Article ID: 1000-8152(2001)06-0847-04

Resource-Constraints Based Optimization Scheduling Approach for Concurrent Activities

YAN Jihong and WU Cheng
(CIMS Center, Department of Automation, Tsinghua University·Beijing, 100084, P.R.China)

Abstract: *A resource-constraints based optimization scheduling approach is proposed for concurrent product development process*. First of all, three definitions for concurrent activities are given. Then an integrated function is proposed in the algorithm for optimization scheduling. Two critical items in concurrent engineering-role allocation and feedback revision are also addressed. Simulation results show feasibility of the algorithm.

Key words: resource-constraints based scheduling; optimization; concurrent engineering; activities
Document code: A

基于资源约束的并行活动优化调度方法

闫纪红 吴 澄
(清华大学自动化系 CIMS 中心·北京,100084)

摘要: 针对并行产品开发过程,提出了一种基于资源约束的优化调度算法.首先给出了并行活动的三个定义,提出了一个使算法能够实现优化调度的综合函数,并研究了并行工程中的两个关键问题——角色分配和反馈修改.仿真实例表明了算法的可行性.

关键词: 基于资源约束调度; 优化; 并行工程; 活动

1 Introduction

Concurrent engineering (CE) as a philosophy aims to address the consideration of different life cycle issues of a product at early stage of the design process in order to analyze the factors affecting manufacturing process. Recently, concurrent engineering has placed greater emphasis on the optimization of development process, which will determine 80% of the total life-cycle cost[1].

So far, there are a few literature which address concurrent scheduling method (see, for instance, [2~4]). But none of them considered the limitations on resource availability in the scheduling process. A resource constrained scheduling strategy for concurrent scheduling was shown in [5], but role allocation and feedback revision - the crux of CE, is not mentioned.

In this research, an optimization algorithm compromising *resources and information constraints* at the same time is achieved by taking advantage of three definitions in Section 3 and several items in Section 4. Role allocation is implemented according to personnel's ability and feedback revision process is realized by considering the revision process as a reschedule cycle.

2 Problem formulation

The problem considered here is minimizing the completion time *for design process submitting* to *all* constraints. Assuming the number of activities in a process is n, the sort of available resource is m, the scheduling problem can be described by the mathematical model as *follows*:

Minimizing

$$t_{e,n+1}, \tag{1}$$

Satisfying

$$t_{e,n+1} \geqslant t_{e,i},\ i \in (1,\cdots,n), \tag{2}$$

$$t_{e,j} - t_{e,i} \geqslant t_{r,ij} + t_j,\ i,j \in (1,\cdots,n), \tag{3}$$

$$\sum r_{ik}(t) < \bar{R}_k(t),\ t = t_{e,i-1},\ k = 1,2,\cdots,m, \tag{4}$$

where, $t_{e,i}$ is completion time of activity i, when $i = 0$, $t_{e,0}$ is initial time, $t_{c,ij}$ is concurrent time between activi-

Received date: 2000-03-27; Revised date: 2001-05-28.

ty i and j, $r_{ik}(t)$ is the quantity of resource k required by task i at time t, $\bar{R}_k(t)$ is available quantity of resource k at time t. Formula (1) is the objective function.

3 Definitions of concurrent activity

Fig. 1 shows intersectant, parallel and feedback natures of relationships between concurrent activities[6]. Activity "END" is a dummy one, standing for process ending.

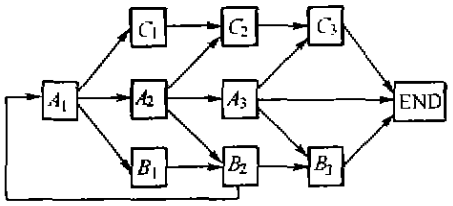

Fig 1 Relationships between activities in one development phase

Definition 1 Layer number of activity. Activities in the bottom layer such as B_3, C_3 shown in Fig. 1, whose layer number is defined as 1, the layer number of activities such as B_2, C_2 is defined as 2. Larger number should be chosen if there are some conflicts in deciding the layer number, for example, the layer number of activity A_3 is 2 rather than 1 because the immediate downstream activities are B_3, C_3, which possess layer 1.

Definition 2 Concurrent time $t_{c,ij}$. $t_{c,ij} \geq 0$ specifies that activity j *cannot start* until at least $t_{c,ij}$ time units after the completion of activity i; $t_{c,ij} < 0$ specifies that there can be $|t_{c,ij}|$ time units overlapping between activity j and activity i.

