

Concept formation in a design optimization tool

Wei Peng and John S. Gero

*Key Centre of Design Computing and Cognition
University of Sydney, Australia*

Key words: Situated Agent, Concept Formation, Knowledge, Design Optimization Tool, Design & Decision Support Systems

Abstract: This paper presents how a situated agent model can wrap around a design optimization tool and construct concepts from interaction between the agent, the design problem and the use of the tool. The agent develops its structure and behaviour specific to what it is confronted with – its experience. As a consequence, designers can integrate their expertise with the learning results from the agent to develop design solutions. We present preliminary results.

1. INTRODUCTION

Design optimization is concerned with identifying optimal design solutions which meet design objectives while conforming to design constraints. The design optimization process involves some tasks that are both knowledge-intensive and error-prone. Such tasks include problem formulation, algorithm selection and the use of heuristics to improve efficiency of the optimization process. Many design optimization tools focus on gathering a variety of mathematical programming algorithms and providing the means for the user to access them to solve design problems. For example, Matlab Optimization Toolbox 3.0¹ includes a variety of functions for linear programming, quadratic programming, nonlinear optimization and nonlinear least squares, etc. Choosing a suitable optimizer becomes a bottleneck in a design optimization process, in which efficiency is

¹ <http://www.mathworks.com/products/optimization/>

related to how many optimization cycles are involved in solving a design optimization problem. Designers rely on their experience to carry out this task. The limitation of such a manual process is that a sub-optimal design solution may result and hence an inefficient design produced. To increase efficiency, knowledge-based design optimization systems (Balachandran, 1988) have been used with the aim of leveraging design tasks that require human expertise. These knowledge-intensive programs respond reflexively to design variables based solely on their predefined knowledge, and are not able to adequately cope with the dynamics of the design process. These systems keep repeating themselves irrespective of their interactions with the design environment. The knowledge and functions are encoded in a “hard-wired” manner during the development stage.

Our objective here is to construct a computational model that is able to capture the knowledge of using the design tool, and hence to aid the tool’s future use. This paper introduces an approach that uses a situated agent to wrap around a design optimization tool and to construct concepts from interactions between the agent, the design problem and the use of the tool. We demonstrate how concepts can be learned in a situated agent-based design optimization tool.

2. CONCEPT FORMATION

Concept formation is essential for an agent to develop its experience in a dynamic environment. Many researchers consider categorization the essence of a concept and its formation. Concept formation has been regarded as a process of incremental unsupervised acquisition of categories and their intentional descriptions (Fisher and Pizzani, 1991). Based on this theory, a broad spectrum of computational models has been developed, including inductive learning methods, explanation-based learning approaches and connectionist algorithms. However, theories of concept formation that merely focus on categorization are not able to address the complexity of the world (Bisbey and Trajkovski, 2005). A concept lacking an understanding of why the object, entity or event has its particular properties is called a protoconcept (Vygotskii, 1986; Bisbey and Trajkovski, 2005). We use the idea that learning a concept inherently involves understanding its influence on its environment. We believe that in the dynamic activity of designing, concepts that incrementally capture the knowledge of a design process are formed as a consequence of “situatedness” (Gero and Fujii, 2000). Interactions in designing play critical roles in shaping the design practice in which new design experiences can be learned.

2.1 Situated concept formation

Situatedness involves both the context and the observer’s experiences and the interactions between them. Situatedness (Clancey, 1997) holds that “where you are when you do what you do matters” (Gero, 1998). A concept formation process can be regarded as the way an agent orders its experience in time, which is referred as conceptual coordination (Clancey, 1999).

A concept is a function of previously organized perceptual categories and what subsequently occurs, *Figure 1(a)*. A concept is generally formed by holding active a categorization that previously occurred (C1) and relating it now to an active categorization C2 (Clancey, 1999). *Figure 1(b)* illustrates a scenario of such a situated concept learning process in which sensory data is augmented into a Gestalt whole. Perceptual category C1 groups sensory sequence “S1 → S2” and activates the agent’s experience to obtain similar organizations. The agent’s experiential response (E1) represents the agent’s expectations about what would happen later in the environment. The agent constructs E1 with environmental changes (S3) into current perceptual category C2. This construction involves a validation process in which environmental changes are matched with the agent’s expectation. “Valid” means the environmental changes are consistent with the agent’s projection of such changes from a previous time.

