Learning To Design Together

Dan L. Grecu & David C. Brown
Artificial Intelligence Research Group
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609, USA
E-mail: dgrecu@cs.wpi.edu, dcb@cs.wpi.edu
URL: http://cs.wpi.edu/~dgrecu, http://www.wpi.edu/~dcb

Abstract

The paper presents experiments in learning in a multi-agent design system. Inductive techniques are used to construct models of other agents' behaviors, by relating design requirements to design proposals. The models are used to anticipate future design proposals and reduce the amount of interaction needed to find solutions for parametric design problems.

1. Introduction

This paper present experiments in learning in a multi-agent design system. We approach non-routine parametric design problems by using small, specialized, knowledge-based design agents called Single Function Agents (SiFAs). The size of the agents makes them very 'competent' in their decisions, but imposes an overhead on the design system due to the complexity of the task of reaching an agreement. The number of interactions between agents can be significantly reduced if the agents learn to predict the responses from the other agents with which they interact.

We first describe what SiFAs are and how they solve design problems, then what the agents learn about each other and how they use that knowledge. The experimental results section will give an overview of the benefits achieved and the difficulties encountered.

2. Design with Single Function Agents

Multi-agent systems are increasingly used for design applications [Klein 1991] [Kuokka et al. 1993] [Lander & Lesser 1991] [Werkman & Barone 1991]. There is also an increasing effort to investigate the learning possibilities in such systems [NagendraPrasad et al 1995] [Sen 1995]. The computational power and flexibility of multi-agent systems suggests attempting design problems of increasing complexity. However, it is important to use agents to investigate the other end of the computational spectrum – the elementary problem-solving steps and interactions which together produce a solution for a design problem [Douglas et al 1993] [Victor & Brown 1994] [Dunskus et al 1995] [Grecu & Brown 1995].

Our experiments address Parametric design problems. This stage of design assumes that the configuration of the design is already decided. The design is to be completed by deciding the values of the parameters associated with the configuration, such as diameters, lengths, material choices, colors, etc. Parametric design is often a non-routine process. Choices are made based on various types of knowledge and from various points of view. Arguments in favor of or against a choice are usually the result of a reasoning process based on considerable domain knowledge.

SiFAs are small, specialized, knowledge-based systems that know how to deal with well delimited portions of a design problem. Each agent has a *target*, the entity about which it reasons. In our case every design parameter represents a target. Each agent performs a single *function* on its target. An agent knows, for example, how to *select* a value for the design parameter, or how to *critique* such a value. Each agent works from a given *point of view* – a direction or aspect of design which the agent tries to favor during its reasoning (e.g., a SiFA could perform *selection* of a *material* from the point of view of *reliability*).

Among the previously enumerated dimensions which characterize an agent, the function is the

most important one. It defines an agent's type, leads to specific types of reasoning, and determines what can be learned about that agent. We use the following agent types:

Selectors: A selector selects a value for a parameter, by picking a value from a prestored or calculated representation of possible values. The values are ordered according to some preferences. The set of values and their order can be context-sensitive, i.e., they can vary depending on the design requirements.

Critics: A critic criticizes the value of a parameter by pointing out constraints or quality requirements that are not met by the current value. Critics are also context-sensitive. They point out different reasons for criticism, depending on the requirements.

Praiser: A praiser praises values of parameters by pointing out their importance from the agent's point of view. For the purposes of this paper, praisers were considered context-insensitive. A material is considered as having good endurance based on general material properties, not on design specifications.

Estimators: An estimator produces an estimate of a parameter's value. Unlike selectors, estimators can produce a result much faster and with insufficient information.

Evaluators: An evaluator evaluates the value of a parameter, producing a measure of goodness, usually represented as a percentage or as a symbol (e.g., "good").

SiFAs use a design board to post their proposals and decisions. The board is also used to store the current design state – the configuration and the parameter values. No external scheduling is used for agent control. Agents get activated based on preconditions describing the availability of information for their task and the need for the result they are supposed to deliver.

Our results are based on the application of SiFAs to the parametric design of helical compression springs. The experiments are limited to agents that have the Spring Material parameter as the *target*. Deciding on a Spring Material involves a large number of criteria, such as temperature ranges, tensile strength, hardness, and elasticity. Each of these criteria represents a separate *point of view.* The agents work together by exchanging and analyzing values depending on their *function*. Selectors provide values, critics provide unacceptable value sets and praisers highlight proposals.

