Workflow Timed Critical Path Optimization

Haibo Li[1], Dechen Zhan[2]

1 Centre of Intelligent Computing of Enterprises, School of Computer Science and Engineering, Harbin Institute of Technology, Harbin, Heilongjiang 150080, China
2 School of Engineer, Northeast Agriculture University, Heilongjiang 150030, China
Email: lihaibo@hit.edu.cn; dechen@hit.edu.cn

Abstract: Approaches to shorten workflow execution time have been discussed in many area of computer engineering such as parallel and distributed systems, a computer circuit, and PERT chart for project management. To optimize workflow model structure of workflow, an approach with corresponding algorithms is proposed to cut timed critical path of workflow schema, which has the longest average execution time path from the start activity to the end activity. Through systematically analyzing the dependency relationships between tasks at build-time, traditional optimization method of critical path is improved through adding selective and parallel control structures into workflow schemas. Data dependency rules are converted to control dependency rules according to semantic rules mined. Further more, consistency between tasks is guaranteed. Finally, to explain validity of the algorithm proposed, an experiment is provided to compare optimized model with original using critical path identification algorithm. [Nature and Science. 2005;3(2):65-74].

Keywords: workflow; critical path; data dependency relationship; control dependency relationship; control structure

1 Introduction

Workflow technology is an effective measure to change business processes in a more direct way. A workflow is the automation of a business process. In whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules (WMFC, 1995). To optimize workflow model and shorten execution duration of workflow is one of the most important way to improve efficiency of business processes. In order to shorten average execution times of workflow, the critical path in a workflow schema should be shorten first.

A simple definition of the critical path of a program is the longest, time-weighted sequence of events from the start of the program to its termination (Jeffrey, 1998). The critical path in a workflow schema is commonly defined as a path with the longest average execution time from the start activity to the end activity (Chang, 2002). The activities in the critical path are called critical activities. The execution times of activities in the critical path directly affect the total workflow completion time. The critical path has been widely discussed in many fields of computer engineering, e.g. evaluating the performance of large dynamic circuits (Lee, 1999), analyzing bottleneck of program in parallel and distributed systems (Jeffrey, 1998), determining the critical path in PERT chart for project management (Cormen, 1994). The concept of the critical path and the critical activity can be effectively utilized in many workflow issues, for example, workflow resource (Jin, 2001, Oh, 2000) and time management (Hagen, 1998; Pozewaunig, 1997; Heinl, 1998).

Many researches for the optimization algorithms of critical path (Gomory, 1961; Lenstra, 1977; Singh, 2000; Lam, 1977) were applied to parallel computing (Meajil, 2000; Hribar, 2001; Singh, 2001). Another approach to shorten critical path is used in project domain. For example, to check the rationality of schedule of every task in critical path, to make tasks on critical path parallel processing by decomposing them,

to support critical path by cutting resources on non-critical path, and to reschedule network structure are measures to shorten critical path (Hu, 1998). Traditional methods to analyze critical path in project domain are PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method). But it is more complicated to shorten a critical path in workflow schema than in PERT chart. Firstly, we cannot use the previous methods (e.g. PERT) to shorten the critical path in a workflow schema, because they cannot support the two common control structures in workflow schema, i.e. selective structure and iterative structure (Chang, 2002). Secondly, optimizing a workflow model must guarantee consistency of all data dependencies and control constraints. For example, as response to change request, all data dependencies and control constraints between activities must be detected whether the problem of missing input or output values or cyclic waits may occur in the modified schema graph. Jin (2005) extended description power of PERT chart. Instead analyzing from workflow model domain, Jin (2005) used PERT chart to shorten workflow critical path as an assistant method. Other researches for the identification of critical path (Aalst, 1998; Cormen, 1994; Jin, 2005; Jin, 2001) only proposed a method to identify critical path in the context of a workflow, however, not gave an approach to shorten it.

To shorten the critical path in a workflow schema and meet all conditions when restructuring workflow graph, we must analyze data dependency systematically and control dependency between activities in workflow at first. We argue that the data dependency can be converted to data dependency between activities. Based on this convertibility, it is possible to shorten average execution time of critical path in a workflow schema by adding selective structure and iterative structure.

The remainder of the paper is organized as follows: In Section 2, we analyze data dependency and control dependency between activities in workflow schema and give some relative definitions. Section 3 gives some basic algorithms to guarantee consistency of workflow schema when changing. Section 4 presents our proposed method that systematically optimizes critical path in workflow schema. Section 5 gives experimental results to compare optimized model with original using critical path identification algorithm. Finally, we discuss the further work in Section 6.