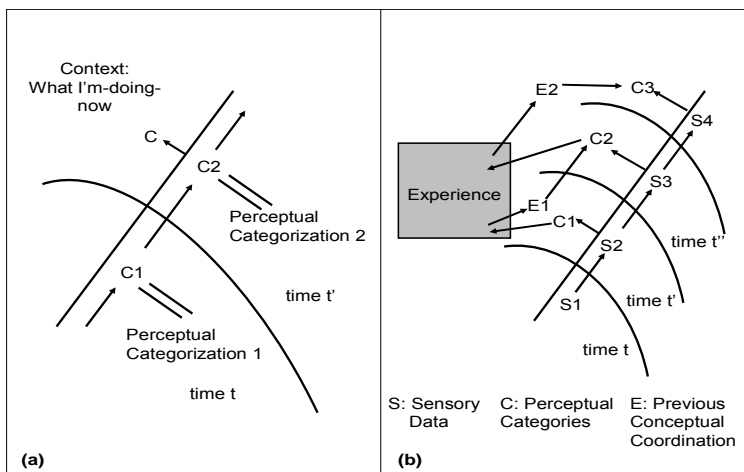


Figure 1. A situated agent learns from its interaction with the environment. (a) shows the conceptual knowledge as a higher order categorization of a sequence (after Fig. 1.6 in Clancey (1999)). (b) illustrates a situated learning scenario

The grounding process then reinforces a valid experience. For invalid expectations, the agent updates its perceptual category (C2) with the latest

environmental changes. The concurrence of “situatedness” and “constructive memory” provides the basis for concept formation in a situated manner.

2.2 A constructive memory model

This notion of “constructive memory” contradicts many views of knowledge as being unrelated to either its locus or application (Gero, 1998). A memory can be regarded as a learning process. “Memories are constructed initially from that experience in response to demands for a memory of that experience but the construction of the memory includes the situation pertaining at the time of the demand for the memory” (Gero, 1999). As shown in *Figure 2*, the original experiences (○) are knowledge structures that represent previous memories. Memories (○,●,▲) are constructed from experiential responses to active cues which are demands for memories of current sensory data.

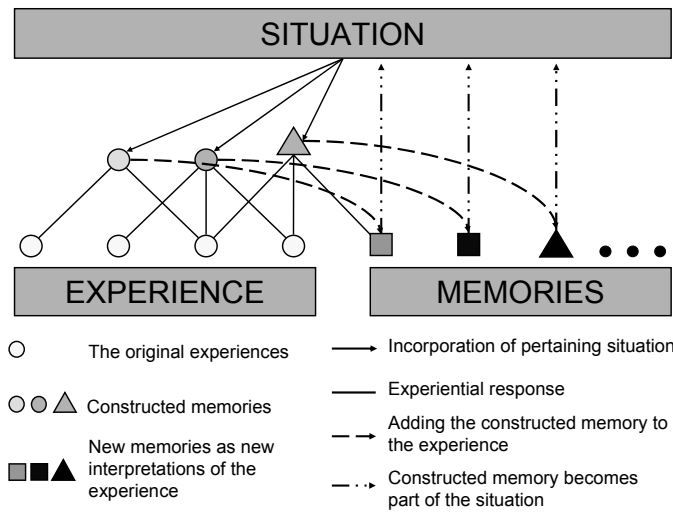


Figure 2. A constructive memory model (after Gero(1999))

This process is biased by current environment cues, experience and the way the agent’s experiential responses are developed into memories from interactions between the agent and its environment. New memories (■,■,▲) which are new interpretations of the augmented experience are added to the experience by an experiential grounding process (Gero, 1999; Liew, 2004). In this way, a newly developed experience (■) can be subsequently used to create a newer memory (▲) which in turn grounds itself into another new experience (▲). An agent constructs its memories through its internal

representation processes triggered both from external “data-driven” and goal-related “expectation-driven” demands (Gero and Fujii, 2000). A “data-driven” process is triggered by environment changes (i.e. events in the environment). Sensory data are the lowest level of descriptions of environmental changes. They capture the real-time environment stimuli in a format that can be processed by the agent. Sensory data are then mapped into various modalities in producing percepts. Percepts are data structures that can be processed by the agent to generate memory cues. According to its experiential responses to memory cues, an agent commences “expectation-driven” learning processes. In its “expectation-driven” learning processes, the agent validates its hypotheses in interactions. “Data-driven” and “expectation-driven” learning processes are coordinated in this constructive memory model, through which conceptual knowledge can be learned. A memory is activated, reactivated or constructed depending on situations encountered. We now introduce a system architecture via which the agent organizes its experience in time to form concepts.

2.3 A situated-agent based design optimization tool

Figure 3 shows the general architecture of a situated agent-based design optimization tool.

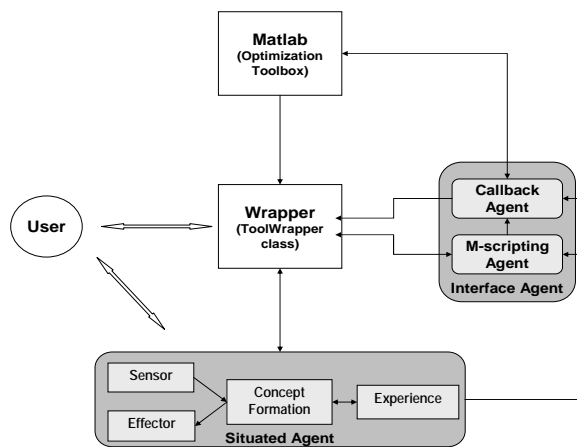


Figure 3. A situated agent-based design optimization tool

A situated agent wraps around an existing design optimization tool. A user accesses a design tool via a wrapper, where the situated agent senses the events performed by that user. The situated agent uses its experience and concept formation engine to generate a concept, which improves the tool’s

behaviour in designing. The user can also directly communicate with the agent to obtain additional information. Such a framework provides the means that allows the agent to incrementally learn new experiences. The system consists of two major components: a situated agent and a tool platform which includes a design optimization tool, a tool wrapper and interface agents.

