文章编号: 1001-0920(2009)03-0423-06

产品开发项目的离散时间/成本/质量平衡问题研究

彭武良[1,2], 王成恩[2]
(1. 沈阳理工大学 经济与管理学院, 沈阳 110168; 2. 东北大学 信息科学与工程学院, 沈阳 110004)

摘 要: 为在项目网络计划技术中集成规划产品开发项目的时间、成本和质量,以多模式资源受限的项目调度问题研究为基础,定义了一个能支持产品开发项目优化控制的离散时间、成本和质量平衡问题模型. 根据产品开发项目的特点,通过人力资源价格计算产品开发项目成本,采用质量功能展开技术对项目质量进行量化计算. 最后给出一个求解该问题模型的分枝剪切算法,并通过一个项目实例对模型和算法进行验证.

关键词: 产品开发; 项目调度; 离散时间/成本/质量平衡; 质量功能展开
中图分类号: TP15 **文献标识码**: A

Discrete time/cost/quality trade-off problem for product development project

PENG Wu-liang[1,2], *WANG Cheng-en*[2]
(1. School of Economics and Management, Shenyang Ligong University, Shenyang 110168, China; 2. College of Information Science and Engineering, Northeastern University, Shenyang 110004, China. Correspondent: PENG Wu-liang, E-mail: peng-wuliang@163.com)

Abstract: To integratedly plan the durations, costs and qualities of project development project by using network planning techniques, a discrete time/cost/quality trade-off model used for optimal scheduling of product development projects is presented on the theoretical basis of multi-mode resource constrained project scheduling problem (MRCPSP). According to the characteristics of product development project, the project cost is calculated by the price of human resources, and the project quality is quantitatively calculated by applying the quality function deployment (QFD) method. A brand and cut algorithm is addressed to solve the model. Finally, the model and its algorithm are verified by a project case.

Key words: Product development; Project scheduling; Discrete time/cost/quality trade-off; Quality function deployment

1 引 言

综合考虑时间、成本和质量 3 个核心项目目标,对产品开发项目进行规划和控制具有重要意义. 这方面研究[1-3]需要建立时间、成本和质量之间的连续函数关系,但这些函数关系不具有通用性,而更符合产品开发过程实践的离散模型还很缺乏. 文献[4]将并行产品开发项目过程建模为资源受限的项目调度问题,但未考虑成本、质量等项目目标. 文献[5]假设项目工期和质量是不可更新资源的离散和非增函数,建立了离散时间/成本/质量平衡的整数规划模型,但该模型没有考虑项目使用可更新资源的情况,也未给出项目计划中各个活动质量的量化关系.

本文根据产品开发项目的特点,定义一个能支持产品开发项目优化控制的离散时间、成本和质量平衡问题模型(DTCQTP). 该模型在 MRCPSP 问题的基础上,给出了基于人力资源利用的项目成本计算方法,并给出一个基于 QFD 的项目质量量化方法. 针对模型的特点,设计出一种求解该问题精确解的分枝剪切算法,并通过实例对模型和算法进行验证.

2 问题模型

2.1 基于 MRCPSP 的过程规划

本文在对产品开发项目过程进行规划时,参考了多模式资源受限的项目调度问题模型[6]. 产品开

收稿日期: 2008-01-20; **修回日期**: 2008-04-14.
基金项目: 国家自然科学基金项目(60604025).
作者简介: 彭武良(1973—),男,内蒙古赤峰人,讲师,博士,从事项目管理、项目调度的研究; 王成恩(1964—),男,黑龙江鸡西人,教授,博士生导师,从事系统工程、制造业信息化等研究.

发过程以一个有向无环图描述,即

$$G = (V, E). \tag{1}$$

其中:V 是开发活动集合,$i \in V$ 为一个项目活动,活动 1 和活动 n 均为虚活动,工期为 0;E 为前置关系集合,$(i, j) \in E$,即 i 为 j 的紧前活动.

项目有可更新资源集合 K,对于每种资源 $k \in K$,在整个项目执行过程有可用资源量 R_k. 对于每个开发活动 i,有执行模式集合 M_i. 当活动按一个特定模式 $m \in M_i$ 执行时,对应的活动工期为 d_{jm},活动占用的资源集合为 K_i. 对于每种资源 $k \in K_i$,有资源消耗 r_{imk}.