2 Dependency Rule between Tasks

In workflow model, a task corresponds to a generic piece of work. A task is not defined for a specific business object, but for a type of objects, i.e., a task may be executed for many business objects. A business process is composed of one or more tasks following a certain order. An activity is one execution of a task. Each task in process is not isolated. The most obvious relationship between activities is logistic order which corresponds to a kind of partial order. This relationship is also called control dependency. These logistic relationships compose the control structures of the workflow. Four kinds of basic control structures, that is sequential, parallel (AND-Split, AND-Join), selective (OR-Split, OR-Join) and iterative (LOOP) structures, are defined in the workflow reference model (WDMC, 1995). Their semantics excerpted from WfMC (1999) are as follows.

(I) **Sequence:** Activities are executed in order under a single thread of execution, which means that the succeeding activity cannot start until the preceding activity is completed.

(II) **AND-Split:** A single thread of control splits into two or more threads which are executed in parallel within the workflow, allowing multiple activities to be executed simultaneously.

(III) **AND-Join:** Two or more parallel executing activities converge into a single common thread of control.

(IV) **OR-Split:** A single thread of control makes a decision upon which branch to take when encountered with multiple alternative workflow branches.

(V) **OR-Join:** Two or more alternative workflow branches re-converge to a single common activity as the next step within the workflow. No synchronization is required because of no parallel activity execution.

(VI) **LOOP:** A workflow cycle involves the repetitive execution of one (or more) workflow activities until a condition is met.

Except for control dependency between tasks, there exists data dependency relationship between tasks which expresses data access rule. All input data must be supplied before executing an activity, and after its successful completion all output data are written correctly. All data dependencies rules between activities should work and detect whether the problem of missing input or output values or cyclic waits may

occur at runtime. So data dependency rule should be well defined in workflow model to restrict the data access in workflow systems. In Figure 1, there exists control dependency (see solid lines) and its rules (predication P4, P5) between tasks T1, T2 and T3. There also exists data dependency (see dot lines which denote input and output relationship) and its rules (predication P1, P2, P3) among tasks T1, T2 and T3. The data dependencies for task T3 are checked. T3 is executable if all its data dependency rules are met. Value d3 produced by T1 must meet rule P3 before d5 is operated (read/written). Value d1 produced by other tasks must satisfy P1 before d4 is operated, and d2 is the same as d1.

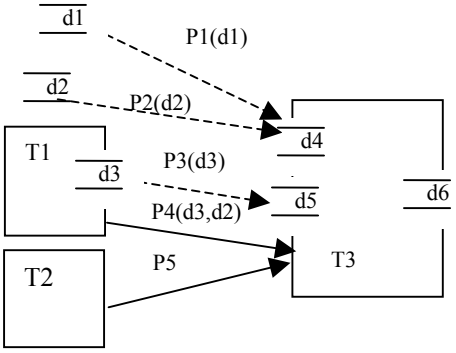


Figure 1. Dependencies relationship between tasks

The automation of business processes needs to be abstracted using a language, namely workflow specification language. The result is called workflow specification which contains information formally describing various aspects of a workflow (Chan, 1997). Considering dependency rules analyzed above, a workflow specification can be defined as follows:

Definition 1 (*Workflow specification*). A workflow specification, ws, is abstracted as a 4-tuple $< TN,CN,D,R >$, where

(i) $TN=\{t_1,t_2,...,t_n\}$ is a set of task nodes.

(ii) $CN=\{cn_1,cn_2,...,cn_n\}$ is a set of control nodes. Each element in CN has one of the above 6 types, that is Sequence, AND-Split, AND-Join, OR-Split, OR-Join or LOOP.

(iii) D is a set of data, used for tasks' input or output.

(iv) R is a superset of rules, $R=\{DR,CR\}$, DR is a set of data dependency rules, and CR is a set of control dependency rules.

Data dependency rule and control dependency role can be defined as follows.

Definition 2 (*Data dependency rule*). Let $t_i.d,..., t_k.d$ and $t_j.d$ be output data of tasks $t_i,...,t_k$ and input data of task t_j respectively, where $t_m.d \in D$. If $t_i.d,..., t_k.d$ need to be read before $t_j.d$ is created, say that there exists data dependency between $t_j.d$ and $t_i.d,..., t_k.d$. If condition $P_j(t_i.d,..., t_k.d)$=TRUE must be met before $t_j.d$ is created, say that P_j is a data dependency rule between data $t_j.d$ and $t_i.d,..., t_k.d$, denoted as $P_j(t_i.d,..., t_k.d) \rightarrow t_j.d$, where $P_j \in$ DR is the rule predication.