2.3.1 The tool platform

In this research, Matlab Optimization Toolbox (version 3.0.1) is chosen as the design optimization platform. It is a collection of functions that extend the capability of the MATLAB numeric computing environment (Release 14). The toolbox includes routines for a variety of optimization classes including unconstrained and constrained nonlinear minimization, quadratic and linear programming, and nonlinear optimization. Via the MATLAB command line, Matlab users use a scripting language called M-file to define and to solve optimization models.

The interface agent, which consists of a Callback agent and a M-scripting agent, enables both users and the situated agent to operate on the engines in the Matlab Optimization Toolbox. A tool wrapper serves as an interface between the user, the tool and agents. It provides a simplified and efficient way to perform design optimization using the Matlab Optimization Toolbox. Unlike the typical Matlab users, who write M-files to define their problems, designers using the tool wrapper save time in correcting syntax errors and searching for terms of standard Matlab functions, since the M-scripting agent that is embedded in the wrapper can automate these mundane tasks.

2.3.2 Situated agent

A situated agent contains sensors, effectors, experience and a concept formation engine, which consists of a perceptor, a cue_Maker, a conceptor, a hypothesizer, a validator and related processes. Sensors gather events from the environment. These events include key strokes of objective functions and the users' selections of design optimization algorithms. Sensor data takes the form of a sequence of actions and their initial descriptions. For example, some sensor data can be expressed as:

- $S(t) \{ \dots \text{"click on objective function text field"} \}$, key stroke of "x", "(", "1", ")", "+", "x", "(, "2", ")" ... }.

The perceptor processes sensor data and groups them into multimodal percepts, which are intermediate data structures of environment states at a particular time. Percepts are structured as triplets:

- E (Object, Property, Values of properties).

For example, perceptual data P1 can be described as (Objective Function Object, Objective_Function, “ $x(1)+x(2)$ ”). The cue_Maker generates cues that can be used to activate the agent’s experience. The conceptor categorizes the agent experience to form concepts. Concepts attach meanings to percepts.

The hypothesizer generates hypotheses from the learned concepts. This is where reinterpretation takes place in allowing the agent to learn in a “trial and error” manner. The validator pulls information from the environment and examines whether the environmental changes are consistent with the agent’s responses. An agent needs to validate its hypotheses in interactions to locate a suitable concept for the current situation. An effector is the unit via which the agent brings changes to environments through its actions.

2.3.3 The agent’s experience and grounding

The agent’s experience is structured as a Constructive Interactive Activation and Competition (CIAC) neural network, which is an extension of a basic IAC network (McClelland 1981, 1995). The CIAC network is implemented in Java using the Repast² library. An IAC network consists of two basic nodes: instance nodes and property nodes. The instance node has inhibitory connections to other instance nodes and excitatory connections to the relevant property nodes. The property nodes encode the special characteristics of an individual instance (Medler, 1998). Property nodes are grouped into cohorts of mutually exclusive values. Each property node represents the perceptual level experience which is processed from sensory data. Instance nodes along with the related property nodes describe an instance of a concept. Knowledge is extracted from the network by activating one or more of the nodes and then allowing the excitation and inhibition processes to reach equilibrium (Medler, 1998). Such knowledge is a dynamic construction in the sense that the agent develops adapted experience as environment stimuli change. The experience grounds via weight adaptation and constructive learning.

Experiential grounding is the process that verifies the usefulness of a related experience in current situation (Liew, 2004). A grounding process tests whether a constructed memory correctly predicts environmental changes. Grounding via weight adaptation adjusts the weights of each excitatory connection of the valid concept, so that those nodes that fired together become more strongly connected. Weight adaptation is formulated similar to a Hebbian-like learning mechanism (Medler, 1998).

Grounding via constructive learning incorporates new instances or reconfigures existing instances. The conceptor performs conceptual labelling

² available from <http://repast.sourceforge.net/>

in the conception process, in which the agent uses its experience and applies various inductive learners to attach meanings to newly formed memories.

2.4 Concept formation from memory activation, reactivation and validation

Memory activation from the agent’s reflexive experience is represented by solid arrowed lines in *Figure 4*. An agent reflexes when its experiential response to the current memory cue is sufficiently strong to directly influence the action. Pulling environmental changes and comparing these changes with the activated memory, the agent is able to evaluate the experience of that memory.

