文章编号: 1001-0920(2008)08-0910-05

基于 DSM 的复杂产品开发流程优化遗传算法

陈冬宇[1], 邱菀华[1], 杨　青[2], 杨　敏[1]
(1. 北京航空航天大学 经济管理学院, 北京 100083; 2. 北京科技大学 经济管理学院, 北京 100083)

摘　要: 为减少产品开发过程中的返工迭代,提出一种基于设计结构矩阵(DSM)理论的多目标流程优化遗传算法. 通过优化任务执行顺序,减少产品开发过程中的返工以压缩进度和降低成本. 该优化算法是一种改进的遗传(GA)算法,在适应度函数中考虑了时间和费用两个指标;在选择、交叉、变异算子中采用了优解保持策略. 仿真结果表明,对于高任务耦合度的产品开发项目,该优化算法能使开发时间压缩 30 %～40 %,费用降低 7 %～20 %.

关键词: 流程优化; 遗传算法; 仿真; 设计结构矩阵
中图分类号: TH168　　**文献标识码**: A

DSM-based complex product development process optimization using genetic algorithm

CHEN Dong-yu[1], *QIU Wan-hua*[1], *YANG Qing*[2], *YANG Min*[1]
(1. School of Economics and Management, Beihang University, Beijing 100083, China; School of Economics and Management, Beijing University of Science and Technology, Beijing 100083, China. Correspondent: CHEN Dong-yu, E-mail: chendongyu@sem.buaa.edu.cn)

Abstract: A multi-object scheduling algorithm based on design structure matrix (DSM) theory is presented for product development process optimization. It is an improved genetic algorithm which takes several measures to have the computation result effective, stable, and reliable. First, it takes account of multi-indicators, such as time and expenditure, in constructing fitness function. Second, several policies are adopted to retain good individuals in the group during the selection, crossover, and mutation process. Computation study results show that this optimization algorithm can effectively reduce interactions, compress time (30 %～40 %) and save expenditure (7 %～20 %).

Key words: Schedule optimization; Genetic algorithm; Simulation; DSM matrix

1 引　　言

现代产品设计的复杂性和知识密集性,使产品开发过程中存在大量相互依赖的设计任务[1]. 采用传统的产品开发思路,一旦在产品开发后期发现产品设计缺陷,将导致大范围的修改和返工,最终导致项目开发失败. 如何设计高效合理的产品开发过程,减少返工造成的损失,已成为产品开发能否成功的关键问题.

Steward[2] 于 1981 年提出了设计结构矩阵(DSM)理论,该理论模型能够描述复杂产品开发过程中的耦合关系以及迭代过程,从而为减少开发任务的复杂程度提供了一个强有力的分析工具. DSM 理论模型在并行工程框架下得到了迅速发展,许多学者提出了基于 DSM 理论模型的产品开发流程优化方法[3]. 例如,张汉鹏等[4]提出了测量任务耦合度的方法,并对任务次序进行了优化; Smith[5,6] 和 Epinger[7] 分别在顺序执行和并行执行的假设前提下,提出了基于设计结构矩阵的优化模型; Browning[8,9] 研究了如何通过削减迭代的次数,从而缩短开发迭代周期循环时间的方法; Ong 等[10] 提出了通过相似状态空间分析和预测任务迭代次数优化产品开发过程的思路.

虽然人们在优化产品开发流程方面作了很多研究,但基本只侧重于减少开发时间,而在降低开发费用方面却很少有人考虑. 本文则同时考虑时间和费用两个指标,研究产品开发多目标流程优化问题,设

收稿日期: 2007-05-21; **修回日期**: 2007-09-17.
基金项目: 国家自然科学基金项目(70701001).
作者简介: 陈冬宇(1981—),男,浙江嵊州人,博士生,从事并行工程、项目管理的研究; 邱菀华(1946—),女,江西临川人,博士生导师,从事管理决策分析、项目管理等研究.

计了基于工序执行优先规则优化产品开发流程的智能算法.

