Task Partition for Function Tree According to Innovative Functional Reasoning

Yiming Tang, Xiaoping Liu
School of Computer & Information, Hefei University of Technology, Hefei, P.R. China
tym109@sina.com; lxp@hfut.edu.cn

Abstract

Task partition is a critical problem of collaborative conceptual design. Aiming at the shortage that current task partition methods don't accord to innovative functional reasoning that is the kernel process and essence embodiment of conceptual design, a new task partition method for function tree according to innovative functional reasoning is proposed. To begin with, the concept of task module is proposed as the basic unit of task pre-partition before task module pre-partition algorithm that generates initial task modules is put forward. Furthermore, based on analyzing and gaining three basic constraint relationships: including, independent and innovative conflict relationship, M-TPG (Module-driven Task Precedence Graph) generated from the former two relationships is introduced into this field as the basic tool of task management. Lastly, task partition algorithm based on M-TPG with three basic principles for innovation is given. The new method is derived from clustering of microcosmic elements of innovative functional reasoning and conflicted task modules, which can obtain deeper cohesiveness and more reasonable results.

Keywords: Innovative Functional Reasoning, Collaborative Conceptual Design, Task Partition, Function Tree, Innovative Conflicts.

1. Introduction

Conceptual design is a vitally significant phase in whole product design process, since design decisions in this phase approximately account for more than 75% of final product costs[1]. The concept of function penetrates every phase in conceptual design[2]. Accordingly, the kernel process of conceptual design is functional reasoning[3], which starts from function requirements and establishes mapping relationship of functions, behaviors, structures and so on to generate and evaluate solutions to specified design problems.

Innovation is the essence and main competing power of conceptual design[3,4], mostly directly determining success or failure of designed products. It is, therefore, vitally important to establish innovative functional reasoning mechanism here.

However, innovation is the most difficult part in conceptual design embodying that it is great possibility that designers can't achieve satisfied innovative solutions with lots of time and money. An ordinary conceptual design task may become a vitally complex task for difficulty of innovation. Furthermore, conceptual design issues are highly interdisciplinary, often involving collaboration from customers, designers, and engineers; and as experienced by many manufactories, not only the resources and equipment, but also the knowledge and expertise are geographically distributed[5]. Hence collaborative conceptual design is of great necessity for conceptual design due to the vital complexity of innovation and current imperative requirement of conceptual design.

Because the essence idea of collaboration is to divide general design task into several design sub-tasks, how to carry through reasonable task partition is a key problem of collaborative conceptual design.

As for collaborative design, there are some papers related to task partition. Paper[6] carries out task partition from combination of structures and functions possessed by many components. Paper[7] works on it conforming to functions, structures and special components, but doesn't consider it in detail. We[8] accord to structures with TPG under background of Cooperative Template[9]. However, there are few literatures for task partition in collaborative conceptual design. Authors[10] divide general task conforming to components first and then divide each obtained sub-task according to its features in collaborative conceptual design, but innovation isn't taken into consideration in this work. Similarly, authors[11] approach the problem based on components and then features which mostly mean shape.

To sum up, structures or functions are the dominant elements for task partition, and innovation is seldom taken into account in task partition, which is embodied in three aspects. First, task partition is carried out at the initial phase of conceptual design where structure information is fuzzy, undetermined; hence task partition according to structures is unreasonable. Second, some task partition methods according to functions are mainly based on structures, always defaulting that

978 -1-4244-1651-6/08/$25.00 © 2008 IEEE

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 15, 2008 at 02:25 from IEEE Xplore. Restrictions apply.

structures in the main have already existed, so it violates spirit of conceptual design in which structures should be eventually achieved by functional reasoning. In addition, functions are the starting point of functional reasoning. Functions are therefore improper standards for it. Lastly, it is the most critical that any literature haven't been found apparently considering innovation in task partition. As a result, it is really demanded for task partition to conform to innovative functional reasoning in collaborative conceptual design.

For difficulty and hugeness of the subject, function tree with and/or decomposition (function tree for short) is chosen as the object of task partition because function tree is a typical, popular representation in conceptual design[12] proved as an effective tool with the ability to support innovative functional reasoning. Paper[13] emphasizes the vital status of innovation in product model of conceptual design, and puts forward Domain Structure Template supporting innovation realized by function tree. Combining with TRIZ theory[14], a hot innovative theory nowadays, paper[15] proposes function means tree based on and/or decomposition as a tool for innovation in conceptual design. We[16] bring forward a method of expanding similarity functions in function tree to enlarge innovative solving space in conceptual design.

