文章编号: 1001-0920(2002)03-0297-04

基于约束逻辑规划和遗传算法的柔性调度

李 岩, 吴智铭
(上海交通大学 自动化研究所, 上海 200030)

摘 要: 根据柔性生产环境的特点, 描述了约束逻辑规划(CLP)和遗传算法(GA)在解决调度问题中的应用框架。CLP 的解作为满足约束的调度问题的起始解, 保证了初始解的合理性。把 CLP 用作计算每一代样本的约束检验手段, 有利于在遗传算法的搜索中获得更好的解和更高的解算效率。最后对一个规模足够大的调度实例进行了计算。

关键词: 遗传算法; 约束逻辑规划; 柔性调度; 启发式规则
中图分类号: TP 18; TP 311 文献标识码: A

Flexible scheduling based on constraint logic programming and genetic algorithms

LI Yan, WU Zhi-ming
(Institute of Automation, Shanghai Jiaotong University, Shanghai 200030, China)

Abstract: The application frame of obtaining a scheduling solution is described by using two methods: CLP and genetic algorithms (GA), with consideration of the characteristics of flexible routes. It generates a set of initial solutions from the results of CLP and uses GA to search in the pruned solution space. At the same time, CLP is adopted to check the feasibility of the solutions in each generation. Thus, it assures a set of reasonable starting points for the searching procedure and provides more chance to get better solutions. A large-scale scheduling example is solved by the methods.

Key words: genetic algorithm; constraint logic programming; flexible scheduling; heuristic rule

1 引 言

由于调度问题的复杂性随着工件和机器数量的增加其计算量按指数规律增长, 经典数学规划方法(如动态规划法和分枝定界法)很难计算规模较大的 NP-完全问题。而启发式规则实现简单, 计算量小, 但都是针对确定问题的求解, 效果不能保证。本文提出一种解决离散组合优化中生产调度问题的新型混合策略。通过在遗传算法(GA)中内嵌约束逻辑规划(核心是逻辑规划(LP)和一致性技术), 在保持解算法效率的同时, 可缩短寻优时间。本文还选用先验的启发式调度规则来帮助获得高效优化的解。

通常的产生式逻辑语言被动地使用约束条件, 通过事后错误检测的方法减小搜索空间, 不能提高调度问题的求解效率[1~5]。LP[6,7]是一种表达离散组合优化问题的语言, 其优点是明晰的语义及以关系形式表达问题。LP 解决问题的不确定性方式[8], 使

收稿日期: 2000-11-20; 修回日期: 2001-06-18
基金项目: 国家自然科学基金项目(59889505; 70071017)
作者简介: 李岩(1972—), 男, 河南郑州人, 博士生, 从事智能软计算等研究; 吴智铭(1936—), 男, 上海人, 教授, 博士生导师, 从事离散事件系统、智能控制与智能软计算等研究。

用户从树搜索程序中解放出来, 有可能通过灵活的解决策略得到一组解。另外, 由人工智能发展而来的一致性技术[5]主动使用问题的约束, 可通过剔除非法解来裁剪搜索空间。这种一致性技术的搜索过程可看作是以下两步的迭代: 1) 尽可能地传播约束; 2) 对变量取值做出假设, 直至问题的解决[6]。但这种解决方案迄今并未用于解决工程应用问题。LP(充分表述问题) 和一致性技术(高效解决问题) 的有效结合, 可产生一种离散组合优化的方法—— 约束逻辑规划(CLP) 。

本文以 CLP 中领域变量设计一种高阶逻辑语言, 从一个调度实例出发阐明了 "选择" 法和高阶谓词化问题, 并把 CLP 应用到 GA 初始解和每一代样本的产生过程中, 实现解的合理性检查程序化, 强化 GA 的搜索速度和有效性, 以获得更高的效率。