2　并行产品开发模型

2.1　基本假设及问题描述

并行产品开发模型的前提假设如下:

1) 产品开发项目可以分解成多个任务,每个任务的完工时间和完工费用服从一定的统计规律;任务之间的关系可以用布尔型信息流矩阵加以表示.

2) 项目完工之后会给其他任务输出信息,导致其他任务开工或返工,返工概率和返工影响可以事先估计,分别用返工概率矩阵和返工冲击矩阵表示,且不随时间变化.

3) 每个任务存在最大返工次数,当某任务的返工次数超过指定值时,即使再有其他任务给它提供信息,该任务也不再返工(这一假设是为了避免无穷次返工而导致项目无法完成的情况发生).

4) 项目执行过程中资源充足,多个项目可以并行执行,不存在资源瓶颈.

在产品开发过程中,任务之间可能存在信息循环. 若某任务对其有信息输入的上游工序都已完工,则该任务可以得到执行. 当某任务完成时,它会给下游的任务输出信息,从而使下游任务可以开工,同时,它也可能会对上游任务提供反馈信息,从而使上游任务以一定的概率修正其所做的工作,即返工;此外,当下游任务的结果达不到指定要求时,也会造成上游工序的返工. 一个任务的返工又有可能导致其他任务返工,从而使整个项目形成一个复杂网络结构.

2.2　复杂产品开发过程建模

由于项目开发过程中存在返工问题,项目总体工期和费用与各项任务工期和费用之间的关系很难用解析方法得到,只能用仿真得到近似解. 该仿真模型为离散事件仿真模型,触发事件为一个或多个任务在某一点上完工;状态包括 3 个方面:各任务的未完工量,各任务的可执行状态以及各任务的返工次数. 给定任务执行优先度序列,产品开发仿真过程如下:

Step1: 遍历所有任务,若某任务不依赖于其他上游任务,或该任务所依赖的上游任务都已完工,且该任务尚未完工,则该任务为可执行任务,表示该任务当前时间可开工. 查找所有可执行的任务,根据这些任务的执行时间和未完工量,将任务中最短完工的任务时间设置为当前的仿真时间片 dt,将其加到累计时间 TT 中. dt 的计算公式为

$$dt = \min_{E_{it}\neq 0}\{V_{it} \cdot T_i\}.$$

Step2: 对于每项可执行任务 i,在 dt 时间内完成的工作量为 dt/T_i,执行成本为 $C_i * dt/T_i$. 其中:T_i 为任务 i 的执行时间,C_i 为任务 i 的执行成本. 将该成本累计到总成本 TC 中,若 t 时刻任务 i 原来的剩余工作量设为 k_{it} %,则经时间 dt 之后,剩余工作量变为 k_{it} % - dt/T_i. 对某项在 dt 时间内完工的任务 i,若该任务输出信息给任务 j,且 j 此时已部分完工,则 i 的完工可能导致已部分完工的任务 j 返工. 当任务 j 的返工次数小于最大返工次数时($n_j < M_{rw}$),生成随机数 r. 若 $r \leq R_{ij}$(R 为返工概率矩阵),则 j 需返工. j 的返工次数加 1,根据返工冲击矩阵 P 重新计算 j 的未完工作量

$$k_{jt}\% + (1 - k_{jt}\%) \times P_{ij}.$$

如果 $r > R_{ij}$,则不返工. 将剩余工作量为 0 的任务可执行状态设置为 0.

Step3: 重新寻找并执行所有可执行的任务,直到所有的任务都完工.

由于返工过程是随机的,每次仿真得到的项目费用和项目工期为一个随机值. 执行多次这样的仿真,若满足以下条件:

$$\begin{cases} \dfrac{\left|\dfrac{1}{L}\sum_{l}^{L}TT_l - \dfrac{1}{L+1}\sum_{l}^{L+1}TT_l\right|}{\dfrac{1}{L}\sum_{l}^{L}TT_l} < 0.005, \\ \dfrac{\left|\dfrac{1}{L}\sum_{l}^{L}TC_l - \dfrac{1}{L+1}\sum_{l}^{L+1}TC_l\right|}{\dfrac{1}{L}\sum_{l}^{L}TC_l} < 0.005, \end{cases} \quad (1)$$

则表明仿真结果趋于稳定,停止仿真程序. 其中:TT_l 和 TC_l 分别为第 l 次仿真得到的项目工期和项目费用,L 为总仿真次数.