Definition 3 (*Control dependency rule*). For $\forall t_i,t_j \in$ TN, under the control of four basic control structures, if a partial order $t_i \prec t_j$ or $t_j \prec t_i$ between t_i and t_j can be determined, say there exists control dependency relationship between t_i and t_j, denoted as $t_i \xrightarrow{P} t_j$, where P is control dependency rule, for short, control rule.

Data dependency relationship between tasks can be classified as semantic and non-semantic rules. Non-semantic rule involves the validity of data chiefly, for example, to validate data format in database table. Semantic rule contains information which can be used to impact the course of business processes. For example, equipment management system in manufacture enterprise will execute different business branches according to different fault codes which origin from different equipment fault type. Due to lacking semantic analysis of business data, though structure of model is correct, data dependency relationships are always neglected. Only those semantic data dependency rules could be converted to control rules, so that we only discuss this sort of rule. Business rules can be embodied in software systems more exactly by mining data relationship.

3 Restructuring Model Schema

To achieve the goal of optimizing critical path of workflow, workflow model structure must be changed, so that execution time of critical path can be cut as short as possible. But traditional methods to analyze critical path such as PERT chart, does not support usual control structures: selective and iterative structure. More over, restructuring workflow schema must guarantee consistency of all data dependencies and control constraints. If task t2 has not completed while task t is ready to execute, in Figure 2 (a), data d2 may be

invalid, so task t and t2 cannot execute synchronously. In order to shorten critical path of workflow by add selective and parallel structures, consistency of all input and output data between tasks must be considered.

3.1 Background about critical path method

Numerous natural physical and organic processes exhibit behavior that is probably meaningfully modeled by Poisson processes. An important application of the Poisson distribution arises in connection with the occurrence of events of a particular type over time. The exponential distribution is frequently used as a model for the distribution of times between the occurrences of successive events such as customers arriving at a service facility. Because of them, the Poisson process and the exponential distribution have been used to analyze many areas of computer engineering. Hence, in this paper it is reasonable to consider workflow schema as a M/M/1 queuing network, as presented, see Chang (2002). We compare optimized model with original using critical path identification algorithm given by Chang (2002).

A workflow schema is represented with a set of nodes and directed edges. Tasks are interconnected by the six types of control structures. In a M/M/1 workflow queuing network, each activity is an independent M/M/1 queuing system at runtime of a workflow. Therefore, the arrival and departure rate in each activity can be specified, as well as the initial request rate to the start node, the service rate in each activity, and the branch selection-probabilities in each workflow control structure. So the average execution time of each task in a workflow schema can be determined. Consequently the path having longest average execution time in a workflow schema can be determined.

The execution time of each control structure is computed according to the following formulas:

(I) Sequence control structure: $W=\sum(1/\mu_i-\lambda_i)$, where λ_i is the arrival rate, μ_i is the service rate of task t_i.

(II) Selective control structure: $W=MAX(\sum(1/\mu_i-p_i\lambda_i))$, where $\sum p_i=1$, p_i is arrival probability of a branch.

(III) Parallel control structure: $W=MAX(\sum(1/\mu_i-\lambda_i))$.

Iterative control structure: $W=(1/p\mu_1-\lambda)+(1/p\mu_2-\lambda)+\ldots+(1/p\mu_n-\lambda)$

3.2 Consistency analysis

The correctness of workflow schema can be guaranteed by some rules, such as soundness property, which states that starting from the initial task node, it is always possible to reach any task nodes in a workflow schema, and for each reachable nodes, it is always possible to reach the finial task node (Aalst, 1998). Considering the six control types of workflow schema, for a split control structure, there must be a join control structure that can be combined with the split control structure whose resulting workflow schema forms a meaningful flow of control. From the meaning of the workflow control structures, when an AND-Split combines with an AND-Join and an OR-Split with an OR-Join, a workflow can be said to have a correct control structure. As a result, a workflow is an activity network in which activities are interconnected by workflow control where some control structures may be contained in some other control structures. In this paper we consider workflows meeting the following rules.

Rule 1: An AND-Split control structure should have its matching AND-Join control structure.

Rule 2: An OR-Split control structure should have its matching OR-Join control structure.

Rule 3: A non-sequential control structure can be completely contained in another non-sequential control structure, but two non-sequential control structures should not be partially overlapped. The rule is also called *nesting rule*. The outermost non-sequential control structure in a workflow is called a *control block*. The workflow depicted in Figure 4 includes a OR control block from node t1 to t7 and a LOOP control block. A control block can be nested in other control block. A control block starting from control node cn can be denoted as $Block(cn)=\{t_1,\ldots,t_m\}$, where t1,…,tm is activities involved in cn.