活动只能任选一种模式执行,且在执行过程中不能中断. 活动 i 的开始时间为 s_i,结束时间为 f_i. 设项目开始时间为 0,则项目工期可用结束任务 f_n 表示,即

$$F_t = f_n. \tag{2}$$

2.2 基于人力资源使用的成本计算方法

产品开发项目是一项知识密集型活动,其所需资源主要是以人力资源为主的可更新资源,其成本支出也主要是基于可更新资源的使用而发生的. 现代企业多采用矩阵式组织结构. 在这种结构中,项目成员有两个上级:一是企业某职能部门的主管,二是所参加项目的项目经理. 对应的项目成本也由两部分组成:一种是企业按月 / 周支付给项目成员的薪金,另一种是由项目经理直接支配的项目支出. 本文定义前一种成本为静态成本,后一种成本为动态成本.

在产品开发项目中,每个设计任务可采用加班的方式执行,以缩短项目工期,但需要付出动态成本,作为加班人员的加班费用. 本文假设项目动态成本全部由加班费构成,同时假设所有不可更新资源在项目内独占,这不改变问题的本质.

项目的可更新资源种类集合 K 代表项目成员的种类,如高级工程师、工程师、技术员等. 对于任意资源种类 $k \in K$,其静态成本 $uc_k = f_n u_k$. 其中:u_k 为资源的单位时间静态成本,可通过人员的月薪进行折算;f_n 代表项目工期.

资源种类 k 有两种使用方式:正常方式和加班方式. 在正常方式下,资源只发生静态成本. 在加班方式下,通过付出额外的动态成本,以缩短活动工期. 动态成本 $ec_k = ed_i c_k$. 其中:ed_i 为加班后的活动工期;c_k 为活动的单位时间动态成本,即人员的单次加班费用.

每个活动 i 给定一个执行模式集 M_i,对应每种执行模式 $m \in M_i$. 在活动正常执行方式下,只发生静态成本. 于是项目成本可描述为

$$F_c = \sum_{i \in V} \sum_{m \in M_i} \left(x_{im} y_i \sum_{k \in K_{im}} (r_{imk} ed_i c_k) \right) + \sum_{k \in K} (R_k u_k f_n). \tag{3}$$

其中:x_{im} 和 y_i 为 0-1 变量,x_{im} 表示活动 i 是否选择模式 m,y_i 表示活动 i 是否选择加班执行方式;K_{im} 为活动 i 按模式 m 执行时,所需的可更新资源种类集合;r_{imk} 为 i 按模式 m 执行时,对可更新资源 k 的需求量;f_n 表示项目工期 F_t. 式(3)中第 1 项为项目动态成本,第 2 项为项目静态成本.

2.3 基于 QFD 的质量计算方法

QFD 是产品开发质量管理的一种重要方法[7],一个基本的 QFD 质量屋如图 1 所示. 其中:左墙为产品的客户需求,天花板为一系列工程措施. 通过 QFD 的层层分解,可明确工程措施与需求之间的对应关系,但 QFD 无法明确工程措施的执行次序、时间和资源,因此不适合对项目进行系统的管理.

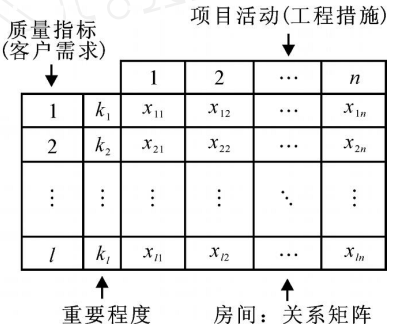

图 1 质量屋示例

本文将 QFD 的客户需求与项目质量相对应,将工程措施与项目网络中的项目活动相对应,使用 QFD 方法对项目活动的质量进行量化定义,并通过网络计划技术对项目时间、成本和质量平衡进行优化调度.

具体做法是将质量指标列在质量屋的左墙上,将所有项目活动列在质量屋的天花板上,在关系矩阵中确定项目活动与质量指标之间的对应关系,通过质量屋的关系矩阵计算每个项目活动的质量在整个项目质量中所占的比重.

设置每个活动的质量值为 1 ~ 10 之间的数值,其中 10 为质量最好,1 为质量最差. 所有项目质量都按统一的比例设置,这样处理可使项目负责人员在定义一个活动不同模式的质量值时,只需确定不同活动模式的质量值对比关系,而不必考虑与其他活动的对比.

如图 1 所示,左墙数字 1,2,…,l 表示客户需求,项目质量由 l 个客户需求组成;$k_i (i = 1, 2, \cdots, l)$ 表示第 i 个客户需求的重要程度;天花板上数字 1,2,…,n 表示 n 个项目活动. 在项目调度时,会为每个活

动选择一个执行模式,每个执行模式都有一个按相同比例设置的质量值.由于不同的活动质量在整个项目质量中所占的比重不同,不能通过所有活动质量的简单相加来获得项目的总质量值,而需要通过关系矩阵来计算.