The process of deciding a parameter involves the following stages:

1. *The selectors negotiate a common acceptable value for the design parameter.* The negotiation consists of an exchange of proposals in decreasing order of preference. Praisers can intervene and point out the desirability of the current proposal. This will alter the course of the negotiation. The agent that responds to a praised proposal has to consider more possibilities than usual (not only the next best) to check whether the proposed value is in an acceptable range. If this process fails the agent counter-proposes the highest ranking value currently available on its list.
2. *The critics post their objections to the agreements reached by selectors*. A critic that rejects a parameter value will announce the entire range of values that are not allowable under the current conditions. The operation is computationally expensive, but it is carried out only when a critic is not satisfied with the current decision.
3. *The selectors start a new negotiation round, during which they avoid use of the values considered not to be acceptable by the critics*.

Steps 2 and 3 are repeated until the system reaches a solution accepted by all the agents.

The experiments used two types of negotiations among selectors. The main difference between them is the number of values an agent offers in a proposal and the number of values which are analyzed when a proposal arrives from another agent. Both strategies converge if a solution exists:

I) Point-to-point negotiations: Each agent proposes a single value at a time. When analyzing a proposal, the agent matches it only against its best value. A disagreement prompts the agent to use its own value as a counter-proposal.

II) Range-to-range negotiations: Each proposal is an ordered set of values. The receiving agent matches this proposal against its best value and, if not successful, against a set of next best values.

Critics and praisers are unaffected by the negotiation strategy used.

3. Learning with Single Function Agents

The number of interactions generated to find an acceptable parameter value is high, and may be a serious system overhead. Our experiments used 11 agents: 2 selectors, 5 critics and 4 praisers. The agents encode knowledge about 20 materials, which makes the material choice difficult, due to the various comparisons which need to be taken into account.

Selectors learn, as they are the only ones that perform a search. Learning is a result of agent interactions and is supposed to reduce the interactions by anticipating responses. The process of agent *A* learning about agent *B* includes two generic phases:

1. *Create a case from the interaction with agent B.* A case is indexed by the design requirements which *B* considered in taking its decision. The contents of a case are the responses of agent *B*. Each case represents a training instance in developing a conceptual description of *B*'s behavior [Gennari et al 1990] [Michalski 1983]. The concept features are the design conditions which led to *B*'s proposal, expressed as allowable design ranges and thresholds. The contents of the proposal itself determines the class in which the concept falls (e.g, the material being objected to, if agent *B* is a critic).
2. *Integrate the case in the knowledge already available about agent B.* The goal is to create a mapping of the options and/or preferences of *B* under various design conditions. Each training instance potentially enhances the intentional description which *A* has developed about *B*'s behavior.

Learning is used to create a model of the other agent based on the connection between design conditions and responses. Modeling the other agent's reasoning would be difficult, due to the specialized agent knowledge and the computational costs of such approaches. Therefore, an inductive technique is likely to be more efficient, even if there is no guarantee of perfect accuracy.

Since the types of domains which have to be covered by the inductive learning are extremely diverse, we have used disjunctive induction methods [Michalski et al 1986]. However, variations were imposed depending on the agent one learns about. The classification features are the same for all agent types since they originate in a set of design conditions which are the same for all agents. However, what exactly is learned *about* an agent *B* depends on its type:

- Praisers are the most easy to learn about. The acquired knowledge captures the values they praised in the current design context. The defined classes correspond to the material types which are used.
- For critics the knowledge includes the entire set of values determined as being unacceptable. Classes correspond to individual materials, as in the praisers' case.
- For a selector the learned information represents a partial preference list. Classes are defined by sets of preferences. The lists are incomplete for two reasons. Preferences are revealed during negotiation. As an agreement is reached the rest of the preferences remain unrevealed. On the other hand, during the negotiation some preferences might not be expressed if the proposing agent anticipates that they will not be accepted.

The knowledge acquired has different degrees of completeness. An agent *A* learns a fixed behavior

about the praisers and critics with which it interacts. That means that under the same conditions, the critics and selectors will act in the same way. This is not true for the selectors with which *A* interacts. Even though the list of preferences of a selector remains the same each time a set of design conditions repeats itself, the perception of these preferences from the outside may be different. Therefore, a case describing a behavior pattern of selector *S* can 'migrate' from one class to another as new information is learned about that case in subsequent negotiation sessions.

