ICED13
INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED13
19-22 AUGUST 2013, SUNGKYUNKWAN UNIVERSITY, SEOUL, KOREA

A MULTI-AGENT BASED FRAMEWORK FOR MULTI-DISCIPLINARY CONCEPTUAL DESIGN SYNTHESIS

Yong CHEN, Zelin LIU, Jian HUANG, Zhinan ZHANG
Shanghai Jiao Tong University, People's Republic of China

ABSTRACT
It is encouraged that designers should explore in wide multi-disciplinary solution spaces for finding novel and promising principle solutions to desired functions during conceptual design. However, due to lack of sufficient multi-disciplinary knowledge, it is often difficult for human designers to explore in such a wide multi-disciplinary solution space. Therefore, it is desirable to have an automated conceptual design system, which can automatically achieve multi-disciplinary conceptual design synthesis for human designers. Based on the multi-disciplinary conceptual synthesis approach developed before, this paper proposes a multi-agent based framework for achieving the conceptual design synthesis of multi-disciplinary systems. The roles of different kinds of agents and the collaborative mechanisms among them are elaborated. A conceptual design case is employed to illustrate how the multi-agent-based design synthesis framework works, which also shows that the proposed multi-agent based framework can achieve better efficiency than our previous approach.

Keywords: conceptual design, computational design synthesis, multi-agent systems, functional modelling, principle solution

Contact:
Prof. Yong Chen
Shanghai Jiao Tong University
School of Mechanical Engineering
Shanghai
200240
People's Republic of China
mechenyong@sjtu.edu.cn

1 INTRODUCTION

Conceptual design is responsible for generating suitable PSs (abbreviation of Principle Solutions) for desired functions (Pahl & Beitz, 1996). Here, PS means the basic physical mechanism of a system for achieving a desired function. During conceptual design, designers are often encouraged to explore in wide multi-disciplinary solution spaces to find novel and promising PSs for desired functions (Pahl & Beitz, 1996). However, it is often very difficult for human designers to fulfill such multi-disciplinary exploration tasks, since they are often cultivated in a single major and therefore have very limited multi-disciplinary PS knowledge (Chen et al, 2010; Chen et al, 2012). Furthermore, even if human designers could have the knowledge about various multi-disciplinary PSs, it would be extremely hard for them to exhaustively synthesize such multi-disciplinary PSs together for finding most promising combinatorial PSs for desired functions, which would be extremely time-consuming and labor-intensive. Such an issue is becoming increasingly worse in an era with rapid technical progresses, since it is becoming more and more difficult for human designers to catch up with those progresses and employ the up-to-date PSs to create novel technical systems.

A possible solution to the above issue is to develop an Automated Conceptual Design (abbreviated as ACD later) system, so that various PSs stored in the knowledge base can be automatically synthesized together for achieving multi-disciplinary design synthesis. Here, a basic premise is that known PSs in various disciplines can be abstracted from existing systems as building blocks for future design synthesis. Based on this idea, we have developed an energy flow-based synthesis approach for achieving the conceptual design of multi-disciplinary systems, with some successful results (Chen et al, 2010; Chen et al, 2012). In contrast with the domain-specific conceptual design synthesis approaches developed before (e.g., Chakrabarti & Bligh, 1996; Chiou & Kota, 1999; Campbell et al, 2000; Chen et al, 2006; etc.), our conceptual design synthesis approach has a salient feature, i.e. its knowledge representation model is independent of any specific disciplines and therefore can be employed to achieve multi-disciplinary conceptual design. However, as the multi-disciplinary PS knowledge base becomes larger and larger, which now contains about 500 PSs from various disciplines, a new issue has come up, i.e., it is becoming more and more inefficient. In addition, the ACD system can even run into failure, when the time required for executing a design synthesis process in a large PS space exceeds the response time limit allowed by the computer. Therefore, a more efficient and robust creative synthesis approach should be developed to achieve conceptual design synthesis in a large multi-disciplinary solution space.

This paper reports a multi-agent based approach that can achieve multi-disciplinary creative synthesis with higher efficiency and better robustness. Here, multi-agent means that a group of loosely connected autonomous agents act in an environment for achieving a common goal (Lander, 1997). This paper is organized as follows. Section 2 briefly reviews the related work. Section 3 introduces our functional knowledge representation approach developed before (Chen et al, 2010; Chen et al, 2012), which is also the foundation of our work to be reported in this paper. Section 4 proposes the multi-agent-based conceptual design synthesis framework, where the roles of the related agents and their collaborative synthesis mechanisms are developed. Section 5 then illustrates how the proposed framework is implemented. Section 6 presents a multi-disciplinary creative synthesis case for illustration. Finally, Section 7 concludes this paper.