There isn't any literature to be found about task partition for function tree in collaborative conceptual design up to now. Aiming at above problems, a new task partition method for function tree is proposed from innovative function reasoning in collaborative conceptual design.

2. Task module pre-partition

2.1. Introduction of task modules

In function tree, its top node is function requirement and its lower nodes are elaboration or realization of upper nodes, and hence its expanding process embodies functional reasoning. As it is of high frequency that many leaf nodes are the same nodes in a function tree, it is feasible to let one basic variable node represent these repeated leaf nodes.

Definition 1 (basic variable nodes): In a tree, let its leaf nodes with the same identity get the same number, and define these leaf nodes as basic variable nodes.

Function tree is able to be regarded as an application of Boolean algebra theory[4]. Consequently any gate node (or tree) can be denoted by Boolean expression composed of basic variable node and relationship of "$\wedge$" (denoting "and") and "$\vee$" (denoting "or"). Then any sub-tree is determined by its basic variable nodes. So the concept of task modules can be achieved by these basic variable nodes. In fact it is natural for repeated elements to cluster together into a sub-task.

Let $H(G_1)$ denote the tree with top node G_1, and let $M(G_1)$ be set of all task modules of $H(G_1)$. Moreover, let $U(G_1)$ stand for subscript set of all of gates in $H(G_1)$, meanwhile let $V(G_1)$ denote subscript set of all of basic variable nodes.

Definition 2(task module): In $H(G_1)$, if a sub-tree $H(G_i)$ doesn't include the same basic variable existing in other part in $H(G_1)$, then $H(G_i)$ is called as a task module, denoted by $H(G_i) \in M(G_1)$.

2.2 Task module identification

Definition 3(traveling time for a node, $T(x_i)$): When travel $H(G_1)$, define traveling time for the first node is 1, then that for the next nodes will be 2, 3, 4, … in turn, in term of the order of traveling.

In order to identify modules, leftist depth-first travel is adopted, with the order of gate-children-gate that means traveling gate node first, and then traveling its children, and lastly traveling it again. As shown in figure 1, $T(x_1) = 2$, $T(G_1) = 1$ or 8.

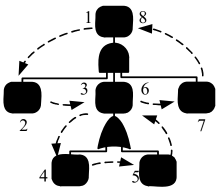

Figure1. Travel order of function tree

Definition 4(minimal / maximal traveling time for a gate node, $T_{min}(G_i)$ / $T_{max}(G_i)$): Since any gate node G_i must be visited twice, its first traveling time is defined as $T_{min}(G_i)$, and the second one is $T_{max}(G_i)$. In Figure 1, $T_{min}(G_2) = 3$, $T_{max}(G_2) = 6$.

Definition 5(minimal / maximal traveling time for a basic variable node, $T_{min}(x_i)$ / $T_{max}(x_i)$): As a basic variable node x_i may be visited for many times, then the first traveling time for it is defined as $T_{min}(x_i)$, and the last one is $T_{max}(x_i)$. In Figure 1, $T_{min}(x_1)= T_{max}(x_1) =2$, but $T_{min}(x_2)= 4$ and $T_{max}(x_2) = 7$.

Definition 6(minimal / maximal traveling time for a tree, $T_{min}(H(G_1))$ / $T_{max}(H(G_1)))$: When tree $H(G_1)$ includes more than one node, define: $T_{min}(H(G_1)) = \min\{\min\{T_{min}(G_i)|i \in U(G_1)-\{1\}\}, \min\{T_{min}(x_i)|i \in V(G1)\}\}$; $T_{max}(H(G_1)) = \max\{\max\{T_{max}(G_i)|i \in U(G_1) - \{1\}\}, \max\{T_{max}(x_i)|i \in V(G1)\}\}$. Else if the tree only includes a leaf node, then define: $T_{min}(H(x_i))= T_{max}(H(x_i))=0$. In Figure 1, $T_{min}(H(G_1))=2$, $T_{max}(H(G_1)) = 7$; $T_{min}(H(G_2)) = 4$, $T_{max}(H(G_2)) = 7$.

Definition 7 (task module partition): The process of getting all of the modules in a tree is defined as task module partition.

Theorem 1 For a gate G_i in tree $H(G_1)$, $H(G_i) \in M(G_1)$ if and only if $T_{min}(G_i) = T_{min}(H(G_i))-1$ and $T_{max}(G_i) = T_{max}(H(G_i))+1$.