The result of learning is used by selectors when making proposals, in order to try to anticipate their acceptance. The anticipation process is carried out only if there is knowledge available about the behavior of all the selectors involved under the current circumstances. If the agent's proposal is unlikely to be accepted the agent prepares a new proposal. The agent posts its proposal if it believes the value will be accepted or if it runs out of knowledge about the other selector(s).

The prediction process may lead to the failure to reach an agreement, due to the fact that a selector has only partial knowledge about the other selector's preferences. After failure, the negotiation is reinitiated and the selectors do not use the learned knowledge about each other, to prevent overlooking some of each other's choices. Knowledge learned about critics is used to filter any invalid proposals or counter-proposals, before posting them. Finally, knowledge about praisers will help anticipate which agent will have to reconsider its proposal in a conflict.

4. Experimental results and conclusions

The experiments used 20 materials considered representative for helical compression springs. The design problems included design constraints that affect each selector and critic. The experiments measured the reduction in interactions and analyzed what the agents learned. An "interaction" is a selector's proposal, a critic's objection, or a praiser's praise.

The first type of analysis used a sequence of 22 generated design problems. One problem was considered a reference problem, while each of the other 21 problems introduced a change in the conditions affecting the reasoning of one selector or one critic, compared to the preceding problem. Three changes were considered for each agent of the 7 selectors and critics. Each run through the set of 21 problems is called an experiment and the set of changes was constant for each experiment, even though their ordering was different.

At first the changes in the design problems were made in random order and the system had to run several times through the sequence of problems. Each experiment was run with and without learning. Table 1 summarizes the average results for the set of experiments:

TABLE 1. Decrease of interactions in point-to-point negotiations (random problem ordering)

	Average number of interactions (rounded to closest integer) after			
Type of analysis	**6 expts.**	**12 expts.**	**17 expts.**	**22 expts.**
without learning	34	36	33	34
with learning	31	29	23	19

The slow initial decrease in the number of interactions is because the selectors don't use predictions if they have no information about the behavior of the other selector in a new situation. The initial decrease in the interactions is due mainly to the information learned about the critics.

The same analysis was repeated by first scheduling the changes in the design conditions affecting selectors, and then making the other changes in random order. The results are shown in Table 2. The numbers in parentheses in the table represent the interactions due to selectors.

The major improvement can be seen towards the end of this run. Selectors learning about selectors

happened mostly during the first 6 design problems and without any changes in the behavior of critics and praisers, as their requirements did not change during that time. Consequently, interactions in the next 5 design problems were reduced mostly due to shorter negotiations between selectors.

TABLE 2. Decrease of interactions in point-to-point negotiations (design problems were ordered such that requirements for selectors were changed first)

Type of analysis	Average number (rounded to closest integer) of interactions after			
	6 expts.	**12 expts.**	**17 expts.**	**22 expts.**
without learning	34	35	33	34
with learning	31 (25)	22 (18)	19 (16)	15 (13)

Similar experiments were carried out for range-to-range negotiations. As the number of interactions is smaller, and the agents convey less information to each other, the learning was significantly slower. However, the number of failures in reaching an agreement was about 3 times as high as in point-to-point negotiations. Using the same type of changes in the points of view in various combinations, the interactions were reduced by 40% after going through 50 experiments.

For all of the previous experiments, the changes in requirements produced problems which were fairly close to the ones in the reference design problem. The primary target was to test the reduction in the number of interactions under the assumption that the variations can be captured relatively quickly. Another type of analysis assumed that the changes in the design requirements affected only one agent, but on a large scale. The goal was to investigate how a selector would build a reliable model of the corresponding agent for a large number of situations.

We targeted how well a selector can learn about the other selector. The main problem affecting learning accuracy is that the knowledge which selector *S* learns about a critic *C* prevents it from revealing preferences that will play a role when the critic isn't present. Note that the critic is active depending on conditions which are different from the ones which influence the selectors.

Having a selector learn without the critics and then running the system on the same (or similar) cases, but with the critics, generated the most accurate learning. Accuracy was measured in terms of distance between the concepts developed and the actual preferences of the targeted agent. The performance is due to the fact that learning without the interference of critics generates an accurate partitioning of the design requirements into concepts. Adding the critics does not bring large changes to the conceptual partitioning of agent behavior, but causes the concepts to be refined.