2 LITERATURE ANALYSIS

Existing Computer-Aided Conceptual Design (abbreviated as CACD later) researches primarily fall into two categories, i.e. reuse-oriented conceptual design approaches and synthesis-oriented conceptual design approaches. A reuse-oriented conceptual design system is primarily responsible for retrieving from its knowledge base some possible design solutions for a design problem at hand, so that a designer can select a suitable solution for reuse, which may further involve some adaptation work. CACD systems of such a kind are for examples, FBS modeler (Umeda et al, 1996) and EKRITIK (Prabhakar and Goel, 1998). A drawback of such systems consists in that they are prone to design bias towards familiar PSs, since the related design synthesis process largely depends on designers' knowledge and experiences. Therefore, the reuse-oriented conceptual design approaches are ineligible for generating creative solutions for multi-disciplinary systems.

The synthesis-oriented CACD systems can generate PSs for desired functions through synthesizing (i.e. combining) some basic elements or known PSs together (Chen et al, 2012). For examples, Welch and

Dixon (1994) have developed a bond graph-based approach for generating PSs of electro-mechanical devices, using some basic electro-mechanical elements (e.g. resistance, capacitance, etc); Chakrabarti and Bligh (1996) have developed a function-based approach for generating PSs of mechanical devices, using some basic mechanical elements (e.g. rod, lever, etc); Chiou & Kota (1999) propose a motional matrix-decomposing approach for generating combinatorial mechanisms with some basic mechanisms (e.g. crank-slider, rack-pinion, etc.); we develop a morphological matrix-based synthesis approach for achieving conceptual design of mechanisms (Chen *et al*. 2006). Such synthesis-oriented CACD systems share a common feature, i.e. they rely on domain-specific knowledge representation models and reasoning algorithms. For example, although the vector, *[kind, orientation, sense, magnitude, position]*, used in Chakrabarti and Bligh (1996) is suitable for representing the input/output flow of a mechanical device, it cannot be employed to represent the functional flows of an electronics component (e.g. an inverter for transforming direct current to alternating current). As a result, such synthesis-oriented CACD systems also cannot be employed to achieve multi-disciplinary conceptual design.

In addition, it should be pointed out that there are two CACD approaches that can probably be used for multi-disciplinary conceptual design. One is the physical principle-chaining approach developed by Zavbi & Duhovnik (2001), which employs some basic physical principles to generate some solution concepts for a desired function. A primary problem with this approach consists in that its output design results are the combinations of physical principles, which are probably difficult for human designers to embody them with concrete components. The other is the physical effect-chaining approach (Chakrabarti, 2004) for achieving the conceptual synthesis of multi-disciplinary sensors. A problem with this approach consists in that it merely employs the input-output variable pair to represent the function of a PS component, which can lead to vague functional representation. For example, since the slider-crank mechanism can achieve the function of transforming rotation into translation, an input-output variable pair, $<\omega, \upsilon>$, could then be employed to represent its function; however, this would make a CACD system difficult to differentiate the slider-crank mechanism from other mechanisms (e.g. the rack-pinion mechanism, the cam-follower mechanism, etc.) regarding their functions.

Therefore, there are still no other suitable approaches for achieving multi-disciplinary conceptual design, except the approach we have recently developed (Chen et al, 2012). Due to the inefficiency issue mentioned before, this paper will therefore develop a multi-agent based approach for achieving the functional synthesis of multi-disciplinary systems. Note that although there have been many multi-agent based design systems (e.g. Zhao et al, 2001; Liu et al, 2004; etc.), most of them are not for conceptual design synthesis. An exception is the A-design approach (Campbell *et al*, 2003), which combines multi-objective optimization, multi-agent systems and automated design synthesis to achieve the conceptual design of electromechanical systems. However, the A-design approach is also based on a domain-specific functional representation model, and therefore cannot be easily adapted to multi-disciplinary conceptual design.

3 THE FUNCTIONAL KNOWLEDGE REPRESENTATION

In order to develop a multi-agent based conceptual design synthesis system, there must be a formal approach for representing the functional knowledge that can be achieved by known PSs. Therefore, we will briefly introduce our functional knowledge representation approach here, which has been developed in our recent research (Chen et al, 2012).

Just like many previous researches, our functional knowledge representation approach also takes a flow transformation-based philosophy. Therefore, a critical issue is how to represent various flows. Our flow representation model is composed of two sections, i.e. a flow name representation and an attribute-value representation. To avoid ambiguity, a flow name representation employs a standard physical variable name to denote a flow. For example, we can employ the variable name, *angular_velocity*, to denote the oral flow, *rotation.* The attribute-value representation is employed to describe the detailed features of a flow. Since the flows in different disciplines may have different sets of detailed features, the attribute-value representation thus allows domain experts to flexibly customize different sets of attributes and values for different kinds of flows. For example, a rotational flow can be represented as, "Angular_Velocity {*magnitude*: Constant; *axial_orientation*: X; *direction*: Clockwise; *intermittence*: Continuous}", while an electrical flow can be as, "Electrical_Current {*type*: DC; *magnitude*: Constant; *intermittence*: Continuous; *direction*: Positive}". This is different from the previous vector-based flow representations, e.g. in Chakrabarti & Bligh (1996) and in Campbell et al

(2000), where all flows must be represented with the same group of features. Therefore, our flow representation approach is more flexible and reasonable than the previous flow representation approaches, when used for multi-disciplinary conceptual design.