Proof (Necessity) If $H(G_i) \in M(G_1)$, then all the gates nodes and basic variable nodes in $H(G_i)$ never

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 15, 2008 at 02:25 from IEEE Xplore. Restrictions apply.

exist at any other place in $H(G_1)$. With leftist depth-first travel, $T_{min}(H(G_i))$ and $T_{max}(H(G_i))$ are only related to $H(G_i)$, and according to traveling order: G_i—offspring of G_i—G_i, clearly get: $T_{min}(G_i) = T_{min}(H(G_i))-1$ and $T_{max}(G_i) = T_{max}(H(G_i))+1$.

(Sufficiency) Suppose $H(G_i) \notin M(G_1)$, then there is at least a basic variable node x_j $(j \in V(G_i))$ where x_j exists at other place in $H(G_1)$, just as well assume x_j is visited before sub-tree $H(G_i)$, then get: $T_{min}(x_j) < T_{min}(G_i) < T_{max}(G_i)$. And, have: $T_{min}(G_i) = T_{min}(H(G_i)) -1 < T_{min}(H(G_i)) \leq \min\{T_{min}(x_k)|k \in V(G_i)\} \leq T_{min}(x_j)$. So $T_{min}(G_i) < T_{min}(x_j)$ which is contradiction, get $H(G_i) \in M(G_1)$. So, the proof is completed. ■

2.3 Task module pre-partition algorithm

Based on Theorem 1, task module pre-partition algorithm is put forward. Before task module pre-partition, it is necessary to travel the total tree to get Tmin and Tmax of all nodes. As Algorithm 1, pTopNode denotes top node pointer of current tree, and nTminInChild is $\min\{T(x_i)\}$ (x_i denotes any node in current tree), and nTmaxInChild is $\max\{T(x_i)\}$. Meanwhile Tmin stands for T_{min} of top node, and Tmax denotes T_{max}, and module list is used for reserving all recognized modules, which is NULL initially. As for its return value, if pTopNode is gate node G_i, then return value nTmin will be $T_{min}(H(G_i))$ and nTmax will be $T_{max}(H(G_i))$. Else if pTopNode is basic variable node x_i, then nTmin will be $T_{min}(x_i)$ and nTmax will be $T_{max}(x_i)$.

Algorithm 1 (task module pre-partition algorithm)

```
TravelForModule(pTopNode,&nTmin,&nTmax)
{if (pTopNode != NULL){
    if (gate == pTopNode.type){        //gate node
        nTmin = Tmin + 1; nTmax = Tmax -1;
        pChild = pTopNode->pFirstChild;
        while (pChild != NULL){
            nTminInChild = nTmaxInChild = 0;
            TravelForModule(pChild, nTminInChild,
            nTmaxInChild);
            if (nTminInChild < nTmin)
                nTmin = nTminInChild;
            if (nTmaxInChild > nTmax)
                nTmax = nTmaxInChild;
            pChild = pChild->pNextBrother;}
        if ((Tmin +1==nTmin)&&(Tmax -1==nTmax)){
            tree with top node pTopNode is a module;
            get the module and insert it to module list;} }
    else{                               //basic variable node
        nTmin = Tmin; nTmax = Tmax;}}}
```

3. Task partition based on M-TPG

3.1 Primary classification for constraint relationships of task modules

Following the initial task modules generated from Algorithm 1, it must be achieved clearly for the general designer to catch the order of executing these modules, which is based on achieving constraint relationships of these task modules. In a word, it is necessary to analysis and gain the basic kinds of constraint relationships.

Suppose the module set is $M(G_1)=\{M_a|a \in X\}$ where X denotes subscript set of modules. Let M_1 be the general module and hence $M_1 \in M(G_1)$. For modules M_i, $M_j \in M(G_1)$ corresponding to tree $H(G_p)$ and $H(G_q)$ in turn:

Definition 8 (Including relationship): If $H(G_q)$ totally includes $H(G_p)$, define this relationship as including relationship, denoted by $M_i \subset M_j$. In it, if there is M_k such that $M_i \subset M_k \subset M_j$, define the relationship as indirect including relationship, else define it as direct including relationship.

Definition 9 (Independent relationship): If $H(G_q)$ has no intersect with $H(G_p)$, define the relationship as independent relationship.

