RE-THINKING OPTIMIZATION AS A COMPUTATIONAL DESIGN TOOL: A SITUATED AGENT APPROACH

SOMWRITA SARKAR, JOHN S. GERO AND ROB SAUNDERS
Key Centre of Design Computing and Cognition
University of Sydney
Email address:{somwrita, john, rob} @ arch.usyd.edu.au

Abstract. This paper presents a situated agent for design optimization. A situated agent captures, learns from and re-uses the interactions which it has with its external environment, forming the basis for experience based knowledge building in an agent. An agent is developed for design modeling, reformulation and algorithm selection – a class of tasks in design optimization traditionally performed by humans based on their experience, and hard to automate.

1. Motivation

Optimization has long been a predominant approach supporting automated design in architecture and engineering. Space layout problems, for example, have been automated using a number of approaches (Liggett, 1985; Gero, 1985; Jo and Gero, 1995; Gero and Kazakov, 1997; Michalek and Papalamros, 2002). Optimization models designing as search. It finds the "best" design for some expected performance from a well structured, fixed solution space (Wilde, 1979; Radford and Gero, 1988; Parmee, 1998; Papalambros and Wilde, 2000). For most automated algorithms, the structure of the model remains unchanged throughout search, as does the behavior of the search process. Activities like design modeling, reformulation and algorithm selection remain primarily human endeavors.

In architectural design, subjective and qualitative criteria are difficult to model mathematically (Michalek and Papalamros 2001). Nonetheless, they are very important for overall design performance. Further, it has been shown (Baya and Leifer, 1995) that designers spend a significant amount of time assessing design information on a non-quantitative level of abstraction. In general, tools fail to support or assist the designer during conceptual design which is when modeling and reformulation are the most pronounced activities. Designers can often produce better designs using heuristics learnt through personal experience. They transfer design knowledge from past experiences into future ones, and treat modeling, reformulation and search as mutually interacting concurrent activities. Suwa et al. (1998) observed from studies conducted on designers that they use sketches not just as external representations, but also as a mode of interaction with the developing design leading to unexpected discoveries and inventions of new design issues. This dynamic, interaction based experiential learning enables a different way of designing, what Schon and Wiggins (1992) refer to as the "interaction of making and seeing". This is in contrast to the behavior of static, non-interactive optimization tools.

Our aim is to develop a computational tool that does not remain static and unchanged by the patterns of its use (Gero, 2003), but learns, builds and modifies design knowledge that arises through the experience of solving design problems. Drawing from cognitive science based approaches of situated learning and action (Clancey, 1997), situatedness (Gero 2003) and constructive memory (Dewey 1896; Gero 1999), this paper presents an agent based system for design optimization. Using this approach, an agent is able to capture, learn from and re-use interactive experiences with its external environment – the design problem and the designer. We focus on three general classes of problems in optimization: modeling, reformulation and algorithm selection. These represent a class of tasks in design traditionally solved by humans, based on experiential knowledge rather than formally encoded declarative knowledge. These have also been the hardest to automate (Papalambros and Wilde, 2000). The result is a general methodology for automated design tools that combines the interactive behavior inherent to conceptual designing with the formal rigor of optimization, along with the learning and reuse of design knowledge captured through design experiences.

2. Situated cognition based designing

The traditional objectivist view of knowledge holds it to be fixed and well defined, independent from the context of its application. It is encoded in formal, descriptive forms (Clancey, 1997) stripped of locus and application, i.e. how, when and in what situations it is used. The empirical evidence in design (Schon and Wiggins, 1992; Gero 1999; Gero 2003) shows that knowledge is not stored descriptions and sets of static rules, but an active construction process in dialogue with the environment, continuously changing and rearranging itself through experience (Clancey, 1999).

Situatedness is the nature of interaction between an embodied design agent in an external environment and the developing design, as a dialogue in which "first-person" intensional interpretations on percepts, arising from a constructive memory, produce actions which affect both the environment and the design agent. Interactions develop on the basis of how experience changes internal knowledge. This makes the path to a solution as interesting as the solution. The sources of learning are the interactive, dynamic processes of modeling, reformulation and search. Automated tools fail to support this interactive behavior.