When logical orders between two task nodes are being changed, in principle, each task receives its input data from those tasks which already have been completed and produces its output data correctly. Data conflict must be avoided as well, i.e. two or more parallel tasks writing the same data item can lead to data conflict. Conflict problem are beyond this article's scope and can be seen in Minkyu (2001). So the following rules are given:

Rule 4 : For $\forall t\in TN$, and task set $T=\{t_1,t_2,\ldots,t_k\}$ which satisfies data dependency $P(t_1.d,\ldots,t_k.d)\rightarrow t.d$, where $t\notin T$, all tasks in T must be completed before t starts.

Rule 5 : Tasks in each branches of a parallel control structure can not write same data item. Let $Out(t_i)$ denote output data set of task t_i, so the rule can

also be described as following: $\cap Out(t_i)= \varnothing, i=1,\ldots,m, t_i \in Block(cn) = \{t_1,\ldots,t_m\}$.

When restructuring workflow graph, the five rules above must be held. The following sections give relevant restructuring steps.

3.3 Parallel control structure

When restructuring a workflow schema, to change sequential order of two tasks in critical order to parallel can cut average execution time. If this modification happens in a parallel control block, a second branch is only created, and if happens in a selective or iterative control block, a new parallel control block is nested in the control block. Adding more semantic rules can improve flexibility of business processes. These semantic rules come from business rules. So new semantic rules added must guarantee consistency of data dependency rule between each task. The following algorithm for restructuring to parallel control structure is given.

First, let TB_{i-1} and TB_i be two tasks or control blocks in path, which is denoted as path=<cn_{i-1}, TB_{i-1}, cn_i, TB_i, cn_{i+1}>. Predecessor(n) and Successor(n) denote predecessor and successor of node n respectively, where $n \in TN$ or $n \in CN$. Type(cn) and Type(Block(cn)) $\in$ {SEQ, AND-Split, AND-Join, AND-Split, AND-Join, LOOP } denote type of cn and Block(cn) respectively. Let type(cn_i)=SEQ where cn_i is involved in the path.

Algorithm 1 CreateParallel (TB_{i-1}, TB_i)
Input:TB_{i-1}, TB_i
Output: control block b
Begin
b:=∅
Predecessor(TB_{i-1}) := cn
Predecessor(TB_i) := cn
Type(cn):= AND_Split
Successor (TBi-1) := cn'
Successor (TBi) := cn'
Type(cn'):= AND_Join
b := Block(cn)
return b
End

3.4 Selective control structure

To change sequential order of two tasks in critical order to selective can still cut average execution time. If this modification happens in a selective control block, a second branch is only created, and if happens in a parallel or iterative control block, a new selective control block is nested in the control block. Changing to selective control structure must keep consistency of data dependency rule between tasks also. The algorithm given below is to restructure selective control structure.

Algorithm 1 CreateParallel (TB_{i-1}, TB_i)
Input:TB_{i-1}, TB_i
Output: control block b
Begin
T':= ∅
Predecessor(TBi) :=cn
Predecessor(T'):=cn
Type(cn):= OR_Split
cn':= OR_Join
Successor (TB_i)) := cn'
Successor (TB_{i-1}) := cn'
Type(Successor (TB_{i+1})) := SEQ
Type(Successor (T')) := SEQ
b := Block(cn)
return b

4 Critical Path Optimization Method

Through mining data dependency relationship between tasks, we can change workflow control structures and shorten the critical path in a workflow schema until the execution time of the path can not be cut any more. The resulting schema is optimum under effect of current set of data dependency rules. The more semantic rules are mined, the more probability to optimize workflow schema by restructuring it. Based on the critical path theory mentioned in section 3.1 , changing more tasks without data dependency in workflow schema to selective or parallel structure can shorten execution time of the critical path.

Let DR be a set of data dependency rules, $D=\{D_1,\ldots,D_n\}$ be set of all tasks' data where $D_i \in D$ is output data set of task T_i. Algorithm 3 begins to deal with all data dependency $P_{ij}(d_p,\ldots,d_q) \rightarrow T_i.d_{ij}$ in DR, where $d_{ij} \in D_i, Din_{ij}=\{d_p,\ldots,d_q\}$, $Din_{ij} \cap D_i=\varnothing$.