Lemma 1[17] Graph G is a tree, if and only if G has no loops and any two distinct nodes of G are joined by one and only one path.

Theorem 2 $\forall$ M_i, $M_j \in M(G_1)$, the relationship between M_i and M_j is either including relationship or independent relationship.

Proof Suppose M_i, M_j corresponds with $H(G_p)$, $H(G_q)$ in turn, And then either $H(G_p)$ has no intersection with $H(G_q)$, or it has intersection. Now proof: when $H(G_p)$ intersects with $H(G_q)$, it is sure to be including relationship, that is: it is impossible to be partly intersecting relationship.

Suppose $H(G_p)$ intersects $H(G_q)$ in part, then $\exists$ x_r such that x_r is the common offspring of G_p and G_q. That is: not only G_p but also G_q has a path to x_r. It is impossible that one of the paths can include other one, then there are two different paths from the top node G_1 to x_r: G_1--G_p--x_r and G_1—G_q--x_r, by Lemma 1, get that there is contradiction, so it is impossible that $H(G_p)$ intersects $H(G_q)$ in part.

In conclusion, there is either no-intersection or totally including relationship in $H(G_p)$, $H(G_q)$. Correspondingly, as for module relationship, get: there is either independent or including relationship. So, the proof is completed. ■

3.2 M-TPG from task modules

For the sake of solving problems of mutual dependence and constraint of task modules in collaborative design, we have proposed TPG[8] (Task Precedence Graph) that has already encapsulated a complete set of task collaboration mechanism proved as an excellent method. Collaborative conceptual design can be regard as a phase of collaborative design with many common characteristics such as precedence

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 15, 2008 at 02:25 from IEEE Xplore. Restrictions apply.

relationship and independent relationship, and it is thus feasible to introduced TPG into the field of collaborative conceptual design where precedence relationship is embodied in including relationship. But, TPG generated from modules with independent and including relationship (Module-driven TPG, M-TPG for short) is a special graph, as Theorem 3.

Theorem 3 M-TPG from $M(G_1)$ is a tree.

Proof If M-TPG from $M(G_1)$ has a loop, there is a sequence in $M(G_1)$: $M_{p_1}, M_{p_2}, \cdots M_{p_n}$ where $M_{p_1} \neq M_{p_2} \neq \cdots \neq M_{p_n}$ and $M_{p_1} \subset M_{p_2} \subset \cdots \subset M_{p_n} \subset M_{p_1}$. From the latter, get $M_{p_1} = M_{p_2} = \cdots = M_{p_n}$, which is a contradiction. Thus, M-TPG from $M(G_1)$ has no loops.

Suppose there are two different paths P_1, P_2 between modules M_i and M_j in $M(G_1)$, then relationship between M_i and M_j is including relationship by Theorem 2, so just as well suppose $M_i \subset M_j$. Now, current state can be separated into two possibilities: (1) M_j directly includes M_i in one path, and there is M_k where $M_i \subset M_k \subset M_j$ in another path; (2) there is M_m where $M_i \subset M_m \subset M_j$ in one path, and in another path, there is M_n where $M_i \subset M_n \subset M_j$, and M_m is independent from M_n. It is clear that contradiction exists in both (1) and (2). By this token, get that any two distinct nodes of M- TPG are joined by one and only one path. So, by Lemma 1, the proof is completed. ■

By the ancestor-offspring relationship of top nodes of trees corresponding to these modules, it is easy to accomplish algorithm of M-TPG generation.

3.3 Innovation embodiment: innovative conflict relationship

The former content is mostly concerned about normal functional reasoning and here innovation factor is introduced and integrated in it. The main research object of innovation is innovative conflict treatment[14,15,16], which is the most difficult part in the whole innovative process[14] differing from conflicts of collaboration issues. Due to the vital importance and difficulty of innovative conflicts, if the modules with innovative conflicts are united into a unit to treat as a whole at the early phase, cohesiveness of sub-task will be great enhanced and so task partition will be more reasonable. As a result, there is the third relationship: innovative conflict relationship. Because it hasn't direct relation with M-TPG, M-TPG is able to be generated before considering innovative conflict.

Function tree is based on Boolean algebra theory, then let $x_i = 1$ or $G_j = 1$ denote realization of basic variable node x_i or gate node G_j, and let $x_i = 0$ or $G_j = 0$ denote no realization of them. The whole function tree can be denoted by Boolean expression $a(G_1)$, and it is easy to get its disjunctive expression composed of only basic variable nodes, denoted as $\underset{1\le j\le m}{\vee}(\underset{i\in F_j}{\wedge} x_i)$ where m is quantity of disjunctive item and F_j is subscript set of basic variable nodes in disjunctive item j.