Since the PSs in different disciplines may deal with different kinds of transformations, how to provide a domain-independent approach for representing the functional knowledge of known PSs is not an easy task. For example, a slider-crank mechanism not only deals with the flow name transformation (i.e. from *angular_velocity* to *linear_velocity*), but also some other feature transformations, e.g. the motional orientation transformation (e.g. from X to Y), the direction transformation (e.g. from the Clockwise rotation to to-and-fro translation), etc. To address this issue, we have developed an integrated model to represent the functional knowledge of known PSs, as shown in Figure 1. The integrated functional knowledge representation model comprises four parts, i.e. the input-output flow name pair, the attribute constraints on the input flow, the attribute constraints on the output flow, and the flow attributes-association rules.

Function
  Principle_Solution
  Input_Output_Flow_Name_Pair
    (*InFlowName*, *OutFlowName*)
  Constraints_On_Input_Flow_Nane
    {(*Attr_Name*, *Attr_Values*)}
  Constraints_On_Output_Flow_Name
    {(*Attr_Name*, *Attr_Values*)}
  Production_Rules
    {[Rule ID:
      IF (*Precondition*) THEN (*Action*)]
    }

(a) The functional representation schema of a PS

Function
  Principle_Solution: *Slider_Crank_Mechanism*;
  Input_Output_Flow_Name_Pair
    (*Angular_Velocity*, *Linear_Velocity*)
  Constraints_On_Input_Flow_Name
  { (*magnitude*, "Constant" || "Variable"),
    (*axial_orientation*, "X" || "Y" || "Z"), …}
  Constraints_On_Output_Flow_Name
  { (*magnitude*, "Variable"), …}
  Production_Rules
  { [Rule 1: IF (*axial_orientation*, "X")
      THEN (*orientation*, "Y" || "Z")]; ...
    [Rule m: IF (*intermittence*, "Continuous")
      THEN (*intermittence*, "Continuous"] ; …}

(b) A functional representation case of a PS

Figure 1. A general schema for representing functional knowledge

The input-output flow name pair, (I_F, O_F), is employed to represent the transformation of the flow's name from input to output achieved by a known PS, where I_F and O_F denote the name of the input flow and that of the output flow, respectively. For example, the flow name transformation achieved by a slider-crank mechanism can be represented as (*angular_velocity*, *linear_velocity*).

The attribute constraints on the input flow of a PS prescribe what values the input flow's attributes should have, which later can be used by a CACD system to judge whether or not the PS can act on an input flow during the design synthesis process. For example, a DC motor requires that the *type* attribute of its input electrical_current flow should be with a value of "DC"; if the *type* attribute of an electrical_current flow has a value of "AC", it is then impossible to employ a DC motor to act on the electrical_current flow.

The attribute constraints on the output flow denote what values the output flow's attributes may have. Such output constraints can be employed to determine the values of the output flow's attributes during the design synthesis process. For example, since a DC motor has an output constraint on the *direction* attribute of its output rotation, i.e. (*direction*, Clockwise||Anti_Clockwise), it is then possible for a CACD system to determine the allowable values of the rotation flow's direction attribute as "Clockwise" or "Anti_Clockwise", while another value, *Bi_Direction*, of the direction attribute can then be ruled out for the output rotation flow.

The attributes-association rules (also called the attributes-mapping rules) of a known PS are employed here to represent the association relations between the values of the input flow's attributes and those of the output flow's attributes. Similar to a production-rule, an attribute-association rule has both a precondition part and an action part. A prediction part denotes what value the related input flow's attribute should have in order for the rule to be activated, while the action part indicates what value(s) the corresponding attribute of the output flow can have after the rule is executed. For example, given the PS, *the slider-crank mechanism*, the perpendicular association relations between the *axial_orientation* attribute of the input rotation flow and the *orientation* attribute of the output translation flow can be represented as: {IF (axial_orientation = "X"), THEN (orientation = "Y||Z"); IF (axial_orientation = "Y"), THEN (orientation = "X||Z"); etc}. Compared with the previous CACD researches, an advantage of the attributes-association rule representation consists in that it allows domain experts to input such complex associations into knowledge base. In previous CACD researches, such attributes-association knowledge was often represented with domain-specific models in the past, e.g. the motional matrix in Chiou & Kota (1999), which must be hard-coded in a CACD system and could not be accessed by domain experts.

4 A MULTI-AGENT BASED DESIGN SYNTHESIS FRAMEWORK

To develop a multi-agent based conceptual design synthesis framework, it is necessary to analyze the problems with our multi-disciplinary conceptual design approach developed before, so that the crux of the inefficiency issue can be found out. Therefore, we will provide a problem analysis at first, prior to introducing our multi-agent based design synthesis framework.

4.1 The problem analysis

Prior to analyzing the inefficiency issue mentioned before, we should briefly introduce our multi-disciplinary design synthesis approach (Chen et al, 2010; Chen et al, 2012), which will also serve as a computational foundation for our multi-agent based conceptual design approach. To facilitate explanation, the agent concept is employed here to elaborate our approach. Our multi-disciplinary design synthesis system developed before can be regarded as an Intelligent Functional Synthesis Agent (abbreviated as IFSA later). It is assumed that there is a knowledge base that contains many known PSs from various disciplines.