The major challenge in this investigation is the development of additional criteria to show how the evolution of agent *B* will influence the accuracy of what agent *A* learns about *B.* The learning of *A* about *B,* and of *B* about *A,* depend on each other. At the same time, *A*'s learning mirrors what *B* has learned about, say, critic *C*. The knowledge that agent *A* learns about *B* does not become obsolete as *B* evolves, but it can obviously be reduced as *B* bases its reasoning on knowledge compiled from previous interactions. Learning can be explored in isolated selector to selector interactions. However, it can be significantly different in the presence of critics. Therefore, besides local investigations, some global criteria are needed to evaluate what happens with a group of agents learning together.

Additional work will explore the phenomena generated by using the system to decide the values of *all* the design parameters in helical compression spring design. Advisors and estimators will need to be included in predictions. Advisors also raise the problem of a new type of learning: Assuming that advisor *A* advises selector *S,* agent *B* which interacts with *S* will receive answers from *S,* which will reflect the reasoning of *A and S*. Estimators are used for preliminary estimates in solving constraints between design parameters. An estimator attached to a parameter provides infor-

mation to a group of agents attached to another parameter. The use of estimators will generate situations in which the learning agent and the agent learned about belong to agent groups associated with different parameters. Another important issue is raised by the use of the methodology for design parameters which span over continuous ranges. Discretization of the domain (which is close to dimensional standardization in design) would be the most likely approach to take.

SiFAs, using their relatively simple negotiation schemes, provide good opportunities to analyze learning in multi-agent systems. The work done so far indicates that learning can help make the interactions smoother for the most frequent design ranges. However, learning experiments will provide more insight into the efficiency of various improvement schemes.

5. References

Douglas, R.E., Brown, D.C., and Zenger, D.C. 1993. A Concurrent Engineering Demonstration and Training System for Engineers and Managers. *International Journal of CADCAM and Computer Graphics*, special issue: "AI and Computer Graphics", I. Costea (ed.), 8(3):263-301.

Dunskus, B.V., Grecu, D.L., Brown, D.C., and Berker, I. 1995. Using Single Function Agents to Investigate Conflicts. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, special issue: "Conflict Management in Design", I. Smith (ed.), 9(4):299-312, Cambridge University Press.

Gennari, J.C., Langley, P., and Fisher, D. 1990. Models of Incremental Concept Formation". In: Carbonell, J. ed. *Machine Learning – Paradigms and Methods*, Cambridge, MA: MIT Press.

Grecu, D.L., and Brown, D.C. 1996. Design Agents That Learn. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, special issue: "Machine Learning in Design", Duffy, A. H. B., Maher, M. L. and Brown, D. C. eds. Cambridge University Press. Forthcoming.

Klein, M. 1991. Supporting Conflict Resolution in Cooperative Design Systems. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6):1379-1390.

Kuokka, D.R., McGuire, J.G., Pelavin, R.N., Weber, H.C., Tenenbaum, H.M., Gruber, T., and Olsen, G. 1993. SHADE: Technology for Knowledge-Based Collaborative Engineering. *Concurrent Engineering: Research and Applications*, 1:137-146.

Lander, S.E., and Lesser, V.R. 1991. Customizing Distributed Search Among Agents with Heterogeneous Knowledge. In Proceedings of the 5th International Symposium on AI Applications in Manufacturing and Robotics, Cancun, Mexico.

Michalski, R.S. 1983. A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, 20:110-156.

Michalski, R.S., Mozetic, I., Hong J., Lavrac, N. 1986. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In Proc. of AAAI-86, 1041-1045. Los Altos, CA: Morgan Kaufmann.

NagendraPrasad, M.V., Lesser, V., and Lander, S.E. 1995. Learning Experiments in a Heterogeneous Multi-Agent System. In Working Notes of the IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems, Sen, S. ed., 59-64, Montréal, Canada.

Sen S. ed. 1995. Working Notes of the IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems, Montréal, Canada.

Victor, S.K., and Brown, D.C. 1994. Designing with Negotiation Using Single Function Agents. In: Rzevski, G., Adey, R.A., and Russell, D.W. eds. *Applications of Artificial Intelligence in Engineering IX*. Proc. of the 9th International AI in Engineering Conference, 173-179, Pennsylvania, USA, Computational Mechanics Publ.

Werkman, K.J., and Barone, M. 1992. Evaluating Alternative Connection Designs Through Multiagent Negotiation. In: Sriram, D., Logcher, R., and Fukuda, S. eds. *Computer Aided Cooperative Product Development*, Lecture Notes Series, 492:298-333, Springer Verlag.