根据项目的任务参数和任务执行序列,可以通过多次仿真得到项目的完工时间和完工费用分布,从而进行风险分析,确定开发项目能在指定时间和费用要求下完工的概率. 产品开发项目能按要求完工的概率计算公式为

$$P(T_E, C_E) = 1 - F(T < T_E, C < C_E). \quad (2)$$

3　产品开发流程优化算法

根据任务执行优先度序列和项目时间费用之间的对应关系,本文设计了改进遗传算法来寻找近似最优的任务执行序列. 在适应度函数的设计中同时考虑了时间和费用两个因素,在遗传操作算子的设计中使用了多种优解保持策略,从而使算法能快速收敛,但又可以突破局部最优解. 该算法的目标是通过合理的项目开发流程设计,减少产品开发过程中不必要的返工迭代,从而降低开发费用,缩短开发时

间.

3.1 染色体编码方案

本文采用任务编码方式. 个体 Id 是一个任务序列 $\mathrm{Id}=\{i_1,i_2,\cdots,i_n\}$,表示任务执行优先顺序,每个 Id 对应一个进度规划方案. 解码过程就是根据任务优先序列 Id 和各任务工期确定各任务的执行顺序以及总的项目完工时间和完工费用.

3.2 适应度函数

项目时间越短,费用越低,个体的适应度就越大. 时间和费用的重要性权重可通过专家打分或调查访谈等方法加以确定. 由于仿真过程的随机性,无法得到项目时间和费用的最大最小值,无法进行项目时间和费用的规一化处理. 为消除量纲的影响,将初始情况下项目的平均执行时间和费用作为标准,适应度函数可定义为

$$\mathrm{Ft}(\mathrm{Seq})=\frac{1}{w_T\dfrac{\frac{1}{L}\sum_{j}^{L}\mathrm{TT}_j(\mathrm{Seq})}{\mathrm{TT}_j(\mathrm{Seq}_0)}+w_C\dfrac{\frac{1}{L}\sum_{j}^{L}\mathrm{TC}_j(\mathrm{Seq})}{\mathrm{TC}_j(\mathrm{Seq}_0)}}. \quad (3)$$

其中:Seq 为任务执行优先序列;$\mathrm{TT}_j(\mathrm{Seq})$ 和 $\mathrm{TC}_j(\mathrm{Seq})$ 为根据优先序列 Seq 在第 j 次仿真中得到的项目总时间和费用;Seq_0 为初始任务优先序列;L 为仿真次数;w_T 和 w_C 分别为时间和费用的重要性权重,有 $w_T+w_C=1$.

3.3 遗传算子及终止策略

假设初始种群个体数目为 M,则随机产生 M 个 1～n 随机排列的整数序列,每个序列为一个任务执行序列,对应一个个体. 在群体中选择生命力强的个体产生新的群体,以实现整个种群的优化. 本文采用比例选择算子(赌盘选择算子),即个体被选中并遗传到下一代群体中的概率与个体的适应度大小成正比.

交叉算法采用单点交叉. 在交叉点选择的问题上,本文对传统的交叉方法做了一定的改进. 对于两个父体 i 和 j,其适应度分别为 $\mathrm{Ft}(i)$ 和 $\mathrm{Ft}(j)$,则交叉点的计算公式为

$$X=\mathrm{Floor}\left(\frac{\mathrm{Ft}(i)}{\mathrm{Ft}(i)+\mathrm{Ft}(j)}*N\right)+1. \quad (4)$$