IFSA takes a SSA (abbreviation of Sensing-Selection-Action) mechanism to achieve multi-disciplinary conceptual design synthesis, as shown in Figure 2. After a designer inputs a desired function, IFSA will generate all possible input flows and then release them to the environment. It then begins to sense the flows in its environment and retrieves an environmental flow that has not been explored before. Thereafter, it will search its PS knowledge base for all suitable PSs that can act on the environmental flow, based on their condition knowledge (i.e. the input flow names and the constraints on the input flow). With all suitable PSs, IFSA will then employ their action knowledge (i.e. the output flow names, the constraints on the output flow and the attribute-association rules) to act on the environmental flow, with a result of some output flows. For details of the action process, interested readers can find it in Chen et al (2010). Such output flows will then be released into IFSA's environment as new input flows for further design synthesis exploration. IFSA will repeat the above sensing-selection-action process, until all flows in its environment have been explored (i.e. processed by all suitable PSs) and the corresponding search depths haven't exceeded a preset search depth. Thereafter, IFSA will then verify all flows in the environment one by one to see if they can satisfy the functional requirements on the output flow. If a flow is identified as satisfying, IFSA then should trace the flow back to find its predecessor flows and the corresponding PSs that have been employed to generate such flows. Such PSs can then be chained together to form a combinatorial PS for achieving the desired function. Since there can be many satisfying flows in the environment, IFSA will then generate a combinatorial PS for each of them through such a backtracking process. Unlike the traditional state space approach that can merely search in a uniform state space, a major advantage of IFSA consists in that it can explore in a heterogeneous state space, which are formed by variables from various disciplines.

According to the above introduction, it can be found that there are at least four major problems that lead to the inefficiency issue. First, the entire conceptual design synthesis is a sequential process, which comprises multiple sub-processes that are sequentially executed. Second, each time IFSA can merely takes one environmental flow for design exploration, making that all other environmental flows have to be kept in the waiting line until the exploration of a previous flow has completed. Thirdly, each time IFSA can merely select one PS to act on an environmental flow, resulting in that all other PSs in the knowledge base have to wait until a previous PS has been processed. Finally, each time IFSA can merely trace one environmental flow back for generating a combinatorial PS, all other environmental flows also have to wait in line until a previous backtracking process has finished.

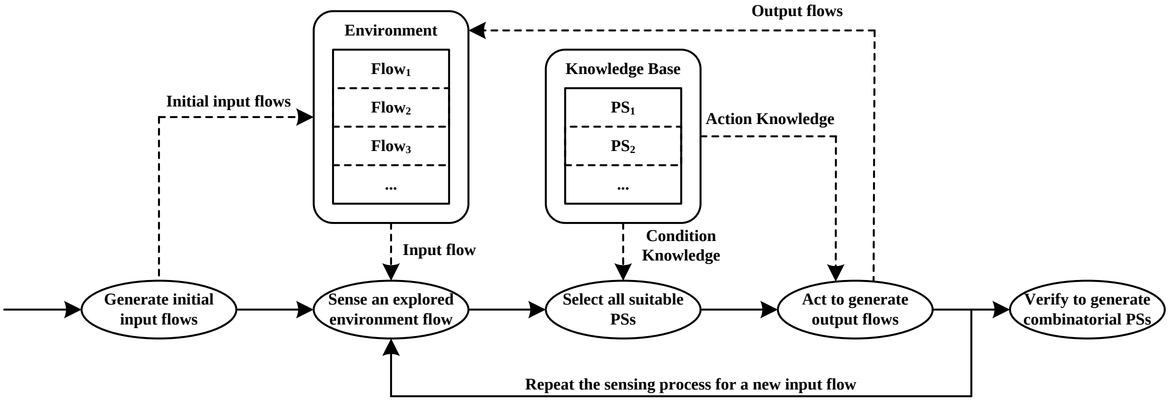

Figure 2. The sensing-selection-action mechanism for multi-disciplinary design synthesis

4.2 A multi-agent based design synthesis framework

According to the problems identified as above, a possible solution to the inefficiency issue can then be to develop a parallel computing framework to achieve conceptual design synthesis. Based on this idea, a multi-agent based conceptual design synthesis framework has been developed, as shown in Fig.3. This framework is primarily composed of two systems, i.e. the PS knowledge management system and the conceptual design synthesis system. Here, the PS knowledge management system is primarily composed of a knowledge management agent, and the conceptual design synthesis system is composed of a task management agent, multiple Sensing-Selection-Action (SSA) agents and some backtracking agents. The roles of such agents are as below.

- Knowledge management agent

The knowledge management agent has two major roles. One is for knowledge engineers to model PS knowledge with some standard user interfaces. The other is to allocate PSs into the sub-knowledge bases, which will be used by different SSA agents (as shown in Fig. 3). When allocating a PS to a sub-knowledge base, the knowledge management agent primarily considers two factors. One is that the PS quantity in each sub-knowledge bases should be in balance. The other is that function-similar PSs should be put into different sub-knowledge bases, so that they can be simultaneously used for design exploration, rather than stay in the waiting line.