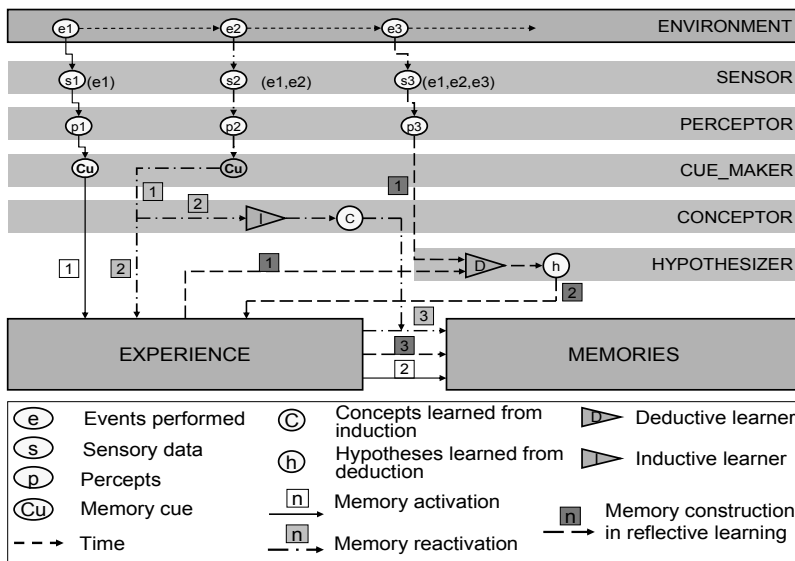


Figure 4. Memory activation, reactivation and construction in a constructive memory model

Memory reactivation occurs when the agent’s initial experiential response to a memory cue fails in the validation process. A memory can be reactivated when a memory cue is able to be subsequently identified in the environment. This process is illustrated as “Long Dash Dot” lines in *Figure 4* where sensor data (e_1, e_2) can reactivate a memory of an experience. A concept is a hierarchical structure of a design instance, related design features and their generalized descriptions. It can be obtained from this reactivated memory. Validation allows a concept to be evaluated in interactions. An agent grounds the activated or reactivated memories with positive validation value into its experience via weight adaptation (“Dash

Dot” lines in *Figure 5*). In this way, a concept is learned from grounded memories that are activated or reactivated from environmental demands of such memories in interactions.

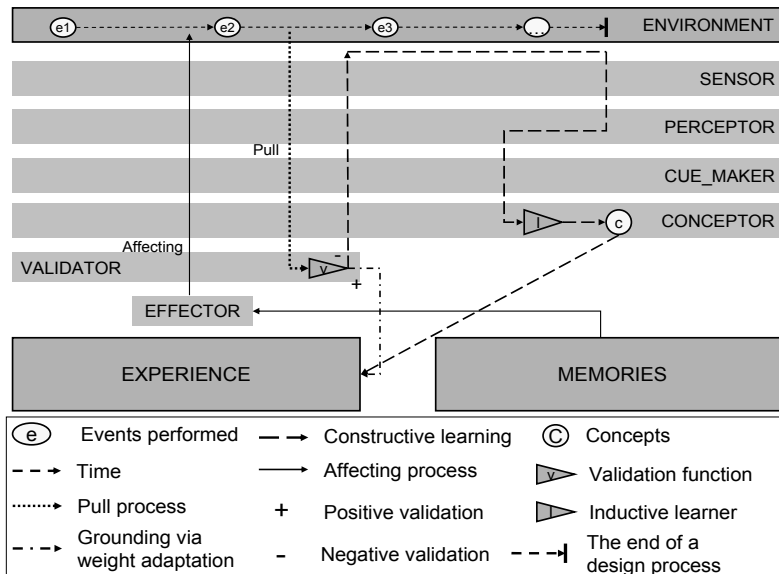


Figure 5. Validation and grounding processes in a constructive memory model

2.5 Memory construction in reflective concept learning

Reflective learning is activated by discrepancies between the agent’s expectation and current environmental changes. In *Figure 4*, “Long Dash” lines show how a memory is formed reflectively. The agent’s perceptual experience captures the knowledge of sequencing, in which lists of design events are grouped into a design instance node. The agent’s conceptual experience (or label) holds regularities or commonalities that share among a group of design instances. Such abstract structures can be re-usable in more than one context without having to repeat all the details in every context. The conceptual experience (or label) provides domain theories for a deductive learner (i.e. an explanation-based learner). A deductive learner is employed to generate hypotheses for the failure of the agent’s experience for environmental changes. The agent draws its goal concept and the explicitly represented domain theories to deduce an explanation for the current sensory data. Backward-chaining reasoning is implemented here.