Definition 10 (Innovative conflict matrix): For basic variable set P, define innovative conflict matrix as $C(P) = (c_{ij})_{n\times n}$, where n=|P| and ($x_i$,$x_j$ are any basic variable nodes) $c_{ij} = \begin{cases} 1, & it's\ impossible\ that\ x_i = x_j = 1 \\ 0, & otherwise \end{cases}$.

In actual design process, innovative conflict matrix is given in advance. If $c_{ij} = 1$, denoting innovative conflict exists between x_i, x_j, then $x_i \wedge x_j = 0$.

Definition 11 (Innovative conflict for Boolean expression): For any Boolean expression a, define mapping $\varphi(a) = \begin{cases} 1, & It's\ impossible\ that\ a = 1 \\ 0, & otherwise \end{cases}$ as innovative conflict mapping.

Theorem 4 For innovative conflict matrix C(P) and Boolean expression a= $\underset{1\le j\le m}{\vee}(\underset{i\in F_j}{\wedge} x_i)\ (\forall i \in P)$:

(1) $\varphi(a) = 1 \Leftrightarrow \forall j, \exists (i,k) \in F_j \times F_j : c_{ik} = 1$;

(2) $\varphi(a) = 0 \Leftrightarrow \exists j, \forall (i,k) \in F_j \times F_j : c_{ik} = 0$.

Proof It is easy to proof the theorem. ■

Definition 12 (Innovative conflict relationship) For modules M_i, $M_j \in M(G_1)$ and given $C(V(G_1))$, M_i, M_j corresponds with $H(G_p)$, $H(G_q)$ in turn; and G_p, G_q have the same parent node G_r. If G_r is "and" gate and $\varphi(G_p \wedge G_q)$ =1, or G_r is "or" gate and $\varphi(G_p \vee G_q)$ =1, define the relationship as innovative conflict relationship, denoted by $\gamma(M_i, M_j)$=1. Else $\gamma(M_i, M_j)$ =0. γ is called as conflict mapping for modules.

3.4 Task partition algorithm

Definition 13 (Innovative conflict group) Define group $\aleph(M_i, M_j) = < M_i, M_j, G_s, G_t, G_r >$ as innovative conflict group where modules M_i, M_j accord with $H(G_s)$, $H(G_t)$ in turn; and G_r is the parent of G_s and G_t.

Algorithm 2 (task partition algorithm)

```
{ TravelForModule(pTopNode,&nTmin,&nTmax);
     //get initial task modules M(G1) of tree H(G1)
  Get M-TPG from M(G1);
  Initiate(Q); //let queue Q be NULL
  Push(Q,m_pTopOfMTPG);
     //m_pTopOfMTPG denotes top node of M-TPG
  if (Empty(Q)) return;
  while (!Empty(Q)){      //queue Q isn't null
     Mp = Pop(Q);
     if (Mp has more than one child){
        m_pCGList = NULL;
           //m_pCGList denote conflict group list
        for (any two child modules M1,M2 of Mp) {
           //according trees are H(Gs),H(Gt) in turn
          if ((Gs->parent==Gs->parent) && ($\gamma$(M_i, M_j)
           ==1))
```

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 15, 2008 at 02:25 from IEEE Xplore. Restrictions apply.

```
            Add(m_pCGList,M1,M2);
              //add ℵ (M1, M2) into m_pCGList
        }
        for (each conflict group without intersection
        with other groups)                        //①
            unite its two modules into a new module;
        for ((intersecting conflict groups)&&(their
        related modules have small weight in Mp))  //②
              unite them into a new module;
      }// if (Mp has more than one child){
      for (each child modules Mc of Mp) Push(Q,Mc);
    }//while (!Empty(Q)){
    for (each leaf module m_leaf in M-TPG) {
      if ((m_leaf is small)&&(m_leaf is fully designed))
        delete m_leaf IN M-TPG;}
    Add all leaf node modules into sub-task list;
    output sub-task list;}
```

Remark 1 ①, ② concerns conflict treatment (supposing trees according with conflicted modules have the same parent node).

