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-based tool 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 been a predominant approach supporting automated design in architecture and engineering. Space layout problems, for example, have been automated using a number of optimization approaches (Liggett, 1985; Jo and Gero, 1995; Gero and Kazakov, 1997; Michalek and Papalambros, 2002). Optimization models designing as search. It finds the "best" design for some expected performance from a well structured, fixed solution space (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, which involve interactive exploration and a dialogue with the problem representation, remain primarily human endeavors.

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 current 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 way of designing that 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. They represent a class of tasks in design traditionally solved by humans, based on experiential knowledge rather than formally encoded knowledge. These have also been the hardest to automate (Papalambros and Wilde, 2000). The result is a general methodology for 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 relates to 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 and concepts, 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. 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". Our current tools fail to learn constructively.

3. System architecture

This section presents the architecture for a situated agent based design optimization support tool. We develop the following computational requirements for a situated agent: (i) interaction of an agent with the problem representation and the human user is the basis for learning; (ii) memory is the basis for all interactions; (iii) experiences in memory represent tacitly gained strategic knowledge over and through the use of pre-coded task knowledge in an agent; (iv) memory of past experiences informs current agent actions by modeling expectations explicitly; (v) based on its current model of expectations and experience, the agent re-constructs, reinterprets and modifies memories of past experience; and (vi) recalled experiences are not copies of original experiences.

3.1. ABSTRACT AGENT ARCHITECTURE

An agent is an autonomous computational entity embodied in an environment that contains the external representation of the design the user and the agent interact with, e.g. sketches, drawings, natural language descriptions, mathematical/ symbolic models, etc. A situated agent represents the external world internally. This internal representation is an interpreted model of the external representation in the form of a *design prototype*, an ontological knowledge representation structure (Gero 1990) encapsulating the agent's current knowledge and expectations about the design. Following from the discussion on situatedness, this has cognitive implications – the situation provides intentionality to the agent's perceptions and conceptions on the external representation, and guides which concepts and interpretations are activated to produce a design prototype.

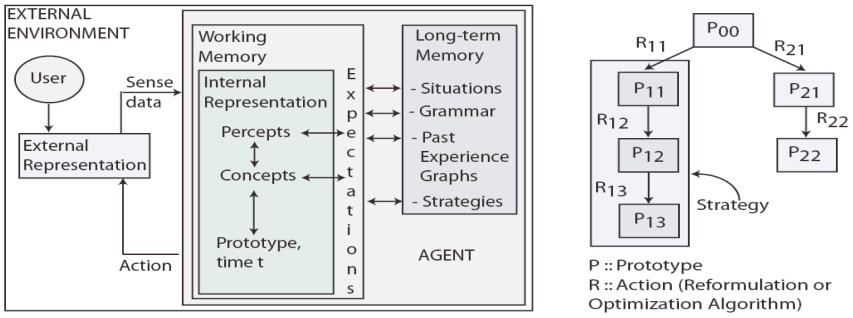
The agent senses the external representation, constructs an internal representation, reasons about the next action(s), and effects it, Figure 1(a). The new external representation now becomes the source for new sense data to the agent in the next cycle. An *interaction* is defined as one cycle of sensing, perceiving, conceiving, hypothesizing and acting on the external representation, Figure 1(a). Such repeatedly occurring interactions produce and modify a design prototype in the agent's working memory. A group of interactions define one design episode or *design experience*, starting with problem modeling and exploration, and ending with a final design solution. Computationally, the agent interacts and experiences through the construct of a situation, and processes "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 are activated, concepts the agent uses, and actions it chooses for the current experience. For an agent, a situation makes the world appear the way it does; it provides intentionality to the agent's actions.

Push is a data driven, bottom up process, similar to forward reasoning. It causes actions driven by the external world data that changes the internal representation. *Pull* is an expectation-based, situation-driven, top-down, constructive process, and results in information seeking, constructive

behavior. In pull, expectations are triggered based on the current situation and past experiences, which affects what data the agent actively looks for in the external or internal worlds to use in the current experience. Push and pull occur as parallel processes between all states, i.e. sense data – percepts, percepts – concepts, and concepts – expectations.



(a) (b)

Figure 1. (a) Abstract agent architecture showing an interaction cycle; double arrows show push-pull processes; single arrows show push processes (b) a design experience in working memory; building of a strategy

3.2.2. Agent working memory

The agent's working memory is represented as a dynamic graph of interactions, Figure 1(b). One interaction is a discrete time step. The two processes of 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. Examples of actions are reformulation actions, calling and applying optimization algorithms. 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.