2.6 Learning scenarios in interactions

We use the following five internal states and their changes to illustrate how an agent constructs concepts from its interactions with the environment:

- Conceptual experience is the agent’s high-level experience which are domain theories it uses to classify and explain its observations;
- Expectations about environmental changes are generated by the agent’s experiential responses to environmental cues;
- Validator status shows whether an agent’s expectation is consistent with the environment changes;
- Hypotheses depict the agent’s reinterpretation about its failures in creating a valid expectation.
- Experience is structured as a Constructive Interactive Activation and Competition (CIAC) neural network, which is composed of instance nodes connecting to a number of property (or feature) nodes.

As shown in *Figure 6*, an agent learns its initial experience from an observation of an instance from agent state Agent (0) to Agent (4). For example, instance 1 has 4 properties, which are named as “A”, “B”, “C” and “D”. Provided a goal concept E, the agent learns its initial experience from a

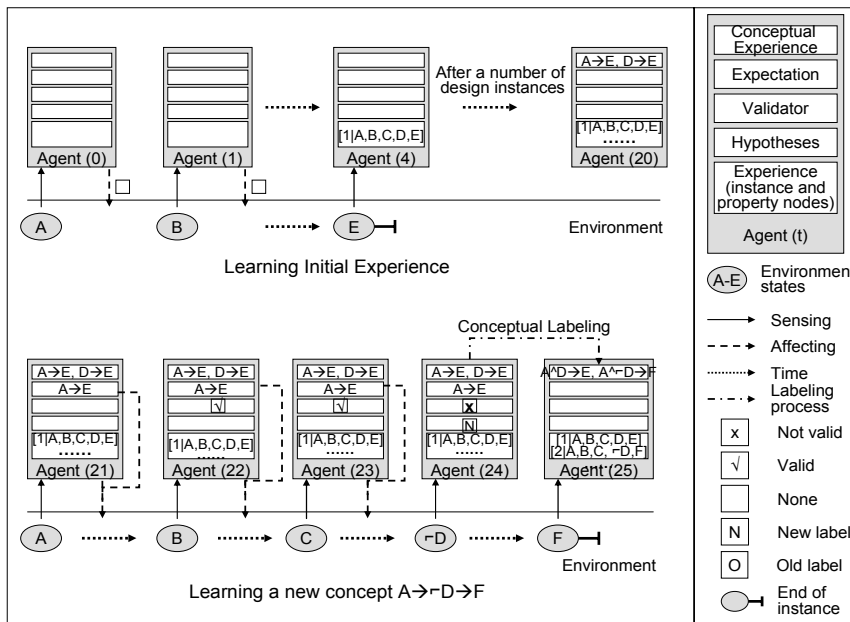


Figure 6. Learning a new concept from interactions

constructive learning process. The learned experience is expressed as “[Instance No| Properties, Goal Concept]”, i.e. “[1|A,B,C,D,E]” for the agent’s state Agent (4). After a number of design instances being observed, the agent is able to generalize a conceptual knowledge from its self-labeling process, i.e., “ $A \rightarrow E$ ” and “ $D \rightarrow E$ ”. “ $A \rightarrow E$ ” means that a feature of “A” leads to a goal concept “E”. These conceptual labels serve as domain theories, upon which the agent is based to create expectations and hypotheses.

The bottom half of *Figure 6* shows the agent constructing new concepts “ $A \wedge D \rightarrow E$ ” and “ $A \wedge \neg D \rightarrow F$ ”. In Agent (21), an experiential response “ $A \rightarrow E$ ” is created and used to affect the environment. In the process of moving from Agent (22) to Agent (24), the agent validates its expectation. When the agent fails in this validation process at Agent (24), it creates a hypothesis for such a failure. In this example, the hypothesis shows that the agent may face a new concept. As a result, the agent subsequently activates a grounding process, in which two new conceptual labels (“ $A \wedge D \rightarrow E$, $A \wedge \neg D \rightarrow F$ ”) and a new instance are incorporated into the agent’s experience.

Figure 7 illustrates a detailed example of how new concepts of selecting design optimization algorithms are formed.