在关系矩阵中,设 z_{ij} 表示第 i 个需求与第 j 个项目活动的关联度,于是每个项目活动的重要程度可通过下式计算:

$$h_i = \sum_{j=1}^{l} k_j z_{ij}. \tag{4}$$

如果第 j 项活动与多项需求密切相关,并且这些客户需求较重要(k_i 较大),则 h_j 取值较大,即该项活动比较重要.基于每个活动质量的重要程度和每个活动质量值,可计算出整个项目的质量值

$$F_q = \frac{\sum_{i=1}^{n}\left(\sum_{m\in M_i}\left(x_{im} q_{im} \sum_{j=1}^{l}(k_j z_{ij})\right)\right)}{\sum_{i=1}^{n}\left(10 x_{im} \sum_{j=1}^{l}(k_j z_{ij})\right)} \times 10. \tag{5}$$

其中:q_{jm} 是当活动 j 选择不同的执行模式 m 时的质量值,为 1 ～ 10 之间的实数;$\sum_{i=1}^{n}\left(10 x_{im} \sum_{j=1}^{l}(k_j z_{ij})\right)$ 是所有项目活动的质量值均为 10 时的项目质量值,式中将其作为分母,并在分子上乘以 10,是为了将项目质量值转化为 1 ～ 10 之间的实数,与项目活动的质量值保持相同的量化标准,便于评价.

可见,通过 QFD 方法,可以多项产品性能指标作为项目质量的判定依据,对活动质量和项目质量进行量化.

2.4　问题的数学模型

综上可建立产品开发项目的时间、成本和质量平衡的优化调度模型.该问题模型可划分为 3 个子问题:1) 工期最短问题:以资源、成本和质量为约束,求解工期最短的项目计划;2) 成本最低问题:以资源、时间和质量为约束,求解成本最低的项目计划;3) 质量最优问题:以资源、时间和成本为约束,求解质量最高的项目计划.

下面分别建立这 3 个问题的数学模型.时间最短问题的数学模型如下:

$$\min\ F_t; \tag{6}$$

$$\text{s.t.} \sum_{m\in M_i} x_{im} = 1,\ i \in V; \tag{7}$$

$$f_j - f_i \geq \sum_{m\in M_j}(x_{jm} d_j),\ (i,j) \in E,$$

$$d_j = \begin{cases} d_{jm}, & y_j = 0 \\ d'_{jm}, & y_j = 1 \end{cases}; \tag{8}$$

$$\sum_{i\in A(t)} \sum_{m\in M_j} x_{im} r_{imk} \leq R_k,$$

$$1 \leq t \leq f_n,\ k = 1,2,\cdots,K; \tag{9}$$

$$F_c \leq C_{\max}; \tag{10}$$

$$F_q \geq Q_{\min}. \tag{11}$$

式(6) 为最小化项目工期,F_t 按式(2) 计算;约束集(7) 要求每个活动只能按一种模式执行;式(8) 要求所有活动需要满足紧前关系约束;式(9) 要求项目计划满足可更新资源约束,$A(t)$ 为 t 时刻正在执行的活动集合;式(10) 要求项目成本必须低于成本上限 $C_{\max}$;式(11) 要求项目质量必须高于质量底线 $Q_{\min}$.

成本最低问题共享时间最短问题约束(7) ～ (9) 和(11),其数学模型为

$$\min\ F_c, \tag{12}$$

$$F_t \leq T_{\max}. \tag{13}$$

式(12) 是最小化项目成本,式(13) 是项目工期约束.

质量最优问题共享时间最短问题和成本最低问题约束(7) ～ (10) 和(13),其数学模型为

$$\max\ F_q. \tag{14}$$

式(14) 是最大化项目质量.

3　分枝剪切算法

分枝剪切法是组合优化问题的一种常用算法,可用于求解项目调度问题的精确解[8].本文采用分枝剪切算法求解 DTCQTP 的精确解.算法的基本思路是:搜索一个满足紧前关系约束的活动优先级列表,将该表输入串行调度方案[9],能产生一个完整的项目计划.在活动优先级列表中,每个活动都需要选择合适的执行模式和执行方式.采用串行调度方案生成项目计划时,优先级列表中排序靠前的活动优先得到调度.算法在搜索过程中,经过的所有节点构成一个不完全活动列表 G_m,$|G_m| = m$.调度 G_m 中的活动可生成一个只包含 m 个任务的不完全项目计划.