Based on grounding and modifying past interactions, recalling a memory is a constructive, situated act, which happens in response to the current situation as well as all past experiences of a similar situation. The current experience, in turn, changes memories for the future. Clancey (1997) paraphrasing Dewey (1896) summarizes the nature of a *constructive memory*: "Sequences of acts are composed such that subsequent experience categorize and give meaning to what was experienced before". Automated tools also fail to learn constructively.

3. System architecture

This section presents the architecture for a computational design support tool based on a situated agent based. We develop the following computational requirements for a situated agent:

1. The interaction of the agent with the problem representation and the human user is the basis for learning.
2. Memory is the basis for all interactions. Experiences in memory represent tacitly gained strategic knowledge over and through the use of pre-coded task knowledge in an agent.
3. Memory of past experiences informs current agent actions by modeling expectations explicitly.
4. Based on its current model of expectations, the agent re-constructs and modifies memories of past experience. Recalled experiences are not copies of original experiences.
5. Current experiences cause the modification and reinterpretation of memories of past experience.

3.1. ABSTRACT AGENT ARCHITECTURE

The agent is embodied in an external environment, which it can sense and effect changes on. The external environment contains the external representation of the design, e.g. sketches, drawings, natural language descriptions, mathematical/ symbolic models, etc. The agent constructs and maintains an internal representation of the external world in the form of a *design prototype*, encapsulating the agent's current knowledge and expectations about the design that it uses to interact with the external representation.

The agent observes the effects of its own actions on the current external representation, and based on its current beliefs and goals, it generates an action to affect the external representation. An *interaction* is defined as a single cycle of sensing and acting on the external representation. A group of interactions define one design *experience*, starting with problem modeling and exploration, and ending with a final design solution. Figure 1 shows the abstract agent architecture, one interaction and the detailed process summary of optimization tasks. The design prototype constructed by the agent in response to the external representation is modeled by the computational constructs of situation, push and pull (Gero and Fujii, 2000).

3.1.1. Situations, push and pull

A *situation* is a construction inside the agent resulting from the agent's past perceptions, interpretations and beliefs about the world. It guides and influences what expectations and beliefs it activates, concepts the agent uses, and actions it chooses for the current experience.

Push ( ) is a data driven, bottom up process, similar to forward reasoning (Figure 1), which takes in data from one state and computes a transformation which outputs the next state. Pull ( ) is an expectation based, situation specific, top down process, similar to hypothesis-driven reasoning and results in information seeking, constructive behavior (Figure 1). In pull, expectations are triggered based on the current situation and past experiences, which affects what data the agent actively looks for in the previous state to compute the transformation to the next state. Push and pull occur as parallel processes between all states, i.e. sense data – percepts, percepts – concepts, and concepts – expectations.

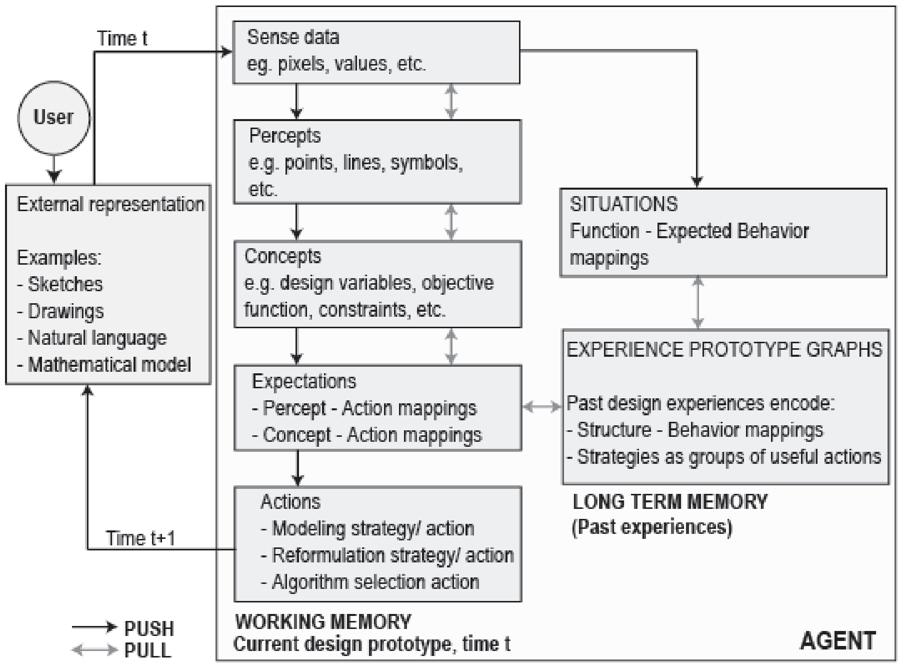
Time t
User
External representation
Examples:
- Sketches
- Drawings
- Natural language
- Mathematical model
Time t+1
PUSH
PULL
Sense data
eg. pixels, values, etc.
Percepts
e.g. points, lines, symbols, etc.
Concepts
e.g. design variables, objective function, constraints, etc.
Expectations
- Percept - Action mappings
- Concept - Action mappings
Actions
- Modeling strategy/ action
- Reformulation strategy/ action
- Algorithm selection action
WORKING MEMORY
Current design prototype, time t
SITUATIONS
Function - Expected Behavior mappings
EXPERIENCE PROTOTYPE GRAPHS
Past design experiences encode:
- Structure - Behavior mappings
- Strategies as groups of useful actions
LONG TERM MEMORY
(Past experiences)
AGENT