Algorithm 3 Optimization(T_i)
Input: Task node T_i
Output: a set of control rule CR
Begin
While DR≠∅
For $P_{ij} \in$ DR do

If $D_{i-1} \cap Din_{ij} = \varnothing$ and $Out(t_{i-1}) \cap Out(t_i) = \varnothing$ then

CreateParallel(T_{i-1}, T_i)

else

CreateAlternative (T_{i-1}, T_i)

End for

CR:= CR $\cup$ { P_{ij} }

DR:= DR- { P_{ij} }

End while

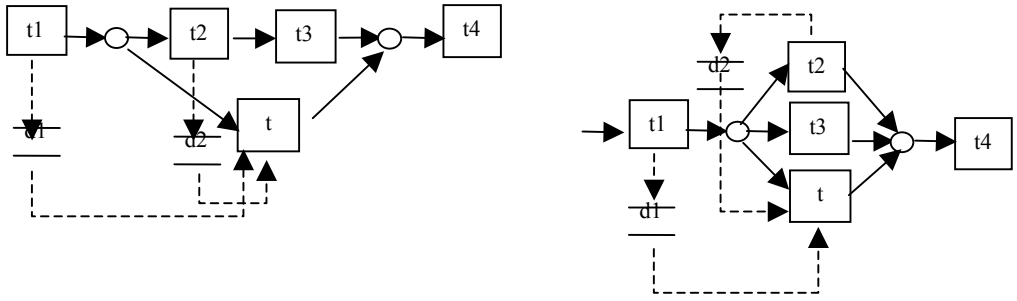
Return DR

End

The statement "if" in algorithm 3 above judges data dependency and data set of written between two neighbouring tasks, which can guarantee satisfying rule 4 and 5. The algorithm can also deal with two neighbouring control blocks to validate data dependency, in which circumstances the granularity is enlarged to control block from task so that control blocks can be combined, see algorithm 1 and 2.

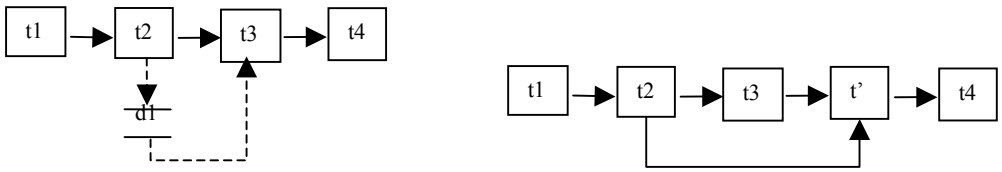
A parallel control structure can be created because no any data dependency between task t2 and t3 in Figure 2 (a) (we adopt expression of workflow schema, see Chang(2002), here we don not give unnecessary details). Data item d1and d2 are input data of task t. d2 may be an invalid data when t begins to execute, therefore restructuring to Figure 2(b) can not happen for not meeting rule 4. In Figure 3(a), there exists data dependency between task t2 and t3, which can be converted to control rule, so a selective branch is added and a new control block is created. It is noted that null task node t' is used to "shortcut" t3. This treatment method can be compatible with the circumstances of nested by other control blocks and meet rule 1, 2 and 3. The example in section 5 shows the soundness of control structure.

After computed by algorithm 3, workflow schema restructured shows a new partial order of different tasks. However the new business logic maybe doesn't match with business custom of enterprises. So some partial orders should be kept artificially when executing algorithm 3.



（a） （b）

Figure 2. Creating parallel control structure



（a） （b）

Figure 3. Creating selective control structure

5 Application and Example

The proposed approach has been applied to the development of our CERP system for a large number of different manufacture enterprises in some cities of China. In development phase of ERP software system and at run time, there are more possibilities of restructuring a workflow schema along with more semantic rules found by domain experts or developers. In order to validate our approach, we give equipment repair processes as an example in most manufacture enterprises. The tasks involved in the process and their meanings are seen in table 1. The content of business process is introduced as following. When some equipment does not work, a requisition is summated from workshop to Department of Equipment Management (DEM). For a small fault, maintenance man of workshop repairs it by himself. If he cannot fix up the machine, another requisition needs to be summated to DEM again and the engineer of DEM completes the business process of repair which involves assigning task, drawing part or dissipative material and repairing. If the failure part is found not to be resumed, another requisition needs to be summated to DEM once more. The repair task is relegated to others out of the enterprise.

Depending on our experience, we assume that the service requests arrive with rate $\lambda=4$. There are three different paths in the workflow Figure 4 when we do not consider outmost iteration structure. Let the sequence of tasks t1, t2, t7, t8 be called *path 1*, the sequence of task t1, t3, t5, t6, t7 and t8 be called *path 2*, the sequence of task t1, t4, t7 and t8 be called *path 3,* and probability of each branch of the selective structure which locates between t1 and t2, t3, t4 is $\alpha1=0.4$, $\alpha2=0.4$ and $\alpha3=0.2$ respectively. Service rate μ of each task is given in Table 1.