- Task management agent

The task management agent has multiple roles. After a designer has input a desired function, the task management will then release the input flows into the common environmental flow pool. It is also responsible for activating SSA agents and backtracking agents, when it receives a conceptual design task. In addition, the task management agent is also responsible for monitoring the whole design synthesis process in case of failure.

- SSA agent

SSA (Sensing-Selection-Action) agents are the primary agents for achieving parallel design synthesis. Each SSA agent has its sub-knowledge base of PSs, and will independently execute the sensing-selection-action mechanism. A SSA agent first senses the common environmental pool for obtaining a batch of flows that it has not employed for design synthesis. Thereafter, it will execute its own sense-selection-action process. The SSA agent will then release the output flows into the common environmental flow pool for other SSA agents to continue the SSA process. After completing a SSA cycle, a SSA agent will then try to obtain from the common environmental pool some new environmental flows for further exploration. Since these agents simultaneously execute the SSA process, the efficiency of the design synthesis process can then be largely improved.

- Backtracking agent

A backtracking agent is primarily responsible for checking whether a flow in the common environmental pool can match the functional requirements on the output flow, and for tracing the design exploration path back to generate combinatorial PSs when a matching environmental flow is found. Note that a backtracking agent will not wait until all SSA agents have ceased their searching processes. Instead, it will begin to work, once the task management has released some input flows into the common environmental pool. Since there are usually multiple backtracking agents working at the same time, the efficiency of the backtracking process can also be largely improved.