Figure 1. Abstract agent architecture, interaction with the problem representation, detailed process summary of optimization tasks carried out by agent

3.2.2. Agent working memory

The working memory is represented as a dynamic graph of interactions. One interaction is a discrete time step. The two processes push and pull act in parallel guiding plan generation and action selection. Each node in the graph represents the design prototype at that time step. Each link in the graph is a transformation or agent action on a design prototype. A full graph represents one design experience, where a design prototype at each time step represents the current state of the design. Working memory is concerned with constructively recalling long term memory for a current design experience.

3.3.3. Agent long term memory

The long term memory is a dynamic graph of experiences. Long term memory represents mappings between design situations, expectations and strategies derived from a number of specific experiences over time. It is the basis for what design situations and past experiences get triggered for each future experience in the working memory. Time is measured in discrete steps in terms of number of experiences. Long term memory is constructive as previous memories are modified in the light of new experiences.

3.2. DESIGN MODELING, REFORMULATION, ALGORITHM SELECTION

A design experience has three parts, modeling, reformulation and algorithm selection, each part comprising a set of interactions between the agent and the design representation. Consider a space layout planning problem as an example, which deals with locating a given number of activities in a given number of locations with the aim of minimizing the costs of allocation (Liggett, 1985). Modeling is a set of interactions dealing with choosing of numbers of activities, defining locations, interaction costs between activities and locations, seeding of a good starting solution, etc. Reformulation is a set of interactions that deal with exploring changes in the model – introduction of new activities or spaces, deletion of old ones, etc. Algorithm selection deals with choosing a particular algorithm and applying it to the model– constructive improvement algorithm, heuristic search, genetic algorithms, etc. If the result satisfies the user, the design experience terminates. If not, this acts as a cue for the agent to change its strategies. The user can act in any of the three stages interactively with the problem representation, and all three stages may be explored in any order of implementation.

The goal for the agent is to develop, by being exposed to many space layout planning experiences, strategies for three tasks that may be applied to future problems. A user specifies a space layout planning problem as the external representation. We present agent activities as examples for the three stages:

- Stage 1: Modeling - The agent interprets the external representation and produces an internal design prototype. Push-pull (Figure 1) cause the sense data to trigger a design situation in the long term memory as a mapping between the function (space layout) and expected behavior (minimization of costs). This triggers past experiences producing expectations in the agent for possible structure – behavior mappings (Figure 1). Structure variables are the number of activities and locations, fixed and interaction cost matrices. The behavior is a measure of the total cost of allocating activities to locations. The agent activates concepts for structure variables and corresponding behavior, pulling in the relevant data from the user specified representation to seed a starting solution. The agent finds a good starting solution by pulling and grouping together activities and locations using any previous useful strategies: iterative improvement, constructive initial placement (Liggett, 1985), or using good activity groups identified in previous experiences (Gero and Kazakov, 1997).
- Stage 2: Reformulation – The user reformulates the model by introducing new activities, redefining location zones, changing the cost matrices. The agent interacts with the problem representation, and changes the internal design prototype in response to user changes. This continues till the user – agent – problem interactions produce a final model.
- Stage 3: Algorithm selection – Depending on the problem size and complexity, the agent chooses an algorithm and applies it to the model. Examples of strategies which it develops in this case are mappings between problem size and algorithm – a small space layout problem is easily solved with iterative improvement or constraint propagation, but a large complex problem needs a stochastic algorithm like genetic algorithms etc.