Grammar components:
1. Terminals: sense data
   - symbols, letters, numbers
2. Non-terminals: percepts, concepts
   - variables, parameters, functions
3. Starting symbol: Design-prototype
4. Grammar rules

Grammar rules:
<Design-prototype> ::= <Str-Var> <Beh-Var> <Str-Par> <Beh-Par>
<ObjFn> <IneqFn> <EqFn> <ReformAct> <ReformStgy> <OptAlgo>
<Str-Var> ::= <Lowercase-list>
<Str-Par> ::= <Uppercase-list>
…
<ReformAct> ::= INEQ-EQ | EQ-INEQ| …
<OptAlgo> ::= GENETIC ALGO | GRAD DESCENT | POWELLS METHOD | …

Figure 2. Design optimization grammar as part of long term memory

3.3.3. Agent long term memory

The long term memory comprises a design optimization grammar that the agent uses to construct prototypes in working memory, Figure 2, past experience graphs and design situations. It represents mappings between design situations, expectations and strategies derived from a number of specific experience graphs over time. Design situations cue and guide concepts and interpretations in past experiences 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. New knowledge is built in terms of the agent extending the grammar by building new concepts and strategies.

3.2. DESIGN MODELING, REFORMULATION, ALGORITHM SELECTION

This section briefly introduces the application of the system in optimization tasks. Three broad classes of optimization tasks are considered, defining three types of involvement for the agent – design modeling, model reformulation and algorithm selection. Table 1 presents 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). The user can act in any of the three stages interactively with the problem representation. The three phases are mutually interacting ones and may flow into each other in any order, i.e. for example, if the user is dissatisfied with the result produced by an optimization algorithm, the agent may return to the reformulation stage. This is the main motivation which guides the system: in a design experience we never know in advance what the most appropriate model or the most appropriate algorithm is. We explore, construct and search in the process of design interaction the best model as well as the best solution within that model. The structure of the solution space or the design model is as fluid as the search for the solution within it; the agent constantly explores both. The goal for the agent is to build, learn and ground, by being exposed to many design experiences, strategies for three phases that may be applied to future problems.

TABLE 1. Example optimization problem and agent-user interactions with problem

Optimization Phase	**Interactions with design**	**Results**
Modeling: identify structure and behavior variables, parameters, objective function, inequalities, equalities	Choosing of numbers of activities, locations, interaction costs between activities and locations, seeding of a good starting solutions, etc.	Design prototype in the optimization grammar representing the design problem
Reformulation: explore conceptual changes in model, reformulate variables, parameters and constraints	Introduction of new activities or locations, deletion of old ones, changing mathematical form of the problem, etc.	Reformulated design prototype
Algorithm selection: select algorithm and apply	Iterative improvement, heuristic search, genetic algorithms, etc.	Interim and final solutions

The set of actions which the agent uses in the interactions are simply choices, and do not explicitly set any path for the agent in guiding it to a solution. They represent third-person knowledge, devoid of any context – simple mathematical-symbolic transformations. The agent or the user can pick and apply any one of these for any problem, if the problem is stated in a standard form in a user interface. The agent builds first-person experience based knowledge through the experience of applying this third-person mathematical knowledge. Over a number of experiences, sequential patterns of actions will begin to emerge as successful ones for classes of problems. We name these as strategies (subsets of design experience graphs), Figure 1(b). In similar situations, these get triggered from past design experiences to

become part of the current one. These recalled strategies are not similar to the past experience, as the prototypes in each case are likely to be different. Since all interactions happen using the grammar, Figure 2, the new strategies which the agent builds through design experiences are continually incorporated into the long term memory from the working memory, extending both the grammar and the long term memory. This knowledge is then available for all the future experiences.

The system is currently being implemented by building a Java-based environment which uses an embedded Jess rule engine for representing agent grammars and performing preliminary reasoning. Mathematica and Matlab will be used as external optimizers, which the Java-based agent will be able to call. We expect that the strategies and concepts which the agent builds will equilibrate for similar classes of design problems over many design experiences. The expected results are conceptual design support, discovering new solution strategies, improvements in the quality of solutions and reductions in the time it takes to find solutions. The system presented in this paper develops the basis for a computational tool that does not remain static and unchanged by the patterns of its use but learns, builds and modifies design knowledge that arises through the experience of 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

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.: 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.

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.