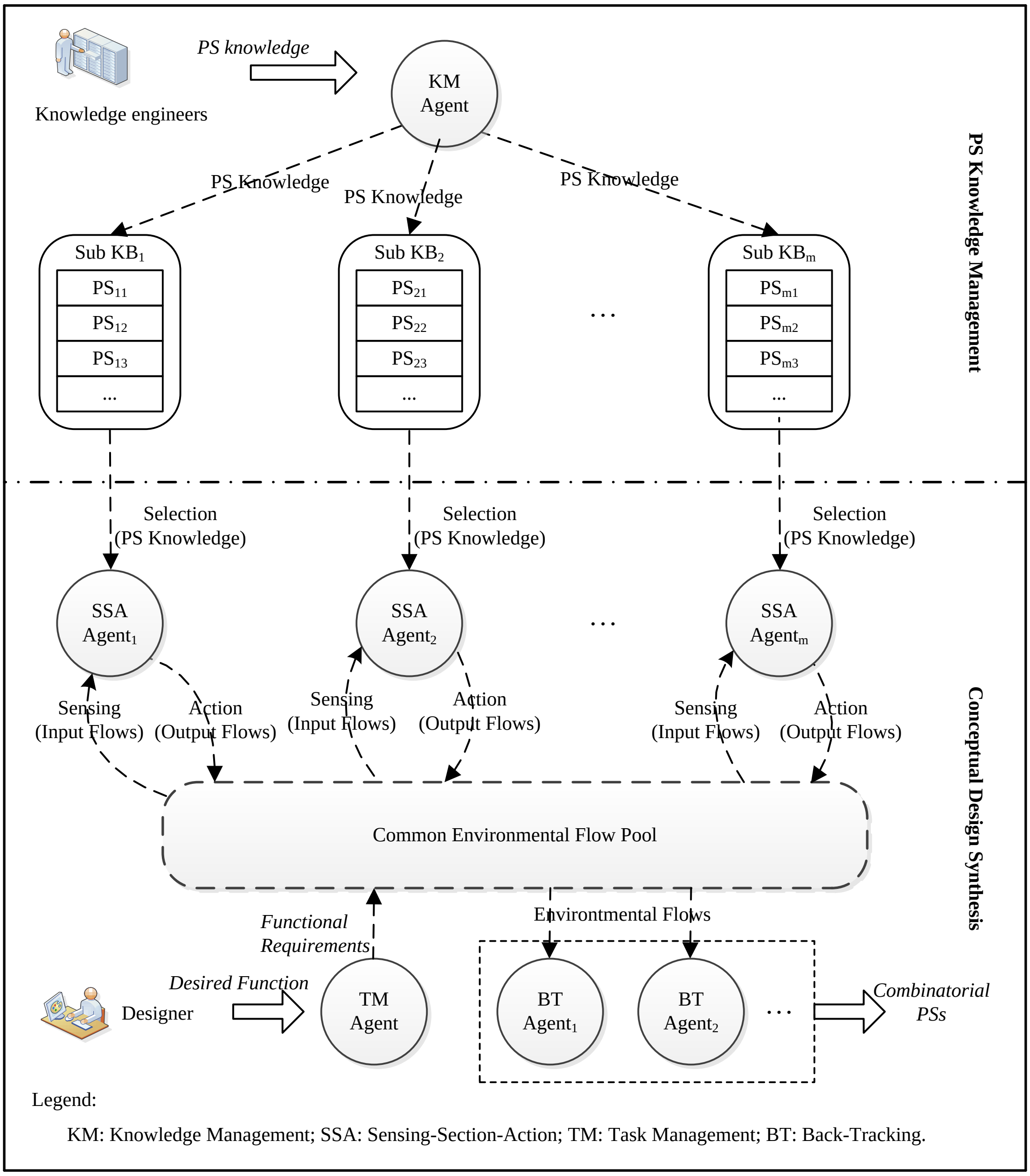

Figure 3. A multi-agent based framework for multi-disciplinary design synthesis

4.3 The multi-agent based collaborative mechanism

As mentioned before, there are four kinds of agents in the multi-agent framework. Since the knowledge management agent is primarily for managing and allocating PSs in the sub-knowledge bases, it will not interact with the three other kinds of agents during the design synthesis process. Therefore, the multi-agent collaborative synthesis here primarily deals with those among the task management agent, the SSA agents and the backtracking agents.

After a designer inputs a desired function, the task management agent will be first activated and will then construct a set of input flows according to the constraints on its input flow. Note that in this research, a desired function is flexibly represented as a set of constraints on the input and output flows (Chen *et al*, 2010). The task management agent will then release these input flows into the common environmental flow pool. The common environmental flow pool here can be conceptualized as a data group, $[id, id_{flow}, (id_{originAgent}, id_{originPS}), \{id_{SSAagent}\}, idepth, ibacktracked]$, where id_{flow} is the id (i.e. identification) reference to a flow, the combination, $(id_{originAgent}, id_{originPS})$, denotes the origin of the flow, i.e. which SSA agent has employed which PS to generate the flow, $\{id_{SSAagent}\}$ refers to a set of SSA agents that have executed the SSA processes with the current flow (Note that an environmental flow can be explored by different SSA agents), *idepth* is a search depth of the flow, and *ibacktracked* is a Boolean element for indicating whether the flow has been backtracked or not. Note that when an environmental flow is derived from a desired function, it doesn't have an origin.

After the task management agent releases some initial flows into the common environmental flow pool, it will then activate SSA agents for conceptual design synthesis. Each SSA agent will then begin to sense the common environmental flow pool. It will copy a batch of environmental flows that it has not processed before into its local environmental (i.e. input) flow pool. The local environmental flow pool can be conceptualized as a data group, $[id, id_{inflow}, id_{cpool}, iprocessed]$, where id_{inflow} is the id reference to an input flow in its local data base, id_{cpool} records the id of the flow in the common environmental flow pool, while *iprocessed* is a Boolean sign for indicating whether it has been explored by the SSA agent. Through the local SSA process, an SSA agent can then select suitable PSs from its own sub-knowledge base for each input flow, and then act on them one by one to generate output flows. Each SSA agent also has an output flow pool, which can be conceptualized as a data group, $[id, id_{outflow}, id_{inflow}, id_{PS}]$, where $id_{outflow}$ is the id of an output flow, id_{inflow} is the local id of the input flow that has been changed to the output flow, id_{PS} is the id of a PS that exerts an action to enable the flow change. When an SSA agent releases an output flow to the common environmental flow pool for further design synthesis, it will also release the related origin information of these flows. In addition, an SSA agent will also add a tag of its own id to the $\{id_{SSAagent}\}$ elements of the corresponding flows in the common environmental pool to indicate that these flows have been explored.

When the task management agent activates the SSA agents, it will also activate backtracking agents. Each backtracking agent will then independently obtain a flow from the common environmental pool and then check whether the flow can satisfy the functional requirements on the desired output flow. If a backtracking agent detects a satisfying flow, it will trace the flow back to find its predecessors and the PSs through the origin information of the related flows, with a result of a combinatorial PS. When a flow has been backtracked, its *ibacktracked* sign in the common environmental flow pool will be set as true. Note that a backtracking agent will not cease its work until the SSA agents have stopped and all flows in the common environmental pool have been backtracked.

5 AN ILLUSTRATIVE EXAMPLE

A toy design case, which has also been used in our previous research (Chen *et al*, 2012), is employed here to illustrate the multi-agent based conceptual design approach. The desired function for this toy can be declared as: "convert continuous or intermittent solar light into a to-and-fro sway motion". With the constraint-based functional representation approach (Chen et al, 2007; Chen *et al*, 2010), this desired function can be represented as a set of input flow constraints, i.e. Visual_Light {Stability: *constant* || *variable*; Intermittence: *continuous* || *intermittent*; Type: *hot_light*}, and a set of output flow constraints, i.e. Angular_Velocity {Stability: *variable*; Axial_orientation: *X*||*Y*||*Z*; Direction: *to-and-fro*; Intermittence: *continuous*}. The multi-agent based conceptual design synthesis process can then be illustrated as below.

At first, the task management agent, according to the input flow constraints, exhaustively constructs four initial input flows as, Visual_Light {Stability: *constant*; Intermittence: *continuous*; Type: *hot_light*}, Visual_Light {Stability: *variable*; Intermittence: *continuous*; Type: *hot_light*}, Visual_Light {Stability: *constant*; Intermittence: *intermittent*; Type: *hot_light*}, Visual_Light {Stability: *variable*; Intermittence: *intermittent*; Type: *hot_light*}. The task management agent then releases these initial input flows into the common environmental pool. Thereafter, the SSA agents and the backtracking agents are then activated.