At the knowledge lean stage, when the agent has no strategic knowledge but only task knowledge, it starts applying strategies and algorithms randomly, or by observing user choices, based on simple forward reasoning. The experience which leads to a good result is taken as a “good” example, and becomes part of long term memory. Now the next time, if a similar problem is presented, the agent perceives a similar situation, and ends up applying the same strategy. But if this time, the problem characteristics are different (say multi-objective instead of single objective) then the expectation based action does not lead to good results. This acts as a cue for the agent to try some other approach. A successful approach again becomes part of the long term memory. The prototype – action experience graphs are dynamic constructs which evolve with experiences, and are the basis for situation perception and expectation computation for the agent for all experiences. It is expected that over time the strategies which it develops will equilibrate for similar classes of problems. Grounding of these strategies with experience will lead to a reduction in time to solving design problems.

Acknowledgements

This work is supported by the KSG Scholarship for Situated Design Optimization and by a Faculty of Architecture International Research Scholarship.

References

Baya, V. and Leifer, L.: 1995, Understanding design information handling behavior using time and information measure, *in* A.C. Ward (ed). *Proceedings of the 1995 Design Engineering Technical Conferences*, ASME DE-83, Vol.2, Pp. 555-562.

Clancey, W.: 1997: *Siutated Cognition – On Human Knowledge and Computer Representations,* Cambridge University Press.

Dewey, J.: 1896/1981, The reflex arc concept in psychology, *Psychological Review* **3,** 357-370 (reprinted in 1981).

Gero, J.S.: 1985, *Design Optimization,* Academic Press, London.

Gero, J.S.: 1990, Design Prototypes: A knowledge representation schema for design, *AI Magazine,* **11**(4), 26-36.

Gero, J.S.: 2003, Situated computing: A new paradigm for design computing, *in* A. Choutgrajank, E. Charoenslip, K Keatruangkamala and W Nakapan (eds), *CAADRIA03*, Rangsit University, Bangkok, Pp 579-587.

Gero, J.S. and Fujii, H.: 2000, A computational framework for concept formation in a situated design agent, *Knowledge-Based Systems,* **13**(6), 361-368.

Gero, J.S. and Kazakov, V.:1997, Learning and re-using information in space layout problems using genetic engineering, *Artificial Intelligence in Engineering,* **11**(3), 329-335.

Jo, J.H. and Gero J.S.: 1995, Space layout planning using an evolutionary approach, *Architectural Science Review*, **36**(1), 37-46.

Liggett, R.S.: 1985, Optimal spatial arrangement as a quadratic assignment problem, *in* J.S. Gero (ed), *Design Optimization,* Academic Press, New York. Pp. 1-40.

Michalek, J.J. and Papalamros, P.Y.: 2002, Interactive design optimization of architectural layouts, Engineering Optimization, 34(5), 485-501.

Papalambros P and Wilde DJ: 2000, Principles of Optimal Design: Modeling and Computation, Cambridge University Press.

Parmee I (ed):1998, Adaptive Computing and Design and Manufacture, Springer.

Radford, AD and Gero, JS: 1988, Design by Optimization in Architecture and Building, Van Nostrand Reinhold, New York.

Schon, D.A. and Wiggins, D.: 1992, Kinds of seeing and their functions in designing, *Design Studies,* **13**(2), 135-156.

Suwa, M., Gero, J.S. and Purcell, T.: 1998, Analysis of cognitive processes of a designer as the foundation for support tools, *Artificial Intelligence in Design '98*, Kluwer, Dordrecht.

Wilde, D.J.: 1979, *Globally Optimal Design*, John Wiley, New York.