其中:N 为染色体长度,即项目任务数;Floor() 为取整函数. 子体 Id′的染色体第 1 部分(前面 X 个基因值) 按顺序取自于父体 i 的前面 X 个基因,第 2 部分 $N-X$ 个基因则取自父体 j. 从父体 j 的第 1 个基因开始顺序选取,如果某基因值与前面第 1 部分的基因重复,则不重复选取该基因值,转而选取父体 j 的下一个基因值,直到选取到父体 j 的第 N 个基因为止. 交叉生成的子体拥有父体 i 的基因比例为

$$\frac{\mathrm{Ft}(i)}{\mathrm{Ft}(i)+\mathrm{Ft}(j)},$$

拥有父体 j 的基因比例为

$$\frac{\mathrm{Ft}(j)}{\mathrm{Ft}(i)+\mathrm{Ft}(j)}.$$

通过这一方法,拥有更高适应度的父体就会有更多的基因传递给下一代,从而更好地实现优胜劣汰. 新生成的个体用以替换种群中适应度低的个体.

本文中变异算法为:若染色体长度为 N,随机生成两个 1～N 之间的整数 i 和 j,将个体 i 位和 j 位上的基因值相互对调. 在种群中随机选取一定比例的个体以概率 P_m 执行变异操作,用新生成的个体替换种群中适应度低的个体.

当上下两代种群的平均适应度变化非常小时,可以认为算法已经收敛,算法停止. 其判断公式为

$$\left|\frac{\sum_{i}^{M}\mathrm{Ft}(\mathrm{Id}_{ki})-\sum_{i}^{M}\mathrm{Ft}(\mathrm{Id}_{(k+1)i})}{\sum_{i}^{M}\mathrm{Ft}(\mathrm{Id}_{ki})}\right|<0.005. \quad (5)$$

其中:k 表示遗传代数,Id_{ki} 表示第 k 代的第 i 个个体,M 表示种群大小. 从最后一代种群中找出适应度最大的个体,解码该个体的基因,得到最优解. 种群规模 P_s,交叉概率 P_c,变异概率 P_m,繁殖代数 G_m 等可根据经验进行估计,或通过多次实验比较加以确定.

4 仿真实例

用一个简单的例子说明该模型在某芯片研发项目中的应用. 该项目粗略可以划分成 A～I 这 9 个开发任务. 这些任务的完工时间服从正态分布,执行时间期望分别为{30, 40, 35, 39, 52, 55, 62, 29, 39},方差为{3.4, 4.2, 3.9, 3.8, 4.6, 4.8, 3.9, 2.4, 3.7}. 各任务单位时间的费用分别为{2, 3, 2, 1, 4, 3, 2, 4, 3},各任务的最大返工次数为 5.

这些任务之间的相互关系如图 1 所示.

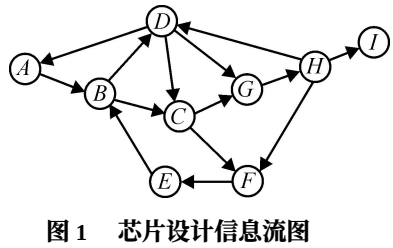


图 1 芯片设计信息流图

该项目的返工概率矩阵和返工冲击矩阵等信息分别如表 1 所示. 如果不存在返工情况,则所有任务的完工费用之和为 1 019.

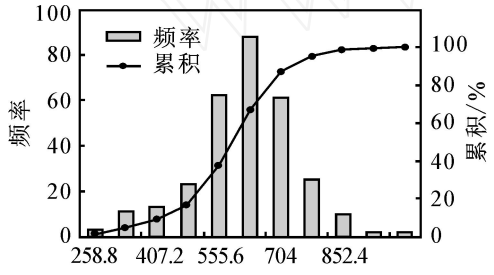
在 Matlab 7.0 平台上编制蒙特卡罗仿真程序,计算该开发项目完工时间和完工费用的分布. 仿真结果显示,当仿真次数超过 200 次时,仿真结果的平