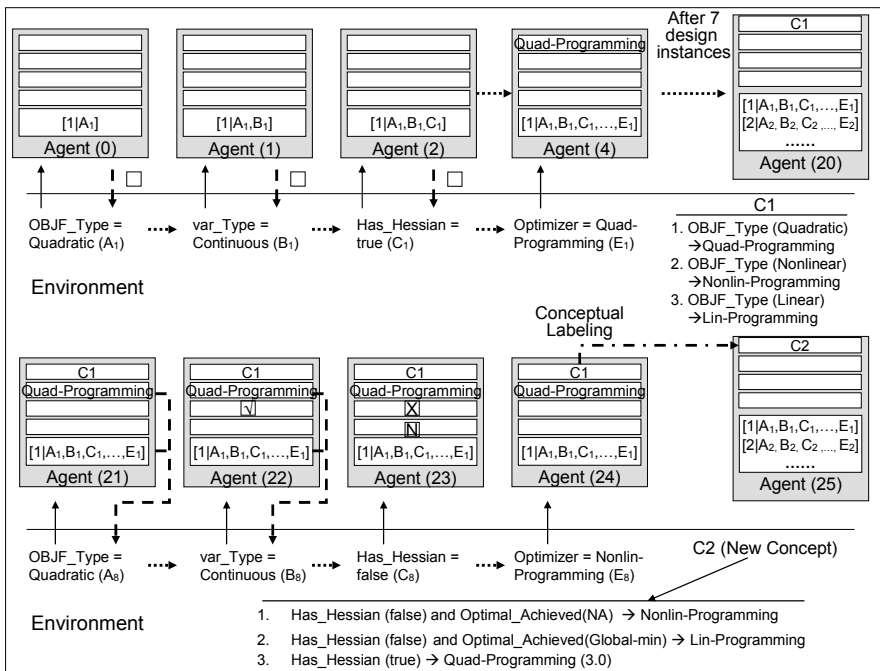


Figure 7. An example of concept formation of in design optimization (“OBJF_Type” refers to objective function type, “var_Type” and “cons_Type” denote variables and constraints type, “Has_Hessian” shows if there is a Hessian function defined)

The upper part of the diagram describes a scenario that an agent learns a new concept “C1” from a constructive learning process. As illustrated in the lower part of *Figure 7*, the agent reinterprets its environment and generates a hypothesis “N” (“N” means the agent faces a new problem) from its deductive reasoning on the feature (“Has_Hessian = false”) at Agent (23). We can see that an agent, who originally holds a concept of “C1”, can learn a new concept of “C2” at state Agent (25).

3. CONCEPT FORMATION IN OPTIMIZATION

We now present how above-mentioned concept formation mechanism can be used in the design optimization domain. *Figure 8* illustrates how a new experience can be learned from validating a hypothesis. We assume the agent’s experience contains 3 design optimization instances that belong to quadratic programming problems. We assume a domain theory of a goal concept “Quadratic Programming Optimizer” includes the following conceptual labels that are shared by these 3 instances:

- Label 1: OBJF_Type (Quadratic) \rightarrow Quad-Programming
- Label 2: Quad-Programming \rightarrow Has_Hessian (true)

“OBJF_Type (Quadratic) \rightarrow Quad-Programming” indicates that the design problem should be solved by a quadratic programming optimizer, provided the objective function is a quadratic type. “Quad-Programming \rightarrow Has_Hessian (true)” describes that a designer needs to define a Hessian matrix in order to use a quadratic programming optimizer.

The training example contains a list of time series events which correspond to an instance of the goal concept. We suppose the agent perceives an event (e₂) that is a quadratic objective function at time t:

- Event 2: OBJF_Type = Quadratic at time t

Matching this event with its experience, an agent is able to create a memory of a quadratic programming experience:

- Memory: {Concept [Label 1: OBJF_Type (Quadratic) \rightarrow Quad-Programming; Label 2: Quad-Programming \rightarrow Has_Hessian (true)], Design instance [OBJF = “xxxxx”, OBJF_Type = Quadratic, Variable_Type = Continuous, Constraint_Type = Linear, Has_Hessian = true, ...]}

As shown in *Figure 8*, such a memory fails the validation process since the event e_m – “Has_Hessian (false)” violates the agent’s memory of a quadratic programming experience. Based on conceptual label 1 and 2, the hypothesizer creates explanations:

- Hypothesis 1: {Not a linear programming concept because [OBJF_Type = Quadratic]}

- Hypothesis 2: {Not a quadratic programming concept because [Has_Hessian = false]}

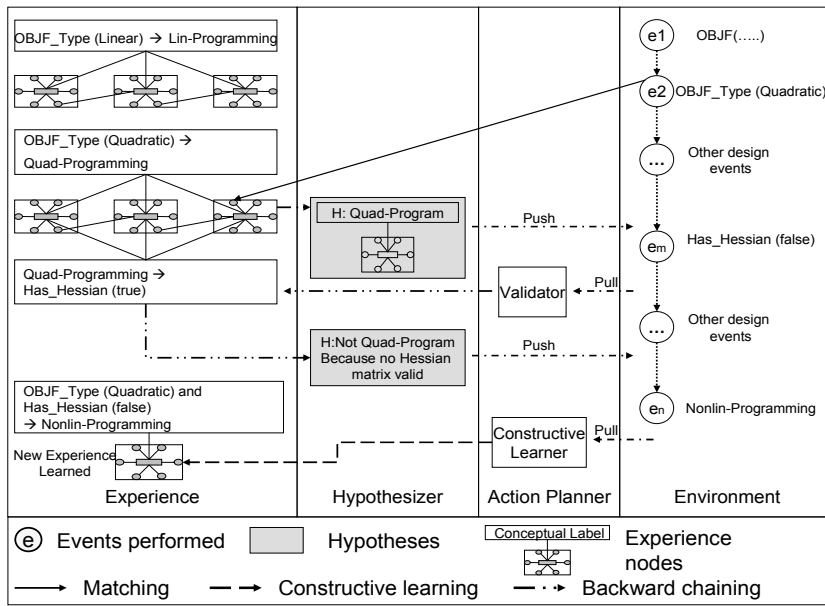


Figure 8. New concept learning from interactions

If no agent experience gives a positive answer, the action planner activates a constructive learner to construct a new memory from these observations. The newly created memory has a design instance with a conceptual label:

- New memory: {New concept [Label: OBJF_Type (Quadratic) and Has_Hessian (false) → Nonlin-Programming]
Design Instance [OBJF = xxxxx, OBJF_Type = Quadratic, Variable_Type = Continuous, Constraint_Type = Linear, Has_Hessian = false, Optimizer = Nonlin-Programming]}

Conceptual labeling can generalize the agent’s experience into conceptual labels. The agent employs an inductive learner that can incrementally generalize the design experience instance. Let us suppose there are 35 design instances stored in the agent’s experience, among which there are 12 instances of design optimization problems that are solved by quadratic programming approaches. These design examples are adopted from design practices that are published in the Matlab Optimization Tool tutorial, Reklaitis and Ravindran, et al. (1983) and Papalambros and Wilde (2000). Figure 9 shows the learning results and performance of applying a decision tree learner to the agent’s experience. Each non-leaf node stands for

a test on an attribute. Edges of the decision tree out of nodes are values of attributes for that node. Leaf nodes are used to represent design decisions for selecting optimizers. Numbers in parenthesis illustrate an observation for the class defined in the leaf node. For example, “3.0/1.0” describes that there are 3 positive observations and 1 negative observation for that class. A conceptual label can be obtained by traversing from the root node to a leaf node.

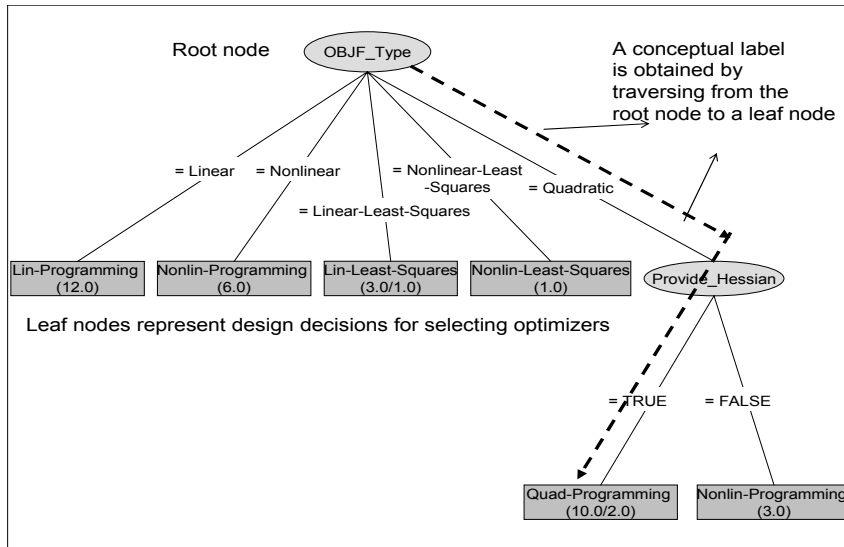


Figure 9. Conceptual labels learned from an inductive learner that uses C4.5 from WEKA³

As indicated in Figure 9, the agent has 83.3% (10 out of 12) confidence that a quadratic programming optimizer is suitable as an objective function is quadratic and a Hessian function is provided. A strong association is thus identified between quadratic objective function and Hessian matrix.

4. SYSTEM DEMONSTRATION

The implemented situated agent is illustrated in Figure 10. The tool wrapper interface allows designers to define problems. Sensors gather a user’s actions that comprise a design optimization process and activate a perceptor to create percepts. A percept cues the agent’s initial experience. Activation diagrams output the neurons winning at the equilibrium state, which represent the activated memory. Based on the responses from the

³ www.cs.waikato.ac.nz/ml/weka/

CIAC neural net, the agent constructs initial concepts and displays the constructed knowledge in the tool wrapper. The grounding process initiates a validation function which matches the initially constructed concepts with environmental changes. Weight adaptation increases connection weights of the valid concept and grounds experience A to experience B. In the agent’s reflective concept learning process, the explanation-based learner is used to form a new concept. A percept at runtime can also be developed as a new concept by a constructive learning process. Experience C is learned from constructive learning and the related self conceptual labeling process.