2 约束逻辑规划

2.1 一致性技术

一致性技术通过对变量值域区间的裁剪使变量得到实例化。嵌入这种概念的逻辑规划 LP 可以看作是一种推理规则, 从而形成 CLP 的框架。

通过设计执行基于以上观点的逻辑规划程序[6], 包括: 1) 对问题求解空间的预裁剪; 2) 对变量取值进行选择; 3) 对用于优化的谓词的高阶扩充。在谓词的参数之间成立的逻辑规划的关系形式, 使其适合表达约束、逻辑中的规划和适应规划风格的约束组成的定义关系。基于领域变量的逻辑规划语言的推理规则包括 Forward 检查推理规则(FCIR) 和 Look Ahead 推理规则(LAIR) , 这些方法体现了对搜索空间的裁剪[1]。

2.2 约束的产生

提交给声明(如 Look Ahead 和 Forward) 或用作选择的逻辑程序是一种基本的约束。解决一个约束在值水平上的推理是不合适的, 最好将其看作一个可以多种互斥的方式得到满足的选择点。这样的逻辑程序有很少的变量, 很少使用递归方法。Look Ahead 检查在大量约束的交互作用场合十分有效, 而 Forward 检查在充分传播其选择之前, 有足够长的等待时间, 可检查出导致巨大惩罚的候选方案。

以一个析取约束作为例子来说明基本约束: disjunctive(S_i, d_i, S_j, d_j) 表示任务 i, j 不能同时执行, 这是一个极易违反的强约束。通过选择策略给出一个任务的开始时间(值约束方法) , 对约束有最小违反。选择了哪一个任务, 便应安排在另一个任务的前面, 但是并未选择一个具体的值。

2.3 优化问题解决的程序结构

```
opt_pb(L, cost) ←
    lower_bound(lower),
    good_solution(upper),
    defining_domains(L, F),
    generating_constraints(L),
    minimize(making_choice(L), F, lower,
upper),
    cost is F.
```

其中, L 表示问题的解, cost 是其成本, 谓词 lower_bound(lower) 用来定义对最优值的估计, 谓词 good_solution(upper) 产生使用某具体启发式解的成本上限。谓词 defining_domains 定义了领域变量表 L 和优化函数 F, 谓词 generating_constraints 产生了作为 "选择" 的约束和变量。

3 CLP 调度的基本原则和实现

柔性 Job-shop 调度(JSS) 提高了生产效率, 但同时增加了调度所考虑的约束数目, 不仅要决定零件加工的次序, 而且还要从多种可能的工艺路线和多台相同的机器间选择合适的路径。在 CLP 方法解决实际的调度问题中, 把每个工艺路径作为零件加工的候选解, 在逻辑程序中作为对变量值域的 "选择" 来处理。

在选择序列约束时, 应首先考虑那些最有可能不成立的约束, 以裁剪搜索空间。处理问题的方法应遵循以下顺序: 1) 基于 FCIR 的内建约束; 2) 基于 LAIR 的内建约束; 3) 提交给 forward 声明的约束; 4) 提交给非 forward 声明的约束; 5) 作为选择的约束。之所以首先应用内建约束是因为它们能以较小的成本对解空间做出较大的裁剪, 最后应用作为选择的约束可以尽可能保留解空间的自由度, 不引入过多对解空间的裁剪。最先应用 FCIR 是因为其计算成本最小, 并且尽可能实现变量的实例化, 从而彻底解决问题。

下面阐述调度中的几种主要约束及其处理方法。

1) 序列约束。

采用表 1 的符号表示, 设有 N 个零件, 零件 i 的第 j 道工序记为(i, j) 。为方便起见, 增加一个持续时间为 0 的虚拟任务(end-task, 结束任务) , 该任务在

表 1　符号定义

符号	定　义	符号	定　义
S_{ij}	(i,j) 的开始时间	S_{end}	加工结束时刻
M	资源总量	T_{ij}	(i,j) 的最晚开始时间
p_{ij}	(i,j) 的加工时间	t_{ij}	(i,j) 的最早开始时间
$I_{i_1j_1-i_2,j_2}$	(i_1,j_1) 和(i_2,j_2) 的时间间隔(一般为 SETUP 时间)	d_{ij}	(i,j) 的松驰时间