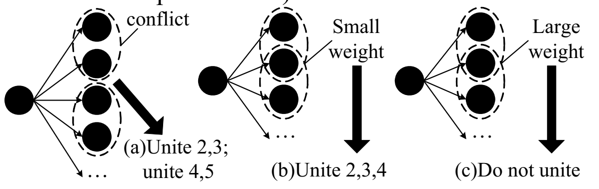

Figure2. Unite modules in conflict in M-TPG

Principle 1 For conflicts without intersects, unite conflicted modules, shown in Figure 2(a).

Principle 2 For intersecting conflicts, if related modules have small weight in its parent module, unite them into a new module. It is clearly inappropriate to unite any two modules and a small weight means it is more a local phenomenon than a comprehensive one, so unite them, shown in Figure 2(b).

Principle 3 For intersecting conflicts, if related modules have large weight in its parent module, don't unite them. It is a comprehensive phenomenon, and then treat with these conflicts as a whole is similar to treat with their parent module, moreover it is necessary to deal with parent module, so don't unite them, shown in Figure 2(c).

Remark 2 Concerning method of modules uniting, suppose united modules are $M_1, \dots, M_k$ with according trees $H(G_{p1}), \dots, H(G_{pk})$ in turn with the same parent gate G_r; their parent module is M_p with top node G_p.

If G_r has only k children, execute: if G_r is G_p, offspring modules of $M_1, \dots, M_k$ are changed to child modules of M_p; else add new module M_N with according tree $H(G_r)$ and change any other offspring modules (of M_p) included by M_N to child modules of M_N.

Else add new middle gate node G_N as parent of G_{p1}, G_{p2}, …, G_{pk} in $H(G_1)$, and add new child module M_N with according tree $H(G_N)$ as child module of M_p, and change any other offspring modules (of M_p) included by M_N to child modules of M_N.

Delete $M_1, \dots, M_k$ in M-TPG.

Remark 3 Task allocation, coordination and composition will be carried through after task partition. Especially with accomplishment and submission of some sub-task, function tree is partly changed and basic variable nodes are altered, so need to update innovative conflict matrix and M-TPG in task composition. Then general designer executes the next task partition for the next task allocation. At last, final task composition is aimed at last module (general task).

4. Experiment and analysis

As shown in Figure 3, it is the conceptual design of multi-function travel teacup. The Initial conflict matrix is given in advance providing the basic variable nodes related to conflicts as Figure 4.

Table 1 shows $T(x_i)$ in part and provides Tmin and Tmax for Algorithm 1, and table 2 shows part final result and gets initial task modules $M(G_1)$. Figure 5 gives initial M-TPG from $M(G_1)$.

In initial M-TPG in Figure 5, modules 1,6,3 aren't united by Principle 3, and by Principle 1, unite modules 8,10 into module 14, and original child module 9 of module 10 is changed to child module of module 14.

Here give innovative conflict detection of modules 1,6 with according trees $H(G_4)$, $H(G_6)$ in turn, and their parent gate node is "and" node G_2 , so:

$$\begin{aligned}
&G_4 \wedge G_6 = (G_9 \vee G_{10}) \wedge (G_{12} \wedge G_{13})\\
&= [(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_4 \wedge x_3)] \wedge [(x_8 \vee x_9) \wedge (x_{10} \wedge x_{11})]\\
&= [(x_1 \wedge x_2 \wedge x_3 \wedge x_9 \wedge x_{10} \wedge x_{11}) \vee (x_1 \wedge x_2 \wedge x_3 \wedge x_8 \wedge x_{10} \wedge x_{11})\\
&\quad \vee (x_1 \wedge x_4 \wedge x_3 \wedge x_9 \wedge x_{10} \wedge x_{11}) \vee (x_1 \wedge x_4 \wedge x_3 \wedge x_8 \wedge x_{10} \wedge x_{11})
\end{aligned}$$

From conflict matrix, $c_{3,10}=1$, then $x_3 \wedge x_{10}=0$, so $\varphi(G_4 \wedge G_6) = 1$ and there is a conflict in module 4,6.

At last, Get sub-task module list "1, 5, 2, 9, 11".

From the content mentioned above and experiment results, conclusions for proposed new method is given.

(1) Current other methods mainly get sub-tasks from main structures or functions, which is difficult to grasp interior relation. However, this method is based on leaf nodes of function tree, and hence sub-tasks are derived from clustering of microcosmic elements of functional reasoning, and it is able to get deeper cohesiveness.

(2) By uniting conflicted task modules to a new task module in terms of three principles for innovation, and due to function tree's characteristic of tying up function reasoning, this method accords to innovative functional reasoning of which current other methods is in lack.