To illustrate the multi-agent design synthesis process, it is assumed that there are two SSA agents. One SSA agent has a sub-knowledge base with 5 PSs, i.e. *Solr_Panel*, *AC_Motor*, *Spur_Gear_Pair*, *DC_To_AC_Inverter*, *Crank_Rocker*; the other also has a sub-knowledge base with 5 PSs, i.e. *DC_Motor*, *Crank_Slider*, *Electrical_Transformer*, *Fluorescent-Lamp*, *Rack_Pinion*. Some detailed explanations about these PSs can be found in (Chen et al, 2010). Through sensing the common environmental flow pool, both the first SSA agent and the second SSA agent can obtain the four environmental flows. For each obtained flow, the first SSA agent will then select suitable PSs for action. In this case, the *Solar_Panel* PS will be selected as a suitable PS for acting on them separately. As a result, four output flows can be generated by the first SSA agent, i.e. "Electrical_Current {Stability: *constant*; Intermittence: *continuous*; Direction: *positive*; Type: *DC*}", "Electrical_Current {Stability: *variable*; Intermittence: *continuous*; Direction: *positive*; Type: *DC*}", "Electrical_Current

{Stability: *constant*; Intermittence: *intermittent*; Direction: *positive*; Type: *DC*}", "Electrical_Current {Stability: *variable*; Intermittence: *intermittent*; Direction: *positive*; Type: *DC*}". For the details about the action process, interested readers can find them in (Chen et al, 2010). For the second SSA, there are no suitable PSs that can be selected to act on the above four flows, and therefore no output flows will be generated. After the output flows generated by the first SSA agent are released into the common environmental flow pool, the two agents each can then begin a new SSA cycle. In this cycle, the first agent can employ the DC_To_AC_Inverter PS to act on its new input DC flows, resulting in some alternating current flows generated, e.g. "Electrical_Current {Stability: *constant*; Intermittence: *continuous*; Direction: null; Type: *AC*}", while the second agent can employ the DC_Motor PS to act on its new input DC flows, resulting in some new rotation flows generated, e.g. Angular_Velocity {Stability: *constant*; Axial_orientation: *X*; direction: *clockwise*; Intermittence: *continuous*}, "Angular_Velocity {Stability: *constant*; Axial_orientation: *Y*; direction: *clockwise*; Intermittence: *continuous*}", "Angular_Velocity {Stability: *constant*; Axial_orientation: *X*; direction: *anti-clockwise*; Intermittence: *continuous*}, etc. Such output flows will then be released in the common environmental flow pool again for further design exploration. For example, when the first SSA agent selects its Crank_Rocker PS to act on the new environmental flow generated by the second SSA agent, "Angular_Velocity {Stability: *constant*; Axial_orientation: *X*; direction: *anti-clockwise*; Intermittence: *continuous*}", it can then generate an output flow, "Angular_Velocity {Stability: *variable*; Axial_orientation: *X*; direction: *bi_direction*; Intermittence: *continuous*}. The two SSA agents will not stop their SSA processes until the maximal search depth has been reached.

To illustrate the backtracking process, it is also assumed that there are two backtracking agents. Each time a backtracking agent can only take one environmental flow for checking and backtracking. Therefore, when the first backtracking agent checks the first environmental flow, "Visual_Light {Stability: *constant*; Intermittence: *continuous*; Type: *hot_light*}", it will find that this flow cannot satisfy the functional requirement on the output flow, which requires a to-and-fro rotation; at the same time, the second backtracking agent will check the second environmental flow, "Visual_Light {Stability: *variable*; Intermittence: *continuous*; Type: *hot_light*}", and will also reach a similar result. Note that unlike the SSA agents, the backtracking agents will not process the same flow(s) in the common environmental pool. The two backtracking agents will continue to check the other flows in the common environmental flow. When a backtracking agent obtains a flow that can satisfy the functional requirements, it then should trace the flow route back to find the combinatorial PS. For example, when a backtracking agent obtains a satisfying flow generated by the Crank_Rocker, "Angular_Velocity {Stability: *variable*; Axial_orientation: *X*; direction: *bi_direction*; Intermittence: *continuous*}, it will then begin to trace the flow route back, with the result of a flow route, i.e. "Angular_Velocity {Stability: *variable*; Axial_orientation: *X*; Direction: *to-and-fro*; Intermittence: *continuous*}" ← "Angular_Velocity {Stability: *constant*; Axial_orientation: *X*; Direction: *clockwise*; Intermittence: *continuous*} ← Electrical_Current {Stability: *constant*; Intermittence: *continuous*; Direction: *positive*; Type: *DC*} ← Visual_Light {Stability: *constant*; Intermittence: *continuous*; Type: *hot_light*}. Correspondingly, a combinatorial PS can then be generated, i.e. "Crank-Rocker ← DC-Motor ← Solar-Array".