定义 1(可行活动集合)　基于不完全优先级列表 G_m,可计算出可行活动集合 M,活动 $j \in M$ 的充分必要条件是 $j \notin G_m$,且 $P_j \subset G_m$,其中 P_j 为 j 的紧前活动集合.

算法依据可行活动集合进行分枝.活动优先级列表的形成是一个多阶段的决策过程,每个不完全优先级列表 G_m 都是搜索树上的一个节点,计算 G_m 的可活动集合 M,对应 M 中的每个活动的执行模式和执行方式都将作为一个新的下级节点.所有这些下级节点又可按同样方法继续形成新的下级节点.算法选择一个下级节点继续搜索,便形成新的不完全优先级列表,并继续重新计算可行活动集合,便形成新的下级节点.如此循环反复,可形成一个 n 层解

空间树(n 为项目活动数目).

从 M 中选取一个活动,并为其暂定执行模式,加入到 G_m 中,便形成一个新的不完全优先级列表 G_{m+1}. 对于一个不完全的优先级列表 G_m,令所有属于 G 但不属于 G_m 的活动集合为 $\overline{G}_m$.

定义 2(最短可能工期) 将所有 $j \in \overline{G}_m$ 的活动设置为工期最短的执行模式,并采用加班方式执行. 基于 G_m 生成不完全项目计划,在该不完全计划的基础上放松资源约束,按照 CPM 调度 $\overline{G}_m$ 中的活动,可生成一个临时完全计划 P'. P 的项目工期 T' 为 G_m 的最短可能工期.

定义 3(最小可能成本) 在定义 2 的 P' 中,若不考虑 $j \in \overline{G}_m$ 的活动加班所带来的动态成本,则该临时计划 P' 所对应的项目成本 C' 为 G_m 的最小可能成本.

定义 4(最大可能质量) 将所有 $j \in \overline{G}_m$ 的活动设置为质量最大的执行模式. 基于 G_m 生成不完全计划,在该不完全计划的基础上放松资源约束,按照 CPM 调度 $\overline{G}_m$ 中的活动,生成一个临时完全计划 P'', P'' 的项目质量 Q' 为 G_m 的最大可能质量.

定理 1 对于一个不完全优先级列表 G_m,若其最短可能工期 $T' > T_{\max}$,则在不考虑资源约束的情况下,基于 G_m 继续调度产生的任意一个完全优先级列表,所生成的项目工期 T'' 皆有 $T'' > T_{\max}$.

定理 2 对于一个不完全优先级列表 G_m,若其最小可能成本 $C' \geq C_{\max}$,则在不考虑动态成本的情况下,基于 G_m 继续调度产生的任意一个完全优先级列表,所生成的项目成本 C'' 皆有 $C'' \geq C_{\max}$.

定理 3 对于一个不完全优先级列表 G_m,若其最大可能质量 $Q' \leq Q_{\max}$,则基于 G_m 继续调度产生的任意一个完全优先级列表,所生成的项目质量 Q'' 皆有 $Q'' \leq Q_{\max}$.

上述 3 个定理都比较直观,通过反证法很容易证明. 限于篇幅,这里不再赘述.

在本文设计的分枝剪切算法中,采用深度优先的策略,从根节点出发搜索解空间树. 算法搜索至任一节点时,总是依据定理 1,定理 2 和定理 3 判断该节点是否肯定不包括问题的最优解,然后决定是继续搜索还是剪枝.

例如对于成本最低问题,在一个决策阶段 G_m,计算其最短可能工期、最小可能成本和最大可能质量,然后判断:若最短可能工期大于工期约束,则依据定理 1,搜索算法在 G_m 上继续搜索已无法产生可行解,故应进行剪枝;若最大可能质量低于质量约束,则依据定理 3,也不可能产生可行解,故应剪枝;若最低可能成本高于目前发现的最低成本,则依据定理 2,在 G_m 的基础上已不可能产生最优解,也应进行剪枝. 若不属于上述 3 种情况之一,则算法可继续搜索该节点的下级节点.

工期最短问题、成本最低问题和质量最优问题的求解原理是相同的,这里以成本最低问题为例对算法进行说明. 设问题的工期约束为 $T_{\max}$,质量约束为 $Q_{\max}$,当前已发现的一个最好解为 $C_{\max}$. 算法的执行步骤如下:

步骤 1: 初始化. 设 $C_{\max} = 999\,999\,999$ 为项目成本的初始下界,令搜索树的层级 $m = 1$. 将项目开始活动 1(虚活动) 加入不完全优先级列表,定义初始优先级列表 $G_m = \{1\}$ 作为搜索树的最顶层节点.