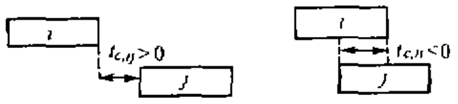

Fig. 2 Concurrent time $t_{c,ij}$

Definition 3 Activity. Each activity is described by a vector $G = (N, A, C, R)$, where

N is ID number of an activity;

A represents aggregation of all the interrelated activities both forward and backward;

$C = (l, t_l, n_d, \text{flag}, t_c)$ states the sequential nature of the relationships between activities, where l is the layer number of current activity, t_l is the lead time of activity i, n_d is the number of immediate downstream activities relevant to the current item; flag is the sign term of pre-release, flag = 1 is a sign bit of pre-release; t_c is a vector consists of all the overlapping time with respect to the immediate downstream activities relevant to the current item;

R is a demand vector of resources, assuming all the resources are discrete, the sort of resources required by an activity is m, and the quantity of each kind of resources is $r_1, \cdots, r_m$ respectively.

4 Scheduling algorithm

4.1 Weighting factors allocation

At first, an integrated priority function is put forward:

$$J = \sum_{i=1}^{5} \lambda_i J_i,$$

where $J_i (i = 1, \cdots, 5)$, J_1 is the layer number of each activity, J_2 is the number of downstream overlapping activities, J_3 is the number of downstream activities, J_4 is the lead time of the task, J_5 represents the resources possessed by the activity. Each J_i is normalized. λ_i is a weighting factor between 0 and 1. Here, the activity possessing larger magnitude of weighting sum is the critical task and, as such requires a greater attention in the scheduling process.

4.2 Resource allocation

In scheduling process, an activity which possesses larger magnitude of J is given priority for resource allocation when activities compete for resources. The rest schedulable activities will keep waiting until the resource requirement is satisfied. Note that, the required resources of an activity are invariable during the performing process, and the possessed resources will be released immediately when the task is ended.

4.3 Role allocation

Role allocation *is the most difficult problem to solve* owing to the fact that it is possible for a person to take charge of different roles in one project and the ability of different roles for a person is also distinct. For the sake of allotting personnel properly and giving free play to every role, four tables are established in our database. The relationships described by the tables are the following:

1) the ability of different roles taken by individuals;

2) roles required by different activities;

3) hardware required by different roles;

4) hardware (such as computer) required by software.

Naturally, the mutual relationships between ability, role, personnel, software, and hardware can be obtained from the above tables. Here, role allocation principle is: choosing the personnel to complete the task which he is adept at, namely, choosing the one who possesses the largest ability value in the ability table of this activity.

4.4 Feedback revision

In practice, normal activity can not be interrupted from the starting time until it is accomplished. In the event of design failure, the failure will be detected by the downstream activities after prerelease. Then revision would take place. If there are some revisions after prerelease and detection, the abnormal activity (activity need to be revised) is divided into two types: the task has been finished and the task is being performed. The latter should be stopped immediately when the feedback revision occurs, and then all the abnormal activities will wait for being rescheduled.

4.5 Scheduling procedure

Note that, it is impossible to schedule all the activities of a whole project due to the fact that the design process consists of a lot of activities and the activities are uncertain. Here, dynamic scheduling scheme is adopted. Therefore, the algorithm must be easier because the number of schedulable activities is limited in one scheduling step under the condition of multi-constraints.

Scheduling procedure is:

Step 1 Updating the vector of available resources to obtain $\tilde{R}$ at a moment.

Step 2 Finding out all the schedulable activities and calculating the priority function J.

Step 3 Choosing the schedulable activities under the limited resources according to the priority.

Step 4 Applying for resources if the available resources are inadequate.

Step 5 Releasing resource immediately when an activity is ended, return to Step1.

5 Simulation example

Simulation example is given based on one phase of a certain product development process. The number of activities for scheduling is 10, relationships between activities are shown in Fig.3. Task 3 represents an activity of another group. Obviously, layer number vector of these activities is (5, 4, 1, 3, 2, 3, 2, 1, 2, 1). Known lead-time vector is (7,3,6,3,4,5,6,5,6,4).

Besides, R, relationship tables can be picked up from the existing database. Here, the required resources are given by R, R = (manpower, software, computer). Manpower = 20, software = (8,9,7,10,9,10,14,9), that is, the sort of software is 8, computer = 20.

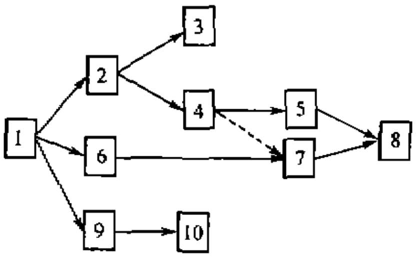

Fig. 3 Relationships between activities

Traditionally, all the activities were executed one by one. Obviously, the completion time would be 43 days (except activity 3) if the activities are performed in the traditional sequential fashion. The scheduling result of these activities using the optimization approach proposed in this paper is shown in Fig.4, and the total completion time is 24 days. Weighing factors are decided according to idiographic problem. Here, the weighing vector is (0.5,0.4,0.3,0.2,0.1) after lots of simulation experiments. The optimization scheduling procedure is shown in Table 1.