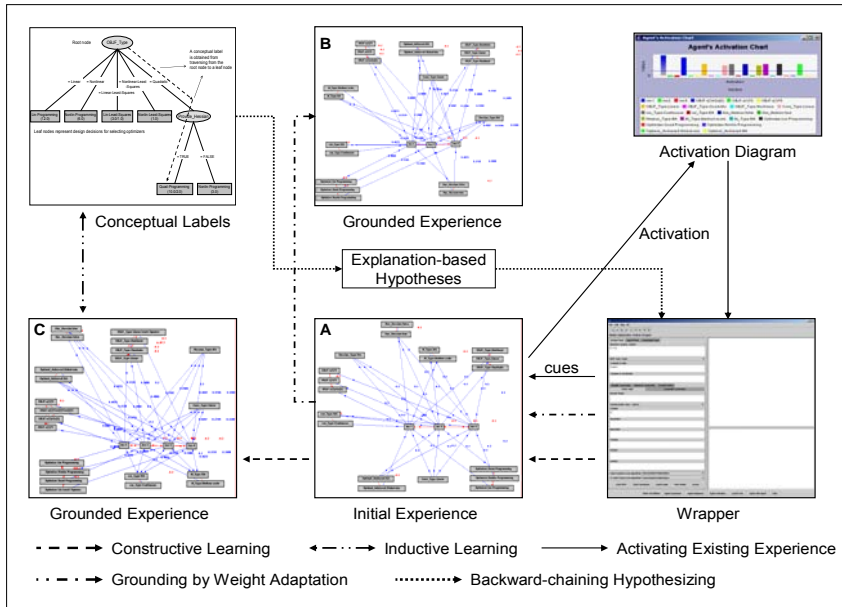


Figure 10. A prototype design optimization system that learns by its use

5. CONCLUSION AND FUTURE RESEARCH

This paper demonstrates a prototype design optimization system that can learn through its use. From its concept formation processes, the agent develops its structure and behaviour specific to what it is confronted with – its experience. Based on the conceptual knowledge learned, the agent can further improve the behaviour of the tool. As a result, designers can integrate their expertise with the knowledge learned from the agent to develop design solutions. A tool can incrementally learn by their uses. Such a tool has the potential to improve the efficiency of a design optimization tool by reducing

the number of design cycles. Future research will focus on training and testing the implemented system.

6. ACKNOWLEDGEMENT

This work is supported by a Cooperative Research Centre for Construction Innovation (CRC-CI) Scholarship and a University of Sydney Sesqui R and D grant.

7. REFERENCES

- Balachandran, M.: 1988, *A Model for Knowledge-based Design Optimization*, a PhD thesis, Key Centre of Design Computing and Cognition, University of Sydney, Sydney, Australia.
- Bisbey, P.R. and G.P. Trajkovski, 2005, "Rethinking Concept Formation for Cognitive Agents", Towson University, Towson, MD.
- Clancey, W., 1997, *Situated Cognition*, Cambridge University Press, Cambridge.
- Clancey, W., 1999, *Conceptual Coordination: How the Mind Orders Experience in Time*, Lawrence Erlbaum Associates, New Jersey.
- Fisher, D.H. and M. Pizzani, 1991, "Computational models of concept learning", in: Fisher, Pazzani and Langley (eds.) *Concept formation: Knowledge and Experience in Unsupervised Learning*, Morgan Kaufmann, San Mateo, CA, p. 3-43.
- Gero, J.S., 1998, "Towards a model of designing which includes its situatedness", in: Grabowski, Rude and Grein (eds.) *Universal Design Theory*, Shaker Verlag, Aachen, p. 47-56.
- Gero, J.S., 1999, "Constructive memory in design thinking", in: Goldschmidt and Porter (eds.) *Design Thinking Research Symposium: Design Representation*, MIT, Cambridge, Cambridge, p. 29-35.
- Gero, J.S. and H. Fujii, 2000, "A computational framework for concept formation in a situated design agent", *Knowledge-Based Systems*, 13(6), p. 361-368.
- Liew, P., 2004, *A Constructive Memory System for Situated Design Agents*, a PhD thesis, Key Centre of Design Computing and Cognition, University of Sydney, Sydney, Australia.
- McClelland, J.L., 1981, "Retrieving general and specific information from stored knowledge of specifics", in: *Proceedings of the Third Annual Meeting of the Cognitive Science Society*, Erlbaum, Hillsdale, NJ, p. 170-172.
- McClelland, J.L., 1995, "Constructive memory and memory distortion: a parallel distributed processing approach", in: Schacter (eds.) *Memory Distortion: How Minds, Brains, and Societies Reconstruct the Past*, Harvard University Press, Cambridge, Massachusetts, p. 69-90.
- Medler, D.A., 1998, "A brief history of connectionism", *Neural Computing Surveys*, 1(1), p. 61-101.
- Papalambros, P.Y. and D.J. Wilde, 2000, *Principles of Optimal Design: Modeling and Computation*, Cambridge University Press 2000, Cambridge, UK.
- Reklaitis, G.V., A. Ravindran, and K.M. Ragsdell, 1983, *Engineering Optimization Methods and Applications*, John Wiley & Sons, Inc.
- Vygotskii, L.S., 1986, *Thought and Language*, MIT Press, Cambridge, Mass.