6 CONCLUSIONS

Engineering designers are often encouraged to explore in wide multi-disciplinary solution spaces to generate novel and promising solution concepts for desired functions. However, due to lack of sufficient multi-disciplinary knowledge, it is often difficult for designers to fulfill such multi-disciplinary design synthesis tasks. Furthermore, even if they had such multi-disciplinary knowledge in mind, it would be impossible for them to manually execute the exhaustive multi-disciplinary PS synthesis processes, which would be very time-consuming and labor-intensive.

Therefore, our research has been devoted to developing an automated approach for assisting designers in achieving the conceptual design synthesis of multi-disciplinary systems. Based on the functional knowledge representation approach and the intelligent functional synthesis approach developed before (Chen et al, 2010; Chen et al, 2012), this paper has developed a multi-agent based approach for achieving multi-disciplinary conceptual design synthesis. The roles of the related agents and their collaborative synthesis mechanisms are introduced. Compared with our functional synthesis approach developed before, this multi-agent based approach has three salient features. The first one is that the multi-disciplinary conceptual design synthesis process has been decomposed into multiple sub-processes, which are then implemented as different kinds of agents (e.g. the SSA agents, the backtracking agents, etc.) that can work independently to decrease the waiting time. The second one is that a large knowledge-base of multi-disciplinary PSs can be decomposed into multiple smaller sub-knowledge bases, which enables multiple SSA (sense-selection-action) agents to simultaneously explore the solution space to speed up the search process. The last one is that the common environmental flow pool allows multiple backtracking agents to work in a parallel way, so that the backtracking time can also be largely shortened. Therefore, the proposed multi-agent based approach can be expected to achieve a higher efficiency than before.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their helpful suggestions. This research is supported by Natural Science Foundation of China (Granted No. 50975173, 51205247, and 50935004), Ministry of Science and Technology of China (Granted No. 2008AA04Z108), and Science and Technology Commission of Shanghai Municipality (Granted No. 09QA1402800).

REFERENCES

Chakrabarti, A. and Bligh, T.P. (1996) An approach to functional synthesis of mechanical design concepts: theory, applications and merging research issues. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, pp. 313-331.

Chakrabarti, A. (2004) A new approach to structure sharing. *Journal of Computing and Information Science in Engineering*, Vol. 4, pp. 11-19.

Campbell, M.I., Cagan, J. and Kotovsky, K. (2000) Agent-based synthesis of electromechanical design configurations. *Journal of Mechanical Design*, Vol. 122, pp. 61-69.

Campbell, M. I., Cagan, J. and Kotovsky, K. (2003) The A-Design approach to managing automated design synthesis. *Research in Engineering Design*, Vol. 14, pp. 12-24.

Chen, Y., Feng, P. E., He, B., Lin, Z. Q. and Xie, Y. B. (2006) Automated conceptual design of mechanisms using improved morphological matrix. *Journal of Mechanical Design*, Vol. 128, pp. 516-526.

Chen, Y., Liu Z. L. and Xie, Y. B. (2007) Understanding and representing function for conceptual design. *Proceedings of the 16th International Conference on Engineering Design (ICED'07)*, Paris.

Chen, Y., Liu Z. L. and Xie, Y. B. (2010) A general knowledge-based framework for conceptual design of multi-disciplinary systems. *Proceedings of the 4th International Conference on Design Computing and Cognition (DCC'10)*, Stuttgart.

Chen, Y., Liu, Z. L. and Xie, Y. B. (2012) A knowledge-based framework for creative conceptual design of multi-disciplinary system. *Computer-Aided Design*, Vol. 44, No. 2, pp. 146-153.

Chiou, S.-J. and Kota, S. (1999) Automated conceptual design of mechanisms. *Mechanisms and Machine Theory*, Vol. 34, pp. 467-495.

Lander, S.E. (1997) Issues in multi-agent design systems. *IEEE Expert*, Vol. 12, No. 2, pp. 18-26.

Liu, H., Tang M.X. and John, H.F. (2004) Supporting dynamic management in a multi-agent collaborative design system. *Advances in Engineering Software*, Vol. 35, pp. 493-502.

Pahl, G. and Beitz, W. (1996) *Engineering design-a systematic approach*. New York: Springer.

Prabhakar, S. and Goel, A. K. (1998) Functional modeling for enabling adaptive design of devices for new environments. *Artificial Intelligence in Engineering*, Vol. 12, pp. 417-444.

Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y. and Tomiyama, T. (1996) Supporting conceptual design based on the function-behavior-state model. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, No. 4, pp. 275-288.

Welch, R. V. and Dixon, J. R. (1994) Guiding conceptual design through behavioral reasoning. *Research in Engineering Design*, Vol. 6, pp. 169-188.

Zavbi, R. and Duhovnik, J. (2001) Conceptual design chains with basic schematics based on an algorithm of conceptual design. *Journal of Engineering Design*, Vol. 12, No. 2, pp. 36-45.

Zhao, G., Deng, J. and Shen, W. (2001) CLOVER: An agent-based approach to systems interoperability in cooperative design systems. *Computers in Industry*, Vol. 45, pp. 261-276.