Table 1 Scheduling procedure for these activities

	Schedulable set- $N(J)$
Step 1	1(1.1192)
Step 2	2(1.0118),6(0.8007),9(0.6796)
Step 3	4(1.0183),9(0.7629)
Step 4	5(1.1158),7(0.6796)
Step 5	10(0.4038)
Step 6	8(0.4369)

In Step 1, only one activity can be scheduled. The result is rational because the others can not be performed until activity 1 is accomplished. After the first activity is scheduled, activities 2, 6, and 9 become schedulable in Step 2. Note that, only two of them (activities 2 and 6) have been scheduled, and activity 9 has to wait for resources due to the smaller magnitude of J it possesses. In Step 3, only one activity (activity 4) emerges in schedulable set newly. It is calculated along with activity 9 for scheduling and wins the higher priority. As such, all the activities are scheduled successfully, based on the principle proposed in this paper.

Ruling the representation (a, b) states that the feedback points to activity a and occurs at time b. If there are design failures, in the interest of realizing the revision process, some items such as time (when feedback occurs), the ID number of the activity feedback points should be input. Fig.5 is the result for these activities with feedback revision (4,12).

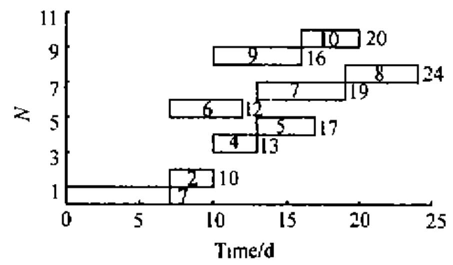

Fig. 4 Scheduling result without feedback revision

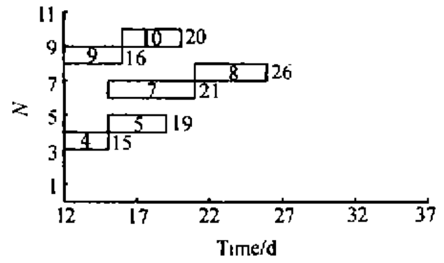

Fig. 5 Scheduling result with feedback revision at (4,12)

When Gannt chart in Fig.4 and Fig.5 are compared, the changes in feedback revision become obvious: the rearrangements take place in the latter scheduling projects, and the accomplished activities irrespective to the revisions need not to be re-implemented in the feedback revision process. The attribute just represents the characteristic and advantage of CE.

6 Conclusions

The natural outcome of this approach is providing an optimization scheduling algorithm for concurrent activities via a priority function defined by us which integrates all the key factors such as time order, information, resources, overlapping time and so on. In addition, definitions for concurrent activities are proposed, which provides the necessary basis for scheduling the activities in concurrent engineering. Moreover, role allocation based on ability is discussed, and then the manpower allocation in CE is mainly according to the ability rather than "person" such as in the existing literature. Finally, pre-release and feedback revision are realized. The scheduling algorithm can ensure utilizing resources effectively and the delivering information optimally, and concurrent development process benefits from these achievements at the same time. Applications have testified that the algorithm is convenient and feasible for solving optimal problem of practical concurrent development process.

References

[1] Hatch M and Badienelli R D. A concurrent optimization methodology for concurrent engineering [J]. IEEE Trans. Eng. Manage., 1999, 46(1):72 - 86

[2] Gayretli A and Abdalla H S. An object-oriented constraints-based system for concurrent product development [J]. Robotics and Computer-Integrated Manufacturing, 1999,15(3):133 - 144

[3] Tappeta V R and Renau E J. A comparision of equality constraints formulation for concurrent design optimization [J]. Concurrent Eng. Res. Appl., 1997,5(2): 253 - 261

[4] Downlatshahi S and Ashok M S Design optimization in concurrent engineering: a team approach [J]. Concurrent Eng. Res. Appl., 1997,5(2):145 - 154

[5] Badiru A B. Scheduling of concurrent manufacturing projects [A]. Parsaei H R, ed. Concurrent Engineering: Contempory Issues and Modern Design Tools [M]. 1st ed. Cambridge: Chapman & Hall, 1993, 93 - 109

[6] Xiong G L, Zhang Y Y and Li B H. The research on collective techniques and executive methods for concurrent engineering [J]. Computer Integrated Manufacturing Systems, 1996,2(3):3 - 8

本文作者简介

闫纪红 1972年生.1999年获哈尔滨工业大学控制工程系博士学位.现在清华大学自动化系CIMS中心做博士后研究.目前感兴趣的领域为智能优化调度算法,过程建模与仿真,并行工程等.

吴 澄 1940年生.中国工程院院士.国家863计划自动化领域首席科学家.清华大学自动化系教授,博士生导师,国家CIMS工程研究中心主任.目前感兴趣的领域为复杂制造系统的建模、分析和智能优化等.