时间上不超前于任何任务。

对于同一零件, 由于没有任务可在时间上超前于它的前序任务约束之前开始, 所以任务(i_1,j_1) 的最早开始时间为$S_{i_1j_1} = \max(S_{i_2j_2} + p_{i_2j_2})$, 对所有超前于任务$(i_1,j_1)$ 的任务(i_2,j_2)。项目的最小持续时间为 t_{end}, 任务(i_1,j_1) 的最晚开始时间为 $T_{i_1j_1} = \min(T_{i_2j_2} - p_{i_2j_2})$。对同一零件所有超前任务$(i_1,j_1)$ 的任务(i_2,j_2): 假定有 $T_{\text{end}} = t_{\text{end}}$ 且松驰为 0 的任务称为关键任务。

2) 距离约束。

相对距离约束 "任务(i_1,j_1) 必须要在任务(i_2,j_2) 结束之后的 10 个时间单位之后开始"可表示为 $S_{i_1j_1} \geq S_{i_2j_2} + p_{i_2j_2} + 10$。绝对距离约束 "$(i,j)$ 必须在 T 时刻之后开始"可表示为 $S_{ij} \geq T$。

3) 析取约束。

调度问题中的复杂性主要来源于析取约束, 一个析取约束只简单地说明了任务(i_1,j_1) 和(i_2,j_2) 不能同时得到执行, 这是由于(i_1,j_1) 和(i_2,j_2) 使用相同的资源(机器) 之故。

4) 资源约束。

保证资源的使用总量不超过资源的保有量。$\sum_{i,j,t} q_{ij}Rt \leq Q_R$。另外, 在某个特定时刻一个任务只能由一个资源处理 $\sum_{i,j,k} A_{ijkt} = 1$。

CLP 提供了表达和解决这些约束的工具, 最典型的实现手段是 CHIP[9]。CHIP 提供了使用户可以自己定义约束, 并控制其执行的方法。在 CHIP 中, 所有约束都可由谓词表达出来。

4　混合算法

定义目标函数为各零件的交货期延迟的加权平方和, 利用 GA 和 CLP 的寻优过程如下:

1) 初始化参数: 读入调度任务数据、确定群体规模、交叉概率 P_c 以及变异概率 P_m。

2) 以一定的启发式规则和 CLP 程序产生初始群体, 其相应的解必然是合理的。

3) 交叉与变异。

4) 依据以上 4 个约束谓词的定义, 对进行交叉变异后的解群体进行合理性检查。

5) 根据编码确定零件工艺路径、各工序的开始时刻及结束时刻, 评估适应度, 选择并形成新群体。

① 确定各工序的开始时间: 如果本工序为当前机器的头道工序且为当前工件的头道工序, 则 $b_{ij} = I_{ij-il} + y_{ij}, l$ 为空工序; 否则, $b_{ij} = \max\{y_{ij}$, 工件上道工序的结束时间, 机器上道工序的结束时间$\} + I_{ij-il}$;

② 确定各工序的结束时间 $e_{ij} = b_{ij} + t_{ij}$。

6) 是否达到指定的迭代次数?未达到则返 3)。

启发式规则由于实现简单、计算量小而得到广泛的应用。本文新解产生中利用了启发式规则库, 每次迭代中随机从库中调出一条启发式规则, 对选定的解进行处理, 同时将启发式规则作为一种变异方式应用于算法。启发式规则的数目相当多, Panwalker[2] 列举了 113 条启发式调度规则, 本文选用了 28 条构成规则库。实验表明, 启发式规则的合理使用将有助于解的优化。

每台机器的染色体由加工队列的工件号和工序号组成, 用 CLP 检查产生个体编码的可行性。如在交叉中, 两个个体 p, q(具有适应度f_q, f_q) 分别对应某工序的一条路径, 分别以 $f_p/(f_p + f_q), f_q/(f_p + f_q)$ 的概率来决定采用哪一个个体的路径方案。处理机器选择的交叉操作同上述路径选择极为类似。变异分为: 1) 同类型机器间处理工序之间的随机交换; 2) 同一台机器内处理工序之间的随机互换; 3) 应用启发式规则的变异。

