所有资源均为可再生资源,在项目整个工期内总量为 6.

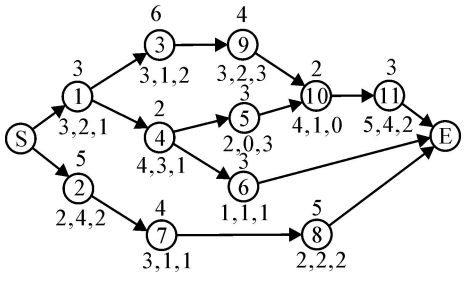

图 4 项目调度问题的例子

采用与文献[5]相同的运行环境对图 4 的问题实例进行测试:Window XP, Pentium IV 2.6G CPU/256 SDRAM 的 PC 机,用 C ++ 实现算法,POP = 30, T_S = 50,运行次数 50,问题实例的运行时间与文献[5]的运行时间的对比如表 1 所示.

表 1 求解算法运行时间的比较

算 法	平均计算时间/s	工期/d
混合遗传算法[5]	2.251	20
粒子群算法	0.509	20

与图 4 对应的最优调度序列为(1,2,3,7,4,5,9,8,10,6,11),最优工期为 20 d,其甘特图如图 5 所示.

对于 Patterson 测试集 110 个问题实例,本文算法所得到的项目调度方案与不可拆分的项目调度方案相比,项目工期可缩短 0 ~ 6 d,平均缩短 2.03 d.

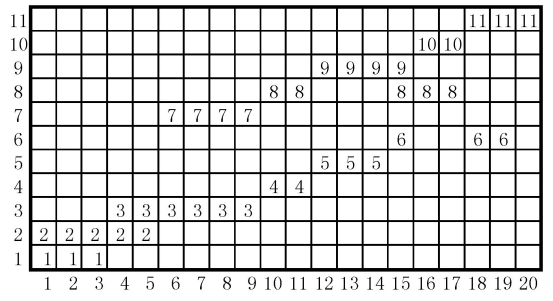

图 5 最佳粒子对应的甘特图

由表 2 可以看出,算法 LSPSO 可进一步缩短项目工期,从而合理调配项目资源,降低项目费用,提高项目的执行效率.

表 2 对 Patterson 测试集计算结果比较

算 法	平均缩短工期/d
粒子群算法	2.03
混合遗传算法[5]	1.86

6 结 论

针对以项目工期最小为目标的任务可拆分的项目调度问题,本文提出了一种粒子群算法 LSPSO. 该算法采用基于任务序列的粒子表示方式. 对 Patterson 问题集中 110 个问题实例进行了测试,结果表明本文算法是可行而有效的.

参考文献(References)

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

[2] Blazewicz J K, Lenstra A H G, Rinnooy Kan. Scheduling subject to resource constraints: Classification and complexity [J]. Discrete Applied Mathematics, 1983: 5(1): 11-24.

[3] Buddhakulsomsiri Jirachi, David S K. Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting [J]. European J of Operational Research, 2006, 175(1): 279-295.

[4] Buddhakulsomsiri Jirachi, David S K. Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resources vacations and activity splitting [J]. European J of Operational Research, 2007, 178(2): 374-390.

[5] 雒兴刚, 汪定伟, 唐加福. 任务可拆分项目调度问题[J]. 东北大学学报, 2006, 27(9): 961-964.
(Luo X G, Wang D W, Tang J F. Project scheduling problem involving time-splittable tasks [J]. J of Northeastern University, 2006, 27(9): 961-964.)

(下转第 688 页)

间的区域即为粒度控制的期望区间.可以看出,人工设定下的磨矿粒度达不到期望的要求,波动范围较大且普遍偏粗,而 MFSC 下的粒度指标基本在目标区间内波动,虽然某些时刻,粒度超出了控制目标区间,但经模糊监督器的调整,粒度又逐渐进入期望区间内,且运行趁于稳定,较好地实现了粒度指标的控制.运行数据还表明,磨机台时处理量可提高约 2.5 t/h,而合格的磨矿产品质量和高的磨矿产量意味着最有效的能耗利用[2].