步骤 2: 基于不完全优先级列表 G_m,计算可行活动集合 M. 定义一个三元组 $z = (j, m_j, y_j)$. 其中: $j \in M$ 为活动标识, m_j 为活动的执行模式标识, y_j 为该活动是否按加班方式执行. 考虑执行模式和加班方式,对 M 中的每个活动可定义若干个 z,形成一个考虑不同执行模式和执行方式的新的活动集合 Z. $z \in Z$ 为 G_m 的一个分枝,每个节点 G_m 对应于一个分枝集合 Z.

步骤 3: 在 Z 中选择一个未被搜索的活动 z(可按活动标识、模式标识或是否加班的次序依次选择):

1) 若 $m \neq 1$ 且所有 $z \in Z$ 均标记为已搜索,说明当前可行活动集合已遍历完成,则将最后加入的 z 从 G_m 中删除,返回上一节点 G_{m-1},递归执行步骤 3;

2) 若 $m = 1$ 且所有 $z \in Z$ 均标记为已搜索,说明算法对整个搜索树搜索完毕,则执行步骤 5;

3) 否则,执行步骤 4.

步骤 4: 将步骤 3 选择的 z 加入到不完全优先级列表 G_m,形成一个新的节点 G_{m+1},并标记 z 被选择:

1) 若已到达搜索树的一个叶子节点(即最后加入的活动是结束活动),说明已形成一个完全活动列表,则计算项目总成本 F_q;若 $F_q < G_{\max}$,则执行 $G_{\max} = F_q$,形成新的下界.

2) 计算 G_m 的最短可能工期 T',最小可能成本 C' 和最大可能质量 Q';若 $T' > T_{\max}$ 或 $C' > C_{\max}$ 或 $Q' < Q_{\text{low}}$,则依据定理 1,定理 2 和定理 3,沿该节点继续搜索已无意义,需要执行剪枝操作. 剪枝的方法是将活动 z 从 G' 中移除,回溯到上一节点 G_{m-1},返回步骤 3;否则执行步骤 2.

步骤 5: 算法结束,输出 $C_{\max}$ 为最终的目标函数值.

4 实例验证

现以一个实际的产品设计项目为例,说明

表 1　球磨机进出料系统设计任务清单

活动	活动名称	前置任务	执行模式 1					执行模式 2				
			资源 1	资源 2	质量	正常工期	加班工期	资源 1	资源 2	质量	正常工期	加班工期
1	开始任务	—	0	0	0	0	0	—	—	—	—	—
2	进料系统设计	1	1	0	9	30	20	0	1	4	42	28
3	进料端盖设计	2	1	0	8	15	10	0	1	5	15	10
4	衬板设计	2	1	0	6	6	4	0	1	9	9	6
5	进料螺旋设计	2	1	0	8	21	14	0	1	5	24	16
6	出料系统设计	2	1	0	9	30	20	0	1	5	42	28
7	出料端盖设计	6	1	0	6	15	10	0	1	10	15	10
8	衬板设计	6	1	0	6	6	4	0	1	10	9	6
9	出料螺旋设计	6	1	0	9	21	14	0	1	6	24	16
10	给料装置	3,4,5	1	0	9	18	12	0	1	6	30	20
11	结束任务	7,8,9,10	0	0	0	0	0	—	—	—	—	—

DTCQTP 的使用过程. 沈阳重型机械厂为适应市场需求,需要对球磨机的进出料系统进行重新设计. 为此,成立一个由 4 名技术人员组成的项目组,其中高级工程师 1 名,工程师 3 名. 项目任务清单见表 1,项目资源的成本情况见表 2. 根据产品的性能指标和项目任务之间的对应关系,建立该项目的质量屋如图 2 所示(未包含虚活动).