表 1　项目任务工期及 DSM 影响矩阵

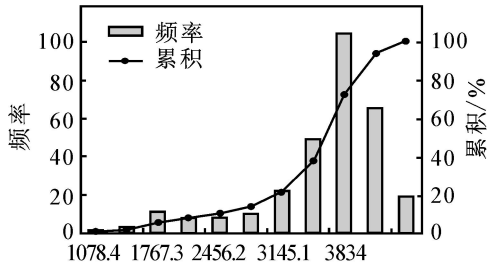
任务	A	B	C	D	E	F	G	H	I
A	—	0.8(0.9)	—	—	—	—	—	—	—
B	—	—	0.6(0.7)	0.7(0.6)	—	—	—	—	—
C	—	—	—	—	—	0.7(0.6)	0.9(0.8)	—	—
D	0.6(0.8)	—	0.7(0.8)	—	—	—	0.7(0.7)	—	—
E	—	0.8(0.9)	—	—	—	—	—	—	—
F	—	—	—	—	0.8(0.7)	—	—	—	—
G	—	—	—	—	—	—	—	0.8(0.8)	—
H	—	—	—	0.7(0.8)	—	0.9(0.7)	—	—	0.9(0.8)
I	—	—	—	—	—	—	—	—	—

注: 括号前的数为返工概率 R_{ij},括号内的数为返工冲击率 P_{ij}.

均值趋于稳定. 事实上,当仿真过程进行到 172 次时,就达到了仿真停止条件. 当该项目中的任务以 $\{A,B,C,D,E,F,G,H,I\}$ 的优先顺序执行时,项目的平均完工时间约为 588.0,方差为 117.7;完工费用为 3 487.7,方差为 710.4. 设定仿真次数为 300 次,项目完工时间和费用的分布情况如图 2 所示.



(a) 工期直方图



(b) 费用直方图

图 2　项目完工统计图

由于任务间的关联作用,项目在执行过程中发生多次返工,使工期和费用大大增加. 如果要求项目完工时间控制在 500 以内,费用在 3 500 以内,则项目能按要求完工的概率仅为 19.3 %.

在 Matlab 7.0 平台上编写改进的遗传算法优化程序,寻找近似最优的项目开发过程. 参数设置为:种群个数 $M = 50$,每次叠代替换的个体数目为 20,交叉概率为 0.8,变异概率为 0.1. 为便于观察,将适应度函数值扩大 10 倍,使其大约位于 1～10 之间. 由于市场情况不同,时间和费用对项目成功的重要程度也会不同. 本案例只考察以下两种特殊情况下的任务执行优化情况.

1) 对开发时间没有要求,只考虑费用的情况,即 $w_C = 1, w_T = 0$. 此时,种群平均适应度的变化情况如图 3 所示. 当遗传算法程序运行到 40 代左右时,种群的适应度函数平均值趋于稳定. 得到近似最优的项目执行优先顺序为 $\{G, C, D, H, B, E, F, A, I\}$; 项目执行时间平均值为 414.0, 方差为 100.7; 执行费用平均值为 2 808.0,方差为 543.6. 相对于原方案,时间大约节省 29.6 %,费用节省 19.4 %. 在这种项目执行顺序安排下,项目完工时间在 500 以内、完工费用在 3 500 以内的概率能达到 75.7 %.

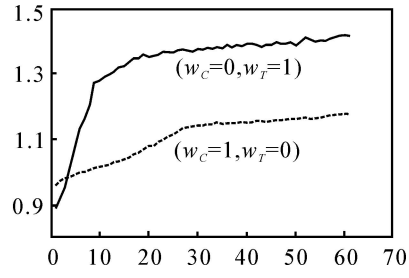


图 3　种群平均适应度变化曲线

2) 只考虑时间,不考虑开发费用的情况,即 $w_C = 0, w_T = 1$. 此时,仿真优化结果如图 3 所示. 当遗传算法程序运行到 20 代左右时,种群的平均适应度趋于稳定. 得到近似最优的项目执行优先度顺序为 $\{G, C, D, B, E, F, H, A, I\}$. 项目执行时间平均值为 370.1, 方差为 70.2; 执行费用平均值为 3 249.8,方差为 474.1. 相对于原方案,时间大约节省 37.1 %,费用大约节省 6.8 %. 在这种项目执行顺序安排下,项目的完工时间在 500 以内、完工费用在 3 500 以内的概率能达到 73.3 %.