Table 1. Service rates ($\mu=1/t$ tis service time)

Tasks	Content	Service Rate μ(s)
t1	Summit requisition for repairing	10
t2	Draw part or material	5
t3	Assign task for dispatching person	10
t4	Relegate repair to other enterprise	8
t5	Assign task for computing part or material	8
t6	Draw part or material from storage	6
t7	Fix up ,check and accept	10
t8	Collect data of repair	8

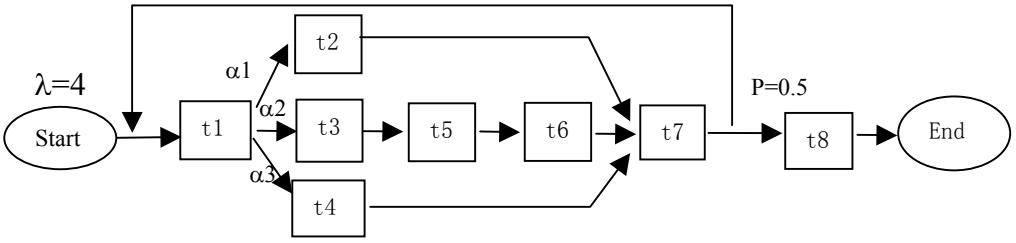


Figure 4. Business process of repair a equipment

The following is the process of computing critical path.

Step 1: Compute average execution time of each branch of the selective structure.

$W_2=1/\mu_2-\lambda_2=1/5-\alpha_1\times\lambda=0.294$s

$W_{356}=1/(\mu_3-\lambda_3)+1/(\mu_5-\lambda_5)+1/(\mu_6-\lambda_6)=1/(10-\alpha_2\times\lambda)+1/(8-\alpha_2\times\lambda)+1/(6-\alpha_2\times\lambda)=0.502$s

$W_4=1/\mu_4-\lambda_4=1/8-\alpha_3\times\lambda=0.139$s

The maximum is $W_{356,}$, then the path with longest average execution time in the selective structure is t3, t5 and t6.

Step 2: Compute average execution time of the iterative structure.

$W_1=1/p\mu_1-\lambda_1=1$; $W_7=1/p\mu_7-\lambda_7=1$

$W_{13567}=W_1+1/(p\mu_3-\alpha_2\times\lambda)+1/(p\mu_5-\alpha_2\times\lambda)+1/(p\mu_6-\alpha_2\times\lambda)+W_7=3.425$s

Step 3: Compute average execution time of the whole process.

$W_8=1/\mu_8-\lambda_8=1/8-4=0.25$s, because task t8 has already been in critical path.

The finial average execution time of critical path is: $W=W_{13567}+W_8=2.752$s, so we get the critical path is t1, t3, t5, t6, t7 and t8.

Then, in order to optimize critical path, semantic rules should be mined first. According to our experience in some manufacture enterprises and analysis to process of equipment repair, we find the fact that some parts are repaired by being relegated to other company. The property of the part should be provided as soon as possible. So the well-timed position is when task t7 is completed for the first execution. The semantic rule provided are described as *IsNotConsign (t7.PartID)→t3.PartID.* A selective branch can be added after task t1 following algorithm 3. In addition, there is not any data dependency between task t3 and t5, so the sequential order can be converted to parallel. Figure 5 shows workflow schema after conversion.

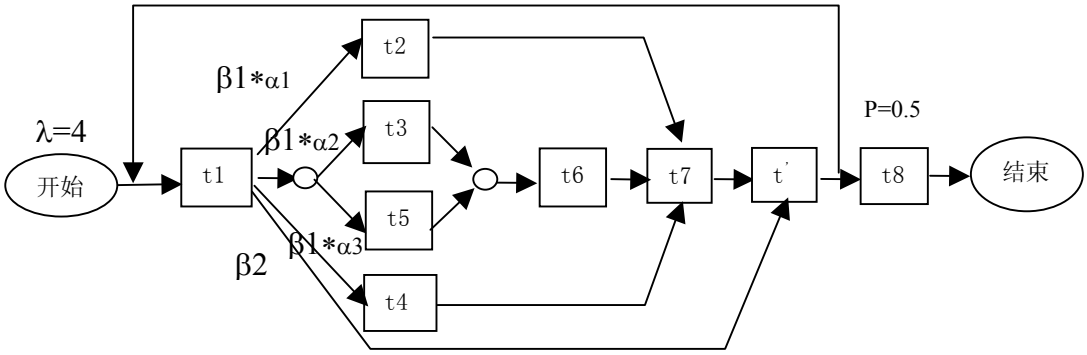


Figure 5. Workflow schema after converting

The following is the process of computing critical path. Let λ=4,α1=0.4, α2=0.4, α3=0.3,β1=0.5, β2=0.5.