表 2　项目资源成本情况

标识	资源类型	数量	单位时间静态成本	单位时间动态成本
1	高级工程师	1	40	100
2	工程师	3	20	50

质量指标	重要程度	活动 2	3	4	5	6	7	8	9	10
系统实用性	5	9	1	1	9	9	1	1	9	7
结构紧凑性	1	5	3	3		7	3	3		7
系统可靠性	4	1	3	3	9	3	3	3	9	5
生产经济性	3				3				3	
使用经济性	1	5	5		5	5	5		3	2
维修方便性	2	3	5	7	2	3	5	8	5	1
造型艺术性	1	9	7			9	9			9
		74	42	34	99	84	44	36	103	75

活动重要程度

图 2　球磨机进出料系统设计质量屋

将项目资源供应情况、任务清单、项目网络以及质量屋中的质量数据输入到 DTCQTP 模型,采用分枝剪切算法进行求解. 在时间、成本和质量 3 个目标中,以其中两个作为约束条件,求解另外一个最优值,可得出一个期望的项目计划. 称该项目计划所对应的项目质量、工期和成本为有效决策值,并以[质量,工期,成本]三元组表示.

假定所有项目活动均采用加班方式执行,并选择工期最短的执行模式,采用关键路径法可确定项目工期的下界为 61. 假定所有项目活动均采用正常的执行方式,并选择工期最长的执行模式,且所有项目活动均串行执行,可确定项目工期的上界为 162. 假定所有项目活动均选择质量最低的模式,可确定项目质量的下界为 5. 41. 假定所有项目活动均选择质量最高的模式,可确定项目质量的上界为 8. 39.

在项目工期上下界之间选择不同的工期,在项目质量上下界之间选择不同的质量值作为约束,求解在各种工期和质量约束下的成本最低问题. 计算结果如表 3 所示,其中数据以有效决策值[质量,工期,成本]的形式给出. 当工期约束大于 100 时,继续放松工期约束已无意义.

表 3 中每个有效决策值均对应一个满足紧前关系约束的活动列表. 基于该列表,通过串行调度方案可得到一个完整的项目计划. 几个主要的有效点及其对应的活动列表如表 4 所示. 列表的每个元素均采用[活动标识,所选模式,是否加班]三元组描述.

5　结　　论

本文提出了产品开发项目成本和质量的量化计算方法,建立了能平衡项目时间、成本和质量的离散问题模型. 该模型无需假定整个项目或具体项目活动的时间、成本和质量之间存在的某种函数关系,具有更广泛的适用性. 通过球磨机进出料系统设计的项目规划可以看出,DTCQTP 能平衡项目的时间、成本和质量 3 个核心项目目标,有效地支持项目决策. DTCQTP 假定活动之间不存在质量关联关系,这在产品模块化较好的项目中是成立的,但在更通用的情况下,活动之间的质量关联关系应予考虑,这也是本文今后进一步的研究方向. 另外在测试中发

表 3 不同工期和质量约束下的成本最低问题计算结果

质量约束	工期约束 61	62	64	68	72	76	80	84	88	92	96	100
5.42	[7.81,61,109000]	[7.81,62,106000]	[7.80,64,104000]	[8.01,65,99000]	[7.80,71,91000]	[8.01,75,89000]	[8.01,75,89000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]
5.60	[7.81,61,109000]	[7.81,62,106000]	[7.80,64,104000]	[8.01,65,99000]	[7.80,71,91000]	[8.01,75,89000]	[8.01,75,89000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]
5.93	[7.81,61,109000]	[7.81,62,106000]	[7.80,64,104000]	[8.01,65,99000]	[7.80,71,91000]	[8.01,75,89000]	[8.01,75,89000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]
6.27	[7.81,61,109000]	[7.81,62,106000]	[7.80,64,104000]	[8.01,65,99000]	[7.80,71,91000]	[8.01,75,89000]	[8.01,75,89000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]
6.61	[7.81,61,109000]	[7.81,62,106000]	[7.80,64,104000]	[8.01,65,99000]	[7.80,71,91000]	[8.01,75,89000]	[8.01,75,89000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]
6.95	[7.81,61,109000]	[7.81,62,106000]	[7.80,64,104000]	[8.01,65,99000]	[7.80,71,91000]	[8.01,75,89000]	[8.01,75,89000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]
7.29	[7.81,61,109000]	[7.81,62,106000]	[7.80,64,104000]	[8.01,65,99000]	[7.80,71,91000]	[8.01,75,89000]	[8.01,75,89000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[7.80,81,81000]
7.63	[7.81,61,109000]	[7.81,62,106000]	[7.80,64,104000]	[8.01,65,99000]	[7.80,71,91000]	[8.01,75,89000]	[8.01,75,89000]	[7.80,81,81000]	[8.01,81,81000]	[7.80,81,81000]	[7.80,81,81000]	[8.01,81,81000]
7.96	—	[8.09,62,106000]	[8.01,64,104000]	[8.01,65,99000]	[8.01,71,91000]	[8.01,75,89000]	[8.01,75,89000]	[8.01,81,81000]	[8.31,87,87000]	[8.01,81,81000]	[8.01,81,81000]	[8.31,87,87000]
8.30	—	—	—	—	[8.31,72,106000]	[8.31,75,99000]	[8.31,77,97000]	[8.31,84,94000]	[7.80,81,81000]	[8.31,87,87000]	[8.31,87,87000]	[7.80,81,81000]