(3) M-TPG has already encapsulated a complete set of task collaboration mechanism which is proved as an excellent method in collaborative design. Moreover, M-TPG without loops leads to easier task treatment. And clearly it adopts gradual composition mechanism.

However, there are some shortages here. The mechanism for innovative conflicts is still simple, and is only effective for some special situation; and after accomplishment of some sub-tasks, it needs the next

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 15, 2008 at 02:25 from IEEE Xplore. Restrictions apply.

task partition, but in fact concerns about more factors and even change of whole task module layout. All of these need further research.

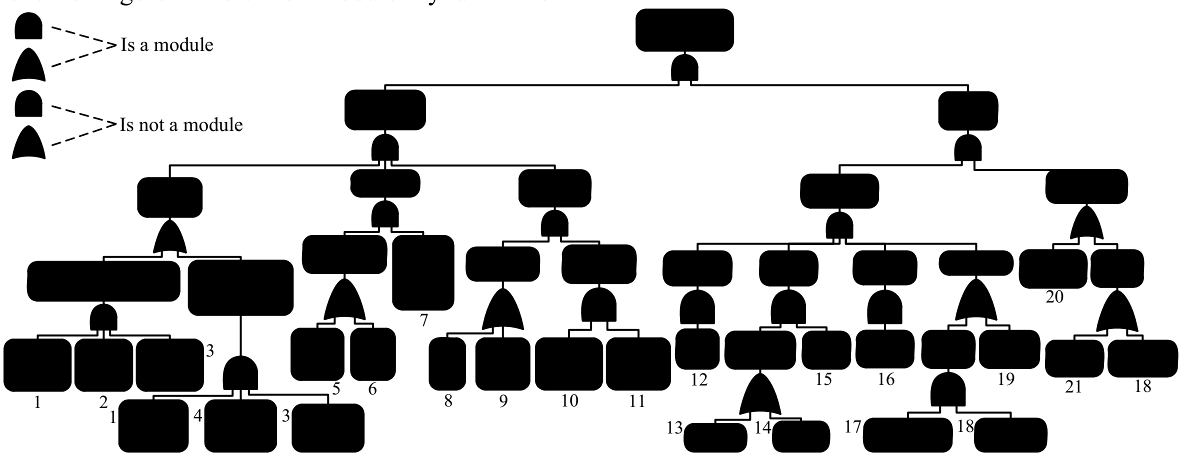

Figure3. Conceptual design of multi-function travel teacup

Table1. Node traveling for pre-partition

time	node	time	node	time	node	time	node
1	G1	6	2	11	4	16	G11
2	G2	7	3	12	3	…	…
3	G4	8	G9	13	G10	62	G8
4	G9	9	G10	14	G4	63	G3
5	1	10	1	15	G5	64	G1

Table2. Gate node traveling and module statistics

Gate(G*)	5	6	7	8	9	10	11
$T_{min}(G_i)$	15	22	34	56	4	9	16
$T_{max}(G_i)$	21	31	55	62	8	13	19
$T_{min}(H(G_i))$	16	23	35	51	5	5	17
$T_{max}(H(G_i))$	20	30	60	61	12	12	18
Module?	Y	Y	N	N	N	N	Y

$$
\begin{array}{c|ccccccccccccccc}
 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 12 & 15 & 17 & 18 & 19 & 21 \\
\hline
2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{array}
$$

Figure4. Initial conflict matrix

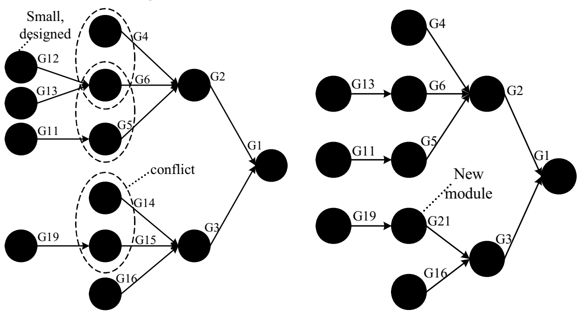

Figure5. Initial (left) and worked (right) M-TPG

5. Conclusions

A new task module partition algorithm based on M-TPG is put forward. It is derived from clustering of microcosmic elements of innovative functional reasoning, which can get the deep cohesiveness. And, tying up function reasoning, the method especially includes consideration of innovative conflicts. Due to clustering conflicted task modules and function tree's characteristic of tying up function reasoning, this method accords to innovative functional reasoning. What's more, it is based on M-TPG without loops, already encapsulating a complete set of task collaboration mechanism proved as an excellent, effective method. And, more research will be carried through about mature mechanism for innovative conflicts and repetitious task partition.