5　计算实例

例 1　下面以一个具体的例子来说明用 CLP 方法解决实际调度问题时, 序列约束的产生与处理过程。表 2 列出了一个调度问题的原始数据, 其中零件 1 拥有两条不同的路径, 领域变量 $S_{11}, \cdots, S_{23}$, 表示项目任务的起始时刻, S_{end} 是结束任务的时刻。

不同的路径用作不同方案的选择。采用总加工

表 2 调度实例原始数据

任务名	工序名	处理机器	处理时间
S_{11}	(1, 1)	1/3	3/5
S_{12}	(1, 2)	2/1	4/6
S_{13}	(1, 3)	3/2	5/1
S_{21}	(2, 1)	2	2
S_{22}	(2, 2)	3	3
S_{23}	(2, 3)	1	4

时间作为完成时间的上限, 程序可分以下 3 步:

1) 约束的前向传播。对一个任务(i_1, j_1), $S_{i_1j_1} \geq \max(S_{i_2j_2} + p_{i_2j_2})$, 对所有超前任务$(i_1, j_1)$的任务$(i_2, j_2)$, 即对每个任务计算其最早开始时刻。给定序列约束 $S_{i_1j_1} \geq S_{i_2j_2} + p_{i_2j_2}$, 可立即推出 $S_{i_1j_1} \geq \min(S_{i_2j_2}) + p_{i_2j_2}$, 其中 $\min(S_{i_2j_2})$ 是 $S_{i_2j_2}$ 中的最小值。以路径 1 为例, 有 $S_{12} \geq 3$(约束 1); $S_{13} \geq 7$(约束 2); $S_{22} \geq 2$(约束 3); $S_{23} \geq 5$(约束 4)。对其它约束进行相似的推理, 可得出各任务的时间变化区间。

2) 结束任务的分配。用 minval 谓词取 S_{end} 的最小值(本例为 12)。

3) 约束的后向传播。后向传播将保证每个对应于任务 i 的变量不能得到一个大于 T_i 的值, 这是通过重新考虑不等式关系来实现的。由于 S_{end} 的实例化, 约束式(3) 和(4) 向后传播这种推理过程, 得到 $S_{11} = 0, S_{12} = 3, S_{13} = 7, S_{21} \in [0, 3], S_{22} \in [2, 5], S_{23} \in [5, 7]$。另一个候选解的传播过程参见表 3。

在 3 个推理过程之后, 可以肯定任务 i 不可能在小于 t_i 大于 T_i 的时刻开始, 而且关键任务被仅存的值所实例化。同时应注意到问题并未得到彻底的解决(还没有找到一个调度方案, 有些约束也没有完全解决)。如 $S_{22} \geq S_{21} + 2, S_{23} \geq S_{22} + 3$, 可以确定存在某种值的分配方案, 使得约束式(3) 和(4) 未得到满足, 但只要每个领域变量取其最小可能的值, 就可以满足这些约束。

表 3 序列约束的传播

	候选解 1	候选解 2	
逻辑	Domain part(0, 21)		
程序	Part($[S_{11}, S_{12}, S_{13}, S_{21}, S_{22}, S_{23}, S_{end}]$) ←		
	$S_{12} \geq S_{11} + 3$(1)	$S_{12} \geq S_{11} + 5$	(1)
	$S_{13} \geq S_{12} + 4$(2)	$S_{13} \geq S_{12} + 6$	(2)
	$S_{22} \geq S_{21} + 2$(3)	$S_{22} \geq S_{21} + 2$	(3)
	$S_{23} \geq S_{22} + 3$(4)	$S_{23} \geq S_{22} + 3$	(4)
	$S_{end} \geq S_{13} + 5$(5)	$S_{end} \geq S_{13} + 5$	(5)
结果	$S_{11} \in [0, 9]$	$S_{11} \in [0, 9]$	
	$S_{12} \in [3, 12]$	$S_{12} \in [5, 14]$	
	$S_{13} \in [7, 16]$	$S_{13} \in [11, 20]$	
	$S_{21} \in [0, 12]$	$S_{21} \in [0, 12]$	
	$S_{22} \in [2, 14]$	$S_{22} \in [2, 14]$	
	$S_{23} \in [5, 17]$	$S_{23} \in [5, 17]$	
	$S_{end} \in [12, 21]$	$S_{end} \in [12, 21]$	