表 4 有效决策值所对应的活动列表

有效决策值	活动列表
7.81,61,109000	[1,0,0] [2,1,1] [3,1,0] [4,1,0] [5,2,1] [6,1,1] [7,2,0] [10,1,0] [9,1,0] [8,1,0] [11,0,0]
7.81,62,106000	[1,0,0] [2,1,1] [3,1,0] [4,1,0] [5,2,0] [6,1,1] [7,2,0] [8,1,1] [9,1,0] [10,1,0] [11,0,0]
8.09,62,106000	[1,0,0] [2,1,1] [3,2,0] [4,1,0] [5,1,0] [6,1,1] [7,2,0] [8,1,1] [9,1,0] [10,1,0] [11,0,0]
7.80,64,104000	[1,0,0] [2,1,1] [3,1,0] [4,1,0] [5,2,0] [6,1,1] [7,2,0] [8,1,0] [9,1,0] [10,1,0] [11,0,0]
7.80,71,91000	[1,0,0] [2,1,1] [3,1,0] [4,1,0] [5,2,0] [6,1,0] [7,2,0] [10,1,0] [9,1,0] [8,1,0] [11,0,0]
7.80,81,81000	[1,0,0] [2,1,0] [3,1,0] [4,1,0] [5,2,0] [6,1,0] [7,2,0] [10,1,0] [9,1,0] [8,1,0] [11,0,0]
8.01,64,104000	[1,0,0] [2,1,1] [3,2,0] [4,1,0] [5,1,0] [6,1,1] [7,2,0] [8,1,0] [9,1,0] [10,1,0] [11,0,0]
8.01,65,99000	[1,0,0] [2,1,1] [3,2,0] [4,1,0] [5,1,0] [6,1,0] [7,2,0] [10,1,0] [9,1,1] [8,1,0] [11,0,0]
8.01,71,91000	[1,0,0] [2,1,1] [3,2,0] [4,1,0] [5,1,0] [6,1,0] [7,2,0] [10,1,0] [9,1,0] [8,1,0] [11,0,0]
8.01,75,89000	[1,0,0] [2,1,0] [3,2,0] [4,1,0] [5,1,0] [6,1,0] [7,2,0] [10,1,0] [9,1,1] [8,1,0] [11,0,0]
8.01,81,81000	[1,0,0] [2,1,0] [3,2,0] [4,1,0] [5,1,0] [6,1,0] [7,2,0] [10,1,0] [9,1,0] [8,1,0] [11,0,0]
8.31,72,106000	[1,0,0] [2,1,1] [3,1,1] [4,1,0] [6,1,0] [5,1,0] [7,2,0] [8,1,1] [9,1,0] [10,1,0] [11,0,0]
8.31,75,99000	[1,0,0] [2,1,1] [3,1,0] [4,1,0] [6,1,0] [5,1,0] [7,2,0] [8,1,1] [9,1,0] [10,1,0] [11,0,0]
8.31,77,97000	[1,0,0] [2,1,1] [3,1,0] [4,1,0] [6,1,0] [5,1,0] [7,2,0] [8,1,0] [9,1,0] [10,1,0] [11,0,0]
8.31,84,94000	[1,0,0] [2,1,0] [3,1,1] [4,1,0] [6,1,0] [5,1,0] [7,2,0] [8,1,0] [9,1,0] [10,1,0] [11,0,0]
8.31,87,87000	[1,0,0] [2,1,0] [3,1,0] [4,1,0] [6,1,0] [5,1,0] [7,2,0] [8,1,0] [9,1,0] [10,1,0] [11,0,0]

现,当项目的任务个数超过 16 时,采用本文的精确算法已经无能为力,需要通过智能优化算法求解问题的次优解. 本文提出的精确算法可用于验证智能优化算法的正确性和算法性能.

参考文献(References)

[1] Cohen M A, Jehoshua E, Ho T H. New product development: The performance and time-to-market trade-off[J]. Management Science, 1996, 42(2): 173-186.

[2] 郑绍濂, 翟丽. 新产品开发的最优战略均衡模型[J]. 管理科学学报, 1998, 1(3): 12-19.
(Zheng S L, Zhai L. The optimal strategy equilibrium model for new product development [J]. J of Management Sciences, 1998, 1(3): 12-19.)