Acknowledgement

The authors are grateful to all the members in VCC (Visualization & Cooperative Computing) division, especially Qiang Lu, Jinting Lu, Jin Qin. This paper is supported by National Nature Science Fund of China (No: 60673028) and Projects of International Cooperation and Exchanges of National Nature Science Fund of China and America (No: 70710045).

References

[1] W. Hsu and B. Liu, "Conceptual design: issues and challenges", *Computer-Aided Design*, 2000, 32(14), 849-850.

[2] Yasushi Umeda and Tetsuo Tomiyama, "Functional reasoning in design", *IEEE Expert*, 1997, 12 (2), 42–48.

[3] Amaresh Chakrabarti and Thomas P. Bligh, "A scheme for functional reasoning in conceptual design", *Design Studies*, 2001, 22(6), 493–517.

[4] Xiaoping Liu, Yiming Tang, Jin Qin and Qiang Lu, "A New Function-Solving Method Based on Extended Function Matrix in Conceptual Design", *Journal of Computer-Aided Design & Computer Graphics*, 2007, 19(12), 1610-1617(In Chinese).

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 15, 2008 at 02:25 from IEEE Xplore. Restrictions apply.

[5] Lihui Wang, Weiming Shen, et al, "Collaborative conceptual design—state of the art and future trends", *Computer-Aided Design*, 2002, 34(13), 981-996.

[6] Xiuli Meng, "Study on the collaborated task management of machine tool product", *Journal of Machine Design*, 2006, 23(7), 30-33 (In Chinese).

[7] Haobo Qiu, Xinyu Shao, et al, "Coordination and Optimization Oriented Integrated Process Management of Collaborative Design", *Machine Tool & Hydraulics*, 2007, 35(2), 36-38 (In Chinese).

[8] Xiaoping Liu, Hui Shi, et al, "Visual Task-driven Based on Task Precedence Graph for Collaborative Design". *Proceeding of CSCWD'2007*, Melbourne, Australia, April 26-28, 2007, pp. 246-251.

[9] Xiaoping Liu, Hui Shi, et al, "Cooperative template mechanism for cooperative design", *Computer Supported Cooperative Work in Design II, Lecture Notes in Computer Science*, Vol.3865, Springer-Verlag, Berlin Heidelberg New York, 2006, 102-111.

[10] Lirong Qiu, Hong Liu and Liping Gao, "A Multi-agent System Supporting Creativity in Conceptual Design", *Proceeding of CSCWD'2004*, Xiamen ,China, May, 26-28, 2004, pp. 362-370.

[11] Shijian Luo, Shouqian Sun, Yunhe Pan, et al, "A Case Study on Product Collaborative Conceptual Design Technology Based on User Implicit Knowledge", *Proceeding of CSCWD'2004*, Xiamen ,China, May, 26-28, 2004, pp. 191-196.

[12] Wynne Hsu and Irene M. Y. Woon, "Current research in the conceptual design of mechanical products". *Computer-Aided Design*, 1998, 30(5), 377-389.

[13] Huijun Song, Zhihang Lin, "Product model supported by axiomatic design in conceptual design", *Journal of Computer-Aided Design & Computer Graphics*, 2002, 14(7), 632-636 (In Chinese).

[14] Xiaoping Liu, Yiming Tang and Jin Qin, "The research and implementation of a computer-aided innovation prototype based on TRIZ", *Journal of engineering graphics*, 2007, 28(6), 6-11 (In Chinese).

[15] Tan Runhua, "The conceptual design of a fast clasping mechanism based on function means tree and TRIZ", http://www.triz-journal.com/archives/2000/10/f/index.htm.

[16] Xiaoping Liu, Jin Qin and Yiming Tang. "An innovative function-tree building method based on similarity theory and extension theory", *Proceeding of CAIDCD'06*, Hangzhou, China, Nov. 17-19, 2006, pp. 199-204.

[17] Michael Townsend, "Discrete Mathematics: Applied Combinatorics and Graph Theory", The Benjamin/Cummings Publishing Company, Inc, 1987

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 15, 2008 at 02:25 from IEEE Xplore. Restrictions apply.