例 **2** Pratt & Whitey Development Operation Shop 的生产任务是一个 33 台机器(包含 14 种机器类型)、127 个工件的问题, 采用加权交货期延迟的平方和作为目标函数, 利用 Lagrangian-Relaxation 法的调度结果最优 J 为 252 730[4], 而本文算法遗传 500 代的调度结果 J 为 240 949。以每代 30 个样本, 交叉 85%, 变异 1.5% 强, 在 Celeron300 上的平均 CPU 时间为 7min(2 000 次)。表 4 为 10 次程序执行的搜索结果的最优个体及其均值和方差, 由此表可看出调度算法的稳健性。图 1 为采用与未采用 CLP 初始化种群的搜索过程的对比, 可以看出, 采用了 CLP 的算法由于起点更低, 中间解的质量更高, 因而收敛速度更快。

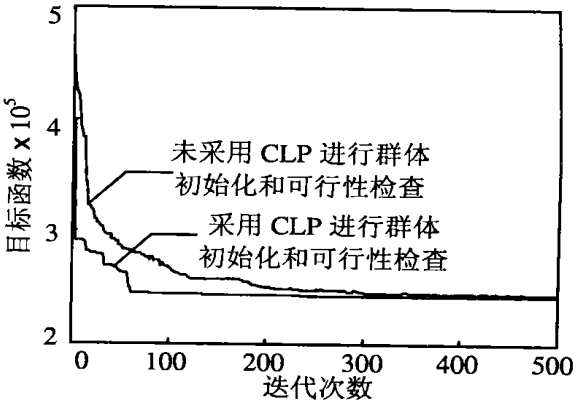

图 **1** 两类 **GA** 搜索过程的比较

6 结 语

离散组合优化问题的求解常运用几种技术的组合。本文以逻辑规划 LP 提高求解问题的效率, 通过应用中导出的 CLP 并结合一致性技术对遗传算法产生的解群体进行合理性检查, 并用规模足够大的例子说明了它在解决实际问题方面的有效性。

表 **4** 本文方法对[**4**]中问题的调度结果

执行次数	1	2	3	4	5	6	7	8	9	10	平均值	标准差
目标函数	240 045	239 796	239 702	239 759	240 024	240 324	239 645	239 786	240 012	239 810	239 890	195

(下转第 305 页)

$x_{2d}(t)$ 有界,$u_{2d}(t)$ 和 $u_{1d}(t)$ 趋于零。

取初值 $x_d(0) = -0.8415$ m, $y_d(0) = 0.4597$ m, $\theta_d(0) = -1.0$ rad, $x(0) = -1.1313$ m, $y(0) = -0.0165$ m, $\theta(0) = 0.8$ rad, 即 $x_{1e}(0) = 1.8$ rad, $x_{2e}(0) = 0.0415$ m, $x_{3e}(0) = -1.2597$ m。采用控制律(7)和(11),这里,$f(t) = e^{-0.5t}$, $f_d = -0.5$, $M_1 = -0.5$, $g(t) = \sin 0.5t$, $G = \frac{1}{4\pi}\int_0^{4\pi}\sin^2 0.5t\,dt = 1/2$。选取参数 $k = 2.5$, $k_3 = 2 > -M_1/G = 1$, $k_4 = 3 > -M_1$,仿真结果如图1～图3所示。由图3可以看出,系统轨迹最终跟踪参考轨迹收敛到原点,实现了逼近参考轨迹基础上的镇定。