(下转第 434 页)

ε应选很小的数值,α_1 和 α_2 应使矩阵 Λ_i 满足 Riccati 方程(27). 控制器参数的选择应满足 $k_i > 0$, $c_i > 0$, $\vartheta > 0$. 仿真中,观测器和控制器参数设置如下:

$$\varepsilon = 0.005,\ \alpha_{i1} = 10,\ \alpha_{i2} = 25,$$

$$k_i = 5,\ c_i = 5,\ \vartheta = 0.01.$$

两种构形的轨迹跟踪性能分别示于图 4 和图 5. 仿真结果表明,所提出的分散控制方案在不修改任何控制参数的情况下,可实现对不同机械臂构形的控制.

5 结 论

本文提出一种基于观测器的可重构机械臂分散自适应模糊控制方案. 该控制方案不需要机械臂的动力学模型,可应用于不同的机械臂构形. 控制器中自适应参数的更新基于 Lyapunov 稳定性理论设计,可保证整个系统的稳定性和轨迹跟踪性能. 通过两个不同构形的二自由度可重构机械臂的数值仿真,验证了所提出的分散控制方案的可行性.

参考文献(References)

[1] Paredis C J J, Brown H B, Khosla P K. A rapidly deployable manipulator system [J]. Robotics and Autonomous Systems, 1997, 21: 289-304.

[2] Melek W W, Goldenberg A A. Neurofuzzy control of modular and reconfigurable robots [J]. IEEE/ASME Trans on Mechatronics, 2003, 8(3): 381-389.

[3] 李英, 朱明超, 李元春. 可重构机械臂鲁棒模糊神经补偿控制仿真研究[J]. 系统仿真学报, 2007, 19(22): 5169-5174.
(Li Y, Zhu M C, Li Y C. Simulation on robust neurofuzzy compensator for reconfigurable manipulator motion control [J]. J of System Simulation, 2007, 19(22): 5169-5174.)

[4] Kirchoff S, Melek W W. A saturation-type robust controller for modular manipulators arms [J]. Mechatronics, 2007, 17: 175-190.

[5] Liu G J, Abdul S, Goldenberg A A. Distributed control of modular and reconfigurable robot with torque sensing [J]. Robotica, 2008, 26: 75-84.

[6] Hsu S, Fu L. A fully adaptive decentralized control of robot manipulators [J]. Automatica, 2006, 42: 1761-1767.

[7] Tang Y, Tomizuka M, Guerrero G, et al. Decentralized robust control of mechanical systems [J]. IEEE Trans on Automatic Control, 2000, 45(4): 771-775.

[8] Zhu M C, Li Y, Li Y C. A new distributed control scheme of modular and reconfigurable robots [C]. Proc of the IEEE Int Conf on Mechatronics and Automation. Harbin, 2007: 2622-2627.

(上接第 428 页)

[3] Dawson D W, Askin R G. Optimal new product design using quality function deployment with empirical value functions [J]. Quality and Reliability Engineering, 1999, 15(1): 17-32

[4] 汪峥, 严洪森, 刘霞岭, 等. 并行工程产品开发过程定量化建模与计划制订[J]. 管理科学学报, 2000, 3(4): 46-59.
(Wang Z, Yan H S, Liu X L, et al. Quantitative modeling and planning of the product development process in concurrent engineering[J]. J of Manegement Sciences, 2000, 3(4): 46-59.)

[5] Tareghian R H, Taheri S H. On the discrete time, cost and quality trade-off problem[J]. Applied Mathematics and Computation, 2006, 181(2): 1305-1312.

[6] 刘士新, 王梦光, 聂义勇. 多执行模式资源受限工程调度问题的优化算法[J]. 系统工程学报, 2001, 16(1): 55-60.
(Liu S X, Wang M G, Nie Y Y. Optimization algorithm for solving multi-mode resource-constrained project scheduling problem[J]. J of Systems Engineering, 2001, 16(1): 55-60.)

[7] 唐加福, 庞士宗, 汪定伟, 等. 利用品质功能展开进行产品优化设计[J]. 机械工程学报, 2003, 39(3): 105-109.
(Tang J F, Pang S Z, Wang D W, et al. Product optimization design using quality function deployment [J]. J of Mechanical Engineering, 2003, 39(3): 105-109.)

[8] Kis T. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities [J]. Mathematical Programming, 2005, 103(3): 519-539.

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