通过比较上述仿真结果可以发现,在优化费用的同时,项目的执行时间也得到了优化;而在优化项目的同时,费用情况也得到了改善. 在不同的权重情

况下,优化的侧重点不同. 不论何种情况,项目能在时间和费用双重要求下顺利完工的概率大大提高.

为验证上述优化结果的稳定性,再次运行优化程序,得到的优化序列分别为{ *G*, *C*, *D*, *H*, *B*, *I*, *E*, *F*, *A*} ,{ *G*, *C*, *D*, *B*, *E*, *F*, *H*, *I*, *A*}. 在上述两种执行优先度序列下,软件开发项目的完工时间分别为 418.7 和 378.6, 完工费用分别为 2 888.5 和 3 187.5, 数据变动都没超过 5 %.

仿真结果表明,根据管理者对进度和费用的偏好,可以得到不同的优化流程. 对于具有高度耦合性的产品开发项目,本算法可有效降低产品开发过程中的返工迭代. 在本案例中,开发时间压缩幅度在 30 % ～ 40 % 之间;费用降低幅度在 7 % ～ 20 % 之间,效果显著.

5 结 论

针对并行产品开发过程中大量存在的任务耦合和返工迭代问题,本文提出了基于遗传算法的多目标流程优化算法. 该算法可通过对项目工期和费用赋予不同的权重得到不同的项目执行优先度序列. 算例分析表明,该优化算法可有效降低项目开发过程中的返工迭代,从而缩短产品开发时间,节约开发费用.

由于本文没有考虑项目开发过程中的资源约束,对于具有严格资源约束要求的开发项目并不适用. 另外,本模型假设返工概率和返工影响是已知的,可通过专家经验或历史数据估算得到,这样的假设对于缺乏历史数据和开发经验的小公司而言通常是不成立的,这也是本文的一个局限. 今后的研究可以进一步探讨返工概率矩阵和返工冲击矩阵的变动对产品开发时间和费用产生的影响.

参考文献(References)

[1] Ali Yassine D B. Complex concurrent engineering and the design structure matrix method [J]. Concurrent Engineering: Research and Applications, 2003, 11(3): 165-177.

[2] Steward D V. Systems analysis and management: Structure, strategy and design [M]. New York: Petrocelli Books, 1981.

[3] 陈希, 王宁生. 虚拟企业环境下的复杂产品并行开发框架模型研究[J]. 控制与决策, 2003, 18(6): 716-719. (Chen X, Wang N S. Study on the framework of complex product concurrent development in a virtual enterprise[J]. Control and Decision, 2003, 18(6): 716-719.)

[4] Zhang H P, Qiu W H. An approach to measuring coupled tasks strength and sequencing of coupled tasks in new product development [J]. Concurrent Engineering: Research and Application, 2006, 14(4): 304-311.

[5] Smith R P, Eppinger S D. A predictive model of sequential iteration in engineering design [J]. Management Science, 1997, 43(8): 1104-1120.

[6] Smith R P, Eppinger S D. Deciding between sequential and parallel tasks in engineering design[J]. Concurrent Engineering: Research and Applications, 1998, 6(1): 15-25.

[7] Epinger S D. Model-based approaches to managing concurrent engineering[J]. J of Engineering Design, 1991, 2(4): 283-290.

[8] Browning T R. Use of dependency structure matrices for product development cycle time reduction[C]. Proc of the 5th ISPE Int Conf on Concurrent Engineering: Research and Applications. Tokyo, 1998: 89-96.

[9] Browning T R. The design structure matrix [C]. Technology Managment Handbook. New York: Boca Raton Press, 1999: 103-111.

[10] Lee S G, Ong K L, Khoo L P. Control and monitoring of concurrent design tasks in a dynamic environment [J]. Concurrent Engineering: Research and Application, 2004, 12(1): 59-66.