5　结　　论

本文对基于运动学模型的非完整移动机器人的全局跟踪问题进行研究,对满足假设的一类参考模型可实现全局指数跟踪。尤其是对参考轨迹趋于静止的情况,对参考轨迹的成功跟踪相当于实现了在逼近给定轨迹基础上的镇定问题。仿真例子证明了该方法的有效性。目前,本文方法只考虑了低阶非完整系统的轨迹跟踪问题,高阶情况将是今后进一步研究的课题。

参考文献(References):

[1] Kolmanovsky I, McClamroch N H. Developments in nonholonomic control problem[J]. IEEE Control System Magzine, 1995, 15(6): 20-36.

[2] Tian Y P, Li S. Smooth exponential stabilization of nonholonomic systems via time-varying feedback[A]. Proc of IEEE Conf on Decision and Control[C]. Sydney: Casual Production Pty Ltd, 2000. 1912-1917.

[3] Kanayama Y, Kimura Y, Miyazaki F, et al. A stable tracking control method for an autonomous mobile robot[A]. Proc of IEEE Int Conf on Robotics and Automation[C]. Cincinnati: IEEE Computer Society Press, 1990. 384-389.

[4] D Andrea-Novel B, Campion G, Bastin G. Control of nonholonomic wheeled mobile robots by state feedback linearization[J]. Int J of Robotics Research, 1995, 14(6): 543-559.

[5] 董文杰, 霍伟. 受非完整约束移动机器人的跟踪控制[J]. 自动化学报(Acta Automat Sinica), 2000, 26(1): 1-6.

[6] 董文杰, 霍伟. 链式系统的轨迹跟踪控制[J]. 自动化学报(Acta Automat Sinica), 2000, 26(3): 310-316.

[7] Jiang Z P, Nijmeijer H. Tracking control of mobile robots: A case study in backstepping[J]. Automatica, 1997, 33(7): 1393-1399.

[8] Jiang Z P, Nijmeijer H. A recursive technique for tracking control of nonholonomic systems in chained form[J]. IEEE Trans on Autom Contr, 1999, 44(2): 265-279.

[9] 李世华, 田玉平. 移动小车的轨迹跟踪控制[J]. 控制与决策(Control and Decision), 2000, 15(5): 626-628.

(上接第 300 页)

参考文献(References):

[1] Van H P. Constraint satisfaction in logic programming [M]. Cambridge: MIT Press, 1989. 61-65.

[2] Panwalker S S, Iskander W. A survey of scheduling rules[J]. Operation Research, 1977, 25(1): 45-61.

[3] Mesghouni K, Pesin P, Hammadi S, et al. GA-constraint logic programming hybrid method for job shop scheduling[A]. Re-engineering for Sustainable Industrial Production[M]. Great Britain: Chapman and Hall, 1997. 151-160.

[4] Hoitomt D J, Luh P B, et al. A practical approach to job-shop scheduling problems[J]. IEEE Trans on Robotics and Automation, 1993, 9(1): 1-13.

[5] Dincbas M, Simonis H, Van Hentenryck P. Solving large combinatorial problems in logic programming[J]. J of Logic Programming, 1990, 8(1-2): 74-94.

[6] Kowalski A R, Keuhner D. The semantic of predicate logic as programming language[J]. Int J of the ACM, 1976, 22: 733-742.

[7] Colmerauer A. Equation and in-equation on finite and infinite trees[A]. Proc of the Int Conf on 5th Generation Computer Systems[C]. Tokyo, 1984. 85-99.

[8] Robsin A J. A machine-oriented logic based on the resolution principle[J]. Int J of the ACM, 1965, 12(1): 23-41.

[9] Fruhwrith T, Herold A, Kuchenhoff V, et al. Constraint logic programming —— An informal introduction[A]. Logic Programming in Action[M]. Berlin: Springer, 1992. 3-35.