Step 1: Computing the average execution time of the innermost parallel structure.

$W_{35}=\mathrm{MAX}(1/(\mu_3-\lambda_3),1/(\mu_5-\lambda_5))=\mathrm{MAX}(1/(\mu_3-\beta1*\alpha2*\lambda),1/(\mu_5-\beta1*\alpha2*\lambda))=0.139$s, task t5 is in critical path.

Step 2: Computing W_{56}.

$W_{56}=1/(\mu_5-\lambda_5)+1/(\mu_6-\lambda_6)=1/(\mu_5-\beta1*\alpha2*\lambda),1/(\mu_6-\beta1*\alpha2*\lambda)=0.331$s

Step 3: Computing W_{2564}.

$W_{2564}=\mathrm{MAX}(1/(\mu_2-\lambda_2),W_{56},1/(\mu_4-\lambda_4))=\mathrm{MAX}(1/(\mu_2-\beta1*\alpha1*\lambda),W_{56},1/(\mu_4-\beta1*\alpha3*\lambda))=0.331$s

Step 4: Computing $W_{1567t'}$

$W_{1567t}=1/(\mu_1-\lambda_1)+W_{56}+1/(\mu_7-\lambda_7))=1/(\mu_1-\beta1*\lambda)+W_{56}+1/(\mu_7-\beta1*\lambda))=0.581$s.

Here the execution time of null task t' is zero, then let $1/\mu\to\infty, W_{t'}\to0$

Step 5: Computing average execution time of iteration structure.

$W_{1567}=1/(p*\mu_1-\lambda)+1/(p*\mu_5-\beta1*\alpha2*\lambda)+1/(p*\mu_6-\beta1*\alpha2*\lambda)+1/(p*\mu_7-\beta1*\lambda)=2.1$s.

The finial average execution time of critical path is: $W'=W_{1567}+W_8=2.35$s, so we get the critical path is t1, t5, t6, t7, t', t8.

By comparing the workflow schema Figure 5 with Figure 4, in optimized critical path, because the order of

task t3 and t5 are changed to parallel structure, the average execution time begin to cut from t1 to t5 (here t3 does not belong to critical path, so its execution time is zero) in Figure 6. At the same time, adding selective branch can result in a reducing probability of all branches, so that the average execution times of t6 and t7 reduce accordingly.

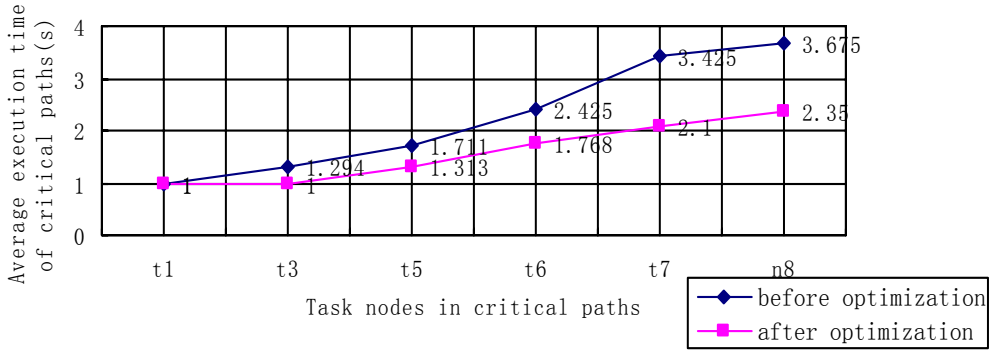


Figure 6. Two critical paths' execution times

6 Conclusion

The primary benefit of critical path method is to help domain experts or models of developers understand where they can most productively spend time modifying their models. However that is not enough only to locate the position of critical path because they still lack an effective approach to optimize it in former researches, such as PERT chat. Though some researchers attempt to optimize workflow model adopting the approach of PERT char, there are also some limitations to the technique in which selective and parallel control structure are not supported, as mentioned above. We show that it is operable to restructure a workflow schema by mine more semantic rules, i.e. data dependency rule called in this paper. The changed model is called optimization in current data dependency rule set.

The paper proposes an approach to optimize workflow construct by shortening average execution time of workflow critical path which has the longest average execution time from the start activity to the end activity. We improve traditional optimization method of critical path, and add selective and parallel control structures by mining more data dependency relationship between tasks. So the approach is also called a conversion from data dependency rule to control dependency.

In addition, this conversion should keep rationality of business process logically, conform to business custom of enterprises, and be handled under human intervention. It is dogmatic to convert all mined semantic rules in order to optimize workflow graph. Unreasonable business logic must result in unpractical business process which can be applied.