5 结 论

本文提出的磨矿过程多变量模糊监督控制方法,可根据磨矿工况的变化,在线自动修改回路控制器的设定值.随着回路控制系统的输出跟踪调整后的设定值,将磨矿粒度控制在期望目标范围内,并有效地提高了磨机处理量.工业试验表明,所提出的方法是有效的,可推广应用于具有类似特性的工业过程控制.

参考文献(References)

[1] Pomerleau A, Hodouin D, Desbiens A, et al. A survey of grinding circuit control methods: From decentralized PID controllers to multivariable predictive controller[J]. Powder Technology, 2000, 108(2/3): 103-115.

[2] Ramasamv M, Narayanan S S, Rao C D P. Control of ball mill grinding circuit using model predictive control scheme[J]. J of Process Control, 2005, 15(3): 273-283.

[3] Najim K, Hodouin D, Desbiens A. Adaptive control: State of the art and application to a grinding process[J]. Powder Technology, 1995, 82(1): 59-68.

[4] Stange W. Using artificial neural networks for the control of grinding circuits[J]. Minerals Engineering, 1993, 6(5): 479-489.

[5] Li H X, Guan S. Hybrid intelligent control strategy: Supervising a DCS-controlled batch process[J]. IEEE Control Systems Magazine, 2001, 21(3): 36-48.

[6] 张钊, 吴爱国, 裴燕玲. 模糊控制的模糊推理分析[J]. 控制与决策, 2005, 20(8): 905-908.
(Zhang Z, Wu A G, Pei Y L. The fuzzy reasoning analysis of fuzzy control[J]. Control and Decision, 2005, 20(8): 905-908.)

[7] 王永富, 柴天佑. 自适应模糊控制理论的研究综述[J]. 控制工程, 2006, 13(3): 193-198.
(Wang Y F, Chai T Y. Present status and future developments of adaptive fuzzy control[J]. Control Engineering of China, 2006, 13(3): 193-198.)

[8] 周平, 柴天佑. 基于案例推理的磨矿分级系统智能设定控制[J]. 东北大学学报, 2007, 28(5): 613-616.
(Zhou P, Chai T Y. Intelligent control setting with CBR for ore grinding classification system[J]. J of Northeastern University, 2007, 28(5): 613-616.)

[9] Del V R G, Thibault J, Del V R. Development of a soft-sensor for particle size monitoring[J]. Minerals Engineering, 1996, 9(1): 55-72.

[10] 周平, 柴天佑. 基于案例推理的磨矿粒度软测量方法及其软件实现[J]. 系统仿真学报, 2007, 19(23): 5397-5400.
(Zhou P, Chai T Y. A case-based reasoning soft-sensor for particle size of grinding process and its software development[J]. J of System Simulation, 2007, 19(23): 5397-5400.)

(上接第 684 页)

[6] Eberhart R, Kenndy J. A new optimizier using particle swarm theory[C]. Proc of the 6th Int Symposium on Micro Machine and Human Science. New York: IEEE, 1995: 39-43.

[7] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. New York: IEEE, 1995: 1942-1948.

[8] Shi Y, Eberhart R. Parameter selection in particle swarm optimization[C]. Proc of the 7th Annual Conf on Evolutionary Programming. New York, 1998: 591-600.

[9] Zhang Hong, Li Heng, Tam C M. Particle swarm optimization for resource constrained project scheduling[J]. Int J of Project Management, 2006, 24(1): 83-92.

[10] Zhang Hong, Li Xiao-dong, Li Heng. Particle swarm optimization-based schemes for resource-constrained project scheduling[J]. Automation in Construction, 2005, 14(3): 393-404.

[11] Maurice Clerc. Discrete particle swarm optimization illustrated by the traveling salesman problem [DB/OL]. 2000. http://www.mauriceclerc.net.

[12] Syserda G. Handbook of genetic algorithms[M]. New York: Van Nostrand Reinhold, 1991.

[13] Kolisch R. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation[J]. European J of Operation Research, 1996, 90(2): 320-333.

[14] Patterson J H. Comparison of exact approaches for solving the multiple constrained resource project scheduling problem[J]. Management Science, 1984, 30(7): 854-867.