Acknowledgments

The authors also wish to acknowledge the financial support of the National High-Tech. R&D Program for CIMS, China, Grant 2003AA4Z3210, 2003AA413023 and 2002AA413310.and another cooperative project from European Union.

Correspondence to:
Haibo Li
Centre of Intelligent Computing of Enterprises
School of Computer Science and Engineering
Harbin Institute of Technology
Harbin, Heilongjiang 150080, China
Working: School of Engineer
Northeast Agriculture University
Harbin, Heilongjiang 150030, China
Telephone: 86-451-86412664
Email: lihaibo@hit.edu.cn

References

[1] Cormen TH, Leiserson CE, Rivest RL. Introduction to Algorithms, 1994 (MIT Press, Cambridge, MA).
[2] Duk-Ho Chang, Jin Hyun Son, Myoung Ho Kim. Critical pPath iIdentification in the cContext of a wWorkflow. Information and Software Technology 2002;44(7):405-17.
[3] Gomory RE, Hu TC. Multi-terminal network flows. SIAM 1961;9:551-70.

[4] Hagen C, Alonso G. Flexible exception handling in the oOpera process support system. Proceedings of the Eighteenth IEEE International Conference on Distributed Computing Systems. 1998.

[5] Heinl P. Exceptions during workflow execution. Proceedings of the Sixth International Conference on Extending Database Technology, workshop on workflow management, Valencia, Spain, 1998.

[6] Hollingsworth D. The workflow reference model. Workflow Management Coalition. 1995.

[7] Hollingsworth JK. Critical Path Profiling of Message Passing and Shared-Memory Programs. Transactions on Parallel and Distributed Systems 1998;9(10).

[8] Hribar MR, Taylor VE, Boyce DE. Implementing parallel shortest path for parallel transportation applications, Parallel Computing 2001;31:1537-68.

[9] Hu Yunquan. Operational research foundation and application (in Chinese). Harbin Institute of Technology Press , Harbin China 1998:147-8.

[10] Jeffrey K. Hollingsworth JK,. Member, IEEE Computer Society. Critical Path Profiling of Message Passing and Shared-Memory Programs. IEEE Transactions on Parallel and Distributed Systems. 1998;9(10):1029-40.

[11] Jin Hyun Son , Myoung Ho Kim. Analyzing the critical path for the well-formed workflow schema. Proceedings of the Seventh International Conference on Database Systems for Advanced Applications (DASFAA .01), 2001.

[12] Jin Hyun Son, Jung Sun Kim, Myoung Ho Kim. Extracting the Workflow Critical Path from the Extended Well-Formed Workflow Schema. Journal of Computer and System Sciences 2005;70(1):86-106.

[13] Lee KT, Abraham JA. Critical path identification and delay tests of dynamic circuits. Proceedings of International Test Conference. 1999:421-30.

[14] Lenstra JK, Rinnooy Kan AHG, Brucker P. Complexity of Machine Scheduling Problems. Ann Discrete Math 1977;1:343-62.

[15] Meajil A, El-Ghazawi T, Sterling T. Characterizing and representing workloads for parallel computer architectures. Journal of Systems Architecture. 2000;46(1).

[16] Minkyu Lee, Dongsoo Han, Jaeyong Shim. Set-based access conflict analysis of concurrent workflow definition. Information Processing Letters 2001;80:189–94.

[17] Oh SK, Son JH, Lee YJ, Kim MH. An efficient method for allocating workflow tasks to improve the performance of distributed workflows. International Conference on Computer Science and Informatics. 2000:445-8.

[18] Pozewaunig H, Eder J, Liebhart W. ePERT: Extending PERT fro workflow management systems. The First European Symposium in ADBIS 1997:217-24.

[19] Shui Lam, Ravi Sethi. Worst Case Analysis of Two Scheduling Algorithms. SIAM J Comput 1977;6(3):518-36.

[20] Singh G, Zinder Y. Worst-case performance of critical path type algorithms. International Transactions in Operational Research. 2000;7:383-99.

[21] Singh G. Performance of critical path type algorithms for scheduling on parallel processors, Operations Research Letters 2001;29:17-30.

[22] Son JH, Kim MH. Improving the performance of time-constrained workflow processing. Journal of Systems and Software 2001;58(3):211-9.

[23] van der Aalst WMP. The application of Petri nets to workflow management. Journal of Circuits, Systems and Computers 1998;8(1):21-66.

[24] Workflow Management Coalition: Process Definition Interchange, Document No. WfMC-TC-1016-P, 1999.